Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Κβαντικός Υπολογισμός: Αλγόριθμοι, Υλοποιήσεις, και Εφαρμογές στην Κρυπτογραφία και στην Θεωρία Γράφων

Quantum Computing: Algorithms, Implementations, and Applications to Cryptography and Graph Theory

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΡΗΣΤΟΣ ΤΣΟΥΦΗΣ

Επιβλέπων : Αριστείδης Παγουρτζής
Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2023

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Κβαντικός Υπολογισμός: Αλγόριθμοι, Υλοποιήσεις, και Εφαρμογές στην Κρυπτογραφία και στην Θεωρία Γράφων

Quantum Computing: Algorithms, Implementations, and Applications to Cryptography and Graph Theory

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΧΡΗΣΤΟΣ ΤΣΟΥΦΗΣ

Επιβλέπων : Αριστείδης Παγουρτζής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 27η Σεπτεμβρίου 2023.

....	....	....
Αριστείδης Παγουρτζής Καθηγητής Ε.Μ.Π.	Δημήτριος Φωτάκης Καθηγητής Ε.Μ.Π.	Νικόλαος Παπασπύρου Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2023

....

Χρήστος Τσούφης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

.

Περίληψη

Σε αυτή τη διπλωματική εργασία, εξετάζουμε τον Κβαντικό Υπολογισμό μέσω ποικίλων υλοποιήσεων και εφαρμογών. Οι κβαντικοί αλγόριθμοι παρέχουν σημαντικά πλεονεκτήματα σε σχέση με τους κλασσικούς αλγόριθμους καθώς μπορούν να βοηθήσουν στην επίλυση προβλημάτων που είναι ακόμη και εγγενώς δύσκολο να λυθούν με τον κλασσικό υπολογισμό.

Συγκεκριμένα, πρώτον, εμβαθύνουμε στην βασική κβαντική σημειογραφία και ύστερα εξετάζουμε τα μαθηματικά εργαλεία που θα χρειαστούμε. Έπειτα, δείχνουμε ορισμένα αμιγώς κβαντικά πρωτόκολλα όπως η Υπερπυκνή Κωδικοποίηση και η Κβαντική Τηλεμεταφορά. Στη συνέχεια, για τους αλγόριθμους Deutsch και Deutsch-Jozsa, τον αλγόριθμο Bernstein-Vazirani, τον αλγόριθμο του Simon, τον αλγόριθμο Shor και τον αλγόριθμο του Grover, αναλύουμε βήμα προς βήμα το πρόβλημα, την επίλυση τόσο στην περίπτωση των κλασσικών υπολογιστών όσο και στην περίπτωση των κβαντικών υπολογιστών, υλοποιούμε και διεξάγουμε αριθμητικά πειράματα και αξιολογήσεις χρησιμοποιώντας την βιβλιοθήκη Qiskit της Python και τρέχουμε τα πειράματά μας σε προσομοιωτές αλλά και σε πραγματικούς Κβαντικούς Υπολογιστές (IBM Quantum, D-Wave, κ.λπ.) και τέλος, εξετάζουμε ορισμένες εφαρμογές στον πραγματικό κόσμο.

Πιο αναλυτικά, ξεκινάμε με τον αλγόριθμο Deutsch & Deutsch-Jozsa ο οποίος προσδιορίζει εάν μια συνάρτηση είναι σταθερή ή ισορροπημένη. Στη συνέχεια, εξετάζουμε τον αλγόριθμο του Bernstein-Vazirani, που είναι μια ειδική περίπτωση του προηγούμενου αλγορίθμου, ο οποίος βρίσκει μια συμβολοσειρά που ικανοποιεί ορισμένες ιδιότητες. Ύστερα, μελετάμε τον αλγόριθμο του Simon ο οποίος βρίσκει επίσης μια συμβολοσειρά κάτω από ορισμένες παραδοχές και μάλιστα, αυτός ο αλγόριθμος αποτέλεσε την έμπνευση πίσω από τον αλγόριθμο του Shor, για τον οποίο αναλύουμε τον τρόπο με τον οποίο επιλύεται το πρόβλημα παραγοντοποίησης ακέραιων αριθμών. Τέλος, ασχολούμαστε με τον αλγόριθμο του Grover, ο οποίος παρέχει έναν αλγόριθμο κβαντικής αναζήτησης και τον χρησιμοποιούμε για την επίλυση κλασσικών προβλημάτων αναζήτησης. Συγκεκριμένα, εξετάζουμε το πρόβλημα Max-Cut και αξιοποιούμε τον αλγόριθμο του Grover αλλά και κάποιες ακόμη τεχνικές για την επίλυση αυτού του προβλήματος. Τέλος, μετά από κάθε ανάλυση από τους παραπάνω αλγορίθμους, ολοκληρώνουμε την μελέτη μας συζητώντας σύγχρονες εφαρμογές, άλλοτε απλές, όπως η εύρεση μιας συμβολοσειράς και άλλοτε πιο σύνθετες, όπως η επίθεση στο κρυπτογραφικό πρωτόκολλο RSA.

Λέξεις-Κλειδιά: Κρυπτογραφία, Κβαντικός Υπολογισμός, Κβαντικοί Αλγόριθμοι, Θεωρία Γράφων

Abstract

In this thesis, we examine Quantum Computing through implementations and applications. Quantum algorithms provide significant advantages over classical algorithms and they can help us solve problems that are even inherently difficult to solve with classical computing.

Specifically, first, we delve into some quantum notation and consider the tools we will need. Next, we show some inherently quantum protocols such as Superdense Coding and Quantum Teleportation. Then, for the Deutsch and Deutsch-Jozsa algorithms, the Bernstein-Vazirani algorithm, Simon's algorithm, Shor's algorithm, and Grover's algorithm, we analyze a step by step solution in both the case of classical and quantum computers and we conduct numerical experiments and evaluations using Python's Qiskit library and we also run our experiments on simulators as well as on real Quantum Computers (IBM Quantum, D-Wave, etc.) and finally, we examine some real-world applications.

In more detail, we start with the Deutsch's & Deutsch-Jozsa's algorithm which determines whether a function is constant or balanced. Next, we look at the Bernstein-Vazirani's algorithm, which is a special case of the previous algorithm, where it finds a string that satisfies certain criteria. After that, we study Simon's algorithm which also finds a string under certain assumptions and in fact, this algorithm led to the inspiration behind Shor's algorithm, which we analyze how it solves the Integer Factorization Problem. Finally, we deal with Grover's algorithm, which provides a Search algorithm, and we use it to solve classical search problems such as the Max-Cut problem, and we even consider additional ways to solve this problem using other settings. Finally, after analyzing each of the above algorithms, we conclude our study by discussing modern applications, sometimes simple, such as finding a secret string, and sometimes more complex, such as attacking the RSA cryptographic protocol.

Keywords: Cryptography, Quantum Computing, Quantum Algorithms, Graph Theory

Ευχαριστίες

Αρχικά, θέλω να ευχαριστήσω από καρδιάς τον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας, κ. Αριστείδη Παγουρτζή, τόσο για την επίβλεψη, όσο και για τη διαρκή βοήθεια, τις συνεχείς συμβουλές και την καθοδήγηση που μου παρείχε, σε όποιο σημείο χρειαζόταν, με εύστοχες παρεμβάσεις. Αποτέλεσε, πέρα από επιβλέποντα της Διπλωματικής μου, μέντορα στα πρώτα μου ερευνητικά βήματα καθώς μέσα από την επικοινωνία μας έμαθα πως να ψάχνω, να αναρωτιέμαι, να απαντώ, να υλοποιώ και να ερευνώ ώστε να έχω το καλύτερο δυνατό αποτέλεσμα που μπορώ. Ένας καθηγητής ο οποίος στάθηκε δίπλα μου και με στήριξε τόσο σε αυτή την εργασία όσο και με τις επιλογές για τις σπουδές μου, βοηθώντας με πάντα να θέτω στόχους για να συνεχίσω το έργο μου και να παίρνω τις καλύτερες δυνατές αποφάσεις και για αυτό τον ευχαριστώ θερμά διότι όλα μου αυτά τα βήματα δεν θα ήταν εφικτά χωρίς την βαρύτητα που έδωσε σε αυτή μου την προσπάθεια. Αυτή η συνεργασία μας είναι ανεκτίμητη για μένα γιατί ξέρω ότι έχω χτίσει γερά θεμέλια υπό την καθοδήγησή του ώστε να μπορέσω να ανταπεξέλθω σε νέες προκλήσεις στο μέλλον.

Επιπλέον, θέλω να ευχαριστήσω θερμά τον κ. Δημήτριο Φωτάκη για την επίσης πολύτιμη βοήθεια και συμβολή του σε όλα μου τα ακαδημαϊκά χρόνια με τις χρήσιμες ιδέες, συμβουλές και παρατηρήσεις του. Ο κ. Φωτάκης είναι ένας άνθρωπος που όποτε τον χρειάστηκα ήταν δίπλα μου και με συμβούλευε, γνωρίζοντας εκ των προτέρων ότι θα μου μιλήσει με ειλικρίνεια και σοβαρότητα. Ένας εκπληκτικός καθηγητής και επιστήμονας, που πάντοτε άκουγα ό,τι μου έλεγε με πλήρη προσοχή. Ένας ευγενέστατος άνθρωπος, ο οποίος πάντοτε είχε χρόνο να με ακούσει αλλά και να μου εξηγήσει τα πάντα λεπτομερώς και με σαφήνεια.

Στο σημείο αυτό, θέλω να απονείμω ιδιαίτερες ευχαριστίες στον κ. Νικόλαο Παπασπύρου, ένα καθηγητή που από τα πρώτα χρόνια σπουδών στην σχολή, με την εξαιρετική μεταδοτικότητα στη διδασκαλία του με μύησε στον κόσμο της Επιστήμης των Υπολογιστών, μέχρι και τα μεγαλύτερα έτη, όπου με έμαθε να προσεγγίζω με σωστό και μεθοδικό τρόπο την λύση σε ένα τεχνικό πρόβλημα και για αυτό νιώθω ευγνώμων που ήμουν μαθητής του.

Επιπροσθέτως, θέλω να ευχαριστήσω όλους τους καθηγητές με τους οποίους συναναστράφηκα και μου μετέδωσαν αρκετά πράγματα καθ' όλη τη διάρκεια των σπουδών μου. Η συμβολή τους είναι ιδιαίτερα πολύτιμη και χαίρομαι πραγματικά που είχα την τιμή να υπάρξω φοιτητής τόσο αξιόλογων επιστημόνων. Η παρούσα εργασία σημαίνει το τέλος ενός πολύ σημαντικού κύκλου της ζωής μου, της φοίτησής μου στην Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου.

Τέλος, θέλω να ευχαριστήσω θερμά την οικογένεια μου και τους φίλους μου - εντός και εκτός του Πανεπιστημίου. Πιο συγκεκριμένα, ένα μεγάλο ευχαριστώ, πρώτα στους γονείς μου, οι οποίοι πέρα από το γεγονός ότι πάντα με στήριζαν σε όλες τις αποφάσεις μου, φρόντισαν να με συμβουλεύουν πάντα για το καλύτερο και να με καθοδηγούν με το σωστότερο τρόπο όπου έκριναν ότι απαιτείται. Θεωρώ ότι χωρίς αυτούς, δε θα ήμουν ο άνθρωπος που είμαι σήμερα, και με έχουν βοηθήσει ιδιαίτερα τόσο στην ακαδημαϊκή και σχολική μου πορεία, όσο και στην διάπλαση του χαρακτήρα μου και του ήθους μου. Επίσης, ευχαριστώ ακόμα όλα τα υπόλοιπα μέλη της οικογένειάς μου που πάντα ήταν και είναι δίπλα μου και νοιάζονται πραγματικά για εμένα. Επιπλέον, ένα ευχαριστώ θα πω και στους φίλους μου, τους οποίους έχω δίπλα μου τόσα χρόνια. Αισθάνομαι βαθιά συγκινημένος για τη στήριξη και την αγάπη που έχω λάβει απο τον περίγυρό μου και ελπίζω να τους κάνω υπερήφανους.

Χρήστος Τσούφης,

Αθήνα, 27η Σεπτεμβρίου 2023

Περιεχόμενα

Κεφάλαιο 1

Εκτεταμένη Περίληψη στα Ελληνικά

Σε αυτό το κεφάλαιο περιγράφονται, πιο συνοπτικά από τα επόμενα κεφάλαια, οι ερευνητικές έννοιες της εργασίας. Έχει γίνει προσπάθεια το περιεχόμενο αυτού του κεφαλαίου να είναι όσο το δυνατόν πιο επεξηγηματικό και να δίνει βάση σε όλες τις έννοιες που αναλύονται στην εργασία. Ωστόσο, σε περίπτωση που περαιτέρω ανάλυση κριθεί απαραίτητη από τον αναγνώστη, συνίσταται η επιπλέον ανάγνωση των επόμενων κεφαλαίων.

1.1 Προηγούμενη Έρευνα & Κίνητρο για αυτή την εργασία

Σημειώνουμε ότι ο τομέας του Κβαντικού Υπολογισμού είναι ένας νέος τρόπος υπολογισμού, ο οποίος διαφέρει από τον τρόπο που πραγματοποεί διεργασίες ο Κλασικός Υπολογισμός, και ο στόχος της παρούσας εργασίας είναι να αναδείξει τα πλεονεκτήματά του. Η έρευνα και η βιβλιογραφία που υπάρχει σε αυτό τον τομέα συνεχώς ανανεώνεται και εμπλουτίζεται και εμείς βασιστήκαμε σε αυτές τις πληροφορίες για να αναλύσουμε τους βασικούς Κβαντικούς Αλγορίθμους και να τους υλοποιήσουμε και εφαρμόσουμε στην Κρυπτογραφία αλλά και στην Θεωρία Γράφων.

1.2 Εισαγωγή στην Επιστήμη Υπολογιστών

Ο **Υπολογιστής** είναι μια φυσική μηχανή που μας βοηθάει να επεξεργαστούμε πληροφορίες εκτελώντας αλγορίθμους.

Ως **Αλγόριθμος** ορίζεται μια πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος.

Με άλλα λόγια, ο αλγόριθμος είναι μια σαφώς ορισμένη διαδικασία, με πεπερασμένα βήματα, υλοποιήσιμα σε υπολογιστή, που επεξεργάζεται πληροφορίες με σκοπό να λυθεί μια κλάση προβλημάτων ή να εκτελεστεί ένας υπολογισμός.

Μια διαδικασία **επεξεργασίας πληροφορίας** μπορεί να μεταφραστεί σε μια φυσική διεργασία.

Είναι σημαντικό να έχουμε υπόψιν ότι όταν σχεδιάζουμε αλγορίθμους, δουλεύουμε με κάποιο ιδανικό/ιδεατό υπολογιστικό μοντέλο αλλά στην πραγματικότητα, όταν το υλοποιούμε, έχουμε περιορισμούς εξαιτίας της σχέσης μεταξύ θεωρητικού υπολογισμού και φυσικής.

Επιπροσθέτως, εξηγούμε ότι η απόδοση και η ταχύτητα των αλγορίθμων μελετάται με όρους **Υπολογιστικής Πολυπλοκότητας**. Έτσι, αν υποθέσουμε ότι το μέγεθος των αντικειμένων της εισόδου είναι n, τότε ένας αλγόριθμος με πλήθος βημάτων ανάλογο του n θεωρείται αλγόριθμος **Γραμμικής πολυπλοκότητας**, ένας αλγόριθμος με πλήθος βημάτων ανάλογο του n^2 θεωρείται αλγόριθμος **Τετραγωνικής Πολυπλοκότητας**, ένας αλγόριθμος με πλήθος βημάτων ανάλογο του $\log n$ θεωρείται αλγόριθμος **Λογαριθμικής Πολυπλοκότητας** κ.ο.κ.. Συνεπώς, ένας αλγόριθμος με πλήθος βημάτων με μεγιστοβάθμιο όρο το πολύ ανάλογο του n^k θεωρείται αλγόριθμος **Πολυωνυμικής Πολυπλοκότητας**. Σημειώνουμε ότι οι αλγόριθμοι Λογαριθμικής Πολυπλοκότητας θεωρούνται αλγόριθμοι Πολυωνυμικής Πολυπλοκότητας, επειδή ο χρόνος εκτέλεσής τους φράσσεται ασυμπτωτικά από χρόνους Πολυωνυμικής Πολυπλοκότητας. Ωστόσο, αλγόριθμοι πλήθους βημάτων ανάλογο του 2^n θεωρούνται αλγόριθμοι **Εκθετικής Πολυπλοκότητας**.

1.3 Κβαντικός Υπολογισμός

1.3.1 Εισαγωγή στις Βασικές Έννοιες

Ο **Κβαντικός Υπολογισμός** είναι ένα υπολογιστικό υπόδειγμα που εκμεταλλεύεται τις ιδιότητες της Κβαντικής Μηχανικής ώστε να πραγματοποιήσει υπολογισμούς.

Στην περίπτωση του Κβαντικού Υπολογισμού, για να πραγματοποιήσουμε διεργασίες, χρησιμοποιούμε την **Κβαντική Θεωρία** και οι υπολογιστές που την υλοποιούν λέγονται **Κβαντικοί Υπολογιστές**, οι οποίοι δέχονται ως δεδομένα τα κβαντικά bits, οι πράξεις που πραγματοποιούν γίνονται με κβαντικές πύλες (ορθομοναδιαίοι/μοναδιακοί μετασχηματισμοί) και τέλος, τα αποτελέσματα προκύπτουν από την πραγματοποίηση μετρήσεων. Έπειτα, σημειώνουμε ότι υπάρχουν διάφορα μοντέλα Κβαντικού Υπολογισμού (Κβαντικές Μηχανές Turing, Κβαντικά Κυκλώματα, Μέτρηση Βασισμένη σε Κβαντικό Υπολογισμό, Αδιαβατικός Κβαντικός Υπολογισμός, Τοπολογικός Κβαντικός Υπολογισμός) και όσον αφορά τις υπολογιστικές τους δυνατότητες είναι ισοδύναμα με τα κλασσικά μοντέλα (Μηχανές Turing). Τέλος, αναφέρουμε ότι υπάρχουν διάφορα εργαλεία που μπορεί να χρησιμοποιήσει κανείς για να προγραμματίσει ένα κβαντικό υπολογιστή. Στην παρούσα εργασία, χρησιμοποιήθηκε το Qasm, η Qiskit βιβλιοθήκη της Python, οι Κβαντικοί Υπολογιστές της IBM, το Cirq της Google, η πλατφόρμα Leap της D-Wave αλλά υπάρχουν και άλλα εργαλεία που μπορεί κανείς να χρησιμοποιήσει.

1.3.2 Χρήσιμα μαθηματικά εργαλεία και συμβολισμοί

Σε αυτό το σημείο, θα αναλύσουμε βασικούς και χρήσιμους συμβολισμούς που θα χρησιμοποιήσουμε από την κβαντική υπολογιστική θεωρία, χρήσιμα εργαλεία από την Γραμμική Άλγεβρα και την Κβαντική Φυσική στα οποία θα βασιστούμε για την ανάλυση των μοντέλων μας.

Κλασσικά Bits & Κβαντικά Bits

Είναι γνωστό ότι τα **κλασσικά bits** είναι διακριτά και μπορούν να πάρουν την τιμή 0 ή 1. Αντιθέτως, τα **κβαντικά bits** ή **qubits** είναι συνεχή και μπορούν να πάρουν άπειρες τιμές. Τα qubits επίσης, εντοπίζονται στο διανυσματικό χώρο του Χίλμπερτ με βάση δύο στοιχεία που θα συμβολίζουμε με $|0\rangle$ και $|1\rangle$, τα οποία αποτελούν την θεμελιώδη μονάδα για την Κβαντική Πληροφορία.

Επιπλέον, αν ένα bit είναι στην κατάσταση 0 με πιθανότητα p_0 και στην κατάσταση 1 με πιθανότητα p_1, τότε μπορούμε να συνοψίσουμε αυτή την πληροφορία χρησιμοποιώντας ένα διδιάστατο διάνυσμα πιθανοτήτων της μορφής $\begin{bmatrix} p_0 & p_1 \end{bmatrix}^T$. Έτσι, μπορούμε να ορίσουμε την κατάσταση 0 όταν έχουμε $p_0 = 1, p_1 = 0$ και την κατάσταση 1 όταν έχουμε $p_0 = 0, p_1 = 1$, και αν χρησιμοποιήσουμε τον **Dirac συμβολισμό bra-ket**, τότε

$$0 \to \text{"ket" συμβολισμός: } |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; 1 \to \text{"ket" συμβολισμός: } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Τα **qubits** λοιπόν, μπορούν να βρίσκονται είτε στην κατάσταση 0 είτε στην κατάσταση 1, αλλά εξαιτίας της **Αρχής της Υπέρθεσης**, τα qubits μπορεί να βρίσκονται και σε υπέρθεση του 0 και του 1 ταυτόχρονα μέχρις ότου κάποιος πραγματοποιήσει μέτρηση. Αυτή η ιδιότητα της Υπέρθεσης επιτρέπει στα qubits να εκτελούν κάποιους τύπους υπολογισμών εκθετικά γρηγορότερα από ότι τα κλασσικά bits, σε ορισμένα προβλήματα. Η **Αρχή της Υπέρθεσης** λέει ότι αν ένα κβαντικό σύστημα μπορεί να είναι σε μια από τις δύο καταστάσεις, τότε μπορεί επίσης να βρίσκεται και σε οποιαδήποτε γραμμική υπέρθεση αυτών των καταστάσεων. Μάλιστα, οποιοδήποτε διάνυσμα $|\psi\rangle$ μπορεί να γραφτεί ως γραμμικός συνδυασμός των διανυσμάτων βάσης (καταστάσεις βάσης) $|0\rangle, |1\rangle$ ως εξής: $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, όπου, οι συντελεστές α, β λέγονται **πλάτη** και γενικά είναι μιγαδικοί αριθμοί που ικανοποιούν την εξίσωση $|\alpha|^2 + |\beta|^2 = 1$.

Μια ακόμη έννοια που θα μας απασχολήσει είναι η έννοια της **κβαντικής κατάστασης**. Έτσι ονομάζεται η μαθηματική περιγραφή που αντιπροσωπεύει την πλήρη πληροφορία για το κβαντικό σύστημα, συμπεριλαμβανομένης της κατάστασης όλων των qubits εντός του συστήματος. Επίσης, περιγράφει τα πλάτη, και επομένως και τις πιθανότητες όλων των δυνατών αποτελεσμάτων, όταν τελικά πραγματοποιηθεί μέτρηση. Μια **κατάσταση βάσης** είναι ένα διάνυσμα στήλης, όπου το $|x\rangle$ αναθέτει το βάρος 1 στην θέση του x και το 0 οπουδήποτε αλλού. Έτσι, για ένα n-bit αριθμό της μορφής $x = x_1 ... x_n \in \{0, 1\}^n$ υπάρχουν 2^n πιθανές συμβολοσειρές. Συνεπώς, για την διανυσματική αναπαράσταση, θα έχουμε ότι $x \to |x\rangle = \begin{bmatrix} 0 & ... & 0 & 1 & 0 & ... & 0 \end{bmatrix}^T$. Το σύνολο όλων αυτών των καταστάσεων αποτελούν μια ορθοκανονική βάση που ονομάζεται και **υπολογιστική βάση**.

Πράξεις και Ιδιότητες

- **"bra" & "ket" συμβολισμός**: $|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix} \quad \langle\psi| = (|\psi\rangle^*)^T = \begin{bmatrix} a^* & b^* \end{bmatrix}$
- **Εσωτερικό Γινόμενο**: $\langle\psi| \, |\psi\rangle = \langle\psi|\psi\rangle = \begin{bmatrix} a^* & b^* \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = a^* a + b^* b = |a|^2 + |b|^2$

- **Τανυστικό γινόμενο**: Έστω δυο qubits A, B. Μπορούμε να περιγράψουμε την κατάσταση A ως $|\psi\rangle_A$ και την B ως $|\psi\rangle_B$. Έτσι, για να γράψουμε την συνδυασμένη κατάσταση $|\psi\rangle_{AB}$ από τα A, B τότε παίρνουμε το τανυστικό γινόμενο (γνωστό και ως γινόμενο Kronecker). Έτσι, για $|\psi\rangle_A = \alpha_A |0\rangle_A + \beta_A |1\rangle_A$ και $|\psi\rangle_B = \alpha_B |0\rangle_B + \beta_B |1\rangle_B$ η συνδυασμένη κατάσταση $|\psi\rangle_{AB} \in \mathbb{C}^2 \otimes \mathbb{C}^2$ μπορεί να εκφραστεί ως το τανυστικό γινόμενο των διανυσμάτων $|\psi\rangle_A, |\psi\rangle_B$:

$$|\psi\rangle_{AB} = |\psi\rangle_A \otimes |\psi\rangle_B = \begin{bmatrix} \alpha_A \\ \beta_A \end{bmatrix} \otimes |\psi\rangle_B = \begin{bmatrix} \alpha_A |\psi\rangle_B \\ \beta_A |\psi\rangle_B \end{bmatrix} = \begin{bmatrix} \alpha_A\alpha_B & \alpha_A\beta_B & \beta_A\alpha_B & \beta_A\beta_B \end{bmatrix}^T$$

  Επίσης, συχνά χρησιμοποιούνται οι ακόλουθοι συμβολισμοί:

  - $|\psi\rangle_A \otimes |\phi\rangle_B = |\psi\rangle_A |\phi\rangle_B$
  - $|0\rangle_A \otimes |0\rangle_B = |0\rangle_A |0\rangle_B = |00\rangle_{AB}$
  - $|\psi\rangle_1 \otimes |\psi\rangle_2 \otimes \cdots \otimes |\psi\rangle_n = |\psi\rangle^{\otimes n}$

  Και ισχύουν οι εξής ιδιότητες:

  - **επιμεριστική**: $|\psi\rangle \otimes (|v_1\rangle + |v_2\rangle) = |\psi\rangle \otimes |v_1\rangle + |\psi\rangle \otimes |v_2\rangle$
  - **προσεταιριστική**: $|\psi\rangle \otimes (|\phi\rangle \otimes |\gamma\rangle) = (|\psi\rangle \otimes |\phi\rangle) \otimes |\gamma\rangle$
  - **όχι αντιμεταθετική** (εκτός κι αν είναι ίδιες καταστάσεις): $|\psi\rangle \otimes |\phi\rangle \neq |\phi\rangle \otimes |\psi\rangle$

Σύνθεση Κβαντικών Συστημάτων - Συνδυασμός Qubits

Για n qubits η κατάσταση $|\psi\rangle \in \mathbb{C}^d$, $d = 2^n$ μπορεί να γραφτεί ως υπέρθεση των στοιχείων την κανονικής βάσης ως $|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$ όπου $\forall x, \alpha_x \in \mathbb{C}$ και $\sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1$.

Επομένως, μπορούμε να γενικεύσουμε λέγοντας ότι αν μας δοθούν δύο διανύσματα $|\psi_1\rangle \in \mathbb{C}^{d_1}$ και $|\psi_2\rangle \in \mathbb{C}^{d_2}$ τότε, το τανυστικό γινόμενό τους θα είναι $|\psi_1\rangle \otimes |\psi_2\rangle = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_d \end{bmatrix} \otimes |\psi_2\rangle = \begin{bmatrix} \alpha_1 |\psi_2\rangle \\ \vdots \\ \alpha_d |\psi_2\rangle \end{bmatrix}$ και $|\psi_1\rangle \otimes |\psi_2\rangle \in \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$.

Σημείωνουμε λοιπόν ότι εάν συνδυάσουμε δύο qubits A, B, τότε η συνδυασμένη κατάσταση δίνεται από το $|\psi\rangle_{AB} = |\psi_1\rangle_A \otimes |\psi_2\rangle_B$. Οπότε, οποιαδήποτε δύο-qubit κατάσταση που είτε είναι αυτής της μορφής είτε είναι μια ανάμειξη καταστάσεων αυτής της μορφής, καλείται διαχωρίσιμη. Ωστόσο, υπάρχει και η περίπτωση της συμπλεγμένης κατάστασης, της κατάστασης δηλαδή που δεν είναι διαχωρίσιμη. Με άλλα λόγια, μια κατάσταση $|\psi\rangle$ είναι συμπλεγμένη αν και μόνο αν $|\psi\rangle \neq |\psi_1\rangle \otimes |\psi_2\rangle$ για οποιαδήποτε πιθανή επιλογή των $|\psi_1\rangle, |\psi_2\rangle$.

Μέτρηση των Qubits

Για να μπορέσουμε να μάθουμε την τιμή ενός qubit, πρέπει να πραγματοποιήσουμε **μέτρηση**, συνήθως στην κανονική βάση. Έτσι, για την κατάσταση $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, θα έχουμε:

$$\text{Αποτέλεσμα } |0\rangle\text{: } p_0 = |\langle\psi|0\rangle|^2 = \left|\begin{bmatrix} a^* & b^* \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right|^2 = |a|^2$$

$$\text{Αποτέλεσμα } |1\rangle\text{: } p_1 = |\langle\psi|1\rangle|^2 = \left|\begin{bmatrix} a^* & b^* \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right|^2 = |b|^2$$

Οι πιθανότητες πρέπει να ικανοποιούν την συνθήκη $p_0 + p_1 = 1 = |a|^2 + |b|^2$. Η νέα κατάσταση μετά την μέτρηση θα είναι $|0\rangle$ ή $|1\rangle$, και θα εξαρτάται από το αποτέλεσμα που προέκυψε (**διάλυση της κυματοσυνάρτησης**).

Επιπλέον, υπάρχει το Θεώρημα No-cloning, σύμφωνα με το οποίο δεν μπορούμε να πραγματοποιήσουμε πολλές ανεξάρτητες μετρήσεις της κατάστασης $|\psi\rangle$ επειδή τα qubits δεν μπορούν να αντιγραφούν. Έτσι, ισχύει ότι καμία κβαντική διαδικασία δεν μπορεί να πετύχει τον μετασχηματισμό $|\phi\rangle \to |\phi\rangle|\phi\rangle, \forall |\phi\rangle$.

Πράξεις με Qubits & Κβαντικές Πύλες

Από ένα qubit $|\psi\rangle$, εφαρμόζοντας μια πράξη, μπορούμε να καταλήξουμε σε ένα άλλο qubit $|\psi_{new}\rangle$. Αυτή η πράξη λέγεται **ορθομοναδιαίος / μοναδιακός μετασχηματισμός** και στην πραγματικότητα, είναι ένας πίνακας U, που ικανοποιεί την εξίσωση $|\psi_{new}\rangle = U|\psi\rangle$. Επίσης, το διάνυσμα ψ θα πρέπει να είναι κανονικοποιημένο, δηλαδή $\langle\psi|\psi\rangle = 1$ και $\langle\psi_{new}|\psi_{new}\rangle = \langle\psi|U^\dagger U|\psi\rangle = 1$. Το U είναι ανεξάρτητο από το διάνυσμα, έτσι $U^\dagger U = \mathbb{I}$. Επίσης, το $U^\dagger$ είναι ο συζυγής ανάστροφος και ισχύει $UU^\dagger = \mathbb{I}$.

Στα **κβαντικά κυκλώματα**, οι πράξεις αυτές μπορούν να πραγματοποιηθούν με την χρήση ορθομοναδιαίων πινάκων και κάθε τέτοιος πίνακας είναι μια πιθανή **κβαντική πύλη**.

Όταν έχουμε 1 qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, μια 1-qubit πύλη μπορεί να γραφτεί ως ένας πίνακας $U = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$. Η κβαντική κατάσταση λοιπόν μετασχηματίζεται σε: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} a\alpha + b\beta \\ c\alpha + d\beta \end{bmatrix} = (a\alpha + b\beta)|0\rangle + (c\alpha + d\beta)|1\rangle$.

Γνωρίζουμε ότι όλες οι πράξεις είναι αντιστρέψιμες και ότι όλες οι πύλες έχουν τον ίδιο αριθμό εισόδων και εξόδων.

Συνεπώς, οποιαδήποτε ορθομοναδιαία πράξη που δρα πάνω σε ένα δισδιάστατο κβαντικό σύστημα, λέγεται **1-qubit κβαντική πύλη**. Τέτοιες πύλες είναι οι I, X, Y, Z, H, S, T. Οι X, Y, Z αντιστοιχούν σε στροφές γύρω από τους x, y, z άξονες αντίστοιχα, στην σφαίρα Bloch και λέγονται πύλες Pauli.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{bmatrix} \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$$

Σημειώνουμε επίσης ότι όταν έχουμε 2 qubits, υπάρχουν και οι **2-qubit πύλες** αλλά και γενικότερα υπάρχουν πύλες και για περισσότερα qubits. Επισημαίνεται σε αυτό το σημείο ότι κάποιες από αυτές τις πύλες μπορούν να γραφτούν ως το γινόμενο δύο 1-qubit πυλών, ενώ κάποιες άλλες όχι.

Στην συνέχεια, θα αναφέρουμε μερικά παραδείγματα για αυτές τις πύλες που περιγράψαμε.

- Η πύλη **Hadamard** H ορίζεται ως εξής $H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Η πύλη Hadamard έχει την ακόλουθη αναπαράσταση σε πίνακα $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. Μια χρήσιμη ιδιότητα της πύλης Hadamard είναι ότι είναι αντιστρέψιμη, δηλαδή $H = H^{-1}$. Έτσι, η H μετασχηματίζει την υπολογιστική βάση $\{|0\rangle, |1\rangle\}$ στην Hadamard βάση $\{|+\rangle, |-\rangle\}$

- Η πύλη **X** ορίζεται ως εξής: $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Υπολογίζουμε την X για 0/1: $X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$, $X|1\rangle = |0\rangle$. Επομένως, παρατηρούμε ότι η X δρα ως μια πράξη **αλλαγής bit** δηλαδή ως NOT.

- Η πύλη **Z** ορίζεται ως εξής: $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. Υπολογίζουμε το Z για 0/1: $Z|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$, $Z|1\rangle = -|1\rangle$. Συνεπώς, παρατηρούμε ότι η Z δρα ως μια πράξη **αλλαγής φάσης**. Επιπλέον, αν υπολογίσουμε πως δρα η Z στις καταστάσεις $|+\rangle$ και $|-\rangle$ (που αποτελούν την βάση Hadamard), θα προκύψει ότι $Z|+\rangle = |-\rangle$, $Z|-\rangle = |+\rangle$. Έτσι, εδώ η Z δρα ως μια πράξη **αλλαγής bit**.

- Η πύλη **controlled-**NOT, ή $CNOT$ έχει την ακόλουθη αναπαράσταση σε πίνακα $CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$. Στην υπολογιστική βάση, η πύλη $CNOT$ αλλάζει την κατάσταση του δεύτερου qubit αν το πρώτο qubit είναι στην κατάσταση $|1\rangle$, ενώ δεν κάνει τίποτα στις άλλες περιπτώσεις. Όπως σε όλες τις ορθομοναδιαίες πράξεις, η πύλη $CNOT$ δρα γραμμικά πάνω σε όλες τις υπερθέσεις. Έτσι, $|00\rangle \to |00\rangle \quad |01\rangle \to |01\rangle \quad |10\rangle \to |11\rangle \quad |11\rangle \to |10\rangle$.

1.3.3 Σύντομη εισαγωγή στην Κβαντομηχανική

Κβαντική σύμπλεξη

Υπάρχουν δύο αρχές στον Κβαντικό Υπολογισμό. Πρώτον, ένα φυσικό σύστημα σε μια ορισμένη κατάσταση μπορεί παρ'όλα αυτά να συμπεριφέρεται τυχαία. Δεύτερον, δύο συστήματα που είναι πολύ μακριά για να επηρεάσει το ένα το άλλο, μπορούν παρ'όλα αυτά να συμπεριφέρονται με τρόπους που, αν και ατομικά είναι τυχαίοι, είναι κατά κάποιον τρόπο ισχυρά συσχετισμένοι.

Η βασική ιδέα πίσω από την δεύτερη αρχή είναι η **κβαντική σύμπλεξη**.

Μέσα σε ένα άτομο υπάρχουν **ηλεκτρόνια** τα οποία χαρακτηρίζονται από τους εξής κβαντικούς αριθμούς: **κύριος κβαντικός αριθμός** (n), **αζιμουθιακός κβαντικός αριθμός** (ο οποίος περιγράφει την τροχιακή στροφορμή του ηλεκτρονίου) (l), **μαγνητικός κβαντικός αριθμός** (ο οποίος περιγράφει τη διεύθυνση του διανύσματος της στροφορμής) (m_l) και **κβαντικός αριθμός του σπιν** (ο οποίος περιγράφει τη διεύθυνση της εσωτερικής στροφορμής του ηλεκτρονίου) (m_s). Εν προκειμένω, μας ενδιαφέρει ο **κβαντικός αριθμός του σπιν** και μάλιστα, αν συσχετίσουμε το σπιν προς τα κάτω με το 0 και το σπιν προς τα πάνω με 1, τότε όταν 2 ηλεκτρόνια αλληλεπιδρούν μεταξύ τους, τότε λέμε ότι είναι σε κβαντική σύμπλεξη και τα σπιν τους είναι συσχετισμένα.

1.3.4 Η σφαίρα Bloch

Εδώ απλώς αναφέρουμε ότι υπάρχει και ένας ακόμη τρόπος να αναπαραστήσουμε τα μονά (single) qubits χρησιμοποιώντας την **σφαίρα Bloch**. Συγκεκριμένα, αν γράφουμε $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, όπου $\alpha, \beta \in \mathbb{Z}$, μπορούμε να βρούμε γωνίες έτσι ώστε: $|\psi\rangle = e^{i\gamma}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\phi}|1\rangle\right)$.

Η γενική φάση (global phase) $e^{i\gamma}$ μπορεί να αγνοηθεί χωρίς βλάβη της γενικότητας επειδή αν πραγματοποιήσουμε μέτρηση της κβαντικής κατάστασης, τότε οι πιθανότητες των αποτελεσμάτων που προκύπτουν από την μέτρηση δεν σχετίζονται με αυτή την φάση οπότε μπορούμε να γράψουμε $|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\phi}|1\rangle$.

1.4 Κβαντικά Πρωτόκολλα

1.4.1 Εισαγωγή στα Κβαντικά Πρωτόκολλα

Σε αυτό το σημείο εξετάζουμε τα πρωτόκολλα **Υπερπυκνή Κωδικοποίηση** & **Κβαντική Τηλεμεταφορά**. Είναι *αμιγώς κβαντικά* και περιλαμβάνουν δύο μέρη που θέλουν να επικοινωνήσουν μεταξύ τους, την "Alice" και τον "Bob".

Ένα **κανάλι** επικοινωνίας είναι ένας μαθηματικός μετασχηματισμός που προκύπτει στα bits/qubits όταν υποβάλλονται σε μια γενική κβαντική λειτουργία.

Ένα **κλασσικό κανάλι επικοινωνίας** είναι ένα κανάλι που μπορεί να μεταφέρει κλασσικά bits.

Ένα **κβαντικό κανάλι επικοινωνίας** είναι μια γραμμή επικοινωνίας που μπορεί να μεταφέρει qubits μεταξύ δύο απομακρυσμένων σημείων.

Επιπροσθέτως, πριν επικεντρώσουμε την προσοχή μας στα πρωτόκολλα, μια βασική προϋπόθεση που πρέπει να ικανοποιείται είναι η Alice και ο Bob αρχικά να μοιράζονται ένα ζεύγος από qubits που βρίσκονται σε κβαντική σύμπλεξη και αυτή η κατάσταση είναι γνωστή ως **κατάσταση Bell / ζέυγος EPR**: $|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

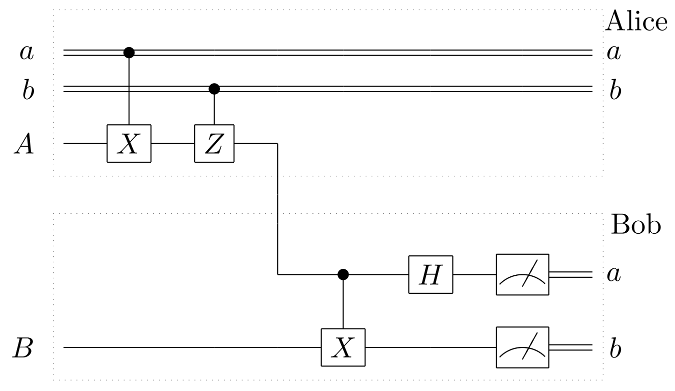
Για να κατασκευάσουμε μια Bell κατάσταση $|\beta_{00}\rangle$ με ένα απλό κύκλωμα, αρκεί να κάνουμε τα ακόλουθα:

1. Ξεκινάμε με 2 qubits $|0\rangle, |0\rangle = |00\rangle$.
2. Έπειτα, εφαρμόζουμε την πύλη H στο πρώτο qubit και παίρνουμε $\frac{|00\rangle+|10\rangle}{\sqrt{2}}$.
3. Μετά, εφαρμόζουμε την πύλη $CNOT$ και προκύπτει $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$.
4. Ύστερα, αν κάνουμε μέτρηση, θα πάρουμε 0 ή 1, το καθένα με πιθανότητα $p = \frac{1}{2}$. Αν υποθέσουμε ότι παίρνουμε 0, τότε η νέα κατάσταση θα είναι η $|00\rangle$. Συνεπώς, αν τώρα μετρήσουμε και το δεύτερο qubit, θα πάρουμε 0 με πιθανότητα $p = 1$. Επίσης, αν πάρουμε 1 στο πρώτο qubit, θα πάρουμε 1 και στο δεύτερο.

Συνεπώς, μετά την ολοκλήρωση των παραπάνω βημάτων, κάθε ένας θα έχει ένα qubit (έστω A, B) και αυτή η κβαντική σύμπλεξη γίνεται μια πηγή (*resource*) από bits την οποία η Alice και ο Bob μπορούν να χρησιμοποιήσουν για να πετύχουν τα πρωτόκολλα που θα περιγράψουμε.

1.4.2 Υπερπυκνή Κωδικοποίηση

Σε αυτή την περίπτωση, η Alice θέλει να επικοινωνήσει στον Bob 2 κλασσικά bits πληροφορίας (a, b) πάνω σε ένα κβαντικό κανάλι και για να το πετύχει αυτό, αρκεί απλώς η Alice να στείλει στον Bob 1 qubit. Το κύκλωμα που το υλοποιεί αυτό είναι το ακόλουθο.

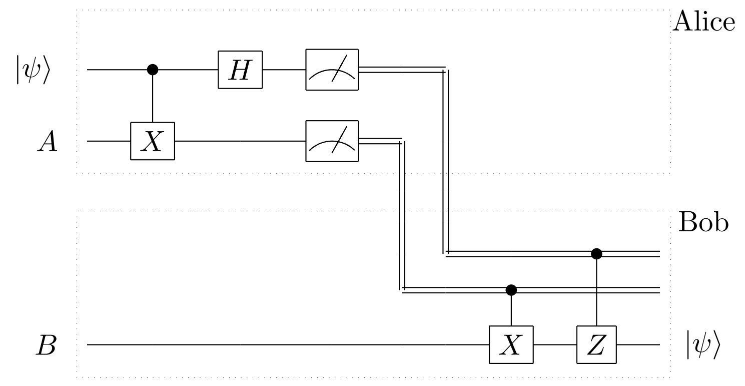


Και μπορούμε να το περιγράψουμε ως εξής:

- Αρχικά, η Alice και ο Bob μοιράζονται την κατάσταση Bell (ζεύγος (A, B) ή $|\beta 00\rangle$).
- Έπειτα, υποθέτουμε ότι η Alice έχει στην κατοχή της το πρώτο qubit και ο Bob το δεύτερο qubit. Εν συνεχεία, εκείνη εφαρμόζει μια από τις τέσσερις πύλες (I για 00, X για 01, Z για 10, $Z \cdot X$ για 11), ανάλογα με τα 2 κλασσικά bits που εκείνη επιθυμεί να επικοινωνήσει στον Bob, και μετά στέλνει το νέο qubit στον Bob.
- Ο Bob, με την σειρά του, κατέχει μια από τις τέσσερις καταστάσεις Bell, και ανάλογα με τα κλασσικό bits που η Alice θέλει να του στείλει, μπορεί να κάνει μέτρηση της ενωμένης 2-qubit κατάστασης, λαμβάνοντας υπόψη την Bell βάση. Αυτό το πετυχαίνει κάνοντας αλλαγή βάσης σε Bell βάση, και μετά κάνοντας μέτρηση στην υπολογιστική βάση.
- Το αποτέλεσμα την μέτρησης δείχνει στον Bob ποια κατάσταση Bell κατέχει και του επιτρέπει να καθορίσει με βεβαιότητα τα δυο κλασσικά bits που εκείνη ήθελε να του επικοινωνήσει.

1.4.3 Κβαντική Τηλεμεταφορά

Σε αυτή την περίπτωση, η Alice θέλει να επικοινωνήσει στον Bob την κατάσταση ενός qubit πάνω σε κλασσικό κανάλι και για να το πετύχει αυτό αρκεί η Alice να στείλει δύο κλασσικά bits πληροφορίας στον Bob. Το πρωτόκολλο αυτό μπορεί να θεωρηθεί ως η *ανεστραμένη εκδοχή* του προηγούμενου. Το κύκλωμα που το υλοποιεί αυτό είναι το ακόλουθο.



Και μπορούμε να το περιγράψουμε ως εξής:

- Αρχικά, μοιράζονται την κατάσταση Bell (ζεύγος (A, B) ή $|\beta 00\rangle$).
- Έπειτα, η Alice επιθυμεί να τηλεμεταφέρει την κατάσταση $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ στον Bob, οπότε η ενωμένη 3-qubit κατάσταση θα είναι

$$|\psi\rangle |\beta_{00}\rangle = \frac{1}{2} |\beta_{00}\rangle |\psi\rangle + \frac{1}{2} |\beta_{01}\rangle (X |\psi\rangle) + \frac{1}{2} |\beta_{10}\rangle (Z |\psi\rangle) + \frac{1}{2} |\beta_{11}\rangle (X \cdot Z |\psi\rangle)$$

- Μετά, η Alice μπορεί να πραγματοποιήσει μέτρηση στα πρώτα 2 qubits στην Bell βάση και η ενωμένη κατάσταση των Alice-Bob μετά την μέτρηση θα είναι μια εκ των $|\beta_{00}\rangle |\psi\rangle$, $|\beta_{01}\rangle (X |\psi\rangle)$, $|\beta_{10}\rangle (Z |\psi\rangle)$, $|\beta_{11}\rangle (X \cdot Z |\psi\rangle)$, καθεμία με πιθανότητα $p = \frac{1}{4}$.

- Τα κλασσικά bits a and b που προκύπτουν από την μέτρηση της Alice θα δείξουν ποια από τις τέσσερις καταστάσεις έχουμε. Έτσι, όταν η Alice στείλει αυτά τα δυο bits στον Bob, αυτός τα μαθαίνει και αναλόγως με το σε ποια κατάσταση βρίσκεται, εφαρμόζει μια πράξη (εφαρμόζει πύλες Z and X ανάλογα με τα a, b) για να μετασχηματίσει την κατάστασή του στην κατάσταση $|\psi\rangle$.
- Ύστερα από αυτό τον μετασχηματισμό, είναι εγγυημένο ότι θα έχει στην κατοχή του την κατάσταση $|\psi\rangle$ και έτσι, η κατάσταση λέμε ότι έχει τηλεμεταφερθεί επιτυχώς.

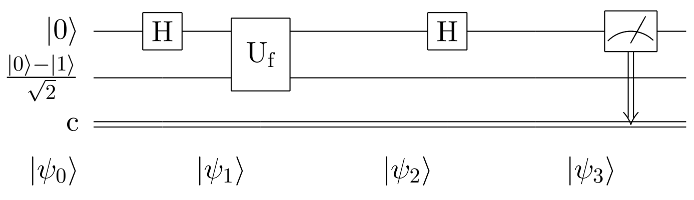
1.5 Κβαντικοί Αλγόριθμοι

Σε αυτή την ενότητα ασχολούμαστε με τους **Κβαντικούς Αλγορίθμους** οι οποίοι παρέχουν σημαντικά πλεονεκτήματα σε σχέση με τους κλασσικούς αλγορίθμους στο μοντέλο υπολογισμού ερωτήσεων (query model). Θα εξετάσουμε μερικούς από αυτούς στην συνέχεια. Συγκεκριμένα, θα ασχοληθούμε με τον αλγόριθμο **Deutsch** & **Deutsch-Jozsa**, τον αλγόριθμο **Bernstein-Vazirani**, τον αλγόριθμο του **Simon**, τον αλγόριθμο του **Shor** και τον αλγόριθμο του **Grover**.

1.5.1 Ο αλγόριθμος Deutsch & Deutsch-Jozsa

Αρχικά, αναφέρουμε το Πρόβλημα του Deutsch.

- **Το Πρόβλημα του Deutsch**:
  Είσοδος: Ένα μαύρο κουτί για τον υπολογισμό μιας συνάρτησης $f : \{0,1\} \to \{0,1\}$.
  Πρόβλημα: Καθορισμός της τιμής $f(0) \oplus f(1)$ κάνοντας ερωτήσεις (queries) στην f.
  Έξοδος: 0 αν f σταθερή (constant), 1 αν f ισορροπημένη (balanced).
- **Η επίλυση**:
  - **Κλασσική**: Χρειαζόμαστε 2 ερωτήσεις ώστε να μπορέσουμε να καθορίσουμε την τιμή του $f(0) \oplus f(1)$, μια για τον υπολογισμό του $f(0)$ και μια για τον υπολογισμό του $f(1)$.
  - **Κβαντική**: Χρειαζόμαστε μόνο 1 ερώτηση και αυτό επιτυγχάνεται με το ακόλουθο κύκλωμα.
- **Ανάλυση Κυκλώματος**:



Και μπορούμε να το περιγράψουμε αναφέροντας μετά από κάθε βήμα την κατάσταση ως εξής:

- Αρχικά, $|\psi_0\rangle = |0\rangle \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$.
- Έπειτα, $|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) = \frac{1}{\sqrt{2}}\left(|0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) + |1\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right)$.
- Οπότε,

$$\begin{aligned}|\psi_2\rangle = \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) + (-1)^{f(1)}|1\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right) =\\ = \left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) =\\ = (-1)^{f(0)}\left(\frac{|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\end{aligned}$$

- Έτσι, θα έχουμε τα εξής:
  * Αν η f είναι μια **σταθερή** (constant) συνάρτηση, $f(0)\oplus f(1) = 0$, τότε προκύπτει: $|\psi_3\rangle = (-1)^{f(0)}|0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. Η τετραγωνική νόρμα της κατάστασης βάσης $|0\rangle$ στο πρώτο qubit είναι 1 και ως αποτέλεσμα, η μέτρηση του πρώτου qubit είναι βέβαιο ότι θα επιστρέψει την τιμή $0 = f(0)\oplus f(1)$.
  * Αν η f είναι μια **ισορροπημένη** (balanced) συνάρτηση, $f(0)\oplus f(1) = 1$, τότε προκύπτει: $|\psi_3\rangle = (-1)^{f(0)}|1\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. Η τετραγωνική νόρμα της κατάστασης βάσης $|1\rangle$ στο πρώτο qubit είναι 1 και ως αποτέλεσμα, η μέτρηση του πρώτου qubit είναι βέβαιο ότι θα επιστρέψει την τιμή $1 = f(0)\oplus f(1)$.

Έπειτα, αναφέρουμε τον αλγόριθμο των Deutsch-Jozsa. Αυτός ο αλγόριθμος είναι μια γενίκευση του προβλήματος που λύνει ο Deutsch αλλά τώρα η f είναι μια συνάρτηση από n-bit συμβολοσειρές προς ένα μόνο bit.

- **Το Πρόβλημα των Deutsch-Jozsa**:
  Είσοδος: Ένα μαύρο κουτί για τον υπολογισμό μιας συνάρτησης $f : \{0,1\}^n \to \{0,1\}$.
  Υπόσχεση: Η f είναι είτε σταθερή είτε ισορροπημένη.
  Πρόβλημα: Καθορισμός του αν η f είναι σταθερή ή ισορροπημένη κάνοντας ερωτήσεις.
  Έξοδος: 0 αν η f είναι σταθερή, 1 αν η f είναι ισορροπημένη.
- **Η επίλυση**:
  - **Κλασσική**: Χρειαζόμαστε $2^{n-1}+1$ ερωτήσεις για τον καθορισμό της f.
  - **Κβαντική**: Και πάλι, χρειαζόμαστε μόνο 1 ερώτηση.

1.5.2 Ο αλγόριθμος Bernstein-Vazirani

- **Το Πρόβλημα Bernstein-Vazirani**:
  Είσοδος: Μια συνάρτηση $f : \{0,1\}^n \to \{0,1\}$
  Υπόσχεση: $\exists$ δυαδική συμβολοσειρά $s = s_{n-1}...s_0$, έτσι ώστε $f(x) = s\cdot x$, $\forall x \in \{0,1\}^n$
  Έξοδος: Καθορισμός της συμβολοσειράς s κάνοντας ερωτήσεις στην f.

- **Η Επίλυση**:
  - **Κλασσική**: Θα πρέπει να κάνουμε τουλάχιστον n ερωτήσεις χρησιμοποιώντας οποιονδήποτε κλασσικό αλγόριθμο ερωτήσεων.
  - **Κβαντική**: Μπορούμε να λύσουμε το Πρόβλημα Bernstein-Vazirani με 1 μόνο ερώτηση.

1.5.3 Ο αλγόριθμος του Simon

- **Το Πρόβλημα του Simon**:
  Είσοδος: Μια συνάρτηση $f : \{0,1\}^n \to \{0,1\}^m$
  Υπόσχεση: $\exists$ συμβολοσειρά $s \in \{0,1\}^n$ έτσι ώστε $[f(x) = f(y)] \Leftrightarrow [(x = y) \vee (x \oplus s = y)]$ $\forall x, y \in \{0,1\}^n$.
  Έξοδος: Η συμβολοσειρά s.
- **Η επίλυση**:
  - **Κλασσική**: Αν έχουμε ένα πιθανοτικό αλγόριθμο ερωτήσεων A και ο A κάνει λιγότερες από $2^{\frac{n}{2}-1} - 1$ ερωτήσεις (εκθετικός αριθμός για το n), τότε ο A θα αποτύχει να λύσει το Πρόβλημα του Simon με πιθανότητα τουλάχιστον $p = \frac{1}{2}$.
  - **Κβαντική**: Ο αλγόριθμος του Simon λύνει το Πρόβλημα με έναν αριθμό ερωτήσεων που είναι γραμμικός για τα bit εισόδου n της συνάρτησής μας. Η κβαντική επίλυση βασίζεται στο κύκλωμα που αναλύουμε στο αντίστοιχο κεφάλαιο.

1.5.4 Ο αλγόριθμος του Shor

Προτού ασχοληθούμε με τον αλγόριθμο του Shor, θα πρέπει να αναφέρουμε λίγα λόγια για τον **Κβαντικό Μετασχηματισμό Fourier** και για την **Κβαντική Εκτίμηση Φάσης**.

Έπειτα, θα προσπαθήσουμε να χρησιμοποιήσουμε τον αλγόριθμο του Shor για να λύσουμε το **Πρόβλημα της Παραγοντοποίησης Ακεραίων** το οποίο αποτελεί ένα πολύ σημαντικό κομμάτι για το κρυπτογραφικό **Πρωτόκολλο RSA**.

Κβαντικός Μετασχηματισμός Fourier (QFT):
Ο N-διάστατος Κβαντικός Μετασχηματισμός Fourier (QFT), ο οποίος περιγράφεται από ένα $N \times N$ πίνακα του οποίου οι στήλες και οι γραμμές σχετίζονται με την κανονική βάση $|0\rangle, ..., |N-1\rangle$: $QFT_N = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \omega_N^{xy} |x\rangle \langle y|$ όπου ω_N είναι μιγαδικός αριθμός και ισχύει ότι $\omega_N = e^{\frac{2\pi i}{N}} = \cos\left(\frac{2\pi}{N}\right) + i \sin\left(\frac{2\pi}{N}\right)$.

Το Πρόβλημα της Κβαντικής Εκτίμησης Φάσης (QPE):
Είσοδος: Μια n-qubit κβαντική κατάσταση $|\psi\rangle$ και ένα ορθομοναδιαίο κβαντικό κύκλωμα για μια n-qubit πράξη U.
Υπόσχεση: Το $|\psi\rangle$ είναι ιδιοδιάνυσμα του U.
Έξοδος: Ο υπολογισμός ή η προσέγγιση ενός πραγματικού αριθμού $\theta \in [0,1)$ που ικανοποιεί την συνθήκη $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$

Η Επίλυση:

- **Η προσέγγιση με το "κλώτσημα" της φάσης (Phase Kick-Back)**: Χρησιμοποιούμε ένα κβαντικό κύκλωμα με μόνο μια πύλη controlled-U και τελικά προκύπτει μια προσέγγιση της τάξης του ενός bit, δηλαδή ενός bit μετά την δυαδική υποδιαστολή.
- **Η προσέγγιση του διπλασιασμού φάσης (Phase doubling)**: Αν δοκιμάσουμε να χρησιμοποιήσουμε δύο controlled-U πύλες, μπορούμε αποτελεσματικά να αποκτήσουμε μια προσέγγιση για το τι θα μπορούσε να είναι το δεύτερο bit μετά την δυαδική υποδιαστολή, αν στρογγυλοποιήσουμε το θ σε δυο bits.
- **Η προσέγγιση των πολλών qubits (Many qubits - Διαδικασία Εκτίμησης Φάσης**: Μπορούμε να επεκτείνουμε τις παραπάνω εκδοχές του ενός και των δύο qubit της QPE και έτσι να πάρουμε την γενικευμένη QPE. Έτσι, για m qubits ελέγχου (control qubits), απαιτούνται συνολικά $2^m - 1$ αντίγραφα controlled-U πυλών, γεγονός που οδηγεί σε ένα υψηλό υπολογιστικό κόστος αλλά επίσης και σε πιο λεπτομερή προσέγγιση του θ. Αν $\theta = \frac{y}{2^m}$, τότε είναι πολύ πιθανό ότι θα πετύχουμε μια προσέγγιση που να είναι εντός $\frac{1}{2^m}$ από την τιμή του θ και η πιθανότητα να είμαστε εκτός περισσότερο από αυτό, μειώνεται εκθετικά με βάση τον αριθμό των επαναλήψεων που τρέχουμε το κύκλωμα.

Το Πρόβλημα της Εύρεσης Σειράς:
Είσοδος: Θετικοί ακέραιοι N και a που ικανοποιούν $gcd(a, N) = 1$.
Έξοδος: Ο μικρότερος θετικός ακέραιος r έτσι ώστε $a^r \equiv 1(\mod N)$.

Η Επίλυση:

Μπορούμε να λύσουμε αυτό το Πρόβλημα εφαρμόζοντας την Διαδικασία Εκτίμησης Φάσης.

Πολυπλοκότητα:

Συνολικά, χρειαζόμαστε $O(n^3)$ αλλά αυτά τα όρια μπορούν να βελτιωθούν χρησιμοποιώντας ασυμπτωτικά γρήγορους αλγορίθμους.

Το Πρόβλημα της Παραγοντοποίησης Ακεραίων:
Είσοδος: Ένας ακέραιος $N \geq 2$.
Έξοδος: Η παραγοντοποίηση σε πρώτους παράγοντες του N.

Η Επίλυση:

Για να παραγοντοποιήσουμε το N, πρέπει να εστιάσουμε την προσοχή μας στο να ξεχωρίσουμε (splitting) το N (για αυτό χρειαζόμαστε την Εύρεση Σειράς), το οποίο σημαίνει ότι θέλουμε να βρούμε οποιουσδήποτε δύο ακεραίους $b, c \geq 2$ για τους οποίους $N = b \cdot c$. Έπειτα, με την εφαρμογή του Αλγορίθμου Εξέτασης Πρώτων Αριθμών (Primality Testing Algorithm) βλέπουμε αν ο N είναι πρώτος, και αν είναι, τότε δεν είναι δυνατό να τον ξεχωρίσουμε αλλά αν δεν είναι, τότε θα προσπαθήσουμε να τον ξεχωρίσουμε. Μόλις το κάνουμε αυτό, μπορούμε να δουλέψουμε αναδρομικά στα b και c έως ότου όλοι οι παράγοντες να είναι πρώτοι αριθμοί και τότε θα προκύψει τελικά η Παραγοντοποίηση σε Πρώτους Αριθμούς του N.

Πολυπλοκότητα:

Ο αλγόριθμος του Shor βρίσκει τους πρώτους παράγοντες σε $O\left((\log n)^2(\log\log n)\right)$, ενώ ο καλύτερος κλασσικός αλγόριθμος που γνωρίζουμε μέχρι τώρα χρειάζεται $O\left(e^{cn^{\frac{1}{3}}(\log n)^{\frac{2}{3}}}\right)$.

Το RSA Πρωτόκολλο

1. **Δημιουργία Κλειδιού**:

   Τα δημόσια και τα ιδιωτικά κλειδιά παράγονται με βάση τις μαθηματικές αρχές των πρώτων αριθμών και ο υπολογισμός τους είναι εύκολος αλλά, το ανάποδο όμως είναι υπολογιστικά δύσκολο. Θα αναφερόμαστε σε αυτά ως *δημόσιο κλειδί* (e, n) και *ιδιωτικό κλειδί* (d, n).

2. **Διανομή Κλειδιού**:

   Το δημόσιο κλειδί (e, n) γίνεται γνωστό σε όσους επιθυμούν να στείλουν ένα μήνυμα ενώ το ιδιωτικό κλειδί (d, n) παραμένει κρυφό.

3. **Κρυπτογράφηση**:

   Η Alice επιθυμεί να στείλει το μήνυμα M στον Bob και για αυτό, χρησιμοποιεί το δημόσιο κλειδί του Bob (e, n) για να κρυπτογραφήσει το μήνυμα στο κρυπτογραφημένο κείμενο (ciphertext) C, $C \equiv M^e(\mod n)$.

4. **Αποκρυπτογράφηση**:

   Ο Bob παραλαμβάνει το κρυπτογραφημένο κείμενο C και χρησιμοποιεί το ιδιωτικό κλειδί (d, n) για να αποκρυπτογραφήσει το κείμενο και να πάρει το μηνυμα M, $M \equiv C^d(\mod n)$.

Εφαρμογή του Shor: Το σπάσιμο του RSA

Οι **επιθέσεις** στον αλγόριθμο του RSA στοχεύουν στην ανάκτηση του ιδιωτικού κλειδιού d μέσω παραγοντοποίησης (βρίσκοντας δηλαδή ποιοι αριθμοί μπορούν να πολλαπλασιαστούν μαζί ώστε να πάρουμε το αποτέλεσμα) σε modulus n πράξη (το υπόλοιπο που μένει μετά την διαίρεση με την συγκεκριμένη τιμή).

- **Κλασσικός Υπολογισμός και RSA**: Η παραγοντοποίση σε πρώτους αριθμούς μεγάλων ακεραίων είναι γνωστό ότι παρουσιάζει σουπερ-πολυωνυμική ή υπο-εκθετική κλιμάκωση και ο πιο γνωστός κλασσικός αλγόριθμος για την παραγοντοποίηση ακεραίων μεγαλύτερων του 10^{100} είναι ο *General Number Field Sieve (GNFS)*. Παρ'όλα αυτά, για τώρα αλλά και για το εγγύς μέλλον, η παραγοντοποίηση τέτοιων ακεραίων μπορεί να είναι ακατόρθωτη σε κλασσικά συστήματα, αλλά η εξέλιξη των κβαντικών υπολογιστών δείχνει ότι αυτό δεν θα ισχύει για πολύ ακόμα.

- **Κβαντικός Υπολογισμός και RSA**: Η παραγοντοποίηση σε πρώτους παράγοντες μπορεί να υπολογιστεί χρησιμοποιώντας τον αλγόριθμο του Shor. Αυτός ο αλγόριθμος προσφέρει $O\left((\log n)^2(\log\log n)\right)$ πολυπλοκότητα, και δίνει ένα υπερ-πολυωνυμικό πλεονέκτημα σε σχέση με τους κλασσικούς αλγορίθμους. Για να το πετύχουμε αυτό, εκμεταλλευόμαστε την περιοδικότητα της modular εκθετικής συνάρτησης $f(x) = a^x (\mod n)$ και δίνουμε ένα κβαντικό αντίστοιχο αλγόριθμο εύρεσης περιόδου που πετυχαίνει αποτελεσματική παραγοντοποίηση πρώτων αριθμών modulus n.

1.5.5 Ο αλγόριθμος του Grover

Τέλος, εξετάζουμε τον αλγόριθμο του Grover, ο οποίος είναι γνωστό ότι είναι τετραγωνικά ταχύτερος από τους αντίστοιχους κλασσικούς αλγορίθμους, όσον αφορά την επίλυση του Προβλήματος Μη-Δομημένης Αναζήτησης. Αυτό σημαίνει ότι απαιτείται ένας αριθμός n από πράξεις σε κλασσικό υπολογισμό, ενώ για την επίλυση με κβαντικό υπολογιστή χρειαζόμαστε $O(\sqrt{n})$.

Αν υποθέσουμε ότι $f : \{0,1\}^n \to \{0,1\}$ είναι μια συνάρτηση από μια δυαδική συμβολοσειρά μήκους n σε bits, τότε θέλουμε να βρουμε μια συμβολοσειρά $x \in \{0,1\}^n$ έτσι ώστε να ισχύει $f(x) = 1$. Ωστόσο, επειδή δεν μπορούμε να βασιστούμε στο γεγονός ότι η f έχει κάποια συγκεκριμένη δομή, αποκαλούμε αυτή τη διαδικασία ως το Πρόβλημα της Μη-Δομημένης Αναζήτησης.

Το Πρόβλημα Αναζήτησης:
Είσοδος: Μια συνάρτηση $f : \{0,1\}^n \to \{0,1\}$.
Έξοδος: Μια συμβολοσειρά $x \in \{0,1\}^n$ που ικανοποιεί $f(x) = 1$, ή το αποτέλεσμα "καμία λύση", αν δεν υπάρχει τέτοια συμβολοσειρά x.

Το Πρόβλημα της Μοναδικής Αναζήτησης:
Είσοδος: Μια συνάρτηση $f : \{0,1\}^n \to \{0,1\}$.
Υπόσχεση: Υπάρει ακριβώς μια συμβολοσειρά $z \in \{0,1\}^n$ για την οποία $f(z) = 1$, with $f(x) = 0, \forall x \neq z$.
Έξοδος: Η συμβολοσειρά z.

Ο αλγόριθμος του Grover

1. Αρχικοποιούμε ένα n-qubit καταχωρητή Q στην κατάσταση $|0^n\rangle$ και μετά εφαρμόζουμε μια πύλη Hadamard σε κάθε qubit του Q.

2. Έπειτα, εφαρμόζουμε t φορές την ορθομοναδιαία πράξη (γνωστή ως πράξη Grover) $G = H^{\otimes n} Z_{OR} H^{\otimes n} Z_f$ στο Q.

   Όπου, $Z_f |x\rangle = (-1)^{f(x)} |x\rangle, \forall x \in \{0,1\}^n$ και $Z_{OR} |x\rangle = \begin{cases} |x\rangle & , \text{αν } x = 0^n \\ -|x\rangle & , \text{αν } x \neq 0^n \end{cases}$

3. Τέλος, μετράμε τα qubits του Q στην κανονική βάση και δίνουμε στην έξοδο την συμβολοσειρά.

Θα αναφερόμαστε στον αριθμό t, ο οποίος αντιστοιχεί στον αριθμό των επαναλήψεων της πράξης Grover αλλά και στον αριθμό των ερωτήσεων στην f που απαιτούνται.

Συνεπώς, τώρα, μπορούμε να εφαρμόσουμε τον αλγόριθμο του Grover στο Πρόβλημα Αναζήτησς ως εξής:

1. Διαλέγουμε τον αριθμό t.
2. Τρέχουμε τον αλγόριθμο του Grover στην f χρησιμοποιώντας το t για να βρούμε την συμβολοσειρά $x \in \{0,1\}^n$.
3. Κάνουμε ερώτηση στην f για το x για να δούμε αν είναι έγκυρη λύση:
    (a) Αν $f(x) = 1$, τότε βρίσκουμε μια λύση και επιστρέφουμε ως έξοδο το x και σταματάμε.
    (b) Αν $f(x) = 0$, τότε είτε τρέχουμε την διαδικασία ξανά, πιθανώς για διαφορετική τιμή του t, ή επιστρέφουμε στην έξοδο "καμία λύση" και σταματάμε.

Πολυπλοκότητα:

Για $t = O(\sqrt{N})$, με μεγάλη πιθανότητα, προκύπτει μια λύση στο Πρόβλημα Αναζήτησης, αν υπάρχει τέτοια. Αυτό είναι **ασυμπτωτικά βέλτιστο**, δηλαδή, δεν είναι δυνατό να βρεθεί αλγόριθμος ερωτήσεων για την επίλυση του *Προβλήματος Αναζήτησης* ή *Μοναδικής Αναζήτησης*, που να χρησιμοποιεί ασυμπτωτικά λιγότερες από $O(\sqrt{N}))$ ερωτήσεις στην χειρότερη περίπτωση, οπότε ακόμα και με αυτόν εδώ τον αλγόριθμο, έχουμε ισοφαρίσει ένα ήδη γνωστό κάτω όριο.

Ειδικότερα, γνωρίζουμε ότι $t = \lfloor \frac{\pi}{4\theta} \rfloor$ έτσι, για $\theta = \sin^{-1}\left(\sqrt{\frac{|A_1|}{N}}\right)$, η εκτίμησή μας $t \approx \frac{\pi}{4\theta} - \frac{1}{2}$ βασίζεται στον αριθμό των συμβολοσειρών στο σύνολο $A_1 = \{x \in \{0,1\}^n : f(x) = 1\}$.

1. Αν υπάρχει μόνο μια συμβολοσειρά x ($s = |A_1| = 1$), τότε $\theta \approx \sqrt{\frac{1}{N}}$ έτσι, $t = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$, οπότε $O(m\sqrt{N})$ ερωτήσεις χρειάζονται για να προκύψει το x με πιθανότητα τουλάχιστον $1 - 2^{-m}$.
2. Αν υπάρχουν πολλαπλές λύσεις $s = |A_1|$, τότε $\theta \geq \sqrt{\frac{s}{N}}$ έτσι, $t \leq \frac{\pi}{4}\sqrt{\frac{N}{s}}$, οπότε χρειάζονται $O\left(\sqrt{\frac{s}{N}}\right)$ ερωτήσεις.
3. Αν το $s = |A_1|$ είναι άγνωστο, τότε επιλέγουμε $t \in \left\{1, ..., \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor\right\}$ ομοιόμορφα τυχαία, βρίσκουμε μια λύση με πιθανότητα μεγαλύτερη από 40%, και επίσης σημειώνουμε εδώ ότι υπάρχει και μια μέθοδος που βρίσκει λύση χρησιμοποιώντας $O\left(\sqrt{\frac{N}{s}}\right)$ ερωτήματα.
4. Αν τα σύνολα A_0 (για $f(x) = 0$), A_1 (για $f(x) = 1$) είναι κενά, τότε ανεξαρτήτως του αριθμού των επαναλήψεων t, οι μετρήσεις πάντα θα αποκαλύπτουν μια ομοιόμορφα τυχαία συμβολοσειρά $x \in \{0,1\}^n$.

Επίλυση του Προβλήματος Max-Cut με Εφαρμογή του Grover

Τώρα, θα εξετάσουμε μια εφαρμογή του αλγορίθμου του Grover στο γνωστό Πρόβλημα Max-Cut και θα δείξουμε ότι πετυχαίνουμε τετραγωνικά καλύτερη ταχύτητα από τους κλασσικούς αλγορίθμους.

Ο ελαφρώς τροποποιημένος αλγόριθμος που θα χρησιμοποιήσουμε είναι ο ακόλουθος:

Ο αλγόριθμος του Grover:

1. Έστω X ένας n-qubit κβαντικός καταχωρητής, δηλαδή ένα σύνολο από n qubits στα οποία αναθέτουμε το όνομα X. Έστω επίσης, ότι η αρχική κατάσταση X είναι $|0^n\rangle$ και εφαρμόζουμε $H^{\otimes n}$ στο X.
2. Έπειτα, εφαρμόζουμε στον καταχωρητή X τον G μετασχηματισμό k φορές, όπου: $G = -H^{\otimes n} Z_0 H^{\otimes n} Z_f$ και $Z_f |x\rangle = (-1)^{f(x)} |x\rangle$, $Z_0 |x\rangle = \mathbb{I} - 2 |0\rangle \langle 0| = \begin{cases} -|x\rangle & \text{, if } x = 0^n \\ |x\rangle & \text{, if } x \neq 0^n \end{cases}$.
3. Τέλος κάνουμε μέτρηση στο X και επιστρέφουμε το αποτέλεσμα.

Επίσης, σημειώνουμε ότι ο τελεστής Grover (Grover's operator) είναι:

$$U = -H^{\otimes n} Z_0 H^{\otimes n} = 2|h\rangle \langle h| - \mathbb{I} = \frac{2}{N} \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix} - \mathbb{I}$$

Και ο τελεστής διάχυσης (diffusion operator) είναι: $|h\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$

Έτσι, αν η μέση τιμή είναι $\mu = \frac{1}{N} \sum_x \alpha_x$, μπορούμε να γράψουμε

$$U\left(\sum_x \alpha_x |x\rangle\right) = \sum_x \alpha_x U |x\rangle = \sum_x \alpha_x \left(\frac{2}{N} \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix} - \mathbb{I}\right) |x\rangle = \sum_x (2\mu - \alpha_x) |x\rangle$$

Θεωρία Γράφων

Definition 1.5.1 (Μη Κατευθυνόμενος Γράφος). *Γράφος καλείται ένα ζεύγος $G = (V, E)$ όπου, $V = \{v_1, ..., v_n\}$ είναι οι κορυφές και E περιλαμβάνει μη ταξινομημένα ζεύγη κορυφών, δηλαδή ακμές $E = \{(v_i, v_j) | v_i, v_j \in V\}$*

Definition 1.5.2 (Τομή (Cut)). *Μια διαμέριση των κορυφών σε δύο μη κενά σύνολα S και $R = V - S$.*

Definition 1.5.3 (Μέγεθος Τομής (Size of a cut)). *Είναι ο αριθμός των ακμών ανάμεσα σε S και R.*

Έστω ότι θέλουμε να χωρίσουμε τις κορυφές ενός γράφου σε δύο σύνολα έτσι ώστε ο αριθμός των ακμών με άκρα και στα δύο σύνολα να είναι ο μέγιστος δυνατός. Έχει αποδειχθεί ότι αυτό το πρόβλημα, γνωστό ως "Μέγιστη Τομή" (Maximum Cut / Max-Cut) είναι NP-hard (επίσης, είναι APX-hard και έτσι, δεν υπάρχει κλασσικό πολυωνυμικού χρόνου σχήμα προσέγγισης (Polynomial-time approximation scheme) $PTAS$ που να προσεγγίζει αυθαίρετα κοντά την λύση (εκτός κι αν $P = NP$)).

Definition 1.5.4 (Το Πρόβλημα Max-Cut). *Θέλουμε να βρούμε τις διαμερίσεις που μεγιστοποιούν τον αριθμό των ακμών στα σύνολα S και R.*

Definition 1.5.5 (Διμερή Γραφήματα (Bipartite graphs)). *Ένας γράφος G είναι διμερής αν ένα σύνολο από ακμές V μπορεί να διαχωριστεί/ζωγραφιστεί σε δύο ξεχωριστά σύνολα S και R, έτσι ώστε να μην υπάρχει καμία ακμή μεταξύ των κορυφών στο S και επίσης, να μην υπάρχει καμία ακμή μεταξύ των κορυφών στο R.*

Τα **Διμερή Γραφήματα** ικανοποιούν την συνθήκη που αναφέραμε για τον διαχωρισμό ενός γράφου σε δύο χρωματισμούς, έναν για κάθε σύνολο. Συνεπώς, το Πρόβλημα Max-Cut είναι απλώς ο συνολικός αριθμός των ακμών, αφού όλες οι ακμές είναι συνδεδεμένες (cross-connecting). Έτσι, το μέγεθος της τομής θα είναι ο αριθμός των ακμών. Επομένως, η νέα ερώτηση που πρέπει να κάνουμε είναι αν ο γράφος είναι διμερής ή όχι.

Πλέον, έχουμε αναγνωρίσει το Πρόβλημα Max-Cut ως ένα Πρόβλημα Βελτιστοποίησης αλλά μπορούμε να το μετατρέψουμε σε ένα ισοδύναμο Πρόβλημα Απόφασης εφόσον μπορούμε να δεχτούμε μια λογαριθμική επιβάρυνση (logarithmic overhead) εξαιτίας της Δυαδικής Αναζήτησης που θα κάνουμε στο τέλος. Έτσι, αντί να ψάχνουμε να βρούμε το Max-Cut, με το νέο Πρόβλημα Απόφασης, δεδομένου ενός G, ψάχνουμε να βρούμε αν υπάρχει κόψιμο μεγέθους τουλάχιστον k και ύστερα κάνουμε μια Δυαδική Αναζήτηση πάνω στο μέγεθος του k ώστε να βρούμε το μέγιστο μέγεθος για το k και αυτή η διαδικασία μας προσθέτει αυτή την επιβάρυνση που αναφέραμε.

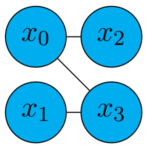
Σχεδιασμός Κυκλώματος (Oracle Design) για Διμερή Γραφήματα

- Αρχικά, για την αναπαράσταση των κορυφών, θα χρησιμοποιήσουμε τον εξής συμβολισμό. Για n κορυφές, χρειαζόμαστε n qubits και επιλέγουμε να αναπαραστήσουμε με $|0\rangle \rightarrow$ χρώμα μπλε και με $|1\rangle \rightarrow$ χρώμα κόκκινο.

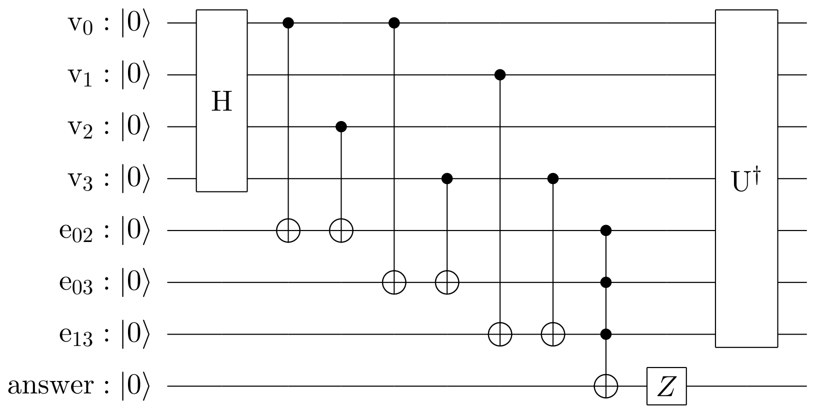
- Έπειτα, για την αναπαράσταση των ακμών, πραγματοποιούμε μια διαδικασία γνωστή ως έλεγχο ακμών (edge checking), κατά την οποία απαντάμε στην ερώτηση για το αν τα άκρα μιας ακμής έχουν κορυφές με διαφορετικό χρώμα. Οπότε, $x \oplus y$ είναι ο έλεγχος ακμής και μας δίνει την απάντηση στο αν και οι δύο κορυφές έχουν διαφορετικό χρώμα. Έτσι, για k ακμές, χρειαζόμαστε k qubits.

- **Κατασκευή Κυκλώματος**:

  Οπότε, για το παρακάτω διμερές γράφημα:



  Το κύκλωμα θα είναι:



- **Πολυπλοκότητα**:

  Παρατηρούμε ότι η εφαρμογή του αλγορίθμου του Grover για την επίλυση του Προβλήματος Max-Cut για την ειδική περίπτωση της εύρεσης Διμερών Γραφημάτων δεν φαίνεται να ήταν καλύτερη από τον καλύτερο κλασσικό αλγόριθμο που ελέγχει αν ένας γράφος είναι διμερής. Συγκεκριμένα, ο κλασσικός αλγόριθμος το πετυχαίνει σε $O(n^2)$ (με BFS) ενώ ο κβαντικός αλγόριθμος που αναπτύξαμε θα λύσει το πρόβλημα σε $O(\sqrt{2^n})$.

Σχεδιασμός Κυκλώματος για Οποιονδήποτε Γράφο

Ωστόσο, για την γενική περίπτωση ενός Μη Κατευθυνόμενου Γράφου που δεν είναι απαραίτητα διμερής, μπορούμε να ακολουθήσουμε τα εξής βήματα:

1. Πρώτα, εφαρμόζουμε τον έλεγχο ακμών για κάθε ακμή.
2. Έπειτα, αθροίζουμε αυτά τα αποτελέσματα που προκύπτουν από τον έλεγχο.
3. Ύστερα, ελέγχουμε αν αυτό το άθροισμα είναι μεγαλύτερο ή ίσο με το k και αποθηκεύουμε το αποτέλεσμα.
4. Μετά, εφαρμόζουμε την πύλη Z στο αποτέλεσμα.
5. Τέλος, εφαρμόζουμε το ισοδύναμο ανάστροφο κύκλωμα.

Για το **Βήμα 3**, η διαδικασία ελέγχου του αριθμού k θα μοιάζει με μια Δυαδική Αναζήτηση, αφού οι ερωτήσεις που κάνουμε έχουν την μορφή: Είναι το $k \geq 4$? Αν όχι, τότε είναι το $k \geq 2$? κ.ο.κ., αλλιώς αν ναι, τότε είναι το $k \geq 6$? Και αν όχι, τότε $k = 5$, ενώ αν ναι, τότε $k = 7$.

Για κάθε ένα από αυτά τα βήματα, χρειάζεται να τρέξουμε τον αλγόριθμο του Grover και επίσης αυτή η διαδικασία του Αν-Τότε μοιάζει με Δυαδική Αναζήτηση οπότε, το βάθος αυτού του δέντρου θα είναι λογαριθμικό στον αριθμό των ακμών, οπότε το πολύ $\log(\#\text{ of edges})$.

Πολυπλοκότητα: Ο κλασσικός αλγόριθμος μπορεί να ελέγξει ένα γράφο σε $O(2^n)$ ενώ ο κβαντικός αλγόριθμος που αναπτύξαμε θα λύσει το πρόβλημα σε $O(\sqrt{2^n})$ οπότε πράγματι παρατηρούμε μια τετραγωνικά καλύτερη βελτίωση στην ταχύτητα.

Σημείωση: Το Πρόβλημα Max-Cut μπορεί να λυθεί και με την χρήση του **Ising Model**.

1.6 Σύνοψη & Μελλοντικές Επεκτάσεις

Με την εμφάνιση των κβαντικών υπολογιστών, η κβαντική κρυπτογραφία αποκτά όλο και περισσότερο ενδιαφέρον τα τελευταία χρόνια. Σε αυτή την εργασία εμβαθύνουμε σε θεμελιώδεις κβαντικούς αλγόριθμους, ενώ παράλληλα εξετάζουμε τις υλοποιήσεις και τις εφαρμογές τους σε βασικούς τομείς όπως η κρυπτογράφηση RSA και το πρόβλημα Max-Cut στη Θεωρία Γραφών.

Μια περαιτέρω ερευνητική κατεύθυνση θα ήταν η διερεύνηση της χρήσης κβαντικών υπολογιστών για πιθανές βελτιώσεις αλγορίθμων σε προβλήματα που πρόσφατα έχουν κάνει σημαντική πρόοδο, όπως το πρόβλημα αθροίσματος υποσυνόλων (subset sum) και για την ανάπτυξη νέων κβαντικών μεθόδων με σκοπό την εκμετάλλευση ορισμένων ιδιοτήτων των προβλημάτων της κλάσης $\sharp P$-hard.

Chapter 2

Basic Notions and Preliminaries

2.1 Introduction

The following is a gentle introduction to Quantum Information. We will introduce the notation that we will use in the next chapters and we will examine the essence of qubits, measurements, and operations and finally we will discuss about some Quantum Mechanics properties that we will use.

A **computer** is a physical device that helps us process information by executing algorithms.

An **algorithm** is a well-defined procedure, with finite description, for realizing an information-processing task.

In other words, an **algorithm** is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation.

An **information-processing task** can be translated into a physical task.

When designing algorithms and protocols it is important to work with some idealized **computing model** but, we should keep in mind that there are computing limitations due to the relationship between computing and physics. Computing devices are embodied in a physical reality that is represented by an idealized computing model. In order to perform tasks that were previously thought impossible or infeasible we use this physical reality to process **Quantum Information** that come from **Quantum Theory** and the devices that perform this processing are known as **quantum computers**.

2.2 Classical bits & Quantum bits

In ordinary computer chips, bits are physically represented by low and high voltages on wires. A bit can be stored, for instance, in a hydrogen atom and the single electron in this atom can either be in the ground state (low energy configuration) or in the excited state (high energy configuration) and we can use these two states to encode for bit values 0 and 1, respectively. Let us now introduce some quantum physics fundamental **notions** of computation theory that will form our basis and after that, we will review the necessary **tools** from linear algebra, and detail the framework of quantum mechanics, as relevant to our model of quantum computation.

2.2.1 Construction of quantum bits (or qubits) from classical bits

Let us consider a single bit that is in state 0 with probability p_0 and in state 1 with probability p_1. We can summarize this information by a 2-dimensional vector of probabilities $\begin{bmatrix} p_0 \\ p_1 \end{bmatrix}$. If we now consider a circuit with wires, then the state is 0 when $p_0 = 1$ and $p_1 = 0$, and similarly, the state will be 1 when $p_0 = 0$ and $p_1 = 1$, so: $0 \mapsto \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $1 \mapsto \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. In fact, we denote the states with the *Dirac* Notation that is called *bra-ket* and it is a vector representation. So, if we denote the ground state of our electron by $|0\rangle$, since it encodes for bit value 0 and likewise, the excited state by $|1\rangle$, then the construction of quantum bits or qubits (also known as standard or computational basis) from classical bits using Dirac Notation will be: $0 \rightarrow$ "ket" notation: $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $1 \rightarrow$ "ket" notation: $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

2.2.2 Mathematical Notation

Complex Numbers

A complex number is: $c = a + ib \in \mathbb{C}$ with $a, b \in \mathbb{R}$ and $i = \sqrt{-1}$. And its complex conjugate is: $c^* = a - ib$. And also, the absolute value of a complex number can be calculated as follows:

Definition 2.2.1. *For a complex number $z \in \mathbb{C}$ expressed as $z = x + iy$, where $x, y \in \mathbb{R}$ are real numbers representing the real and imaginary part of z respectively, the absolute value, or modulus of z is given by $|z| = \sqrt{z^* z} = \sqrt{x^2 + y^2}$*

Vectors in "bra-ket" notation

We will write $|v\rangle \in \mathbb{C}^2$ to denote a vector in a 2-dimensional vector space. For, $|v\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

The "bra" of this vector is the conjugate transpose, so: $\langle v| = ((|0\rangle)^*)^T = \begin{bmatrix} 1^* \\ 0^* \end{bmatrix}^T = \begin{bmatrix} 1 & 0 \end{bmatrix}$.

Definition 2.2.2 (Ket notation)**.** *A "ket", denoted $|\cdot\rangle$, represents a d-dimensional column vector in the complex vector space $\mathbb{C}^d$.*

Definition 2.2.3 (Bra notation)**.** *A "bra", denoted $\langle\cdot|$, represents a d-dimensional row vector equal to the complex conjugate of the corresponding "ket", namely: $\langle\cdot| = (|\cdot\rangle^*)^T$, where * denotes the entry-wise conjugate, and T denotes the transpose.*

Definition 2.2.4 (Inner product). *Given two d-dimensional vectors* $|v_1\rangle = \begin{bmatrix} a_1 & \cdots & a_d \end{bmatrix}^T$*,* $|v_2\rangle = \begin{bmatrix} b_1 & \cdots & b_d \end{bmatrix}^T$*, their inner product is given by* $\langle v_1|v_2\rangle := \langle v_1|\,|v_2\rangle = \sum_{i=1}^{d} a_i^* b_i$*.*

Definition 2.2.5 (Length of a ket vector). *Consider a ket vector:* $|v\rangle = \begin{bmatrix} a_1 & \cdots & a_d \end{bmatrix}^T$*. The length of* $|v\rangle$ *is given by* $\||v\rangle\|_2 = \sqrt{\langle v|v\rangle} = \sqrt{\sum_{i=1}^{d} a_i^* a_i} = \sqrt{\sum_{i=1}^{d} |a_i|^2}$*. If* $\||v\rangle\|_2 = 1$*, we say that* $|v\rangle$ *has norm 1, or that it is normalized.*

Finally, we will remember the notion of an orthonormal basis from linear algebra, $\{|b\rangle\}_b$, that is true when $\langle b|\hat{b}\rangle = \delta_{b\hat{b}}\ \forall b, \hat{b}$. Where, $\delta_{ab} = 0$ if $a \neq b$, and 1 otherwise.

2.2.3 Mathematical Description of qubits

Hence, any vector can be written as a superposition of $|0\rangle$ & $|1\rangle$: $|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha|0\rangle + \beta|1\rangle$ where, α and β are called amplitudes and in general, they can be complex numbers. Thus some element of a complex vector space of dimension two.

A qubit can be described by a vector $|v\rangle \in \mathbb{C}^2$, where $\mathbb{C}^2$ is known as the state space of the qubit. In order to have a valid qubit $|v\rangle$, it must be normalized, just like $|0\rangle$ and $|1\rangle$ corresponding to classical bits were also normalized.

Definition 2.2.6 (Qudit). *A qudit or a d-dimensional quantum system can be represented as a d-dimensional "ket" vector* $|\psi\rangle \in \mathbb{C}^d$*:* $|\psi\rangle = \sum_{i=0}^{d-1} \alpha_i |i\rangle$ *where,* $\forall i, \alpha_i \in \mathbb{C}$ *and* $\sum_{i=0}^{d-1} |\alpha_i|^2 = 1$*. The condition on the coefficients* α_i *means that* $|\psi\rangle$ *is a vector of length of 1.*

Definition 2.2.7 (Qubit). *A (pure) state of a qubit can be represented as a 2-dimensional "ket" vector* $|\psi\rangle \in \mathbb{C}^2$*:* $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ *where,* $\alpha, \beta \in \mathbb{C}$ *and* $|\alpha|^2 + |\beta|^2 = 1$*. The condition on* α, β *means that* $|\psi\rangle$ *is normalized and they are also called amplitudes of* $|\psi\rangle$*.*

In fact, vectors $|0\rangle, |1\rangle$ are orthonormal and they can be expressed as $\langle 1|0\rangle = 0$ and $\langle 1|1\rangle = \langle 0|0\rangle = 1$. They form a basis for $\mathbb{C}^2$ and any vector $|v\rangle$ in it can be written as $|v\rangle = \alpha|0\rangle + \beta|1\rangle$ for some coefficients $\alpha, \beta \in \mathbb{C}$. This basis corresponding to "classical" bits is called *standard basis.*

Definition 2.2.8 (Standard basis). *Consider the 2-dimensional complex vector space* $\mathbb{C}^2$*. The standard basis or computational basis* $\{|0\rangle, |1\rangle\}$ *is an orthonormal basis for this vector space and the basis vectors are:* $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ *and* $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

There are also other bases and another useful one is the *Hadamard basis* which is also orthonormal.

Definition 2.2.9 (Hadamard basis). *The Hadamard basis is an orthonormal basis* $\{|+\rangle, |-\rangle\}$ *consisting of two elements:* $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$*,* $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$*.*

2.2.4 Operations

Let us suppose we have two qubits A and B and we saw that we can describe them as $|\psi\rangle_A$ and $|\phi\rangle_B$, correspondingly but now we will see how we can describe a combined state $|\psi\rangle_{AB}$ of A and B together. The rule of computing the joint state is given by the **tensor product** or **Kronecker product**.

Definition 2.2.10 (Tensor Product)**.** *Given two vectors* $|\psi_1\rangle \in \mathbb{C}^{d_1}$ *and* $|\psi_2\rangle \in \mathbb{C}^{d_2}$ *respectively, the tensor product is given by* $|\psi_1\rangle \otimes |\psi_2\rangle = \begin{bmatrix} a_1 \\ \vdots \\ a_d \end{bmatrix} \otimes |\psi_2\rangle = \begin{bmatrix} a_1 |\psi_2\rangle \\ \vdots \\ a_d |\psi_2\rangle \end{bmatrix}$ *and* $|\psi_1\rangle \otimes |\psi_2\rangle$ *lies in the state space* $\mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$*.*

And, below we have summarized the basic notations and operations:

- **"bra" & "ket" notation**: $|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix} \quad \langle\psi| = (|\psi\rangle^*)^T = \begin{bmatrix} a^* \\ b^* \end{bmatrix}^T = \begin{bmatrix} a^* & b^* \end{bmatrix}$

- **Inner product**: $\langle\psi|\,|\psi\rangle = \langle\psi|\psi\rangle = \begin{bmatrix} a^* & b^* \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = a^*a + b^*b = |a|^2 + |b|^2$

- **Tensor product**: Let us suppose that we have two qubits, A, B. We can describe the state of A as $|\psi\rangle_A$ and the state of B as $|\psi\rangle_B$. Thus, in order to write down the combined/joint state $|\psi\rangle_{AB}$ of A and B together, we use the tensor product (also known as Kronecker product). So, for two qubits $|\psi\rangle_A = \alpha_A |0\rangle_A + \beta_A |1\rangle_A = \begin{bmatrix} \alpha_A \\ \beta_A \end{bmatrix}$ and $|\psi\rangle_B = \alpha_b |0\rangle_B + \beta_B |1\rangle_B = \begin{bmatrix} \alpha_B \\ \beta_B \end{bmatrix}$ the joint state $|\psi\rangle_{AB} \in \mathbb{C}^2 \otimes \mathbb{C}^2$ can be expressed as the tensor product of individual vectors $|\psi\rangle_A$ and $|\psi\rangle_B$ as

$$|\psi\rangle_{AB} = |\psi\rangle_A \otimes |\psi\rangle_B = \begin{bmatrix} \alpha_A \\ \beta_A \end{bmatrix} \otimes |\psi\rangle_B = \begin{bmatrix} \alpha_A |\psi\rangle_B \\ \beta_A |\psi\rangle_B \end{bmatrix} = \begin{bmatrix} \alpha_A\alpha_B & \alpha_A\beta_B & \beta_A\alpha_B & \beta_A\beta_B \end{bmatrix}^T$$

  Some simplified notations are the following:

  - Omitted tensor product symbol: $|\psi\rangle_A \otimes |\phi\rangle_B = |\psi\rangle_A |\phi\rangle_B$
  - Classical bits as a string: $|0\rangle_A \otimes |0\rangle_B = |0\rangle_A |0\rangle_B = |00\rangle_{AB}$
  - Combining several identical states: $|\psi\rangle_1 \otimes \cdots \otimes |\psi\rangle_n = |\psi\rangle^{\otimes n}$

  Properties of the **tensor product**:

  - **distributive**: $|\psi\rangle \otimes (|v_1\rangle + |v_2\rangle) = |\psi\rangle \otimes |v_1\rangle + |\psi\rangle \otimes |v_2\rangle$
  - **associative**: $|\psi\rangle \otimes (|\phi\rangle \otimes |\gamma\rangle) = (|\psi\rangle \otimes |\phi\rangle) \otimes |\gamma\rangle$
  - **not commutative** (unless same states): $|\psi\rangle \otimes |\phi\rangle \neq |\phi\rangle \otimes |\psi\rangle$

2.2.5 Examples

- **Example 1**: The $|00\rangle_{AB} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $|01\rangle_{AB} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $|10\rangle_{AB} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $|11\rangle_{AB} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$ is the standard basis for two qubits AB. It can be constructed by taking the tensor product of standard bases for individual qubits: $|0\rangle_A \otimes |0\rangle_B$, $|0\rangle_A \otimes |1\rangle_B$, $|1\rangle_A \otimes |0\rangle_B$, $|1\rangle_A \otimes |1\rangle_B$. For instance, $|1\rangle_A \otimes |0\rangle_B = \begin{bmatrix}0\\1\end{bmatrix} \otimes |0\rangle_B = \begin{bmatrix}0\,|0\rangle_B\\1\,|0\rangle_B\end{bmatrix} = \begin{bmatrix}0\cdot 1\\0\cdot 0\\1\cdot 1\\1\cdot 0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} = |10\rangle_{AB}$.

- **Example 2**: Consider the state: $|\psi\rangle_{AB} = \frac{1}{\sqrt{2}}(|+\rangle_A |+\rangle_B + |-\rangle_A |-\rangle_B)$. Let us express the state in terms of the standard basis, by expanding the terms: $|+\rangle_A |+\rangle_B = \frac{1}{2}(|0\rangle_A + |1\rangle_A)(|0\rangle_B + |1\rangle_B) = \frac{1}{2}(|00\rangle_{AB} + |10\rangle_{AB} + |01\rangle_{AB} + |11\rangle_{AB})$ and $|-\rangle_A |-\rangle_B = \frac{1}{2}(|0\rangle_A - |1\rangle_A)(|0\rangle_B - |1\rangle_B) = \frac{1}{2}(|00\rangle_{AB} - |10\rangle_{AB} - |01\rangle_{AB} + |11\rangle_{AB})$. By substituting this into the initial state, we get $|\psi\rangle_{AB} = \frac{1}{\sqrt{2}}(|+\rangle_A |+\rangle_B + |-\rangle_A |-\rangle_B) = \frac{1}{2\sqrt{2}}(|00\rangle_{AB} + |10\rangle_{AB} + |01\rangle_{AB} + |11\rangle_{AB} + |00\rangle_{AB} - |10\rangle_{AB} - |01\rangle_{AB} + |11\rangle_{AB}) = \frac{1}{\sqrt{2}}(|00\rangle_{AB} + |11\rangle_{AB}) = |EPR\rangle_{AB}$. We can see that the coefficients of $|EPR\rangle_{AB}$ are the same whether we write it in the Hadamard basis or the standard basis.

- **Example 3**: The state that is in the mid-vertical of $|0\rangle$ and $|1\rangle$ is called $|+\rangle$ and it is $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$ so, $\langle +|+\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{\sqrt{2}}(1+1) = 1$.

It is noted that the length of a qubit vector is still the same as the length of the classical vector. For classical bits, vectors have a length equal to one. Thus, it is demanded that the qubits should also correspond to the vectors that have length exactly equal to one. For instance, for real vectors it is: $\langle\psi|\psi\rangle = 1 \Rightarrow a^2 + b^2 = 1$.

Intuition of what does this mathematical expression (the superposition) mean: When we consider a maze and we are to associate 0 with left and 1 with right, then, classically, we could send a particle either to the left or to the right. However, quantumly, we could do both left and right at the same time. This means that the particle would be in a superposition of being on the left and on the right.

Physical implementation:

A physical implementation of the qubits could be labeled as the ground and the excited state of an atom respectively. A bit can be encoded by preparing the atom in the ground state $|0\rangle$ or the excited state $|1\rangle$. In this implementation, we represent bits by energy levels in an atom. We can say that the ground state, or the lower one, is called zero, and the higher one is called one. A classical bit means that out system will be in one of these two states, while a qubit can be in a superposition between zero and one and that means that it is effectively in the ground and the excited state at the same time.

2.3 Combining Qubits

2.3.1 Multiple Qubits

In classical computing, if we have a number with n bits, we can write this down as a string of numbers $x = x_1, \ldots, x_n \in \{0,1\}^n$. There are $d = 2^n$ possible strings. As far a vectors are concerned, we will make them equal to zero everywhere except of the position indexed by x. Thus, $x \to |x\rangle = \begin{bmatrix} 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \end{bmatrix}^T$. Hence, there are 2^n possible strings. So, we can think of x as a number between 1 and 2^n. For instance, for $n = 1$, $|0\rangle = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$, $|1\rangle = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$. Note that the resulting vectors are in $\mathbb{C}^d$ with dimension $d = 2^2 = 4$, where the dimension corresponds to the number of possible strings. It turns out that one can write a 2-qubit state $|\psi\rangle_{AB} \in \mathbb{C}^4$ as a superposition of these vectors, where we again demand that $|\psi\rangle_{AB}$ is normalized.

2.3.2 Quantum state of n qubits

To address multiple qubits, we first look at the vector representation for multiple classical bits. For binary strings of length n, consider the vector space $\mathbb{C}^{2^n}$ where each coordinate is labelled by a string $x = x_1, ..., x_n$. We can express the string x as a vector $|x\rangle$ that is 0 everywhere, except at the position labelled by i.

Definition 2.3.1. *An n-qubit state $|\psi\rangle \in \mathbb{C}^d$, with $d = 2^n$ can be written as a superposition of standard basis elements: $|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$ where, $\forall x, \alpha_x \in \mathbb{C}$ and $\sum_{x\in\{0,1\}^n} |a_x|^2 = 1$ where, α_x are the coefficients and they are called amplitudes.*

The space $\mathbb{C}^d$ where $d = 2^n$ is called the *state space* of n qubits and will need an exponential number of parameters α_x to keep track of only n qubits, in contrast to the n parameters $x_1, ..., x_n$ to describe n classical bits.

Definition 2.3.2 (Standard basis for n qubits)**.** *Consider the state space of n qubits $\mathbb{C}^d$, where $d = 2^n$. For each distinct string $x \in \{0,1\}^n$, associate x with a distinct integer $i \in \{1, 2, ..., d\}$. The standard basis for $\mathbb{C}^d$ is an orthonormal basis given by $\{|x\rangle\}_{x\in\{0,1\}^n}$, where $|x\rangle$ are d-dimensional vectors $|x\rangle = \begin{bmatrix} 0 & \cdots & 1 & \cdots & 0 \end{bmatrix}^T \to$ i-th position.*

Therefore, given two vectors $|\psi_1\rangle \in \mathbb{C}^{d_1}$ and $|\psi_2\rangle \in \mathbb{C}^{d_2}$ respectively, the tensor product is given by $|\psi_1\rangle \otimes |\psi_2\rangle$ and $|\psi_1\rangle \otimes |\psi_2\rangle \in \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$.

Hence, we should note that if we combine two qubits A, B, each of which is in a pure state, then the joint state of the two qubits is given by $|\psi\rangle_{AB} = |\psi_1\rangle_A \otimes |\psi_2\rangle_B$. Any two-qubit state that either has directly this form or has a mixture of states of this form is called separable. Entangled states are states which are not separable. In other words, a pure state $|\psi\rangle$ is entangled if and only if $|\psi\rangle \neq |\psi_1\rangle \otimes |\psi_2\rangle$ for any possible choice of $|\psi_1\rangle, |\psi_2\rangle$.

2.3.3 Examples

- **Example 1**: 2 qubits in equal Superposition: For 2 qubits, we have $n = 2$ so $2^2 = 4$ possible classical strings: $|00\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, |01\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, |10\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, |11\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$. Equal superposition: $|\psi\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}^T$, $\langle\psi|\psi\rangle = 1$.

- **Example 2**: 2 qbits in an E.P.R. pair: It is noted that E.P.R. stands for *Einstein, Podolsky, Rosen* and it is a superposition between $|00\rangle$ and $|11\rangle$ with a normalization $|\psi\rangle = |EPR\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 0 & 1\end{bmatrix}^T$. It is true that they are entangled since $\langle\psi|\psi\rangle = 1$.

- **Example 3**: Another 2 qubits state $|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |11\rangle)$ that is normalized but not entangled.

- **Example 4**: Calculation of a new state $|\psi\rangle_{AB}$ from 2 separated qubits $|\psi\rangle_A = \alpha_A|0\rangle_A + \beta_A|1\rangle_A$ and $|\psi\rangle_B = \alpha_B|0\rangle_B + \beta_B|1\rangle_B$. For this operation we will use the **tensor product** with which we can compute the joint state $|\psi\rangle_A \otimes |\psi\rangle_B = \begin{bmatrix}\alpha_A\\\beta_A\end{bmatrix} \otimes |\psi\rangle_B = \begin{bmatrix}\alpha_A|\psi\rangle_B\\\beta_A|\psi\rangle_B\end{bmatrix} = \begin{bmatrix}\alpha_A\alpha_b & \alpha_A\beta_B & \beta_A\alpha_B & \beta_A\beta_B\end{bmatrix}^T$. Same can be proven about the basis of 2 qubits.

- **Example 5**: Combining 2 qubits: $|+\rangle_A = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|1\rangle_B$. In order to combine these 2 qubits, we will take the tensor product $|+\rangle_A \otimes |1\rangle_B = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \otimes |1\rangle_B = \frac{1}{\sqrt{2}}\begin{bmatrix}1\,|1\rangle_B\\1\,|1\rangle_B\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}0 & 1 & 0 & 1\end{bmatrix}^T$

  $$|+\rangle_A \otimes |1\rangle_B = \frac{1}{\sqrt{2}}(|0\rangle_A + |1\rangle_A) \otimes |1\rangle_B = \frac{1}{\sqrt{2}}(|0\rangle_A|1\rangle_B + |1\rangle_A|1\rangle_B) = \frac{1}{\sqrt{2}}(|01\rangle_{AB} + |11\rangle_{AB})$$

- **Example 6**: Reconstructing equal superposition $|+\rangle_A \quad |+\rangle_B$. So, $|+\rangle_A \otimes |+\rangle_B = \frac{1}{\sqrt{2}}(|0\rangle_A + |1\rangle_A) \otimes \frac{1}{\sqrt{2}}(|0\rangle_B + |1\rangle_B) = \frac{1}{2}(|00\rangle_{AB} + |01\rangle_{AB} + |10\rangle_{AB} + |11\rangle_{AB}) = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}^T$.

2.4 Measuring Qubits

In this section we will discuss what is a measurement. Classically, for a bit it is a readout, because we have a system that encodes the states ”0” and ”1” and we make a measurement to find out which one it is.

2.4.1 Measuring qubits in the standard basis

If someone wants to measure a qubit, it is usual to measure it in the standard basis. Quantum measurements can result in probabilistic outcomes. For example, if the state $|\psi\rangle \in \mathbb{C}^2$ is a superposition between $|0\rangle$ and $|1\rangle$, then upon measuring $|\psi\rangle$, we obtain different measurement outcomes corresponding to some probability distribution. Intuitively, this means that we will have to find out whether the state is 0 or 1 or if it is in superposition (meaning, somewhere in the middle). This probability of different outcomes can be computed by looking at the amount of 0 that exists in the qubit vector and it can be quantified by the inner product between $|\psi\rangle$ and $|0\rangle$.

So, if $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$, where a and b are complex numbers, then by doing the measurement in the standard basis, the probabilities of outcomes will be:

$$\text{Outcome ''0''}\ : p_0 = |\,\langle\psi|0\rangle\,|^2 = \left| \begin{bmatrix} a^* & b^* \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right|^2 = |a|^2$$

$$\text{Outcome ''1''}\ : p_1 = |\,\langle\psi|1\rangle\,|^2 = \left| \begin{bmatrix} a^* & b^* \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right|^2 = |b|^2$$

So, evidently, if we ask "is it zero or one?" then these probabilities should add up to one. Thus, $p_0 + p_1 = 1 = |a|^2 + |b|^2$. So, the sum of the probability of seeing a zero plus the probability of seeing a one is given by alpha squared plus beta squared and since the qubit was normalized, this was precisely one. Also, it can be seen that there was a very good reason why qubits were normalized vectors of length one and that is that with the measurement, in the standard basis, the sum of the probabilities adds up to one, as it should.

However, what happens to the qubit after the measurement? Let us assume that $|\psi\rangle$ is our qubit and that after the measurement the outcome is zero, with some probability. If we obtained outcome zero, the qubit will now be in the state zero and we say that the state has collapsed to zero. Similarly, if we saw the outcome one, then the final state would be one. Therefore, we have collapsed the superposition and we have lost the information about alpha and beta and we are just in one of the two classical basis states.

Application: Randomness from a deterministic machine

In the classical world, like a classical computer, it is not possible to produce true random numbers using a purely deterministic process. So, if there is no randomness to begin with, there is no randomness at the end.

In the quantum world, on the other hand, we can produce true random numbers from a deterministic process.

Let us imagine that we build a machine:

- First, the machine will prepare the state of equal superposition between zero and one: $|\psi\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

- Next, we take this qubit and we measure it in the standard basis. Thus, we can get two outcomes, zero or one. The probability p_0 of getting outcome zero occurs when we compute how much zero is contained in the state. So,

$$p_0 = |\langle\psi|0\rangle|^2 = \left|\frac{1}{\sqrt{2}}(\langle 0| + \langle 1|)|0\rangle\right|^2 = \left|\frac{1}{\sqrt{2}}(\langle 0|0\rangle + \langle 1|0\rangle)\right|^2 = \frac{1}{2}$$

  Also, the probability p_1 of getting outcome one occurs similarly:

$$p_1 = |\langle\psi|1\rangle|^2 = \left|\frac{1}{\sqrt{2}}(\langle 0| + \langle 1|)|1\rangle\right|^2 = \left|\frac{1}{\sqrt{2}}(\langle 0|1\rangle + \langle 1|1\rangle)\right|^2 = \frac{1}{2}$$

Thus, we saw that we can build a deterministic machine that prepares a qubit in an equal superposition between zero and one and if we subsequently measure it in the standard basis, we produce random numbers zero and one with equal probabilities. In contrast, that is not possible with classical deterministic machines.

Definition 2.4.1 (Measuring n-qubit states)**.** *Consider an n-qubit state $|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$. The probability outcome when x is measured in the standard basis $\{|x\rangle\}_x$ is given by $p_x = |\langle x|\psi\rangle|^2 = |\alpha_x|^2$.*

2.4.2 Measuring qubits in another basis

If someone wants to measure a qubit in a different basis, it is also possible. First, we will analyze the probabilities we used earlier. In fact, when we measure in the standard basis, the probabilities are given by the squared amplitudes when writing out the state in terms of the standard basis. Respectively, when we measure in a different orthonormal basis, given by the vectors $\{|v\rangle, |v^\perp\rangle\}$, it is intuitive that we would have to express the qubit in the new basis. Thus, we have to find amplitudes $\hat{a}$ and $\hat{b}$, such that $|\psi\rangle = a|0\rangle + b|1\rangle = \hat{a}|v\rangle + \hat{b}|v^\perp\rangle$.

Definition 2.4.2 (Rule for finding probabilities for multiple qubits)**.** *Consider that we measure a quantum state $|\psi\rangle$ in the orthonormal basis $\{|b_j\rangle\}_{j=1}^d$. The probability of observing outcome b_j can be found by computing $p_j = |\langle b_j|\psi\rangle|^2$. The post-measurement state when obtaining outcome b_j is given by $|b_j\rangle$.*

Definition 2.4.3. *Consider we want to measure the state of multiple qubits, where we think of measuring each qubit in a separate basis. The joint state space $\mathbb{C}_A^{d_A} \otimes \mathbb{C}_B^{d_B}$ can be obtained from bases for the individual state spaces $\mathbb{C}_A^{d_A}$ and $\mathbb{C}_B^{d_B}$. In particular, if $\{|b_j^A\rangle\}_j$ is a basis for $\mathbb{C}_A^{d_A}$ and $\{|b_j^B\rangle\}_j$ is a basis for $\mathbb{C}_B^{d_B}$, then the set of vectors $\{\{|b_j^A\rangle \otimes |b_k^B\rangle\}_{j=1}^{d_A}\}_{k=1}^{d_B}$ gives a basis for $\mathbb{C}_A^{d_A}$ and $\mathbb{C}_B^{d_B}$.*

2.4.3 Examples

- **Example 1**: Let us consider that the basis is given by the state plus and the state minus: $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. If we want to measure in this basis, we ask how much of plus or minus is in our state. Thus, just as before, we basically compute the inner product between $|\psi\rangle$ and $|+\rangle$ to see how much plus is in $|\psi\rangle$. So, the measurement outcomes that we will identify by the element of the basis are: "+" if $p_+ = |\langle\psi|+\rangle|^2$ and "-" if $p_- = |\langle\psi|-\rangle|^2$.

- **Example 2**: Consider the qubit $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ and we want to measure the qubit in the Hadamard basis $\{|+\rangle, |-\rangle\}$. The probabilities of obtaining "+", "-" can be calculated as:

$$p_+ = |\langle\psi|+\rangle|^2 = \left|\frac{1}{2}\left(\left(\langle 0| - i\langle 1|\right)\left(|0\rangle + |1\rangle\right)\right)\right|^2 = \frac{1}{4}|\langle 0|0\rangle + \langle 0|1\rangle - i\langle 1|0\rangle - i\langle 1|1\rangle|^2 =$$
$$= \frac{1}{4}|1 - 0 - i + 0|^2 = \frac{1}{4}|1-i|^2 = \frac{1}{4}(1-i)(1+i) = \frac{1}{4}(1+1) = \frac{2}{4} = \frac{1}{2}$$

$$p_- = |\langle\psi|-\rangle|^2 = \left|\frac{1}{2}\left(\left(\langle 0| - i\langle 1|\right)\left(|0\rangle - |1\rangle\right)\right)\right|^2 = \frac{1}{4}|\langle 0|0\rangle - \langle 0|1\rangle - i\langle 1|0\rangle + i\langle 1|1\rangle|^2 =$$
$$= \frac{1}{4}|1 - 0 - 0 + i|^2 = \frac{1}{4}|1+i|^2 = \frac{1}{4}(1+i)(1-i) = \frac{1}{4}(1+1) = \frac{2}{4} = \frac{1}{2}$$

- **Example 3**: Consider a *qutrit*, a 3-dimensional quantum system represented by the vector $|v\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}0\\1\\0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}0\\0\\1\end{bmatrix}$. And we want to measure in the basis $\{|b_1\rangle, |b_2\rangle, |b_3\rangle\}$, where $|b_1\rangle = \begin{bmatrix}1\\0\\0\end{bmatrix}$, $|b_2\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}0\\1\\1\end{bmatrix}$, $|b_3\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}0\\1\\-1\end{bmatrix}$. The probabilities of each outcome will be: $p_{b_1} = |\langle b_1|v\rangle|^2 = \frac{1}{2}$

$$p_{b_2} = |\langle b_2|v\rangle|^2 = \langle b_2|v\rangle\langle v|b_2\rangle = \frac{1}{2\sqrt{2}}(1+1)\cdot\frac{1}{2\sqrt{2}}(1+1) = \frac{1}{2}$$

$$p_{b_3} = |\langle b_3|v\rangle|^2 = \langle b_3|v\rangle\langle v|b_3\rangle = \frac{1}{2\sqrt{2}}(1-1)\cdot\frac{1}{2\sqrt{2}}(1-1) = 0$$

- **Example 4**: Let us consider measuring a single qubit $|\psi\rangle = \sqrt{\frac{1}{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$ which is in the superposition between zero and one. Now, let us apply the rule to compute the probability that when we measure in the standard basis, we will get outcome zero.

$$p_0 = |\langle\psi|0\rangle|^2 = \left|\left(\sqrt{\frac{1}{3}}\langle 0| + \sqrt{\frac{2}{3}}\langle 1|\right)|0\rangle\right|^2 = \left|\sqrt{\frac{1}{3}}\langle 0|0\rangle + \sqrt{\frac{2}{3}}\langle 1|0\rangle\right|^2 = \frac{1}{3}$$

$$p_1 = |\langle\psi|1\rangle|^2 = ... = \frac{2}{3} \quad p_0 + p_1 = 1$$

- **Example 5**: Measuring two qubits (EPR pair): We want to measure the following qubit in the standard basis $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. So, $p_{00} = |\langle\psi|00\rangle|^2 = \left|\frac{1}{\sqrt{2}}\left(\langle 00| + \langle 11|\right)|00\rangle\right|^2 = \left|\frac{1}{\sqrt{2}}\left(\langle 00|00\rangle + \langle 11|00\rangle\right)\right|^2 = \left|\frac{1}{\sqrt{2}}(1+0)\right|^2 = \frac{1}{2} = p_{11}$, $p_{01} = |\langle\psi|01\rangle|^2 = ... = 0 = p_{10}$.

2.4.4 General rule for measuring qubits in a basis

- *Basis we measure in*: If the basis we are measuring in is given by the basis vectors labeled b: $\{|b\rangle\}_b$ where $|b\rangle \in \mathbb{C}^d$ (remember that for n qubits the dimension is $d = 2^n$)
- *Probabilities of measurement outcomes*: The probability of outcome b is given $p_b = |\langle\psi|b\rangle|^2$
- *What happens after the measurement?*: If outcome b occurred, then the state collapsed to b. We will also call the state after the measurement, the post-measurement state.

2.5 Transformations on Qubits

2.5.1 Performing operations on qubits

Remember that a qubit is a vector $|\psi\rangle = a|0\rangle + b|1\rangle$ and if we perform an operation on a qubit, we want to end up with another qubit $|\psi_{new}\rangle$. So, the transformation that we apply is a linear operator U that is applied to the starting qubit $|\psi\rangle$ and maps vectors to vectors, thus $|\psi_{new}\rangle = U|\psi\rangle$. If $|\psi\rangle \in \mathbb{C}^d$, then U is a $d \times d$ matrix with complex entries.

Now, the question is whether we can use any matrix or does the U have to satisfy certain properties. In order to still have a qubit after the transformation, the vector should be normalized, meaning $\langle\psi|\psi\rangle = 1$, because only in this case the sum of the probabilities, if we are going to make a measurement, are going to be equal to one. Thus, it is important for the new qubit the inner product to be one, meaning $\langle\psi_{new}|\psi_{new}\rangle = 1$. However, we must firstly compute what this means in terms of matrix U. Remember that, $\langle\psi_{new}| = \langle\psi|U^+$. Therefore, $\langle\psi_{new}|\psi_{new}\rangle = \langle\psi|U^+U|\psi\rangle = \langle\psi|UU^+|\psi\rangle = 1$.

We want our operation U to be independent of the vector and we would like to perform this operation on potentially any qubit that we begin with. Thus, the operation U should preserve the length of any vector, i.e. the previous term must be 1 $\forall |\psi\rangle$ and this is only possible when $U^+U = \mathbb{I}$. Also, it is true that $\mathbb{I}|\psi\rangle = |\psi\rangle$. Thus, we can see that this is a condition that we want matrix U to satisfy. Moreover, U^+ is an inverse operation, and $UU^+ = \mathbb{I}$ meaning that they are interchanged. It is reminded that "+" sign means dagger and gives the conjugate transpose of a matrix.

Definition 2.5.1 (Identity)**.** *The identity $\mathbb{I}$ is a diagonal, square matrix where each diagonal element is equal to 1, i.e.* $\mathbb{I} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$. *For any dimension d, we denote the $d \times d$ identity matrix as $\mathbb{I}_d$.*

Definition 2.5.2 (Unitary matrices)**.** *U is unitary $\iff U^+U = UU^+ = \mathbb{I}$, if $U^+ = (U^*)^T$ conjugate transpose U*

2.5.2 Quantum Gates

Briefly, any unitary operator acting on a 2-dimensional quantum system (a qubit) is called a **1-qubit quantum gate**. They are rotation gates, because they correspond to rotations about the x,y, and z axes of the Bloch sphere. Also, they are defined as **Pauli gates**, and they are: $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. Apart from the **1-qubit quantum gates**, there are also **2-qubit gates**. Additionally, just as there are 2-qubit states that *can* be written as the product of two 1-qubit states, there are 2-qubit gates that *cannot* be written as a tensor product of two 1-qubit gates.

2.5.3 Examples

- **Example 1**: the Hadamard (H) transform: The unitary transformation that we will consider is often written as H, for the Hadamard transform and his unitary transformation is given by the following matrix $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. Note that the conjugate transpose of H^+ is H itself and such a matrix is called a hermitian. So, if we apply this unitary on the zero of the standard basis, we get: $H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) = |+\rangle$. And, we can verify that if we apply Hadamard to the state one, we will get the state minus $H|1\rangle = |-\rangle$. Finally, it is noted that if we apply a unitary to an element of the standard basis, it also means that the transformation preserves the inner product. So, we knew that zero and one are orthogonal at the beginning, thus, if we apply the Hadamard transform to them, we will also end up with two vectors that are orthogonal. Therefore, if $U|0\rangle$ gives as $|+\rangle$, then $U|1\rangle$ should give us $|-\rangle$ along with a global pre-factor in front. All in all, the Hadamard gate, H, is defined as gate mapping the computational basis states as follows $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. One useful property of the Hadamard gate is that it is self-inverse, meaning that $H = H^{-1}$ and so, $H(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) = |0\rangle \quad H(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)) = |1\rangle$.

- **Example 2**: the X operation: X is defined as $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. First, we verify that X is unitary. So, $X^+ = X$, $X^+X = XX = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbb{I}$. So, X is unitary. It is also useful to think what does this operation actually do to an element of the standard basis. So, we will compute X applied to zero. Thus, $X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$. So, it is easy to understand that the unitary operation has to preserve the inner product and zero and one are orthogonal, so we will get $X|1\rangle = |0\rangle$. Therefore, we can see that X acts as a **bit flip** operator as it exchanges the two elements of the standard basis and it is called the Pauli X matrix.

- **Example 3**: the Z operation: Z is defined as $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. First, we verify that Z is unitary. So, $Z^+ = Z$, $Z^+Z = ZZ = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbb{I}$. So, Z is unitary. It is also useful to think what does this operation actually do to an element of the standard basis. So, we will compute Z applied to zero. Thus, $Z\left|0\right\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \left|0\right\rangle$. And also, $Z\left|1\right\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -\left|1\right\rangle$. Therefore, we can see that Z transformation when applied to $\left|0\right\rangle$ it preserves the zero state but when it is applied to $\left|1\right\rangle$ it adds a minus in front of the one and we will call this a **phase**. Thus, Z is a **phase flip** operator and it is called the Pauli Z matrix: $\left|0\right\rangle \to \left|0\right\rangle \quad \left|1\right\rangle \to -\left|1\right\rangle$. Furthermore, if calculate what Z does to the $\left|+\right\rangle$ and $\left|-\right\rangle$ basis (the Hadamard basis), we will get $Z\left|+\right\rangle = \left|-\right\rangle \quad Z\left|-\right\rangle = \left|+\right\rangle$. So, we can see that in the Hadamard basis, the Z acts as a **bit flip** operator, just like X acted for the standard basis.

- **Example 4**: Combining X and Z: Finally, we will see that if we apply two unitary transformations consecutively, the resulting transformation will also be unitary. In fact, let us first do a unitary U_1 and then do U_2 which will give us some overall unitary U. Now, we will prove that it is unitary $U^+U = U_1^+U_2^+U_2U_1 = \mathbb{I} = UU^+$. So, we proved that if we combine two unitary matrices, we get a unitary operation. In particular, if we combine $U_1 = X$ and $U_2 = Z$, we can get a new operator, called $Y = iXZ$. So now, we will see the effect it has on the standard basis. We already know what X and Z do to the standard basis. So, for Y, $Y\left|0\right\rangle = iXZ\left|0\right\rangle = iX\left|0\right\rangle = i\left|1\right\rangle$, $Y\left|1\right\rangle = iXZ\left|1\right\rangle = -iX\left|1\right\rangle = -i\left|0\right\rangle$.

- **Example 5**: The controlled-*NOT*: The controlled-*NOT* or *CNOT* gate behaves as follows: $\left|00\right\rangle \to \left|00\right\rangle \quad \left|01\right\rangle \to \left|01\right\rangle \quad \left|10\right\rangle \to \left|11\right\rangle \quad \left|11\right\rangle \to \left|10\right\rangle$. In the computational basis, the *CNOT* gate flips the state of the second qubit if the first qubit is in state $\left|1\right\rangle$, and does nothing otherwise. As with all unitary operators, the *CNOT* gate acts linearly over superpositions. The matrix representation: $CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$.

2.6 No Cloning Theorem: The reason why we cannot copy qubits

One of the most famous principles of quantum mechanics is the ”no-cloning principle”, which states that a quantum state cannot be copied.

Definition 2.6.1 (No-Cloning Theorem)**.** *No quantum procedure transforms* $\left|\phi\right\rangle \to \left|\phi\right\rangle\left|\phi\right\rangle, \forall\left|\phi\right\rangle$.

- **Proof** N_o 1: If a copying unitary C existed, it would give us $C(\left|\psi\right\rangle \otimes \left|0\right\rangle) = \left|\psi\right\rangle \otimes \left|\psi\right\rangle$ for any input qubit $\left|\psi\right\rangle$. In contrast, let us consider such a unitary exists, so that:

$C(|\psi_1\rangle \otimes |0\rangle) = |\psi_1\rangle \otimes |\psi_1\rangle \quad C(|\psi_2\rangle \otimes |0\rangle) = |\psi_2\rangle \otimes |\psi_2\rangle$. Since C is unitary, we get $C^{+}C = \mathbb{I}$ and hence, $\langle\psi_1|\psi_2\rangle = \langle\psi_1|\psi_2\rangle \langle 0|0\rangle = (\langle\psi_1| \otimes \langle 0|)C^{+}C(|\psi_2\rangle \otimes |0\rangle) = (\langle\psi_1| \otimes \langle\psi_1|)(|\psi_2\rangle \otimes |\psi_2\rangle) = (\langle\psi_1|\psi_2\rangle)^2$. However, when $0 < |\langle\psi_1|\psi_2\rangle| < 1$ the above cannot hold and hence, such a copying unitary C cannot exist (e.g. $|\psi_1\rangle = |0\rangle$ and $|\psi_2\rangle = |+\rangle$). This property also means that we cannot determine a and b from a single copy of a qubit $|\psi\rangle = a|0\rangle + b|1\rangle$, else we could copy the qubit by making a machine that prepares a qubit in the superposition $a|0\rangle + b|1\rangle$

- **Proof** N_o 2: If a copy procedure existed, then there would be a copy transformation U that takes a qubit $|\psi\rangle$ and an empty register and an empty qubit, let us say in zero state and it would produce two copies of the state $|\psi\rangle$. Also, U should work for any state $|\psi\rangle$ independently of what that state is (it should work for any $|\psi\rangle_A$) $U|\psi\rangle_A |0\rangle_B = |\psi\rangle_A |\psi\rangle_B$. So, particularly, an example of qubit $|\psi\rangle$ that we would like to copy are actually the elements of the standard basis. So, we should be able to copy zero and one: $|\psi\rangle_A = |0\rangle_A, U|0\rangle_A |0\rangle_B = |0\rangle_A |0\rangle_B \quad |\psi\rangle_A = |1\rangle_A, U|1\rangle_A |0\rangle_B = |1\rangle_A |0\rangle_B$. We know that classical bits can be copied, so certainly there is no surprise so far that we should definitely be able to find such a copy operation U. In the quantum case, the copy operation should work for any qubit. Also, it should work if $|\psi\rangle$ is the plus state, the Hadamard basis. This means that such a copy procedure, if it existed, would turn the state plus and zero into plus plus $|\psi\rangle_A = |+\rangle_A, U|+\rangle_A |0\rangle_B = |+\rangle_A |+\rangle_B = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}^T$. As we can see, we get the equal superposition. So, if such a copy machine, such an operation U, existed, then this should be true. However, now we will see what happens if we compute the action of U on the plus state in another way, namely by first expanding the plus state $U|+\rangle|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(U|0\rangle|0\rangle + U|1\rangle|0\rangle) = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}\left(\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^T + \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T\right) = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 0 & 1\end{bmatrix}^T \neq \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}^T$. So, there is a contradiction because not both things can be true at the same time. Thus, this means that there cannot be a copy operation U that copies all three (0, 1, +) of these states. So, there cannot be any unitary that can copy qubits. That is a fundamental difference between classical and quantum information that is quite important for quantum cryptography.

2.7 The Bloch sphere

Sometimes it is useful to use a different representation to write qubits. Hence, here, we will present the Bloch sphere representation. It only works for single qubits and not for many. Let us imagine that instead of using the standard basis, zero and one, we associate the zero to the up direction and the one to the down direction. Also, instead of writing the state as usual in terms of complex numbers, α and β, we write out such states in terms of a global phase, e to the i times γ, an angle θ away from the Z-axis and an angle ϕ away from the X-axis. So, $|\psi\rangle = e^{i\gamma}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\phi}|1\rangle\right)$.

It is noted that the global phase $e^{i\gamma}$ is not interesting to us because if we measure the state, then the probabilities of getting measurement outcomes do not depend on this phase, so we cannot see it (it has no observable effects) and we can write: $|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\phi}|1\rangle$. In this form, we can think of every state as being described by two numbers, theta and phi, which give us a point on the sphere. This is known as the Bloch sphere representation.

Definition 2.7.1. *The parameterization (θ, ϕ) of $|\psi\rangle = e^{i\gamma}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\phi}|1\rangle\right)$ is called the Bloch sphere representation and a qubit can be described by a Bloch vector $\vec{r} = (\cos\phi\sin\theta, \sin\phi\cos\theta, \cos\theta)$*

Visual representation:

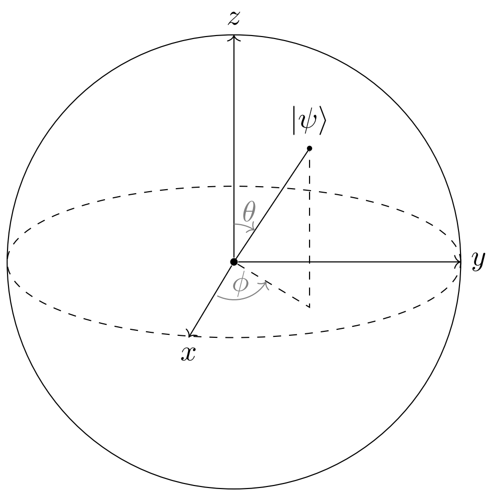


Figure 2.1: Bloch sphere

It is also useful to mention that there are some single qubit unitaries U that can be expressed as rotations on the Bloch sphere. A rotation matrix $R_S(\theta)$ is a unitary operation that rotates a qubit Bloch vector around the axes $s \in \{x, y, z\}$ by an angle θ. Such matrices are: $R_x(\theta) = e^{-i\theta X/2}$ $\quad R_y(\theta) = e^{-i\theta Y/2}$ $\quad R_z(\theta) = e^{-i\theta Z/2}$ where, X, Y, Z are the Pauli matrices. Any arbitrary single qubit operation U can be expressed as $U = e^{i\alpha}R_z(\beta)R_y(\gamma)R_z(\delta)$ for some real numbers $\alpha, \beta, \gamma, \delta$.

Finally, we will mention an example. In particular, for the state plus, zero and one, this representation is particularly easy as e.g. $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ so ϕ is zero and we can easily see that the point is on the X-axis.

As a next step, we will focus on a qubit for which the angle ϕ is zero. Since we ignore the global phase, this means that the qubit is characterized by only one number, the angle θ. So now, the Y direction does not matter and thus, we can have a simplified picture where we just have the up direction Z and the direction X and the qubit is parameterized by the angle θ. This simplified version is known as the XZ plane. If we say that a qubit is in the XZ plane, it means that we are precisely in the case where ϕ is equal to zero and we again ignore the global phase and we get $|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)|1\rangle$.

2.8 Quantum Mechanics

2.8.1 Double Slit/Young Experiment - Interference Pattern & Superposition

- **Interference**: **Thomas Young** in nineteenth century conducted an experiment using a candle-lighter, a card and a whiteboard. As it is known, light consists of photons, which are tiny structures (subatomic particles). With this experiment, Young wanted to examine the debate of whether light is particle or wave and how photons behave. The steps and the conclusions of the **Young's double-slit / interference experiment** are mentioned below:

  - If the card has no slit, the light does not pass through it, so empty whiteboard.

  - If the card has one double slit, the light will create a pattern of multiple rays of light concentrating on the whiteboard.

  - If the card has a double slit, the light will create a pattern of multiple rays of light concentrating on the whiteboard.

  - If the photons behave like a wave, then this interference pattern makes sense but if they behave like a particle then it does not, because they have to pass through those slits and then the result would be only two rays. Thus, someone could believe that they are waves but that is not completely true. On the one hand, if light is a wave, why do we have an interference pattern. Photons pass through these slits and they collide with each other and they create this pattern. On the other hand, if we try to see from which slit, they are passing through (if we try to measure it), the physicists put an observer to alarm them but then they saw that they lost this pattern as if the photons knew that they were being watched and they behaved like a particle and it is a result of the collapse of the wave function! Furthermore, when they decided to not observe but leave the observer over there, the pattern emerged again. Later, they tried to send photons one by one and mark them on the whiteboard so that they would know where those photons landed. However, even though they send them one by one, the interference pattern emerged again! The only logical explanation is that that photon is interfering with itself, and that is called superposition. In other words, photons collide with each other and have a constructive or destructive interference and then create this pattern.

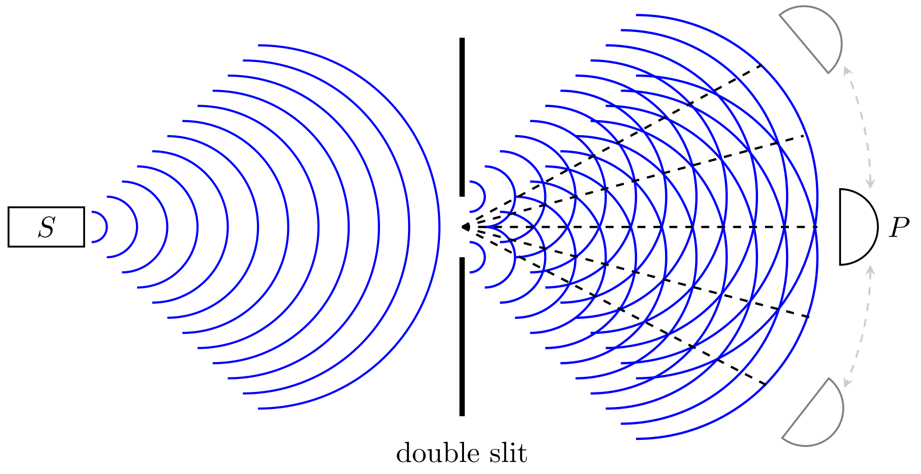


Figure 2.2: Interference pattern

- **Superposition**: Many of the most counterintuitive aspects of quantum physics arise from the superposition principle which states that if a quantum system can be in one of two states, then it can also be in any linear superposition of those two states. For instance, the state of the electron could be $\frac{1}{\sqrt{2}}\left|0\right\rangle+\frac{1}{\sqrt{2}}\left|1\right\rangle$ or $\frac{1}{\sqrt{2}}\left|0\right\rangle-\frac{1}{\sqrt{2}}\left|1\right\rangle$ or an infinite number of other combinations of the form $a_0\left|0\right\rangle+a_1\left|1\right\rangle$ (the coefficients a_0, a_1 are called amplitudes and in general they can be complex numbers as long as $|a_0|^2+|a_1|^2=1$) and such a superposition is the basic unit of encoded information and it is called qubit.

  In other words, this means that it is in the left slit as well as in the right slit at the same time and only when we try to observe it, we break the superposition. So, it does not mean that we have multiple photons but it is actually the possibilities of being everywhere when we do not observe it and at the time it reaches the destination, it lands somewhere and it creates this interference pattern. In other words, a qubit can have 0 and 1 and anything in between at the same time, not the value but the possibilities. So, even a single photon can pass through the left or the right slit, interfere with itself and land anywhere on the whiteboard. A thought experiment that illustrates the paradox of quantum superposition is *Schrödinger's cat*.

2.8.2 Entanglement

- Inside an atom there are electrons in specific positions when we try to measure them. We can find them by using principal (n), orbital angular momentum (l), magnetic (m_l), electron spin (m_s).

  We are interested in the electron spin. An electron can either have spin up or spin down and we can measure this value in order to find out which one it is. Before measurement, we cannot determine the spin since it is in superposition. However, when we measure it, even though we can find out the spin, the whole system collapses into some kind of state. In fact, if we try to measure again, we will get the same result.

  We can associate spin down with 0 and spin up with 1.

Physicists have discovered that when 2 electrons interact with each other, they can be in a set up called entanglement, meaning that they are entangled with each other and their spins have a correlation between them. So, when we measure one electron with spin up, then the other one will immediately be spin down, no matter where they are located. It can be used to make very large calculations in very little time.

- In other words, if we recall the two principles of quantum computing. Firstly, a physical system in a definite state can still behave randomly. Secondly, two systems that are too far apart to influence each other can nevertheless behave in ways that, though individually random, are somehow strongly correlated. The core idea behind the second principle is entanglement. Upon reading the principle, one might be inclined to think that entanglement is simply strong correlation between two entities – but entanglement goes well beyond mere perfect (classical) correlation. If two people read the same paper, they will have learned the same information. If a third person comes along and reads the same paper, they also will have learned this information. All three people in this case are perfectly correlated, and they will remain correlated even if they are separated from each other.

  Quantum entanglement is a bit more subtle. In the quantum world, two people could read the same quantum paper, and yet they will not learn what information is actually contained in the paper until they get together and share their information. However, when they are together, they find that they can unlock more information from the paper than they initially thought possible. Thus, quantum entanglement goes much further than perfect correlation. To demonstrate this, we will define the **controlled-NOT ($CNOT$) gate** and the composition of two systems. The convention is to label states by writing the first qubit's name in the rightmost position, so that we can easily convert from binary to decimal. As a result, we define the tensor product between operations q_0 and q_1 by $q_1 \otimes q_0$. Taking q_0 as the control and q_1 as the target, then we can get the $CNOT$ as we described it before. This is non-standard in the quantum community, but more easily connects to classical computing, where the least significant bit (LSB) is typically on the right. An entangled state of the two qubits can be made via an H gate on the control qubit, followed by the CNOT gate. This generates a particular maximally entangled two-qubit state known as a Bell state, named after John Stewart Bell. All in all, the controlled-NOT gate is a 2-qubit quantum gate that conditionally applies the NOT (known as X) gate on the second (target) qubit when the first (control qubit) is in state $|1\rangle$. Such a gate acts on quantum states in quantum superposition. In the computational basis, the $CNOT$ gate flips the state of the second qubit if the first qubit is in state $|1\rangle$ and does nothing otherwise. As with all unitary operators, the $CNOT$ gate acts linearly over superpositions.

  In conclusion, suppose we have two qubits, the first $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ and the second $\beta_0 |0\rangle + \beta_1 |1\rangle$, then the joint state of the two qubits is the tensor product of the two. However, given an arbitrary state of two qubits, we cannot specify the state of each individual qubit in this way because the two qubits are entangled and cannot be decomposed into the states of the individual qubits (e.g. the Bell states). Entanglement is one of the most mysterious aspects of quantum mechanics and is ultimately the source of the power of quantum computation.

2.9 Superdense Coding and Quantum Teleportation

This section examines the first protocols for quantum information that are inherently quantum and involve two parties who wish to perform some communication tasks between them. In descriptions of such communication protocols, it is very common to name the two parties "Alice" and "Bob", for convenience.

A **channel** is a mathematical transformation that occurs on bits/qubits when undergoing a general quantum operation.

A **classical communication channel** can carry classical bits.

A **quantum communication channel** is a communication line that can carry qubits between two remote locations.

The protocols require that Alice and Bob will initially share an entangled pair of qubits in *Bell state / EPR pair*: $|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. This state should be created ahead of time and it should be made in order to give rise to the entanglement between them. This state should be created ahead of time and it should be made in order to give rise to the entanglement between them. After that, each party takes one qubit and this entanglement becomes a *resource* which Alice and Bob can use to achieve the protocols we will describe.

2.9.1 Superdense Coding

Let us first mention the setting. Suppose Alice wishes to send Bob two classical bits of information. A way to achieve this task is **Superdense Coding** over a quantum channel, because it requires Alice to send only one qubit to Bob.

The Superdense Coding Protocol in which Alice sends two bits of classical information by sending one physical qubit to Bob can be seen below:

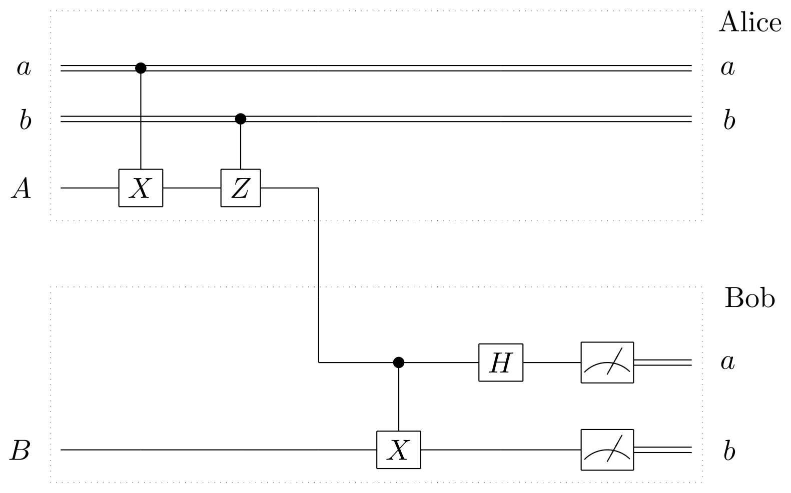


Figure 2.3: Superdense coding

The steps that we need to follow are described below:

- First of all, Alice and Bob must share the Bell state that we described above.
- Also, let us suppose Alice possesses the first qubit and Bob the second qubit. She performs one of the four given 1-qubit gates, depending on the 2 classical bits she wishes to communicate to Bob. Thus, if Alice wants to send some bits, she has to apply the following:
  - For 00: $I \otimes I : \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \mapsto \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = |\beta_{00}\rangle$
  - For 01: $X \otimes I : \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \mapsto \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) = |\beta_{01}\rangle$
  - For 10: $Z \otimes I : \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \mapsto \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) = |\beta_{10}\rangle$
  - For 11: $Z \cdot X \otimes I : \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \mapsto \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) = |\beta_{11}\rangle$
- So, after the application of the appropriate gate, Alice sends her qubit to Bob.
- Bob possesses one of the four Bell states, depending on the classical bit Alice wished to send to him, and he can now perform a measurement of the joint 2-qubit state, with respect to the Bell basis, by performing a change of basis to the Bell basis, and then performing a measurement in the computational basis.

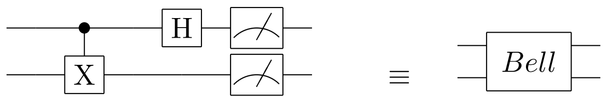


Figure 2.4: Bell basis

- The result of the Bell measurement shows to Bob which Bell state he possesses and it allows him to determine with certainty the two classical bits she wanted to communicate to him.

2.9.2 Quantum Teleportation

We use Quantum Teleportation to transform some state from one qubit to another. The setting is that Alice wishes to communicate the state of a qubit to Bob, by sending only two bits of classical information to him.

Note: The Teleportation protocol can be thought of as a flipped version of the Superdense Coding protocol, in the sense that Alice and Bob merely “swap their equipment.”

A circuit implementation of the Quantum Teleportation is the following:

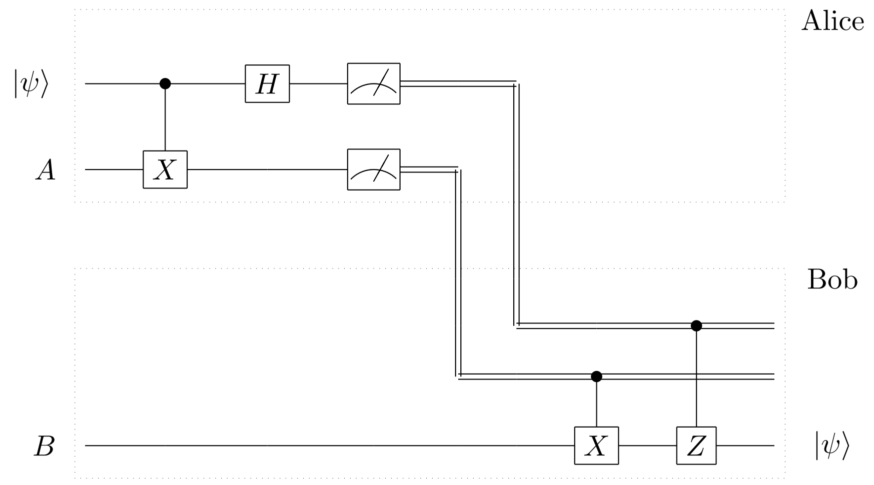


Figure 2.5: Quantum teleportation

The steps that we need to follow in order to achieve this are described below:

- First of all, let us suppose that Alice has only a classical channel linking her to Bob.
- To send the state of a qubit exactly, it would seem that Alice would either have to send the physical qubit itself, or she would have to communicate the two complex amplitudes with infinite precision. However, if Alice and Bob possess an entangled state this intuition is wrong and a quantum state can be sent exactly over a classical channel. Thus, it is required that both initially share the Bell state. Suppose
- Alice wants to teleport the state $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ to Bob.
- The 3-qubit state possessed jointly by Alice and Bob is initially $|\psi\rangle |\beta_{00}\rangle$
- By regrouping the qubits, the state can be written as

$$|\psi\rangle |\beta_{00}\rangle = \frac{1}{2} |\beta_{00}\rangle |\psi\rangle + \frac{1}{2} |\beta_{01}\rangle (X |\psi\rangle) + \frac{1}{2} |\beta_{10}\rangle (Z |\psi\rangle) + \frac{1}{2} |\beta_{11}\rangle (X \cdot Z |\psi\rangle)$$

- Alice can now make a measurement of the first two qubits in the Bell basis and the joint Alice-Bob state after this measurement is one of $|\beta_{00}\rangle |\psi\rangle$, $|\beta_{01}\rangle (X |\psi\rangle)$, $|\beta_{10}\rangle (Z |\psi\rangle)$, $|\beta_{11}\rangle (X \cdot Z |\psi\rangle)$ each with probability $\frac{1}{4}$.
- The classical bits a and b resulting from Alice's measurement indicate which of the four states is obtained. When Alice sends these two bits to Bob, he learns and depending on which state he has, he performs an operation to transform his state into $|\psi\rangle$
- So, Bob conditionally applies Z and X to his qubit conditioned on the values a and b, respectively.
- After this transformation, it is guaranteed that he will have the state $|\psi\rangle$ and so the state has been successfully teleported from Alice.

For *deeper understanding* of the protocol, here is a step-by-step explanation:

Classical bits can be easily cloned because they only possess 0 or 1 but with qubits it is not easy because we cannot clone quantum states (**No Cloning Theorem**) since we do not know the state before we measure it and when we measure it, we lose the information. So, we call it $|\psi\rangle$ and it is $|\psi\rangle = a|0\rangle + b|1\rangle$. Also, we have 2 qubits (q_0, q_2) and 1 helper qubit (q_1) and we want to transfer the information from $|\psi\rangle$ to q_2.

The total quantum state will be:

$$|\psi\rangle = a|0\rangle + b|1\rangle \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle)$$

Later, Alice applies the $CNOT$, so: $|\psi\rangle = \frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|101\rangle + b|110\rangle)$ After that, she applies the H:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + a|100\rangle + a|111\rangle + b|001\rangle + b|010\rangle - b|101\rangle - b|110\rangle)$$

Then. she measures this and the results are:

q_0, q_1	00	01	10	11
q_2	$a\|0\rangle + b\|1\rangle$	$a\|1\rangle + b\|0\rangle$	$a\|0\rangle - b\|1\rangle$	$a\|1\rangle - b\|0\rangle$
	I	X	Z	$X \cdot Z$

Depending on these values, Bob will have to adjust his own qubit in order to have $|\psi\rangle$. Alice also does not know this state so, Bob will have to make some operations to adjust his qubit. This will help us eventually clone this $|\psi\rangle$ state into q_2. When Alice measures q_0, q_1 she can get the values above and depending on them Bob can adjust his own q_2 to find the state. So, from the measurement of Alice and by applying some gates, we can find $|\psi\rangle$. That is why Bob will apply I, X, Z, $X \cdot Z$ gates. Bob will decide which gate to implement by looking at what Alice has measured. We do this by applying CZ, CX gates. Eventually, when we measure q_2 it will be in the $|\psi\rangle$ notation.

A more detailed depiction of Quantum Teleportation is the following:

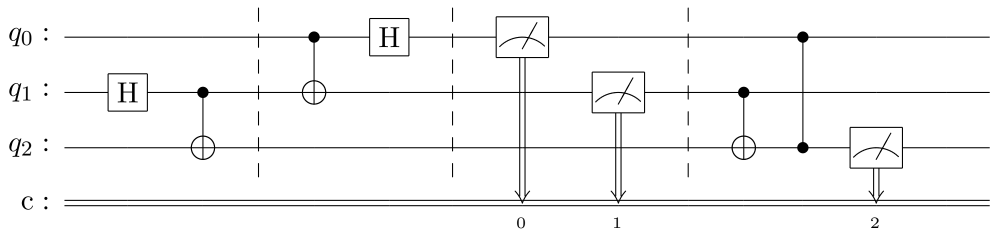


Figure 2.6: Quantum Teleportation

Implementation:

An implementation of the Superdense Coding can be seen in this **link** and another more detailed implementation of Quantum Teleportation run on IBM Hardware can be seen in this **link**.

Chapter 3

Query Model of Computation

3.1 Introduction

During our analysis, we will use the **query model** of computation as a tool for developing quantum algorithmic techniques. In Mathematics, when we want to model computations we think of a process where information is given as input, we make a computation and we get an output. Classical computers work in this way and they interact with both us and with other computers.

In the query model of computation, however, the entire input is not provided to the computation but, it is given in a form of a *function*, which the computation accesses by making *queries*. So, we might view that computations in the query model have random access to bit of the input and we say that the input is provided by an **oracle** or **black box** (we cannot look inside the function and understand how it works, we can only evaluate it on arguments we select) in the context of the query model. As a result, a complete description of the input is hidden from the computation and the only access to it is by asking questions. An analogous example, is the case where ancient Greeks consulted the Oracle at Delphi: she did not tell them everything see knew, but only answers to specific questions.

We will work with binary strings so we will write $\{0,1\} = \{0,1\}$ to refer to the binary alphabet, thus the input will be represented by a function $f : \{0,1\}^n \to \{0,1\}^m$, for two positive integers n, m. Moreover, when we will say that a computation makes a query, it will mean that string $x \in \{0,1\}^n$ is selected and then the string $f(x) \in \{0,1\}^m$ is made available.

Finally, in order to measure efficiency of our query algorithms, we will count the number of queries to the input they require. Although, that is related to the time required to perform a computation, it is not entirely true because we will ignore the time required for operations other than the queries and we will treat them as if they each have unit cost.

Furthermore, we will examine query problems where we have a promise on the input, we will be given some guarantee on the input and we will not be responsible for what happens when this guarantee is not met. In other words, some input functions (the ones for which the promise is not satisfied) are considered as "do not care" inputs. Some examples:

- **Parity**: The input function is $f : \{0,1\}^n \to \{0,1\}$. We can think of f as a sequence of 2^n bits to which we have random access. The goal is to determine whether the number of strings $x \in \{0,1\}^n$ for which $f(x) = 1$ is even or odd, or, in other words, to compute the parity (or XOR) of these bits. The desired output is 0, if the set $\{x \in \{0,1\}^n : f(x) = 1\}$ has an even number of elements and 1, if the set has an odd number of elements.
- **Unique search**: The input function is $f : \{0,1\}^n \to \{0,1\}$. The promise is that there is exactly one string $z \in \{0,1\}^n$ for which $f(z) = 1$, with $f(x) = 0, \forall x \neq z$. The goal is to find this unique string z.

It should be noted that there might exist some query problems of interest that are not so naturally easy to describe and we might come up with very complicated and highly controverted problems where it might be difficult to imagine that someone would ever actually pursue to solve them in practice. However, it is part of the process to seek problems that reveal potential advantages of quantum computing and maybe one day these problems will inspire new ideas.

3.2 Query Gates

In the query model of computation the entire input is not provided to the computation but, it is given in a form of a *function*, which the computation accesses by making *queries*. Regarding the circuit models of computation, those queries are made by special gates called query gates. For classical Boolean circuits, we can use query gates that compute the input function f directly as it is depicted in the figure. When we create such a Boolean circuit the input function f is accessed through these gates and the number of queries that the circuit makes is the number of query gates that appear in the circuit and also, the wires are initialized to fixed values and they are part of the algorithm and not the input.

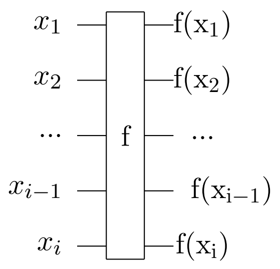


Figure 3.1: Classical Boolean circuit

However, for quantum circuits, this definition of query gates does not work very well because there are some non-unitary gates for some functions f and we will not be able to apply them to quantum states. For that reason, we use *unitary query gates* that operate as it depicted in the figure. Here, the assumption is that $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ are arbitrary strings. Moreover, the notation $y \oplus f(x)$ refers to the bitwise exclusive OR of strings (of length m, e.g. $001 \oplus 101 = 100$). Furthermore, the gate U_f echos the top input string x and XOR the function value $f(x)$ onto the bottom input string y and it is a unitary operation for every function f. As we saw previously, U_f is a permutation matrix, meaning a matrix with a single 1 in each row and each column and with all other entries being 0 and by applying such a matrix to a vector, we simply shuffle the entries of the vector without changing the vector's Euclidean norm.

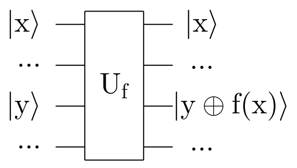


Figure 3.2: Quantum circuit

Note that we will examine query algorithms by simply counting the number of queries that a query algorithm makes and we will ignore the difficulty of physically constructing the query gates because we are focusing on defining the theoretical model that will reveal the potential advantages of quantum computing.

In the next sections, we are going to examine and analyze some of the first and most basic quantum algorithms that exist. We will first set the problem that we are trying to solve and then we will find the solution both in a classical and in a quantum setting and we are going to compare them and implement them.

We will show the computational advantages that quantum computers have to offer over classical computers. On the one hand, quantum computers might provide faster solutions to some computational problems that classical computers are too slow to solve. On the other hand, regarding computer memory, quantum computers have a limited potential to offer advantages in space usage over classical ones.

What is more, it should be clear that quantum computers cannot provide computational solutions to problems that classical computers cannot solve, irrespective of the resources required (e.g. the halting problem). Indeed, quantum computers can be simulated by classical computers, thus any problem that can be solved quantumly, can also be solved classical, even though it might take longer to find a solution.

Chapter 4

Quantum Algorithms: Deutsch & Deutsch-Jozsa Algorithm

4.1 Introduction

As we discussed, quantum algorithms provide significant advantages over classical algorithms in the query model. The classical computers that exist today are incredibly fast, and some might be inclined to believe that computers are so fast that no computational problem is beyond their reach. However, this belief is false because there are some computational problems that are so inherently difficult (e.g. integer factorization) that, although there exist algorithms to solve them, no computer on the planet Earth today is fast enough to run these algorithms to completion on even moderately sized inputs within our lifetime.

4.2 Deutsch & Deutsch-Jozsa Algorithm

In this section, we have analyzed the Deutsch & Deutsch-Jozsa Algorithm in two ways and we are going to present them both below.

The **Deutsch algorithm** can solve the parity problem for the special case that $n = 1$. To be precise, the input is represented by a function $f : \{0, 1\} \to \{0, 1\}$ form one bit to one bit and there are 4 such functions:

a	$f_1(a)$
0	0
1	0

a	$f_2(a)$
0	0
1	1

a	$f_3(a)$
0	1
1	0

a	$f_4(a)$
0	1
1	1

The first and last are called **constant** and the middle two are called **balanced**, meaning that the two possible output values for the function occur the same number of times as we range over the inputs. The Deutsch problem is to determine which of these two categories the input function belongs to. We can view the input function f in Deutsch's problem as representing random access to a two-bit string $f(0)f(1)$: $f_1 \mapsto 00$, $f_2 \mapsto 01$, $f_3 \mapsto 10$, $f_4 \mapsto 11$. And that is the same as computing the parity (or XOR) of the two bits. Also, knowing just one of two bits does not provide any information at all about their parity (e.g. if $f(1) = 1$ then if

$f(0) = 1$ we get 0 and if $f(0) = 0$ we get 1). Thus, the Boolean circuit is the best we can do in terms of the number of queries required to solve this problem.

For the **Deutsch-Jozsa problem**, the input function is $f : \{0,1\}^n \to \{0,1\}$ and the goal is to determine the whether f is constant and output 0 or f is balanced and output 1. Is should be clear that when $n > 1$, there are functions of the form $f : \{0,1\}^n \to \{0,1\}$ that are neither constant nor balanced (e.g. for $n = 2$, $f(00) = 0, f(01) = 0, f(10) = 0, f(11) = 1$) but for the Deutsch-Jozsa problem we consider them to be "do not care" inputs.

The Deutsch-Jozsa algorithm with its single query says that if every one of the n measurement outcomes is 0, then f is constant, else if at least one of them is 1, then f is balanced.

4.2.1 Theoretical Analysis

Phase Kick-Back

In the classical basis, the $CNOT$ gate appears to do nothing to the control qubit but in reality, it affects the control qubit as much as it does the target qubit. However, in the Hadamard basis, the $CNOT$ gate appears to switch the role of the control and target qubit: $CNOT\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \mapsto \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. We also notice that the second value is an eigenvector of the NOT gate with eigenvalue -1 and an eigenvector of the identity gate with eigenvalue $+1$. So, the $CNOT$ applies the NOT gate to the target qubit if the first qubit is in state $|1\rangle$ and we get: $CNOT$: $|1\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \mapsto |1\rangle\left(NOT\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right) = |1\rangle\left((-1)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right) = -|1\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. However, if the $CNOT$ applies the identity gate, meaning that it does nothing to the target qubit when the first qubit is in state $|0\rangle$, then we get: $CNOT$: $|0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \mapsto |0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. Thus, we can conclude that when the target qubit is in an eigenstate, it is not affected and we can effectively treat the eigenvalue as being *"kicked back"* to the control register, so: $CNOT$**:** $|b\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \mapsto (-1)^b|b\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$ where $b \in \{0,1\}$. Finally, when the control qubit is in a superposition of $|0\rangle$ and $|1\rangle$, then: $CNOT$**:** $(a_0|0\rangle + a_1|1\rangle)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \mapsto (a_0|0\rangle - a_1|1\rangle)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. And if we remind ourselves that $Z = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$ this corresponds to applying the Z gate to the control qubit.

Generalization:

Now, let us suppose that we have a more general 2-qubit gate U_f implementing an arbitrary function f:$\{0,1\} \to \{0,1\}$ by mapping U_f: $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$. Let us now fix the target register to the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ and analyze the action of U_f on an arbitrary basis in the control qubit: U_f: $|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \mapsto \left(\frac{U_f|x\rangle|0\rangle-U_f|x\rangle|1\rangle}{\sqrt{2}}\right) = \left(\frac{|x\rangle|0\oplus f(x)\rangle-|x\rangle|1\oplus f(x)\rangle}{\sqrt{2}}\right) = |x\rangle\left(\frac{|0\oplus f(x)\rangle-|1\oplus f(x)\rangle}{\sqrt{2}}\right)$. It is known that the action of $\oplus f(x)$ has no effect on a single bit if $f(x) = 0$ and it flips the state of the bit if $f(x) = 1$. Therefore, the expression $\frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle)$ in each case will be: $f(x) = 0$ **:** $\left(\frac{|0\oplus f(x)\rangle-|1\oplus f(x)\rangle}{\sqrt{2}}\right) = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$,

$f(x) = 1$ **:** $\left(\frac{|0\oplus f(x)\rangle - |1\oplus f(x)\rangle}{\sqrt{2}}\right) = \frac{|1\rangle - |0\rangle}{\sqrt{2}} = -\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$. Those two possibilities differ by a factor of (-1), thus we can write: $\frac{|0\oplus f(x)\rangle - |1\oplus f(x)\rangle}{\sqrt{2}} = (-1)^{f(x)}\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$. So, the overall state can be written as: $(-1)^{f(x)}|x\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$. Moreover, if we associate the $(-1)^{f(x)}$ factor with the first qubit, we get: U_f: $|x\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \mapsto (-1)^{f(x)}|x\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$. Finally, when the control qubit is in a superposition of $|0\rangle$ and $|1\rangle$, we get: U_f: $(a_0|0\rangle + a_1|1\rangle)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \mapsto \left((-1)^{f(0)}a_0|0\rangle + (-1)^{f(1)}a_1|1\rangle\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$. In conclusion, we can think of U_f as a 1-qubit operator, $|b\rangle \mapsto |b \oplus f(x)\rangle$ acting on the second qubit, controlled by the state $|x\rangle$ of the first register. So, the equivalent circuit will be:

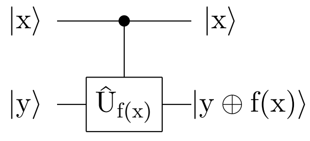


Figure 4.1: U_f equivalent circuit

It is noted that the state $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ of the second register is an eigenvector of U_f. In other words, this state of the target register is an eigenstate of U_f and the eigenvalue $(-1)^{f(x)}$ can be *"kicked back"* in front of the target register. The technique of inputting an eigenstate to the target qubit of an operator like U_f and associating the eigenvalue with the state of the control register will be used in the next algorithm of Deutsch.

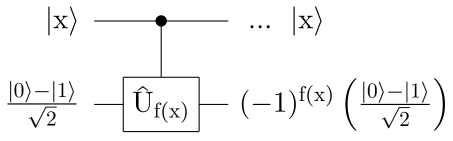


Figure 4.2: Kick Back

Additional note: Let us take $b, c \in \{0, 1\}$. It is true that $|b \oplus c\rangle = X^c|b\rangle$ (e.g. $|b \oplus 0\rangle = |b\rangle = 1|b\rangle = X^0|b\rangle$, or $|b \oplus 1\rangle = |\neg b\rangle = X|b\rangle = X^1|b\rangle$). Thus, we can write $U_f(|b\rangle|a\rangle) = |b \oplus f(a)\rangle|a\rangle = (X^{f(a)}|b\rangle)|a\rangle$. So, $U_f(|\psi\rangle|a\rangle) = (X^{f(a)|\psi\rangle})|a\rangle$. Thus, $U_f(|-\rangle|a\rangle) = (X^{f(a)|-\rangle})|a\rangle = (-1)^{f(a)}|-\rangle|a\rangle$. Because $X|-\rangle = -|-\rangle$ and the vector $|-\rangle$ is called an eigenvector of the matrix X with eigenvalue -1.

Deutsch Algorithm

- **The Problem**:

  Suppose we are given a reversible circuit for computing an unknown 1-bit function $f : \{0,1\} \to \{0,1\}$. We treat this reversible circuit as an oracle". This means that we can apply the circuit to obtain values of $f(x)$ for given inputs x, but we cannot gain any information about the inner workings of the circuit to learn about the function f. The problem is to determine the value of $f(0) \oplus f(1)$. If we determine that $f(0) \oplus f(1) = 0$, then we know that $f(0) = f(1)$ but we do not know the value and we say that f is "**constant**". If we determine that $f(0) \oplus f(1) = 1$, then we know that $f(0) \neq f(1)$ and we say that f is "**balanced**". Thus, determining $f(0) \oplus f(1)$ is equivalent to determining whether the function f is **constant** or **balanced**.

  The Deutsch Problem:
  Input: An oracle for computing an unknown function $f : \{0,1\} \to \{0,1\}$.
  Problem: Determination of the $f(0) \oplus f(1)$ by making queries to f.
  Output: 0 is f is constant, 1 if f is balanced.

- **The Solution**:

  - **Classical**:

    Classically, how many queries for f should be made to the oracle in order to determine $f(0) \oplus f(1)$? The correct answer is 2. Let us suppose we compute $f(0)$ using one classical query. Then the value of $f(1)$ could be 0, making $f(0) \oplus f(1) = 0$ or the value of $f(1)$ could be 1, making $f(0) \oplus f(1) = 1$. Without making a second query to the oracle to determine the value of $f(1)$, we can make no conclusion about the value of $f(0) \oplus f(1)$.

  - **Quantum**:

    The Deutsch algorithm is a quantum algorithm capable of determining the value of $f(0) \oplus f(1)$ by making only a single query to a quantum oracle for f. This implementation can be achieved by replacing every reversible classical gate in the given circuit with the analogous unitary quantum gate in the given reversible circuit for f in order to make it a quantum circuit. The quantum circuit can be expressed as a unitary operator: $U_f : |x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle$. The next step now is to have quantum bits as inputs. We define U_f so that if we set the second input qubit to be in the state $|y\rangle = |0\rangle$, then:

    * $|x\rangle = |0\rangle$ in the first input qubit will give $|0 \oplus f(0)\rangle = |f(0)\rangle$ and we can think of it as a quantum version of the classical input bit 0
    * $|x\rangle = |1\rangle$ in the first input qubit will give $|f(1)\rangle$ and we can think of it as a quantum version of the input bit 1.

    Another possibility is that the state of the input qubit to be some superposition of $|0\rangle$ and $|1\rangle$. Let us suppose that the first qubit is in the superposition state, meaning $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and the second input qubit is $|y\rangle = |0\rangle$. Then, the two qubit inputs to U_f will be: $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|0\rangle)$. Thus, the output of U_f will be: $U_f\left(\frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|0\rangle)\right) = \frac{1}{\sqrt{2}}(U_f|0\rangle|0\rangle + U_f|1\rangle|0\rangle) =$

$\frac{1}{\sqrt{2}}(|0\rangle|0 \oplus f(0)\rangle + |1\rangle|0 \oplus f(1)\rangle) = \frac{1}{\sqrt{2}}(|0\rangle|f(0)\rangle + |1\rangle|0 \oplus f(1)\rangle)$. So, it occurs that U_f has simultaneously computed the value of f on both possible inputs 0 and 1 in superposition. Regarding the measurement, if we now measure the output state in the computational basis, we will get either $|0\rangle|f(0)\rangle$ with probability 50% or $|1\rangle|0 \oplus f(1)\rangle$ with probability 50%. After the measurement, the output state will be either $|f(0)\rangle$ or $|f(1)\rangle$, respectively, and so any other measurements of the output state will yield the same result. Thus, although we have successfully computed two values in superposition, only one of those values is accessible through a quantum measurement in the computational basis. However, we are only interested in individual values of $f(x)$, but in reality, we are interested in the value of $f(0) \oplus f(1)$. In conclusion, the Deutsch algorithm shows that by using interference we can obtain such global information about the f more efficiently than the classical way.

- **Implementation**:

The implementation of the Deutsch algorithm by a quantum circuit is the following:

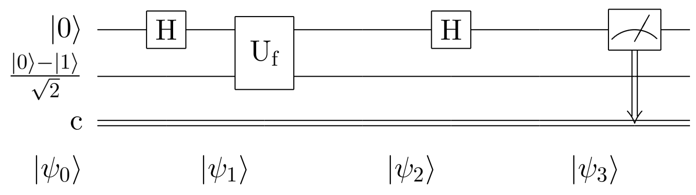


Figure 4.3: Deutsch algorithm

It is noted that the second input bit has been initialized to the state $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ and this can easily be created from the state $|1\rangle$ by applying a single Hadamard gate but it is not shown in the figure because we want to emphasize a certain symmetry that is characteristic of these algorithms. In order to analyze the behavior of the quantum algorithm more easily, we will work though the state at each stage of the circuit.

First, the input state is $|\psi_0\rangle = |0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. Then, the first Hadamard gate is applied to the first qubit and we get $|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) = \frac{1}{\sqrt{2}}\left(|0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) + |1\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right)$. After that, we apply the U_f gate and we get:

$$\begin{aligned}
|\psi_2\rangle = \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) + (-1)^{f(1)}|1\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\right) &= \\
= \left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) &\overset{(*)}{=} \\
= (-1)^{f(0)}\left(\frac{|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)&
\end{aligned}$$

$$(*) : (-1)^{f(0)}(-1)^{f(1)} = (-1)^{f(0)\oplus f(1)}$$

- If f is a **constant** function, $f(0) \oplus f(1) = 0$, then we get: $|\psi_2\rangle = (-1)^{f(0)} \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. And the final Hadamard gate on the first qubit transforms the state to: $|\psi_3\rangle = (-1)^{f(0)} |0\rangle \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. The squared norm of the basis state $|0\rangle$ in the first qubit is 1 and as a result, the measurement of the first qubit is certain to return the value $0 = f(0) \oplus f(1)$.
- If f is a **balanced** function, $f(0) \oplus f(1) = 1$, then we get: $|\psi_2\rangle = (-1)^{f(0)} \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. And the final Hadamard gate on the first qubit transforms the state to: $|\psi_3\rangle = (-1)^{f(0)} |1\rangle \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. The squared norm of the basis state $|1\rangle$ in the first qubit is 1 and as a result, the measurement of the first qubit is certain to return the value $1 = f(0) \oplus f(1)$.

Therefore, a measurement of the first qubit at the end determines the value $f(0) \oplus f(1)$ and thus, whether the function is **constant** or **balanced**.

Note: The operator U_f can be viewed as a single-qubit operator $\hat{U}_{f(x)}$ whose action on the second qubit is controlled by the state of the first qubit. The state $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ is an eigenstate of with eigenvalue $(-1)^{f(x)}$ and by encoding these eigenvalues in the phase factors of the control qubit, we are able to determine $f(0) \oplus f(1)$ by determining the relative phase factor between $|0\rangle$ and $|1\rangle$. Finally, the distinction between $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ is done using the Hadamard gate.

Deutsch-Jozsa Algorithm

- **The Problem**:

  This algorithm is a straight forward generalization of the problem solved by the Deutsch algorithm. In particular, we are given a reversible circuit implementing an unknown function f, but this time f is a function from n-bit strings to a single bit. Thus $f : \{0,1\}^n \to \{0,1\}$. Moreover, it is promised that f is either **constant** (meaning $f(x)$ is the same $\forall x$), or f is **balanced** (meaning $f(x) = 0$ for exactly half of the input strings x and $f(x) = 1$ for the other half of the inputs). The problem is to determine whether f is **constant** or **balanced** by making queries to the circuit for f.

  The Deutsch-Jozsa Problem:
  Input: An oracle for computing an unknown function $f : \{0,1\}^n \to \{0,1\}$.
  Promise: f is either a constant or a balanced function.
  Problem: Determination of whether f is constant or balanced by making queries to f.
  Output: 0 if f is constant, 1 if f is balanced.

- **The Solution**:

  - **Classical**:

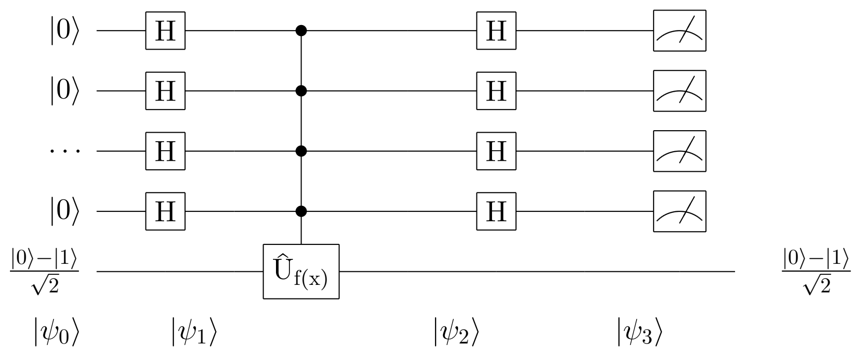
    Classically, we suppose that we have used the oracle to determine $f(x)$ for exactly half of the possible inputs x (meaning that we have made 2^{n-1} queries to f) and that all queries have returned $f(x) = 0$. Thus, we would strongly suspect that f is constant but, it is possible that if we queried f on the remaining 2^{n-1} inputs, we might get $f(x) = 1$ each time. So, it is possible that f is balanced. Therefore, in the worst case, using a classical algorithm, we cannot decide with certainty whether f is constant or balanced using any less than $2^{n-1}+1$ queries.

- **Quantum**:

  A quantum algorithm can take advantage of quantum superposition and interference to determine this global property of f. It is also noted that the property of being constant or balanced is a global property of f. Thus, the Deutsch-Jozsa algorithm will determine whether f is constant or balanced by making only one query to a quantum version of the reversible circuit for f.

- **Implementation**:

The quantum operation can be expressed as $U_f : |x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$. It is noted that x above refers to an n-bit string. We can think of U_f as a 1-qubit operator controlled by the register of qubits in the state $|x\rangle$. We can also see that $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ is an eigenstate of $\hat{U}_{f(x)}$ with eigenvalue $(-1)^{f(x)}$. Thus, the circuit will be as follows:



$|0\rangle$
$|0\rangle$
$\cdots$
$|0\rangle$
$\frac{|0\rangle-|1\rangle}{\sqrt{2}}$
H
$\hat{\text{U}}_{\text{f(x)}}$
$\frac{|0\rangle-|1\rangle}{\sqrt{2}}$
$|\psi_0\rangle$
$|\psi_1\rangle$
$|\psi_2\rangle$
$|\psi_3\rangle$

Figure 4.4: Deutsch-Jozsa algorithm

There are some similarities between the circuit of the Deutsch algorithm and the circuit for the Deutsch-Jozsa algorithm. In fact, in a place of a simple 1-qubit Hadamard gate, we now have tensor products of n 1-qubit H gates and it is denoted as $H^{\otimes n}$. We will also use $|0\rangle$ or $|0\rangle^{\otimes n}$ to denote the state that is the tensor product of n qubits, each in the state $|0\rangle$.

Now, we will try to analyze the states through each step of the circuit. Initially, the state is $|\psi_0\rangle = |0\rangle^{\otimes n}\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. Next, we consider the action of an n-qubit Hadamard transformation on the state $|0\rangle^{\otimes n}$, so $H^{\otimes n}|0\rangle^{\otimes n} = \left(\frac{1}{\sqrt{2}}\right)^n (|0\rangle+|1\rangle)\otimes(|0\rangle+|1\rangle)\otimes \cdots \otimes (|0\rangle+|1\rangle)$. And, by expanding out the tensor product, this can be written as: $H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle$. The n-qubit H gate is acting on the n-qubit state of all zeros and gives a superposition of all n-qubit basis states with the amplitude $\frac{1}{\sqrt{2^n}}$, called an "equally weighted superposition". As a result, the state immediately after the first $H^{\otimes n}$ in the algorithm is: $|\psi_1\rangle = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$. We notice now that the query register is now in an equally weighted superposition of all the possible n-bit input strings. After that, we consider the state immediately after the U_f gate and we get: $|\psi_2\rangle = \frac{1}{\sqrt{2^n}} U_f \left(\sum_{x\in\{0,1\}^n} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \right) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$. It is noted that we have associated the phase shift of $(-1)^{f(x)}$ with the first qubit. For the analysis of the state after the interference is completed by the second H gate, we consider the action of the n-qubit H gate on an n-qubit basis state $|x\rangle$. It is easy to verify that the effect of the 1-qubit H gate on a 1-qubit basis state $|x\rangle$ can be written as $H|x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^x |1\rangle) = \frac{1}{\sqrt{2}} \sum_{z\in\{0,1\}} (-1)^{xz} |z\rangle$. Thus, the action of H transformation on an n-qubit basis state $|x\rangle$ is:

$$\begin{aligned} H^{\otimes n}|x\rangle = H^{\otimes n}(|x_1\rangle |x_2\rangle \dots |x_n\rangle) = H|x_1\rangle H|x_2\rangle \dots H|x_n\rangle = \\ = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_1}|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_2}|1\rangle)\dots\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_n}|1\rangle) = \\ = \frac{1}{\sqrt{2^n}} \sum_{z_1 z_2 \dots z_n \in\{0,1\}^n} (-1)^{x_1 z_1 + x_2 z_2 + \dots x_n z_n} |z_1\rangle |z_2\rangle \dots |z_n\rangle = \\ = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{z}\in\{0,1\}^n} (-1)^{\mathbf{xz}} |\mathbf{z}\rangle \end{aligned}$$

And it is noted that **xz** denotes the bitwise inner product of x and z, $\mod 2$. In fact, we know that addition mod 2 is the same as XOR, thus the state after the final n-qubit H gate will be:

$$\begin{aligned} |\psi_3\rangle = \left(\frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(x)} \frac{1}{\sqrt{2^n}} \sum_{\mathbf{z}\in\{0,1\}^n} (-1)^{\mathbf{xz}} |\mathbf{z}\rangle \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = \\ = \frac{1}{2^n} \sum_{\mathbf{z}\in\{0,1\}^n} \left(\sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})+\mathbf{xz}} \right) |\mathbf{z}\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \end{aligned}$$

Finally, a measurement of the first register is made in the computational basis and for that, we consider the total amplitude of $|\mathbf{z}\rangle = |0\rangle^{\otimes n}$ in the first register of state $|\psi_3\rangle$. This is $\frac{1}{2^n} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})}$. Thus, we consider the amplitude in two cases: f is **constant** and f is **balanced**.

- If f **constant**, the amplitude of $|0\rangle$ is either $+1$ or -1. So, a measurement of the first register is certain to return all zeros (000...0).
- If f **balanced**, then the positive and negative contributions of the amplitudes cancel out and the overall amplitude of $|0\rangle$ is 0. So, a measurement of the first register is certain not to return all zeros.

Therefore, for the determination of f as **constant** or **balanced**, the first register is measured. If the result is all zeros, then the algorithm outputs "**constant**" and otherwise it outputs "**balanced**".

Note: It is again noted that **deterministic classical algorithms** would require $2^{n-1} + 1$ queries in the worst case. A **probabilistic classical algorithm** could solve the Deutsch-Jozsa problem with probability of error at most $\frac{1}{3}$ using 2 queries. This error can be reduced to less than $\frac{1}{2^n}$ with only $n+1$ queries. Thus, there is an **exponential gap** between deterministic classical and "exact" quantum query complexity.

However, the gap between **classical probabilistic query complexity** and the **quantum computational query complexity** is a **constant gap** in the case of constant error and can be amplified to a **linear gap** in the case of exponentially small error.

4.2.2 Further Analysis

Deutsch algorithm

For the **quantum** solution, we will analyze the following circuit step by step for deeper understanding:

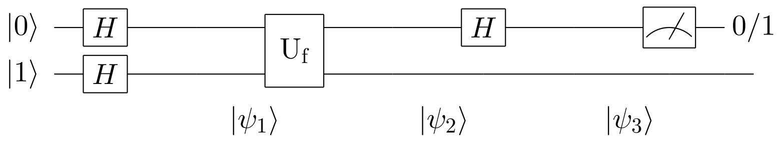


Figure 4.5: Deutsch circuit

First, let us assume that the initial state is $|1\rangle|0\rangle$ and after the application of the two H gates, the state will be: $|\psi_1\rangle = |-\rangle|+\rangle = \frac{1}{2}(|0\rangle - |1\rangle)|0\rangle + \frac{1}{2}(|0\rangle - |1\rangle)|1\rangle$. After that, we apply the U_f gate and the value of the function f for the classical state of the top/rightmost qubit is XORed onto the bottom/leftmost qubit, so: $|\psi_2\rangle = \frac{1}{2}(|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle)|0\rangle + \frac{1}{2}(|0 \oplus f(1)\rangle - |1 \oplus f(1)\rangle)|1\rangle$. And if we use the formula for $a \in \{0,1\}$: $|0 \oplus a\rangle - |1 \oplus a\rangle = (-1)^a(|0\rangle - |1\rangle)$. We get: $|0 \oplus 0\rangle - |1 \oplus 0\rangle = |0\rangle - |1\rangle = (-1)^0(|0\rangle - |1\rangle)$ and $|0 \oplus 1\rangle - |1 \oplus 1\rangle = |1\rangle - |0\rangle = (-1)^1(|0\rangle - |1\rangle)$. So, overall: $|\psi_2\rangle = \frac{1}{2}(-1)^{f(0)}(|0\rangle - |1\rangle)|0\rangle + \frac{1}{2}(-1)^{f(1)}(|0\rangle - |1\rangle)|1\rangle = |-\rangle\left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right)$. We observe that the state of the top/rightmost qubit has changed while the state of the bottom/leftmost qubit remains the same and it is $|-\rangle$ before and after the U_f and that phenomenon is called Phase Kick-back, as we analyzed earlier. So, we get the following: $|\psi_2\rangle = U_f(|-\rangle|+\rangle) = \frac{1}{\sqrt{2}}U_f(|-\rangle|0\rangle) + \frac{1}{\sqrt{2}}U_f(|-\rangle|1\rangle) = (-1)^{f(0)}|-\rangle\left(\frac{|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle}{}\right) = \begin{cases} (-1)^{f(0)}|-\rangle|+\rangle & \text{, if } f(0) \oplus f(1) = 0 \\ (-1)^{f(0)}|-\rangle|-\rangle & \text{, if } f(0) \oplus f(1) = 1 \end{cases}$ We use $f(0) \oplus f(1)$ in the exponent instead of $f(1) - f(0)$ because we can obtain the same value either way, since the value $(-1)^k, k \in \mathbb{Z}^*$ depends only on whether k is odd or even. Next, after the application of H gate to the top qubit we get:

$$|\psi_3\rangle = \begin{cases} (-1)^{f(0)}|-\rangle|0\rangle & \text{, if } f(0) \oplus f(1) = 0 \\ (-1)^{f(0)}|-\rangle|1\rangle & \text{, if } f(0) \oplus f(1) = 1 \end{cases}$$

Finally, we make a measurement and with probability 1 we get the correct outcome.

For the **classical** solution, we need to ask two queries, one to learn $f(1)$ and one for $f(0)$.

Deutsch-Jozsa algorithm

For the **quantum** solution, we will analyze the following circuit step by step for deeper understanding:

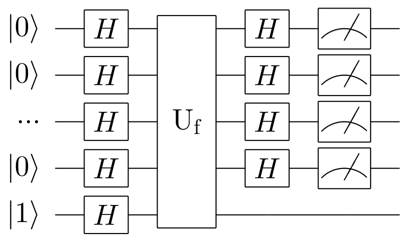


Figure 4.6: Deutsch-Jozsa circuit

First, we are reminded of the single qubit formula for the Hadamard gate: $H|a\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^a|1\rangle = \frac{1}{\sqrt{2}}\sum_{b\in\{0,1\}}(-1)^{ab}|b\rangle$. So, the n qubit formula will be the combined operation of H on n qubits and it is described by the tensor product $H \otimes H ... \otimes H = H^{\otimes n}$ and so, for the binary strings of length n $x_{n-1}...x_0$ and $y_{n-1}...y_0$ we get:

$$H^{\otimes n}|x_{n-1}...x_1x_0\rangle = (H|x_{n-1}\rangle) \otimes ... \otimes (H|x_0\rangle) =$$
$$= \left(\frac{1}{\sqrt{2}}\sum_{y_{n-1}\in\{0,1\}}(-1)^{x_{n-1}y_{n-1}}|y_{n-1}\rangle\right) \otimes ... \otimes \left(\frac{1}{\sqrt{2}}\sum_{y_0\in\{0,1\}}(-1)^{x_0y_0}|y_0\rangle\right) =$$
$$= \frac{1}{\sqrt{2^n}}\sum_{y_{n-1}...y_0\in\{0,1\}^n}(-1)^{x_{n-1}y_{n-1}+...+x_0y_0}|y_{n-1}...y_0\rangle$$

Thus, after the application of H, the state of $n+1$ qubits will be: $(H|1\rangle)(H^{\otimes n}|0...0\rangle) = |-\rangle \otimes \frac{1}{\sqrt{2^n}}\sum_{x_{n-1}...x_0\in\{0,1\}^n}|x_{n-1}...x_0\rangle$. Next, after the application of U_f, we get: $|-\rangle \otimes \frac{1}{\sqrt{2^n}}\sum_{x_{n-1}...x_0\in\{0,1\}^n}(-1)^{f(x_{n-1}...x_0)}|x_{n-1}...x_0\rangle$. And we can see the Phase Kick-back phenomenon. Then, after the application of the second layer of H, we get $|-\rangle\otimes\frac{1}{2^n}\sum_{x_{n-1}...x_0\in\{0,1\}^n}\sum_{y_{n-1}...y_0\in\{0,1\}^n}($ Finally, we need to know only the probability that every one of the measurement outcomes is 0 because only then f is constant. So,

$$p = \left|\frac{1}{2^n}\sum_{x_{n-1}...x_0\in\{0,1\}^n}(-1)^{f(x_{n-1}...x_0)}\right|^2 = \begin{cases} 1 & \text{, if } f \text{ constant} \\ 0 & \text{, if } f \text{ balanced} \end{cases}$$

- if f **constant**, then:
  - for $f(x_{n-1}...x_0) = 0$ for every string $x_{n-1}..x_0$ the sum is 2^n and thus gives 1.
  - for $f(x_{n-1}...x_0) = 1$ for every string $x_{n-1}..x_0$ the sum is -2^n and thus dividing by 2^n and taking the square of the absolute value gives 1.
- if f **balanced**, then, f is 0 on half of the strings $x_{n-1}..x_0$ and 1 on the rest, so $+1$ and -1 terms cancel out and thus gives 0.

Therefore, we get 100% the correct outcome when the promise is met, with just one query. In the case where f is neither constant nor balanced, then all bets are off and we cannot say too much.

For the **deterministic classical** solution, the algorithm must make exponentially many more queries, in particular $2^{n-1} + 1$ in the worst case. The reason is that if it queries f on 2^{n-1} different input strings and gets the same result every time, we are not sure yet if f is constant of balanced, that is why we need one more. For the **probabilistic classical** solution, the algorithm can solve the Deutsch-Jozsa problem with very high probability using just a few queries. The reason is that if we choose some inputs to f randomly and calculate f on those strings, it is very unlikely for the output to all be the same when f is balanced. In particular, if we choose k input strings $x^1, ..., x^k \in \{0,1\}^n$ uniformly at random, calculate $f(x^1), ..., f(x^k)$ and output 0 if they are all the same and 1 if not, then we will always be correct when f is constant and wrong in the case that f is balanced with probability 2^{-k+1} (e.g. for $k = 11$, we get the correct answer with $p = 99.9\%$).

Implementation:

For the development of this algorithm, there is a complete implementation of Deutsch algorithm in this **link** and another implementation of both Deutsch and Deutsch-Jozsa algorithm in this **link**.

Chapter 5

Quantum Algorithms: The Bernstein-Vazirani Algorithm

5.1 Introduction

In this chapter, we will examine the **Bernstein-Vazirani Problem** or, also known as the **Fourier Sampling Problem**.

5.2 Theoretical Analysis

- **The Problem**:

  To begin with, we will use the following notation for binary strings of length n, $x = x_{n-1}...x_0$, $y = y_{n-1}...y_0$. And the binary dot product as: $x \cdot y = x_{n-1}y_{n-1} \oplus ... \oplus x_0y_0 = \begin{cases} 1 & \text{, if } x_{n-1}y_{n-1} + ... + x_0y_0 \text{ is odd} \\ 0 & \text{, if } x_{n-1}y_{n-1} + ... + x_0y_0 \text{ is even} \end{cases}$. It is noted that the result does not change if we swap x and y because it is a symmetric operation and we can see that it equals the parity of those bits of x in positions where the string y has 1 and vice versa.

<u>**The Bernstein - Vazirani Problem**</u>:
Input: A function $f : \{0,1\}^n \to \{0,1\}$
Promise: There exists a binary string $s = s_{n-1}...s_0$, s.t. $f(x) = s \cdot x, \forall x \in \{0,1\}^n$
Output: Determination of string s by making queries to f.

  It is noted that in fact, we do not need a new algorithm to solve this problem because the Deutsch-Jozsa algorithm can solve it with some modifications.

- **The Solution**:

  - **Classical**:

    In order to solve the Bernstein - Vazirani problem, we have to make at least n queries using any classical query algorithm. Information-theoretic argument: We know that by making a classical query, we can only reveal a single bit of information about the solution and since there a n bits of information, we will need n queries. In fact, it is possible to solve this problem classically by querying the function on each of the n strings having a single 1, in each possible position, and 0 for all other bits, and reveal the bits of s one at a times.

– **Quantum**: First, we can describe the action of n Hadamard gates on the standard basis states of n qubits by using the following: $H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle$. Because, the value $(-1)^k$, $k \in \mathbb{Z}$ depends only on whether k is odd or even. After the application of the H gates, the state of the $n+1$ qubits will be: $|-\rangle \otimes \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle$. Next, we apply the query gate and gives us: $|-\rangle \otimes \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle$. Because of the Phase Kick-back phenomenon. Then, after we apply the second layer of Hadamard gates, the state will be: $|-\rangle \otimes \frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{f(x)+x\cdot y}|y\rangle$. Since we are promised that $f(x) = s\cdot x$ for some string $s = s_{n-1}...s_0$, we can write: $|-\rangle\otimes\frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{s\cdot x+x\cdot y}|y\rangle$. Also, since $s\cdot x$ and $x\cdot y$ are binary values, we can replace addition with XOR, since we are only interested whether the exponent of -1 is even or odd. So we get: $|-\rangle \otimes \frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{(s\cdot x)\oplus(y\cdot x)}|y\rangle$. Moreover, after an expansion of the binary dot product and bitwise XOR, we get: $(s\cdot x)\oplus(y\cdot x) = (s_{n-1}x_{n-1})\oplus...(s_0x_0)\oplus(y_{n-1}x_{n-1})\oplus...\oplus(y_0x_0) = (s_{n-1}\oplus y_{n-1})x_{n-1}\oplus(s_0\oplus y_0)x_0 = (s\oplus y)\cdot x$. So, we can write the state before the measurement like this: $|-\rangle \otimes \frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{(s\oplus y)\cdot x}|y\rangle$. Also, we can use the fact that $\forall z = z_{n-1}...z_0$ it is true that: $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{z\cdot x} = \begin{cases} 1 & \text{, if } z = 0^n \\ 0 & \text{, if } z \neq 0^n \end{cases}$ where 0^n is the all-zero string of length n. This formula works, because:

  * if $z = 0^n$, then $z\cdot x = 0, \forall x \in \{0,1\}^n$, so the value of each term in the sum is 1 and thus we get 1 by summing and dividing by 2^n
  * if $z \neq 0^n$ (meaning that any one of the bits of z is equal to 1), then $z\cdot x = 0$ for exactly half of the possible choices for $x \in \{0,1\}^n$ and $z\cdot x = 1$ for the other half, because the value of the $z\cdot x$ flips from 1 to 0 and from 0 to 1 if we flip the bit of x in any position where z has a 1.

  So, if we use this formula to simplify the state before measurement, we get $|-\rangle \otimes \frac{1}{2^n}\sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}^n}(-1)^{(s\oplus y)\cdot x}|y\rangle = |-\rangle \otimes |s\rangle$ because $s\oplus y = 0^n$ iff, $y = s$. Finally, we make the measurement and we get exactly the string s.

  Therefore, we can solve the Bernstein - Vazirani problem with a single query.

Implementation:

For the development of this algorithm, there is a simple implementation in this **link** and a more complete one in this **link**

A more complicated problem is known as the recursive **Fourier sampling problem** where solutions to different instances of the problem unlock new levels of the problem arranged in a tree-like structure and the Bernstein-Vazirani problem is just a special case of it. It is one of the query problems where we can talk about the so-called super-polynomial advantage of quantum algorithms over the probabilistic ones. To give an intuition to the problem, the recursive version effectively amplifies the 1 versus n advantage of quantum algorithms to an even larger class.

Chapter 6

Quantum Algorithms: Simon's Algorithm

6.1 Introduction

The case of **Simon's problem** that we will examine in this section provides a simple example of this super-polynomial (actually exponential) advantage of quantum over classical algorithms.

6.2 Theoretical Analysis

- **The Problem**:

  Let us consider a function $f : \{0,1\}^n \to X$, where the $\{0,1\}^n$ is the vector space Z_2^n over Z_2 (we can treat as additive group), which consists of n-tuples of zeros and ones and has dimension n and X is some finite set $X \subseteq \{0,1\}^n$. Also, we are promised that there is a hidden string $\mathbf{s} = s_1 s_2 ... s_n$ so that $f(x) = f(y)$, if and only if, $x = y$ or $x = y \oplus s$.

  <u>The Simon's Problem</u>:
  Input: An oracle for computing a function $f : \{0,1\}^n \to X$, X being a finite set.
  Promise: There exists a string $s = s_1 s_2 ... s_n$, s.t. $f(x) = f(y)$, iff, $x = y$ or $x = y \oplus s$.
  Problem: Determination of string s by making queries to f.

  We saw at the analysis of Deutsch-Jozsa algorithm that the n-qubit Hadamard transformation is: $H^{\otimes n} |\mathbf{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{z} \in \{0,1\}^n} (-1)^{\mathbf{xz}} |\mathbf{z}\rangle$. So, if we apply the $H^{\otimes n}$ to a superposition of two basis states $|0\rangle + |s\rangle$, then $H^{\otimes n} \left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |\mathbf{s}\rangle \right) = \frac{1}{\sqrt{2^{n+1}}} \sum_{\mathbf{z} \in \{0,1\}^n} |\mathbf{z}\rangle + \frac{1}{\sqrt{2^{n+1}}} \sum_{\mathbf{z} \in \{0,1\}^n} (-1)^{\mathbf{sz}} |\mathbf{z}\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{\mathbf{z} \in \{0,1\}^n} (1 + (-1)^{\mathbf{sz}}) |\mathbf{z}\rangle$. Thus, if $\mathbf{sz} = 1$ we get $(1 + (-1)^{\mathbf{sz}}) = 0$ and the basis $|\mathbf{z}\rangle$ disappears, else, it remains with amplitude $\frac{1}{\sqrt{2^{n-1}}}$. Now, if we define $\mathbf{s}^{\perp} = \{\mathbf{z} \in \{0,1\}^n | \mathbf{sz} = 0\}$, which is a vector subspace of Z_2^n that is orthogonal to the subspace $S = \{\mathbf{0,s}\}$, the orthogonal complement of S', denoted $S^{\perp}$, we get that $dim(S) + dim(S^{\perp}) = dim(Z_2^n) = n$ and hence, $\mathbf{s}^{\perp}$ has dimension $n-1$, so $H^{\otimes n} \left(\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |\mathbf{s}\rangle \right) = \frac{1}{\sqrt{2^{n-1}}} \sum_{\mathbf{z} \in \{\mathbf{s}\}^{\perp}} |\mathbf{z}\rangle$. Consequently, in the case that we want to map $\frac{1}{\sqrt{2}} |\mathbf{x}\rangle + \frac{1}{\sqrt{2}} |\mathbf{x} \oplus \mathbf{s}\rangle$ to a uniform superposition of states $\mathbf{z} \in \{\mathbf{s}\}^{\perp}$ we can use the following formula for $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ and $s = x \oplus y$: $H^{\otimes n} \left(\frac{1}{\sqrt{2}} |\mathbf{x}\rangle + \frac{1}{\sqrt{2}} |\mathbf{y}\rangle \right) = \frac{1}{\sqrt{2^{n-1}}} \sum_{\mathbf{z} \in \{\mathbf{s}\}^{\perp}} (-1)^{\mathbf{xz}} |\mathbf{z}\rangle$.

- **The Solution**:
  - **Classical**:

    This problem requires an exponential number of queries f. In fact, any classical algorithm that solves the Simon's problem with probability at least $\frac{2}{3}$ for any such f, must evaluate f a number of times in $\Omega(2^{n/3})$.
  - **Quantum**:

    Let us suppose that we have a reversible oracle for implementing f: $U_f : |\mathbf{x}\rangle |\mathbf{b}\rangle \mapsto |\mathbf{x}\rangle |\mathbf{b} \oplus f(x)\rangle$. The circuit for the quantum part of Simon's algorithm, where the values of the measured bit correspond to a string $\mathbf{w}_i$ from $\mathbf{s}^{\perp}$.

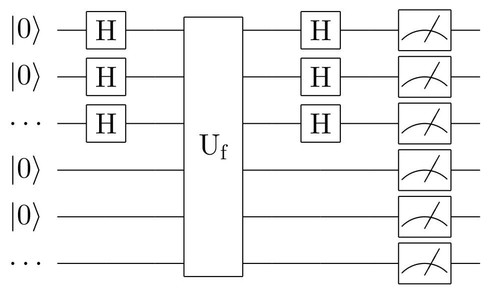


Figure 6.1: Simon's algorithm

First, set a counter $i = 1$. Prepare $\frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle |0\rangle$. Apply U_f to produce the state $\sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle |f(\mathbf{x})\rangle = \frac{1}{\sqrt{2^{n-1}}} \sum_{\mathbf{x}\in I} \frac{1}{\sqrt{2}}(|\mathbf{x}\rangle + |\mathbf{x} \oplus \mathbf{s}\rangle) |f(\mathbf{x})\rangle$ where, $\{0,1\}^n$ can be partitioned into 2^{n-1} pairs of string of $\{\mathbf{x}, \mathbf{x} \oplus \mathbf{s}\}$ and I is a subset of $\{0,1\}^n$ consisting of one representative from each pair. (Optional) Measure the second register to obtain $f(\mathbf{x})$ and thus, the first register is left in superposition $\frac{1}{\sqrt{2}}(|\mathbf{x}\rangle + |\mathbf{x} \oplus \mathbf{s}\rangle)$. Apply $H^{\otimes n}$ to the first register after that the first register will be in an equally weighted superposition of elements of $\mathbf{s}^{\perp}$. Measure the first register and record the values $\mathbf{w}_i$ (they are elements of $\mathbf{s}^{\perp}$) selected uniformly at random. If the dimension of the span of $\{\mathbf{w}_i\}$ is $n-1$, then $\mathbf{span}\{\mathbf{w}_i\} = \mathbf{s}^{\perp}$ and thus, go to the next step, else increment i and go to the second step. Solve the linear equation $\mathbf{W}\mathbf{s}^T = 0^T$ and let $\mathbf{s}$ be the unique non-zero solution (9 and $\mathbf{s}$ are the only solutions and they can be found by Gaussian elimination modulo 2 in time polynomial in n). The expected number of samples from $\mathbf{s}^{\perp}$ before the algorithm stops is less than $m + 1 = n$. Finally, output $\mathbf{s}$.

The above algorithm finds the hidden string $\mathbf{s}$ in Simon's Problem. The expected number of evaluations of f in the execution of the algorithm is less than n, and the expected number of other elementary gates is in $O(n^3)$. However, there is another way to have a polynomial worst-case running time as long as we accept a small probability of not getting an answer. This is true because it possible to generically convert an algorithm with expected running time T into one with a definite running time in $O(T)$ and with a bounded probability of successfully getting an output and such algorithms are called "**zero-error**" algorithms (when they provide an answer, it is always correct).

Using Markov's inequality, i.e., let X be a discrete random variable that takes on non-negative values, then $Pr(X \geq c\mu_X) \leq \frac{1}{c}$ where $\mu_X = E(X) = \sum_{x \in S} xPr(X = x)$ and S is a finite or countable subset of the real numbers, it occurs that any algorithm that terminates with an expected number of queries equal to T will terminate after at most $3T$ queries, with probability at least $\frac{2}{3}$. Thus, if we abandon Simon's algorithm if it has not stopped after $3n$ queries, then with probability at least $\frac{2}{3}$ the algorithm will successfully solve Simon's problem.

However, for any particular algorithm, it might be possible to do better than what Markov's inequality provides. Here for example, $n + 3$ uniformly random samples from $\mathbf{s}^{\perp}$ will generate $\mathbf{s}^{\perp}$ with probability at least $\frac{2}{3}$. As a result we can have a zero-error version of Simon's algorithm that has a bounded running time. So, the Simon's problem can be generalized as:

The Simon's Problem Generalization:
Input: An oracle U_f implementing some $f : \{0,1\}^n \to X$, X being a finite set.
Promise: $f(\mathbf{x}) = f(\mathbf{y})$, iff, $\mathbf{x} - \mathbf{y} \in S$ for some subspace $S \leq Z_2^n$.
Problem: Find a basis $\mathbf{s}_1, ..., \mathbf{s}_m$ for S, where m is the dimension of subspace S.

If $S = \{0, \mathbf{x}_1, ..., \mathbf{x}_{2^m-1}\}$ is an m-dimensional subspace of $Z_2^n = \{0,1\}^n$ over Z_2, then the set $\{0,1\}^n$ can be partitioned into 2^{n-m} subsets of the form $\{\mathbf{y}, \mathbf{y} \oplus \mathbf{x}_1, \mathbf{y} \oplus \mathbf{x}_2, ..., \mathbf{y} \oplus \mathbf{x}_{2^m-1}\}$ (also denoted as $\mathbf{y} + S$) and I is a subset of $\{0,1\}^n$ consisting of one representative from each of the disjoint subsets.

In the case where we do not know m, we still know that $m+4$ samples are enough in order to generate $S^{\perp}$ with probability at least $\frac{2}{3}$ and hence, $n+4$ samples are certainly adequate. Thus, we can run the zero-error algorithm for the generalized Simon's problem. So, first, set a counter $i = 1$. Prepare $\frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in \{0,1\}^n} |\mathbf{x}\rangle |0\rangle$. Apply U_f, to produce the state $\sum_{\mathbf{x} \in \{0,1\}^n} |\mathbf{x}\rangle |f(\mathbf{x})\rangle = \frac{1}{\sqrt{2^{n-m}}} \sum_{\mathbf{y} \in I} |\mathbf{y} + S\rangle |f(\mathbf{y})\rangle$ where $|\mathbf{y} + S\rangle = \sum_{\mathbf{s} \in S} \frac{1}{\sqrt{2^m}} |\mathbf{s}\rangle$. (Optional) Measure the second register. Thus, the first register is left in a state of the form $|\mathbf{y} + S\rangle$ for a random $\mathbf{y}$. Apply $H^{\otimes n}$ to the first register. After that, the first register contains a uniform superposition of elements of $S^{\perp}$. Measure the first register and record the value $\mathbf{w}_i$ which is sampled uniformly at random from $S^{\perp}$. If $i = n+4$, then go to the next step, else increment i and go to the second step. Solve the linear equation $\mathbf{W}\mathbf{s}^T = 0^T$ and let $\mathbf{s}_1, \mathbf{s}_2, ...$ be the generators of the solution space. Evaluate $f(\mathbf{0}), f(\mathbf{s}_1), f(\mathbf{s}_2), ...$ and if the outputs all equal $f(\mathbf{0})$, then output $\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_m$, otherwise output "Fail".

6.3 Further Analysis

Simon's algorithm is a quantum query algorithm for the **Simon's Problem** and it provides an exponential advantage of quantum over classical and probabilistic algorithms.

- **The Problem**:

  Let us assume that we have a function $f : \{0,1\}^n \to \{0,1\}^m$, where $n, m \in \mathbb{Z}^*$.

  Then, we are promised that there is a string $s \in \{0,1\}^n$ such that $[f(x) = f(y)] \Leftrightarrow$

$[(x = y) \vee (x \oplus s = y)] \forall x, y \in \{0,1\}^n$. Thus, there is always a unique correct answer to the problem for functions that satisfy the promise. So, we have to examine two cases:

- The first case is that s is the all-zero string 0^n, so $s = 0^n$ and so, we can simplify the "if and only if" as follows: $[f(x) = f(y)] \Leftrightarrow [x = y]$ because it must be true $\forall x, y \in \{0,1\}^n$ and we can see that only when f is a one-to-one function, the promise is met.
- The second case is that s is not the all-zero string 0^n, so $s \neq 0^n$ and so, the promise implies that for every possible output string of f, there are exactly two input strings that cause f to output the string and they must take the form x and $x \oplus s$ for some string x, so it is implied that f is two-to-one (e.g. $f : \{0,1\}^3 \to \{0,1\}^5$ where there are 8 different inputs and 4 different outputs, each of which occurs twice, and the promise for the string $s = 011$ is met since for any two different inputs that produce the same output, the bitwise XOR is equal to s).

<u>Simon's Problem</u>:
Input: A function $f : \{0,1\}^n \to \{0,1\}^m$
Promise: There is a string $s \in \{0,1\}^n$ s.t. $[f(x) = f(y)] \Leftrightarrow [(x = y) \vee (x \oplus s = y)] \forall x, y \in \{0,1\}^n$
Output: The string s

- **The Solution**:
  - **Classical**:
    In a classical setting we need a lot queries to solve Simon's problem. In particular, if we have a probabilistic query algorithm A and A makes less than $2^{\frac{n}{2}-1} - 1$ queries (number exponential in n), then A will fail to solve Simon's problem with probability at least $p = \frac{1}{2}$. In order to prove this, we will provide the intuition. So, we try to find the hidden string s, On the one hand, so long as we do not query the function on two strings having the same output value, we will get very limited information about s. In fact, all we can learn is that s is not the XOR of any two distinct strings we have queried. On the other hand, if we query less than $2^{\frac{n}{2}-1} - 1$ strings, there will still be a lot of choices of s that we have not ruled out.
  - **Quantum**:
    As we saw in the theoretical analysis, the first n qubits are acted on by H gates and the rest m qubits go directly to the query gate. So, after the first layer of H gates applied to the first n qubits, the state will be: $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |0^m\rangle |x\rangle$. Next, the U_f is applied and output of f is XORed onto the all-zero state of the bottom m qubits. So, the $n+m$ qubits will be in the state: $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |f(x)\rangle |x\rangle$. After that, the second layer of H gates is applied and we get: $\frac{1}{2^n} \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}^n} (-1)^{y \cdot x} |f(x)\rangle |y\rangle$. Then, for the measurement, we are interested in the probability of those that result in each possible string $y \in \{0,1\}^n$ and that is: $p(y) = \left\| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{y \cdot x} |f(x)\rangle \right\|^2$. And, with the following notation:
    * the range of function f is the set containing all of its outputs: $range(f) = \{f(x) : x \in \{0,1\}^n\}$.

∗ for each string $z \in range(f)$, we express the set of all input strings that cause the function to evaluate to this output z as $f^{-1}(\{z\})$: $f^{-1}(\{z\}) = \{x \in \{0,1\}^n : f(x) = z\}$. It is noted that this is not the inverse of f, since f does not have to be invertible. Also, $\{z\}$ is a set and $f^{-1}(\{z\})$ is known as the pre-image of $\{z\}$ under f and it is the set of all elements that f maps to that set.

We can split up the sum as follows:

$$p(y) = \left\| \frac{1}{2^n} \sum_{z \in range(f)} \left(\sum_{x \in f^{-1}(\{z\})} (-1)^{y \cdot x} \right) |z\rangle \right\|^2$$

Hence, every string $x \in \{0,1\}^n$ is represented exactly once by the two summations (it is like splitting these strings into separate buckets depending on which output string $z = f(x)$ they produce when we evaluate f and then we sum separately over all buckets). Finally, we can compute the Euclidean norm squared to obtain:

$$p(y) = \frac{1}{2^{2n}} \sum_{z \in range(f)} \left| \sum_{x \in f^{-1}(\{z\})} (-1)^{y \cdot x} \right|^2$$

To compute this, first we will look at the value $\left| \sum_{x \in f^{-1}(\{z\})} (-1)^{y \cdot x} \right|^2$ for any arbitrary selection of $z \in range(f)$:

∗ In the first case, when $s = 0^n$, f is one-to-one and thus there is always just a single element of $x \in f^{-1}(\{z\})$, for every $z \in range(f)$, so the value is equal to 1.

∗ In the second case, when $s \neq 0^n$, f is two-to-one and thus there are exactly two strings in the set $f^{-1}(\{z\})$. In particular, if w is one of these two strings, then the other must be $w \oplus s$ according to the promise.

$$\left| \sum_{x \in f^{-1}(\{z\})} (-1)^{y \cdot x} \right|^2 = \left| (-1)^{y \cdot w} + (-1)^{y \cdot (w \oplus s)} \right|^2 = \left| (-1)^{y \cdot w} \left(1 + (-1)^{y \cdot s}\right) \right|^2 =$$

$$= |1 + (-1)^{y \cdot s}|^2 = \begin{cases} 4 & \text{, if } y \cdot s = 0 \\ 0 & \text{, if } y \cdot s = 1 \end{cases}$$

So, this value is independent of the specific choice of $z \in range(f)$ in both cases. Finally, to get the overall probability we will see again what happens in each case:

∗ In the first case, $s = 0^n$, f is one-to-one, so there are 2^n strings $z \in range(f)$, so we get: $p(y) = \frac{1}{2^{2n}} \cdot 2^n = \frac{1}{2^n}$ which means that the measurement results in a string $y \in \{0,1\}^n$ chosen uniformly at random.

∗ In the second case, $s \neq 0^n$, f is two-to-one, so there are 2^{n-1} strings $z \in range(f)$, so we get:

$$p(y) = \frac{1}{2^{2n}} \sum_{z \in range(f)} \left| \sum_{x \in f^{-1}(\{z\})} (-1)^{y \cdot x} \right|^2 = \begin{cases} \frac{1}{2^{n-1}} & \text{, if } y \cdot s = 0 \\ 0 & \text{, if } y \cdot s = 1 \end{cases}$$

which means that the measurement results in a string $y \in \{0,1\}^n$ chosen uniformly at random from $\{y \in \{0,1\}^n : y \cdot s = 0\}$, which contains $\frac{2^n}{2} = 2^{n-1}$ stings (half of the binary strings of length n have binary dot product 1 with s and the other half have 0).

Thus, we calculated the probabilities for the possible measurement outcomes when we run the quantum circuit for Simon's algorithm and that information is enough to determine s but we should repeat the process several times and also there is a probability that it could fail (which is almost eliminated if we run it enough times). If we run the circuit k times, where e.g. $k = n + 10$. Then we obtain strings $y^1, ..., y^k \in \{0,1\}^n$ (these superscripts are part of the names of these strings), so $y^1 = y^1_{n-1}...y^1_0 \quad ... \quad y^k = y^k_{n-1}...y^k_0$ and by inserting the bits of these string in a matrix M, we get: $M = \begin{bmatrix} y^1_{n-1} & \cdots & y^1_0 \\ y^2_{n-1} & \cdots & y^2_0 \\ \vdots & \ddots & \vdots \\ y^k_{n-1} & \cdots & y^k_0 \end{bmatrix}$.

Next, even though we do not know s yet, if we try to imagine that we know it and we form a vector v as follows: $v = \begin{bmatrix} s_{n-1} \\ \vdots \\ s_0 \end{bmatrix}$. We can perform the multiplication Mv mod 2 and get: $Mv = \begin{bmatrix} y^1 \cdot s \\ y^2 \cdot s \\ \vdots \\ y^k \cdot s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$. Hence, the string s will always be an element of the null space of the matrix M, provided that we do the arithmetic mod 2, regardless of whether $s = 0^n$ or $s \neq 0^n$. Next, by using Gaussian elimination for linear algebra, we can efficiently compute a description of the null space of M. Also, it is noted that it is very unlikely when $k = n + 10$ that there will be any other vectors in the null space of M besides the one corresponding to 0^n and s. In particular, in both cases, with probability greater than $1 - 2^{-10}$ the vectors in the null space of M will correspond exactly to the set $\{0^n, s\}$ and thus, we have a chance greater than 99.9% to determine s from the null space of M. In general, for $k = n + r$, for any $r \in \mathbb{Z}^*$, the probability of success will be at least $1 - 2^{-r}$, thus the probability of failure can be extremely small (e.g. for $k = 2n$, the probability of failure gets exponentially small in n). Therefore, Simon's algorithm solves Simon's problem with a number of queries that is linear in the number of input bits n of our function, while any classical or probabilistic algorithm need to make a number of queries that is exponential in n in order to solve the problem with a reasonable probability of success.

Implementation:

For the development of this algorithm there is a simple implementation in this and a more complete one in this **link**.

The technique used in this algorithm inspired Peter Shor to discover his quantum algorithm for factoring integers.

Chapter 7

Quantum Algorithms: Shor's Algorithm

7.1 Introduction

In this section, we have analyzed the Shor's Algorithm which solves the Problem of Integer Factorization. This procedure is used to break the RSA Protocol as we will see in this chapter.

7.2 Theoretical Analysis

7.2.1 Phase Estimation

Definition 7.2.1 (Normal matrices)**.** *A $n \times n$ matrix M with complex number entries is a normal matrix if it commutes with is conjugate transpose $MM^\dagger = M^\dagger M$.*

Examples:

- If U is a unitary matrix, then it is also normal because $UU^\dagger = \mathbb{I} = U^\dagger U$.
- If H is a hermitian matrix (a matrix that is equal to its own conjugate transpose), then it is also normal because $HH^\dagger = H^2 = H^\dagger H$.
- But, e.g. if $M = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, then $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}^\dagger = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}^\dagger \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}^\dagger \neq \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}^\dagger \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

  So, not every square matrix is normal.

Then, there is the **Spectral theorem** which is used for simpler expression of *normal matrices* and says that if M is a normal $N \times N$ complex matrix, then there exists an orthonormal basis of N dimensional complex vectors $\{|\psi_1\rangle, ..., |\psi_N\rangle\}$ along with complex numbers $\lambda_1, ..., \lambda_N$ such that the spectral decomposition is: $M = \lambda_1 |\psi_1\rangle \langle\psi_1| + ... + \lambda_N |\psi_N\rangle \langle\psi_N| = \sum_{k=1}^{N} \lambda_k |\psi_k\rangle \langle\psi_k|$.

It is noted that if M is a normal matrix expressed in a spectral decomposition, then it is true for every $j = 1, ..., N$: $M |\psi_j\rangle = \left(\sum_{k=1}^{N} \lambda_k |\psi_k\rangle \langle\psi_k|\right) |\psi_j\rangle = \sum_{k=1}^{N} \lambda_k |\psi_k\rangle \langle\psi_k|\psi_j\rangle = \lambda_j |\psi_j\rangle$. So, each number λ_j is an eigenvalue of M and $|\psi_j\rangle$ is an eigenvector.

Examples:

- Let $\mathbb{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, which is normal. Then, $\mathbb{I}$ can be written as a spectral decomposition for $\lambda_1 = 1, \lambda_2 = 1, |\psi_1\rangle = |0\rangle, |\psi_2\rangle = |1\rangle$, so $\mathbb{I} = |0\rangle\langle 0| + |1\rangle\langle 1|$. We can even choose another orthonormal basis and in that case we will have $\mathbb{I} = |+\rangle\langle +| + |-\rangle\langle -|$.
- Let $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ be Hadamard operation and also a unitary normal matrix. Then, H can be written as a spectral decomposition, using $|\psi_\theta\rangle = \cos(\theta)|0\rangle + \sin(\theta)|1\rangle$ as $H = |\psi_{\frac{\pi}{8}}\rangle\langle\psi_{\frac{\pi}{8}}| - |\psi_{\frac{5\pi}{8}}\rangle\langle\psi_{\frac{5\pi}{8}}|$ where $|\psi_{\frac{\pi}{8}}\rangle = \frac{\sqrt{2+\sqrt{2}}}{2}|0\rangle + \frac{\sqrt{2-\sqrt{2}}}{2}|1\rangle \quad |\psi_{\frac{5\pi}{8}}\rangle = -\frac{\sqrt{2-\sqrt{2}}}{2}|0\rangle + \frac{\sqrt{2+\sqrt{2}}}{2}|1\rangle$.

It is noted that we have some freedom in how eigenvectors are selected but there is no freedom at all in how the eigenvalues are chosen, except for their ordering. So, for any matrix M, the same n complex numbers $\lambda_1, ..., \lambda_N$ will always occur in the decomposition. Moreover, for unitary matrices, if we have a complex number λ and a non-zero vector $|\psi\rangle$ ($\||\psi\rangle\| \neq 0$) that satisfies the equation $U|\psi\rangle = \lambda|\psi\rangle$, then λ is an eigenvalue of U and $|\psi\rangle$ is an eigenvector. These matrices preserve the Euclidean norm, so $\||\psi\rangle\| = \|U|\psi\rangle\| = \|\lambda|\psi\rangle\| = |\lambda|\||\psi\rangle\| \rightarrow |\lambda| = 1$. Thus, eigenvalues of unitary matrices must always have absolute value equal to one, so they must lie on the unit circle $\mathbb{T}$ or S^1: $\mathbb{T} = \{\alpha \in \mathbb{C} : |\alpha| = 1\}$.

The Problem: Now, we are ready to state the problem of Phase Estimation. In this problem, we are given a state $|\psi\rangle$ of n qubits along with a unitary quantum circuit that acts on n qubits and we are promised that $|\psi\rangle$ is an eigenvector of the unitary matrix U that describes the action of the circuit, and lastly, the goal is to either identify or approximate the eigenvalue $\lambda = e^{2\pi i\theta}, 0 \leq \theta < 1$ to which $|\psi\rangle$ corresponds, or more precisely this real number θ.

Phase Estimation Problem:
Input: An n qubit quantum state $|\psi\rangle$ and a unitary quantum circuit for an an n-qubit operation U.
Promise: $|\psi\rangle$ is an eigenvector of U.
Output: The computation or approximation of the real number $\theta \in [0, 1)$ satisfying $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$

It is noted that this problem differs from others because it includes a quantum state as an input and it is a subproblem that will appear later in the context of integer factorization. We will formulate more precise problem statements later in order to specify the approximation of θ. When we go from $\theta = 0$ to $\theta = 1$, we are going all the way around the unit circle, starting from $e^{2\pi i0} = 1$ and moving counter-clockwise, we are moving towards to $e^{2\pi i1} = 1$, which is back where we started. Thus, choices near $\theta = 1$, should be considered as being near to $\theta = 0$ (e.g. $\theta = 0.999$ is within $1/1000$ of $\theta = 0$).

7.2.2 Quantum Fourier Transform

It is a unitary operation that can be defined for any positive integer dimension N. The matrices that describe this operation are derived from an analogous operation on N dimensional vectors known as the **Discrete Fourier Transform** (DFT). We can think of the DFT as being given an N dimensional vector of complex numbers (use binary notation to encode the real and imaginary parts) and the goal is to calculate the result which is an N dimensional vector. An efficient algorithm that accomplishes that is the **Fast Fourier Transform** but we want to consider the case where this transform is a unitary operation that can be transformed on a quantum system.

Definition 7.2.2 (Quantum Fourier Transform)**.** *The N-dimensional Quantum Fourier Transform, which is described by an $N \times N$ matrix whose rows and columns are associated with the standard basis states $|0\rangle, ..., |N-1\rangle$ is $QFT_N = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \omega_N^{xy} |x\rangle \langle y|$ where ω_N is a complex number and $\omega_N = e^{\frac{2\pi i}{N}} = \cos\left(\frac{2\pi}{N}\right) + i \sin\left(\frac{2\pi}{N}\right)$.*

In fact, ω_N is the number on the complex unit circle that we obtain if we start at 1 and move counter-clockwise by an angle of $\frac{2\pi}{N}$ and for example, $\omega_1 = 1, \omega_2 = -1, \omega_3 = -\frac{1}{2} + \frac{\sqrt{3}}{2} i, \omega_4 = i, ...$ So, for the case of Phase Estimation that we are studying, we are going to focus on $N = 2^m$. What is more, we can write QFT as a matrix as:

$$QFT_1 = \begin{bmatrix} 1 \end{bmatrix} \quad QFT_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = H$$

$$QFT_3 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \frac{-1+i\sqrt{3}}{2} & \frac{-1-i\sqrt{3}}{2} \\ 1 & \frac{-1-i\sqrt{3}}{2} & \frac{-1+i\sqrt{3}}{2} \end{bmatrix} \quad QFT_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

In order to check for **unitarity** of QFT_N, we have to show that the columns form an orthonormal basis. Thus, if we define a column number y, then $|\phi_y\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \omega_N^{xy} |x\rangle$. And if we take the inner product between any two columns, we get: $\langle \phi_Z | \phi_y \rangle = \frac{1}{N} \sum_{x=0}^{N-1} \omega_N^{x(y-z)}$. This way, we are summing a bunch of points that are evenly spread around the unit circle, so that they cancel each other out and leave 0 when we sum them.

In addition, it is true that for a geometric series that:

$$1 + \alpha + \alpha^2 + ... + \alpha^{N-1} = \begin{cases} \frac{\alpha^N - 1}{\alpha - 1} & \text{, if } \alpha \neq 1 \\ N & \text{, if } \alpha = 1 \end{cases}$$

If we take $\alpha = \omega_N^{y-z}$, then:

- When $y = z$, then $\alpha = 1$, so $\langle \phi_y | \phi_y \rangle = 1$.
- When $y \neq z$, then $\alpha \neq 1$, so $\langle \phi_z | \phi_y \rangle = \frac{1}{N} \frac{\omega_N^{N(y-z)} - 1}{\omega_N^{y-z} - 1} = \frac{1}{N} \frac{1-1}{\omega_N^{y-z} - 1} = 0$ because, $\omega_N^N = e^{2\pi i} = 1$, so $\omega_N^{N(y-z)} = 1^{y-z} = 1$.

Thus, we have an orthonormal set $\{|\phi_0\rangle, ..., |\phi_{N-1}\rangle\}$: $\langle\phi_z|\phi_y\rangle = \begin{cases} 0 & \text{, if } y = z \\ 1 & \text{, if } y \neq z \end{cases}$ and that reveals that QFT_N is **unitary**.

Next, we need to make a **controlled-phase gate**. For that, we have to recall that a phase operation is a single-qubit quantum operation that is: $P_\alpha = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{bmatrix}$. For any real number α. So, a controlled version of this gate is the following: $CP_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\alpha} \end{bmatrix}$. And, it is noted that it does not matter which qubit is the control and which is the target because the two possibilities are equivalent. In fact, we can use these gates to transform a bit α and a number encoded in binary notation as a string of $m-1$ bits $y \in \{0, ..., 2^{m-1}-1\}$ as $|y\rangle|\alpha\rangle \mapsto \omega_{2^m}^{ay}|y\rangle|\alpha\rangle$. For instance, for $m = 5$, we can write (from left to right $\frac{\pi}{16}, \frac{\pi}{8}, \frac{\pi}{4}, \frac{\pi}{2}$):

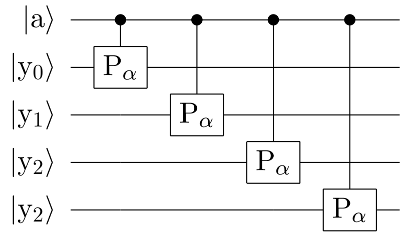


Figure 7.1: Controlled-phase gate for $m = 5$

The phase gates start from $\frac{\pi}{2}$ on the most significant bit of x down to $\frac{\pi}{2^{m-1}}$ on the least significant bit.

Now, we have the tools to implement our circuit for the QFT. The implementation is recursive in nature, and so that is how it is most naturally described. The base case is that the QFT on a single qubit is a Hadamard operation. So, for m qubits, $m \geq 2$, we can follow the steps below that we will describe for standard basis states of the form $|x\rangle|a\rangle$, where $x \in \{0, ..., 2^{m-1}-1\}$ is an integer encoded as $m-1$ bits using binary notation and a is a single bit.

First, we apply the 2^{m-1}-dimensional QFT to the bottom/leftmost $m-1$ qubits to get the state: $(QFT_{2^{m-1}}|x\rangle)|a\rangle = \frac{1}{\sqrt{2^{m-1}}}\sum_{y=0}^{2^{m-1}-1}\omega_{2^{m-1}}^{xy}|y\rangle|a\rangle$. This is achieved by recursively applying the method we described for one less qubit, using H operation on a single qubit as the base case. Next, we use the top/rightmost qubit as a control to inject the phase $\omega_{2^m}^{y}$ for each standard basis state $|y\rangle$ of the remaining $m-1$ qubits and get the state: $\frac{1}{\sqrt{2^{m-1}}}\sum_{y=0}^{2^{m-1}-1}\omega_{2^{m-1}}^{xy}\omega_{2^m}^{ay}|y\rangle|a\rangle$. Then, we apply the H gate on the top/rightmost qubit to get the state: $\frac{1}{\sqrt{2^m}}\sum_{y=0}^{2^{m-1}-1}\sum_{b=0}^{1}(-1)^{ab}\omega_{2^{m-1}}^{xy}\omega_{2^m}^{ay}|y\rangle|b\rangle$. Finally, we permute the order of the qubits so that the LSB becomes the MSB, with all others shifted: $\frac{1}{\sqrt{2^m}}\sum_{y=0}^{2^{m-1}-1}\sum_{b=0}^{1}(-1)^{ab}\omega_{2^{m-1}}^{xy}\omega_{2^m}^{ay}|b\rangle|y\rangle$.

And, to visualize it, here is the quantum circuit for $N = 2^5 = 32$:

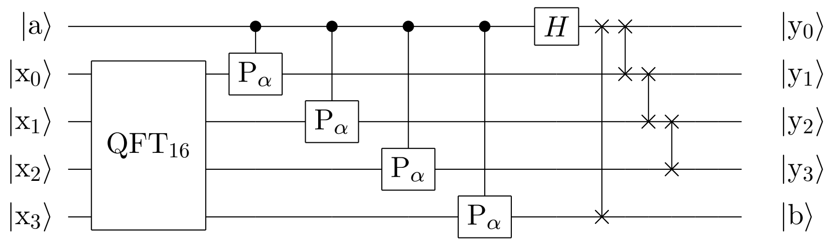


Figure 7.2: 2^5-dimensional QFT

Now, we will verify that the circuit actually implements the 2^m-dimensional QFT.

First, we can see that if we choose $a, b \in \{0, 1\}$ and $x, y \in \{0, ..., 2^{m-1} - 1\}$, and also, if we use $\omega_{2^m}^{2^m xb} = \left(\omega_{2^m}^{2^m}\right)^{xb} = 1^{xb} = 1$. Then, $\omega_{2^m}^{(2x+a)(2^{m-1}b+y)} = \omega_{2^m}^{2^m xb}\omega_{2^m}^{2xy}\omega_{2^m}^{2^{m-1}ab}\omega_{2^m}^{ay} = (-1)^{ab}\omega_{2^{m-1}}^{xy}\omega_{2^m}^{ay}$. Next, if we write $u = 2x+a, v = 2^{m-1}b+y$ for the a, b, x, y that we defined earlier, and using the definition for the 2^m-dimensional $QFT \forall u \in \{0, ..., 2^m - 1\}$: $QFT_{2^m} |u\rangle = \frac{1}{\sqrt{2^m}} \sum_{v=0}^{2^m-1} \omega_{2^m}^{uv} |v\rangle$. Then, $QFT_{2^m} |2x + a\rangle = \frac{1}{\sqrt{2^m}} \sum_{y=0}^{2^{m-1}-1} \sum_{b=0}^{1} \omega_{2^m}^{(2x+a)(2^{m-1}b+y)} |b2^{m-1} + y\rangle = \frac{1}{\sqrt{2^m}} \sum_{y=0}^{2^{m-1}-1} \sum_{b=0}^{1} (-1)^{ab}\omega_{2^{m-1}}^{xy}\omega_{2^m}^{ay} |b2^{m-1} + y\rangle$. Lastly, if we think about the standard basis state as binary encodings of integers in the range $\{0, ..., 2^m - 1\}$, then $|x\rangle |a\rangle = |2x + a\rangle$ and $|b\rangle |y\rangle = |2^{m-1}b + y\rangle$ and thus, we can verify that the circuit we implemented actually performs this operation.

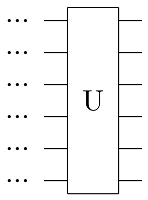
Now, we will calculate the computational cost. First, we will start by counting the number of gates g_m we used in the circuit for each possible choice of m. We will consider the controlled-phase gates as single gates. For $m = 1$, QFT is just a Hadamard, and $g_1 = 1$, so we will see the case $m \geq 2$. In that case, we need g_{m-1} gates for the QFT on $m-1$ qubits, $m-1$ controlled-phase gates, a H gate and $m-1$ swap gates, thus $g_m = g_{m-1} + (2m-1) = \sum_{k=1}^{m}(2k-1) - m^2$ and hence, we will have $O(m^2)$ for performing QFT_{2^m}.

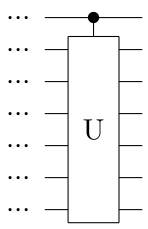
7.2.3 Phase Estimation tools

Approximating phases with low precision: Now, we will start seeing some low-precision solutions to the Phase Estimation Problem in order to steadily scale the algorithm into its general form.

First approach: Phase Kick-back:

As an input, we have a unitary quantum circuit that implements the operation U and we will use it to create a circuit for a controlled-U operation. To accomplish that, first we add a control qubit to the circuit for U and then we replace every gate in the circuit for U with a controlled version of that gate, so that every single gate in U will be controlled by the new control qubit. We can always build circuits for these controlled operations rather than insisting that they're single gates. So,


(a) operation U


(b) operation controlled-U

Next, we will examine the following circuit, where all qubits except the top one is the quantum state eigenvector of U and the eigenvalue of U corresponding to the eigenvector $|\psi\rangle$ determines the measurement outcome probabilities.

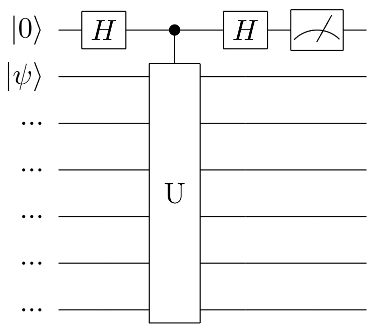


Figure 7.4: Quantum circuit with controlled-U

To better understand its proceedings, we will analyze it at each step:

1. First, the initial state is: $|\phi_0\rangle = |\psi\rangle |0\rangle$

   This gate sets the control qubit to a uniform superposition, so that when the Phase Kick-back occurs, it happens for the $|1\rangle$ state and not the $|0\rangle$ state, creating a relative phase difference that affects the measurement outcomes. Otherwise, the Phase Kick-back would not have any effect on the probabilities of obtaining different measurement outcomes.

2. Then, we apply the H gate and we get: $|\phi_1\rangle = |\psi\rangle |+\rangle = \frac{1}{\sqrt{2}} |\psi\rangle |0\rangle + \frac{1}{\sqrt{2}} |\psi\rangle |1\rangle$

3. After that, we apply the controlled-U gate, so: $|\phi_2\rangle = \frac{1}{\sqrt{2}} |\psi\rangle |0\rangle + \frac{1}{\sqrt{2}}(U |\psi\rangle) |1\rangle = \frac{1}{\sqrt{2}} |\psi\rangle |0\rangle + \frac{e^{2\pi i\theta}}{\sqrt{2}} |\psi\rangle |1\rangle = |\psi\rangle \otimes \left(\frac{1}{\sqrt{2}} |0\rangle + \frac{e^{2\pi i\theta}}{\sqrt{2}} |1\rangle\right)$. And we can observe that the Phase Kick-back phenomenon is happening.

4. Then, the second H gate is applied and we write: $|\phi_3\rangle = |\psi\rangle \otimes \left(\frac{1+e^{2\pi i\theta}}{2} |0\rangle + \frac{1-e^{2\pi i\theta}}{2} |1\rangle\right)$. This gate allows us to learn something about the number θ through the Interference

phenomenon. Before this gate, the state of the top qubit is $\frac{1}{\sqrt{2}}|0\rangle + \frac{e^{2\pi i\theta}}{\sqrt{2}}|1\rangle$ and if we did a measurement on this state, we would get 0 and 1 with probability half each and we would learn nothing about θ but with its application, we can affect the output probabilities.

5. Finally, we measure and we get: $p_0 = \left|\frac{1+e^{2\pi i\theta}}{2}\right|^2 = \cos^2 \pi\theta$ and $p_1 = \left|\frac{1-e^{2\pi i\theta}}{2}\right|^2 = \sin^2 \pi\theta$ and to visualize it, we can see the plot:

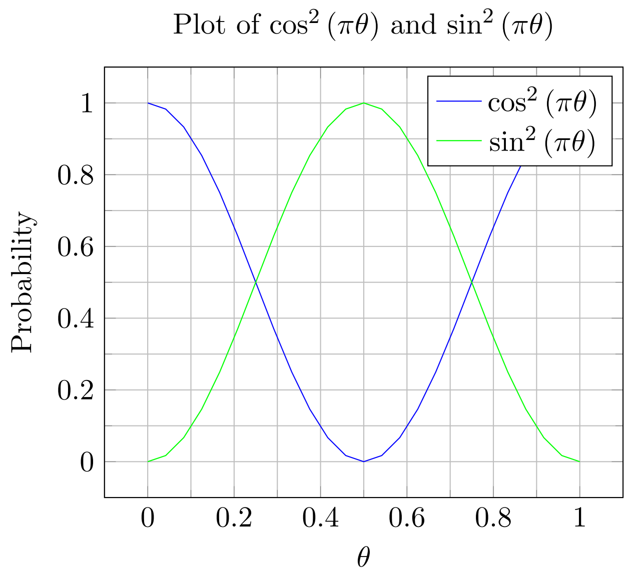


Figure 7.5: Plot of $\cos^2(\pi\theta)$ and $\sin^2(\pi\theta)$

Notice that, when $\theta = 0$, the outcome is 0 and when $\theta = \frac{1}{2}$, the outcome is 1, so, even though the result of the measurement does not reveal exactly the value of θ, it provides us with some information about it and thus, if we are promised that either $\theta = 0$ or $\theta = \frac{1}{2}$, then we could learn from the circuit which one is correct without error. To take it a step further, we can think of the circuit's measurement outcome as being a guess for θ to "one bit of accuracy", so if we write it in binary notation and round it off to one bit after the binary point, we would get $0.\alpha = \begin{cases} 0 & , \text{if } \alpha = 0 \\ \frac{1}{2} & , \text{if } \alpha = 1 \end{cases}$. Therefore, the measurement outcome can be viewed as a guess for the bit α and whenever θ is neither 0 or $\frac{1}{2}$, then there is a non-zero probability that the guess will be wrong and the probability of making an error becomes smaller and smaller as we get closer to 0 or $\frac{1}{2}$.

Second approach: Phase Doubling:

In order to achieve more than one bit accuracy, like the case of factoring, we have to customize the previous circuit a little differently. One thing we can do is to replace the controlled-U gate with two copies of this operation and get a controlled-U^2 operation. If $|\psi\rangle$ is an eigenvector of U with eigenvalue $\lambda = e^{2\pi i\theta}$ then, this state is also an eigenvector of U^2 and the eigenvalue is $\lambda^2 = e^{2\pi i(2\theta)}$ and it determines the measurement outcome probabilities.

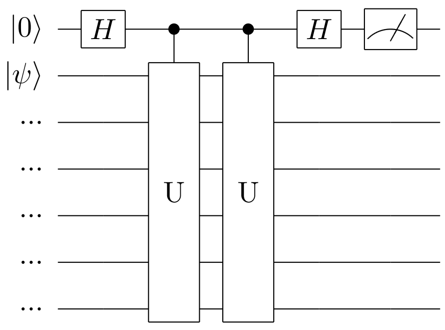


Figure 7.6: Quantum circuit with controlled-U^2

Thus, we are performing the same computation as before, but instead of θ we have 2θ. And if we visualize the result, it will be:

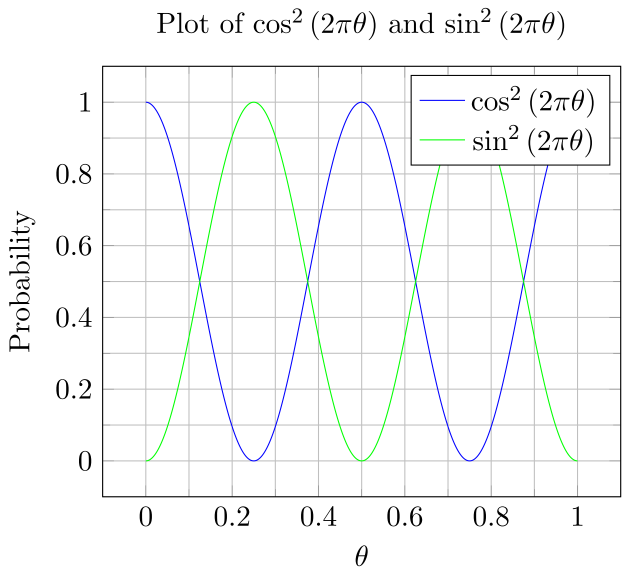


Figure 7.7: Plot of $\cos^2(2\pi\theta)$ and $\sin^2(2\pi\theta)$

And to see the improvement, if we assume that the binary representation of θ is $\theta = 0.\alpha_1\alpha_2\alpha_3...$ then doubling θ effectively shifts the binary point one position to the right, so $2\theta = \alpha_1.\alpha_2\alpha_3....$ Hence, because we are equating $\theta = 1$ with $\theta = 0$ as we move around the unit circle, we see that the bit α_1 has no influence on our probabilities and as a result, we are effectively obtaining a guess for what we would get for the second bit after the binary point if we were to round θ to two bits (e.g. if we know in advance that $\theta = 0$ or $\theta = \frac{1}{4}$, then we can fully trust the measurement outcome to tell us which).

Example 2-qubit Phase Estimation:

In order to understand how this estimation is better than the previous circuit, we will run an example. For that purpose, we will combine these two options into a single circuit as it is depicted below:

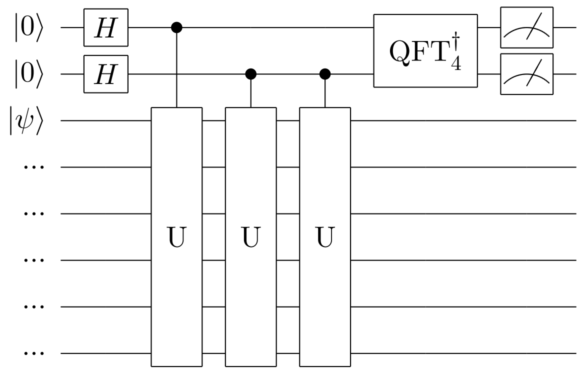


Figure 7.8: Quantum circuit for 2-qubit Phase Estimation

Next, we will examine the circuit, by analyzing each step until the last controlled-U.

- First, the state of the bottom qubits will remain $|\psi\rangle$ throughout the entire circuit and the phases will be kicked into the state of the top two qubits and also $|\psi\rangle$ will be an eigenvector of U.
- Then, we apply the H gate and we get: $|\psi\rangle \otimes \frac{1}{2}\sum_{a_0=0}^{1}\sum_{a_1=0}^{1}|a_1a_0\rangle$.
- Next, the first controlled-U operation is performed and the eigenvalue $\lambda = e^{2\pi i\theta}$ gets kicked into the phase when a_0 (the top qubit) is equal to 1 but not when it is equal to 0. So, we get: $|\psi\rangle \otimes \frac{1}{2}\sum_{a_0=0}^{1}\sum_{a_1=0}^{1}e^{2\pi i a_0\theta}|a_1a_0\rangle$.
- After that, the second and third controlled-U gates do behave similarly but with a_1 instead of a_0 and 2θ instead of θ. So, we can write: $|\psi\rangle\otimes\frac{1}{2}\sum_{a_0=0}^{1}\sum_{a_1=0}^{1}e^{2\pi i(2a_1+a_0)\theta}|a_1a_0\rangle$.

  And if we think the binary string a_1a_0 as representing an integer $x = 2a_1 + a_0 \in \{0, 1, 2, 3\}$ in binary notation, we can write: $|\psi\rangle \otimes \frac{1}{2}\sum_{x=0}^{3}e^{2\pi i x\theta}|x\rangle$.

Now, if we consider a special case, where we are promised that $\theta = \frac{y}{4}$ for some integer $y \in \{0, 1, 2, 3\}$, then $\theta \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$, so we can express this number exactly using binary notation with two bits as $.00, .01, .10, .11$. We should mention that it is not necessary for θ to take one of these four values in general but it is just a helpful way of thinking.

So, we will define one two-qubit state vector for each possible value $y \in \{0, 1, 2, 3\}$ so that: $|\phi_y\rangle = \frac{1}{2}\sum_{x=0}^{3}e^{2\pi i x\frac{y}{4}}|x\rangle = \frac{1}{2}\sum_{x=0}^{3}e^{2\pi i\frac{xy}{4}}|x\rangle$.

And if we take each case, we get: $|\phi_0\rangle = \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle + \frac{1}{2}|2\rangle + \frac{1}{2}|3\rangle$, $|\phi_1\rangle = \frac{1}{2}|0\rangle + \frac{i}{2}|1\rangle - \frac{1}{2}|2\rangle - \frac{i}{2}|3\rangle$, $|\phi_2\rangle = \frac{1}{2}|0\rangle - \frac{1}{2}|1\rangle + \frac{1}{2}|2\rangle - \frac{1}{2}|3\rangle$, $|\phi_3\rangle = \frac{1}{2}|0\rangle - \frac{i}{2}|1\rangle - \frac{1}{2}|2\rangle + \frac{i}{2}|3\rangle$. Thus, since they are orthonormal, we can choose any pair of them, compute their inner product and get 0. Also, each one is a unit vector so $\{|\phi_0\rangle, |\phi_1\rangle, |\phi_2\rangle, |\phi_3\rangle\}$ is an orthonormal basis. Hence, there is a measurement that can help us if we are given one of them but we do not know which, to discriminate which one it is without error. To accomplish that with a quantum circuit, we first have to define a unitary operation V that transforms standard basis states into the four states, so that: $V|00\rangle = |\phi_0\rangle \quad V|01\rangle = |\phi_1\rangle$, $V|10\rangle = |\phi_2\rangle \quad V|11\rangle = |\phi_3\rangle$. And V is a 4×4 matrix that is associated with the 4-dimensional *discrete Fourier transform.* For the purpose of this analysis, we will call it Quantum Fourier Transform (QFT) and in this case QFT_4 rather than V. QFT is the discrete Fourier transform but viewed as a quantum operation. So, the operation QFT_4 maps standard basis states to the four possible states we discussed above. So: $V = QFT_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$. This operation is reversible, meaning that we can transform the states $|\phi_0\rangle, ..., |\phi_3\rangle$ into the standard basis states $|0\rangle, ..., |3\rangle$. Using that, we can make a measurement and learn which value $y \in \{0,1,2,3\}$ describes θ as $\theta = \frac{y}{4}$. Therefore, if we run the above circuit when $\theta = \frac{y}{4}$ for $y \in \{0,1,2,3\}$, immediately before the measurement happens, the state will be $|\psi\rangle|y\rangle$ (for y being a two-bit binary string) and the result will reveal the value y without error. We focused on the special case of θ that we mentioned but we can run this circuit for any choice of U and $|\psi\rangle$ and get any value of θ. Consequently, we managed to improve our chances. It is still not perfect because it can give us the wrong answer, but the answer is directed towards values of y for which $\frac{y}{4}$ is close to θ. More precisely, the most likely result corresponds to the closest value of $\frac{y}{4}$ to θ (equating $\theta = 0$ and $\theta = 1$ as before). It occurs that the closest value for x always appears with probability above 40%. Also, when θ is exactly halfway e.g. between $\theta = 0.375$, the two equally close values of y are equally likely.

Third approach: Many qubits - Phase Estimation Procedure:

Now, we want to extend the single and double qubit versions of phase estimation and it is natural to think that when we add more control qubits, we obtain the general Phase Estimation procedure. For each new control qubit we add on the top, we double the number of times the U operation is performed. In general, for m control qubits, a total of $2^m - 1$ copies of the controlled-U operation are required which leads to a computational cost but also to a more accurate approximation of θ.

So, if we start with $|0^m\rangle$ qubits, then the state prior to QFT is: $\frac{1}{\sqrt{2^m}}\sum_{x=0}^{2^m-1}(U^x|\psi\rangle)|x\rangle = |\psi\rangle \otimes \frac{1}{\sqrt{2^m}}\sum_{x=0}^{2^m-1} e^{2\pi i x\theta}|x\rangle$. Thus, as we did for $m = 2$, for arbitrary values of θ, we will see that the outcomes after we measure will be certain if $\theta = \frac{y}{2^m}$ and they will not be certain for other values of θ but, we can prove certain bounds on the probabilities for different outcomes. So, in general, for $0 \le \theta < 1$, the state after QFT will be: $|\psi\rangle \otimes \frac{1}{2^m}\sum_{y=0}^{2^m-1}\sum_{x=0}^{2^m-1} e^{2\pi i x(\theta - y/2^m)}|y\rangle$. Hence, after the measurement, we get each outcome y with probability p_y: $p_y = \left|\frac{1}{2^m}\sum_{x=0}^{2^m-1} e^{2\pi i x(\theta-y/2^m)}\right|^2$. And if we use the geometric series: $1 + \alpha + \alpha^2 + ... + \alpha^{N-1} = \begin{cases} \frac{\alpha^N - 1}{\alpha - 1} & \text{, if } \alpha \neq 1 \\ N & \text{, if } \alpha = 1 \end{cases}$, we can take $\alpha = e^{2\pi i(\theta - y/2^m)}$ and have: $\sum_{x=0}^{2^m-1} e^{2\pi i x(\theta - y/2^m)} = \begin{cases} \frac{e^{2\pi(2^m\theta - y)} - 1}{e^{2\pi(\theta - y/2^m)} - 1} & \text{, if } \theta \neq \frac{y}{2^m} \\ 2^m & \text{, if } \theta = \frac{y}{2^m} \end{cases}$. Hence, $\frac{1}{2^{2m}}\left|\frac{e^{2\pi i(2^m\theta - y)} - 1}{e^{2\pi i(\theta - y/2^m)} - 1}\right|^2$.

And, in the special case where $\theta = \frac{y}{2^m}$ for $y \in \{0, ..., 2^m - 1\}$, we can write: $|\psi\rangle \otimes \frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} e^{2\pi i \frac{xy}{2^m}} |x\rangle = |\psi\rangle \otimes \frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} \omega_{2^m}^{xy} |x\rangle = |\psi\rangle \otimes QFT_{2^m} |y\rangle$.

Thus, after the application of the inverse QFT, the state becomes $|\psi\rangle |y\rangle$.

Finally, the measurement returns the y in binary encoding with $p_y = 1$. In fact, we can find out more about these probabilities by taking a unit circle and thinking the relationship between arc length and chord length. For example, for $\delta \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ we get a chord from $e^{2\pi i\delta}$ to 1 with length $|e^{2\pi i\delta} - 1|$ and an arc with length $2\pi|\delta|$ as:

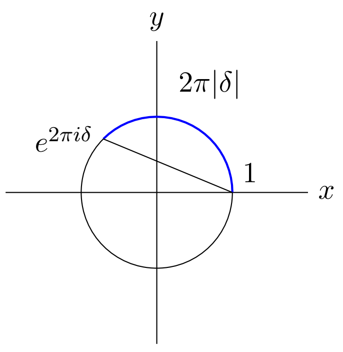


Figure 7.9: Unit circle illustration

From trigonometry we know that: $\left|e^{2\pi i\delta} - 1\right| \leq 2\pi|\delta|$. And if we relate these lengths in the other direction, we can write $4|\delta| \leq \left|e^{2\pi i\delta} - 1\right|$.

And if we use these relations, we can get the following about the probabilities:

- If we assume that θ is a real number and $y \in \{0, ..., 2^m - 1\}$ satisfies $\left|\theta - \frac{y}{2^m}\right| \leq 2^{-(m+1)}$. Then, $\frac{y}{2^m}$ is either the best m-bit approximation to θ or it is one of the two best approximations if θ is exactly halfway between $\frac{y}{2^m}$ and $\frac{y-1}{2^m}$ or $\frac{y+1}{2^m}$. In that case, $|2^m\theta - y| \leq \frac{1}{2}$, so two things are true: $\left|e^{2\pi(2^m\theta - y)} - 1\right| \geq 4|2^m\theta - y| = 4 \cdot 2^m \cdot \left|\theta - \frac{y}{2^m}\right|$ so $\left|e^{2\pi(\theta - y/2^m)} - 1\right| \leq 2\pi \left|\theta - \frac{y}{2^m}\right|$. Therefore, we can get $p_y \geq \frac{1}{2^{2m}} \frac{16 \cdot 2^{2m}}{4\pi^2} = \frac{4}{\pi^2} \approx 0.405$. Which means that the best outcome happens with probability greater than 40%, as we saw in $m = 2$ as well.

- If we assume, on the other hand, that $y \in \{0, ..., 2^m - 1\}$ satisfies: $\frac{1}{2^m} \leq \left|\theta - \frac{y}{2^m}\right| \leq \frac{1}{2}$. Then, there is a better approximation $\frac{z}{2^m}$ to θ in between θ and $\frac{y}{2^m}$. In that case, two things are true: $\left|e^{2\pi(2^m\theta - y)} - 1\right| \leq 2 \quad \left|e^{2\pi(\theta - y/2^m)} - 1\right| \geq 4 \left|\theta - \frac{y}{2^m}\right| \geq \frac{4}{2^m}$. Therefore, we can get $p_y \leq \frac{1}{2^{2m}} \frac{4}{16 \cdot \frac{1}{2^{2m}}} = \frac{1}{4}$. Which means that the approximation off by more than $\frac{1}{2^m}$ are likely to happen with probability at most 25%.

In conclusion, we can run the circuit several times and take the most commonly appearing outcome of the runs and it is very likely that we will obtain an approximation $\frac{y}{2^m}$ that is within $\frac{1}{2^m}$ of the value of θ and the chance that we are off by more than that decreases exponentially in the number of times we run it.

7.2.4 Shor's algorithm

We will divide our analysis in two parts, the Order-finding Problem and the reduced problem of Integer Factorization Problem before we examine Shor's algorithm.

Order-Finding

We will start our analysis with a brief revision of **Number Theory**. So, for any given positive integer N, we will define a set $\mathbb{Z}_N = \{0, 1, ..., N-1\}$ so that $\mathbb{Z}_1 = \{0\}, \mathbb{Z}_2 = \{0, 1\}, ...$ Also, we can have arithmetic operations of $\mathbb{Z}_N$ such as addition and multiplication and if we always take our answers modulo N, then we will always stay within this set when we perform these operations (and we will have a *ring*).

For instance, 3 and 5 are elements of $\mathbb{Z}_7$ and if we do $3 \cdot 5 = 15$ and divide by 7 we get 1 as a remainder and we can write it as $3 \cdot 5 \equiv 1 (\mod 7)$. In fact, the elements $a \in \mathbb{Z}_N$ that satisfy $gcd(a, N) = 1$ are denoted as: $\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N : gcd(a, N) = 1\}$.

Moreover, for the operation of multiplication, the set $\mathbb{Z}_N^*$ forms an *abelian group* which means that if we pick any element $a \in \mathbb{Z}_N^*$ and repeatedly multiply a to itself, we will always get the number 1.

For example, if we take $N = 6$, we have $5 \in \mathbb{Z}_6^*$ because $gcd(5, 6) = 1$ and we get $5 \cdot 5 = 5^2 = 1$ working within $\mathbb{Z}_6$. Also, if we take $N = 21$, we have the following numbers that have gcd equal to 1 with 21: $\mathbb{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$. And if we raise these number to a positive integer power, we get 1: $1^1 = 2^6 = 4^3 = 5^6 = 8^2 = 10^6 = 11^6 = 13^2 = 16^3 = 17^6 = 19^6 = 20^2 = 1$.

So, we can now define the problem as being given $a \in \mathbb{Z}_N^*$ and looking for the smallest positive integer r so that $a^r = 1$, where r is the *order* of a modulo N. And formally,

Order Finding Problem:
Input: Positive integers N and a satisfying $gcd(a, N) = 1$.
Output: The smallest positive integer r such that $a^r \equiv 1 (\mod N)$.

Next, in order to connect this problem with Phase Estimation, we will examine an operation defined on a system whose classical states correspond to $\mathbb{Z}_N$ multiplied by a fixed element $a \in \mathbb{Z}_N^*$: $M_a \left|x\right\rangle = \left|ax\right\rangle, x \in \mathbb{Z}_N$. It is implicit that we are taking the remainder modulo N inside of the ket on the right term.

For example, for $N = 15$ and $a = 2$, we have $M_2 \left|0\right\rangle = \left|0\right\rangle$, $M_2 \left|1\right\rangle = \left|2\right\rangle$, $M_2 \left|2\right\rangle = \left|4\right\rangle$, $M_2 \left|3\right\rangle = \left|6\right\rangle$, $M_2 \left|4\right\rangle = \left|8\right\rangle$, $M_2 \left|5\right\rangle = \left|10\right\rangle$, $M_2 \left|6\right\rangle = \left|12\right\rangle$, $M_2 \left|7\right\rangle = \left|14\right\rangle$, $M_2 \left|8\right\rangle = \left|1\right\rangle$, $M_2 \left|9\right\rangle = \left|3\right\rangle$, $M_2 \left|10\right\rangle = \left|5\right\rangle$, $M_2 \left|11\right\rangle = \left|7\right\rangle$, $M_2 \left|12\right\rangle = \left|9\right\rangle$, $M_2 \left|13\right\rangle = \left|11\right\rangle$, $M_2 \left|14\right\rangle = \left|13\right\rangle$

Thus, $gcd(a, N) = 1$ is a unitary operation which shuffles the elements of the standard basis $\{\left|0\right\rangle, ..., \left|N-1\right\rangle\}$ and acts like a *permutation matrix.* This deterministic operation is also invertible, so if we think about the order r of a modulo N, then the inverse M_a^{r-1} is:

$M_a^{r-1}M_a = M_a^r = M_{a^r} = M_1 = \mathbb{I}$. We can also compute the inverse. If we assume $a \in \mathbb{Z}_N^*$, there is always a unique element $b \in \mathbb{Z}_N^*$ that satisfies $a \cdot b = 1$. If we denote b as a^{-1}, then we can compute it efficiently, so $M_{a^{-1}}M_a = M_{a^{-1}a} = M_1 = \mathbb{I}$. Consequently, M_a is deterministic, invertible and unitary. So, there are N eigenvalues of M_a (even including the same but repeated multiple times) and there is some freedom for selecting eigenvectors respectively.

- Let us choose one eigenvector of M_a: $|\psi_0\rangle = \frac{|1\rangle + |a\rangle + ... + |a^{r-1}\rangle}{\sqrt{r}}$. Then, the eigenvalue corresponding to this eigenvector is 1, because $a^r = 1$: $M_a|\psi_0\rangle = \frac{|a\rangle + ... + |a^{r-1}\rangle + |a^r\rangle}{\sqrt{r}} = \frac{|a\rangle + ... + |a^{r-1}\rangle + |1\rangle}{\sqrt{r}} = |\psi_0\rangle$. So, it seems like we are slowly stirring $|\psi_0\rangle$ but it is already fully stirred so nothing changes, since each standard basis state $|a^k\rangle$ gets shifted to $|a^{k+1}\rangle$ for $k \leq r-1$ and $|a^{r-1}\rangle$ gets shifted back to $|1\rangle$.
- Let us choose another eigenvector of M_a: $|\psi_1\rangle = \frac{|1\rangle + \omega_r^{-1}|a\rangle + ... + \omega_r^{r-1}|a^{r-1}\rangle}{\sqrt{r}} = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\omega_r^{-k}|a^k\rangle$. Then, the eigenvalue corresponding to this eigenvector is ω_r, because $\omega_r^{-r} = 1 = \omega_r^0$ and $|a^r\rangle = |1\rangle = |a^0\rangle$: $M_a|\psi_1\rangle = \sum_{k=0}^{r-1}\omega_r^{-k}M_a|a^k\rangle = \sum_{k=0}^{r-1}\omega_r^{-k}|a^{k+1}\rangle = \sum_{k=1}^{r}\omega_r^{-(k-1)}|a^k\rangle = \omega_r\sum_{k=1}^{r}\omega_r^{-k}|a^k\rangle = \omega_r\sum_{k=0}^{r-1}\omega_r^{-k}|a^k\rangle = \omega_r|\psi_1\rangle$.
- In general, we can find eigenvector/eigenvalue pairs for M_a for any choice of $j \in \{0, ..., r-1\}$ as $|\psi_j\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\omega_r^{-jk}|a^k\rangle \quad M_a|\psi_j\rangle = \omega_r^j|\psi_j\rangle$.

Now, we are ready to solve the Order Finding Problem for a given choice of $a \in \mathbb{Z}_N^*$ by applying the Phase Estimation Procedure to the operation M_a. For that purpose, we need to implement $M_a, M_a^2, M_a^4, ...$ with a quantum circuit in order to obtain a precise estimation from the Phase Estimation procedure.

For starters, we will deal with M_a. We will use binary notation to encode the numbers between 0 and $N-1$, so the number of bits we need is $n = log(N-1) = \lfloor log(N-1) \rfloor + 1$ (e.g. for $N = 21$, we have $n = 5$). So, M_a is defined as an n-qubit operation as follows:

$$M_a|x\rangle = \begin{cases} |ax(\mod N)\rangle & \text{, if } 0 \leq x < N \\ |x\rangle & \text{, if } N \leq x < 2^n \end{cases}$$

Also, it is mentioned that we need to define what it does to the remaining $2^n - N$ standard basis states in order to still have a unitary operation. We can achieve that by defining M_a to do nothing to the remaining standard basis states.

Thus, if we use the algorithms for integer multiplication, division and also methodologies for reversible, garbage-free implementation of them, we can build a quantum circuit for M_a at cost $O(n^2)$. We will analyze each step:

1. First, we build a circuit to perform the operation $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f_a(x)\rangle$. Where, $f_a(x) = \begin{cases} ax(\mod N) & \text{, if } 0 \leq x < N \\ x & \text{, if } N \leq x < 2^n \end{cases}$. This gives us a circuit of size $O(n^2)$.
2. Then, we swap the two n-qubit systems using n swap gates.

3. Next, similarly to the first step, we build a circuit for the inverse operation of a, a^{-1}: $|x\rangle |y\rangle \mapsto |x\rangle |y \oplus f_{a^{-1}}(x)\rangle$.

Hence, if we begin with the bottom n qubits and run the steps above, we get: $|x\rangle |0^n\rangle \mapsto |x\rangle |f_a(x)\rangle \mapsto |f_a(x)\rangle |x\rangle \mapsto |f_a(x)\rangle |x \oplus f_{a^{-1}}(f_a(x))\rangle = |f_a(x)\rangle |0^n\rangle$.

For $M_a^2, M_a^4, ...$ we can use the same methodology by replacing a with $a^2, a^4, ...$ Therefore, for any power k, we can create a circuit M_a^k by computing $b = a^k \in \mathbb{Z}_N^*$ and then using the circuit for M_b. This computation is the *Modular Exponentiation Problem* and it can be done classically using the power algorithm from Number Theory. In terms of the quantum circuit we are running, the cost of M_a iterated k times is simply the cost of M_b for $b = a^k$ and thus the cost is $O(n^2)$ but using this classical algorithm, the cost is linear in m.

As a next step, we will choose different pairs of eigenvectors/eigenvalues to solve the Order Finding Problem using Phase Estimation.

- Choosing a convenient pair:

  We will choose eigenvector $|\psi_1\rangle$ and the corresponding eigenvalue $\omega_r = e^{2\pi i\theta}$ for $\theta = \frac{1}{r}$ of M_a. If we run the Phase Estimation procedure on M_a, we will get an approximation to $\frac{1}{r}$ and by computing the reciprocal, we will be able to learn r if our approximation is decent.

  In particular, if we run Phase Estimation using m control qubits, we get a number $y \in \{0, ..., 2^m - 1\}$ and we take $\frac{y}{2^m}$ as a guess for θ, which is $\frac{1}{r}$ here. So, in order to find r, we need to calculate the reciprocal of our approximation and round to the nearest integer, so $\left\lfloor \frac{2^m}{y} + \frac{1}{2} \right\rfloor$.
  In order to understand this better, let us assume $r = 6$, and we perform Phase Estimation on M_a with eigenvector is $|\psi_1\rangle$ and $m = 5$ control bits. Then, the best 5-bit approximation to $\frac{1}{r} = \frac{1}{6}$ is $\frac{5}{32}$ and we have 68% chance to obtain the outcome $y = 5$ from Phase Estimation, so $\frac{2^m}{y} = \frac{32}{5} = 6.4$ and the rounding to the nearest integer gives 6, which is correct.

  On the other hand, if we do not use enough precision, we might not get the right answer. For example, if we take $m = 4$ control qubits in Phase Estimation, we might get the best 4-bit approximation to $\frac{1}{r} = \frac{1}{6}$ which is $\frac{3}{16}$, so $\frac{2^m}{y} = \frac{16}{3} = 5.333...$ and the rounding gives 5, which is incorrect.

  Therefore, the question that occurs is how much precise do we need to be in order to get the right answer. In particular, we need enough precision to distinguish $\frac{1}{r}$ from nearby possibilities, including $\frac{1}{r+1}, \frac{1}{r-1}$ and for the former, we have $\frac{1}{r} - \frac{1}{r+1} = \frac{1}{r(r+1)}$. So, in order to tell them apart, we need to use enough precision to guarantee that a best approximation $\frac{y}{2^m}$ to $\frac{1}{r}$ is closer to $\frac{1}{r}$ than it is to $\frac{1}{r+1}$, so $\left|\frac{y}{2^m} - \frac{1}{r}\right| < \frac{1}{2r(r+1)}$. and the error is less than half of the distance between $\frac{1}{r}$ and $\frac{1}{r+1}$ and $\frac{y}{2^m}$ is closer to $\frac{1}{r}$ than to any other possibility.

  We can verify this by taking $\frac{y}{2^m} = \frac{1}{r} + \epsilon$, where $|\epsilon| < \frac{1}{2r(r+1)}$. Then we take the reciprocal and we get $\frac{2^m}{y} = \frac{1}{\frac{1}{r}+\epsilon} = \frac{r}{1+\epsilon r} = r - \frac{\epsilon r^2}{1+\epsilon r}$. and if we maximize the numerator and minimize the denominator, we can bound how far away we are from r, $\left|\frac{\epsilon r^2}{1+\epsilon r}\right| \leq \frac{\frac{r^2}{2r(r+1)}}{1-\frac{r}{2r(r+1)}} = \frac{r}{2r+1} < \frac{1}{2}$.

Which means that we are less than $\frac{1}{2}$ away from r, so we will get r when we round.

However, since we do not know what r is yet, we cannot use it to tell us how much accuracy we need. Nevertheless, r must be smaller than N to ensure that we use enough precision, so we need $\left|\frac{y}{2^m} - \frac{1}{r}\right| \leq \frac{1}{2N^2}$.

In that case, we will have decent precision to find r correctly when we take the reciprocal. In fact, taking $m = 2\log N + 1$ ensures that we have a high chance to get an estimation with this precision using the methodology we described.

- Choosing another pair:

  We will now choose an eigenvector $|\psi_k\rangle$ of M_a for any choice of $k \in \{0, ..., r-1\}$. Then, the outcome of the Phase Estimation procedure will be an approximation $\frac{y}{2^m} \approx \frac{k}{r}$. However, since we do not know either k or r, this might or might not (e.g. for $k = 0$) allow us to identify r. Nevertheless, in order to identify r, we will use the *Continued Fraction Algorithm* to turn our approximation $\frac{y}{2^m}$ into nearby fractions or even $\frac{k}{r}$.

  We know that given an integer $N \geq 2$ and a real number $\alpha \in (0, 1)$, there is at most one choice of integers $u, v \in \{0, ..., N-1\}$ with $v \neq 0$ and $gcd(u, v) = 1$ satisfying $|\alpha - \frac{u}{v}| < \frac{1}{2N^2}$. So, given α, N the *Continued Fraction Algorithm* finds u, v or outputs that they do not exist. It can be implemented as a Boolean circuit having size $O((\log{(N)})^3)$.

  So, if we run this algorithm for $N, \alpha = \frac{y}{2^m}$ and we have a very close approximation $\frac{y}{2^m}$ to $\frac{r}{k}$, then we will conclude that $\frac{u}{v} = \frac{k}{r}$. Thus, if we learn u, v from this equation for $k \in \{0, ..., r-1\}$ chosen uniformly at random, then we are very likely to recover r after just a few samples. In particular, if our guess for r is the least common multiple of all the values for v that we observe, then we will be correct with high probability. Also, some values of k are not good because they might share common factors with r, but random choices of k are not likely to hide factors of r and the probability that we do not guess r correctly drop exponentially in the number of samples.

- Working without an eigenvector:

  Finally, we will run the Phase Estimation procedure on the input state $|1\rangle$ (the n-bit binary encoding of the number 1), in place of an eigenvector $|\psi\rangle$ of M_a. This is not an eigenvector of M_a for $a \in \mathbb{Z}_N^*$ unless $a = 1$ which is a choice we will avoid. The state $|1\rangle$ can be also written as:

$$|1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |\psi_k\rangle$$

  So, after the QFT, the state will be:

$$\frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |\psi_k\rangle |\gamma_k\rangle$$

where, $|\gamma_k\rangle$ represents the state of the top m qubits after the inverse of the QFT is performed:

$$|\gamma_k\rangle = \frac{1}{2^m} \sum_{y=0}^{2^m-1} \sum_{x=0}^{2^m-1} e^{2\pi i x(\frac{k}{r} - \frac{y}{2^m})} |y\rangle$$

Thus, when the top m qubits are measured, we get an approximation $\frac{y}{2^m}$ to the value $\frac{k}{r}$ where $k \in \{0, ..., r-1\}$ is chosen uniformly at random and hence, we can learn r with a high probability after some runs.

Regarding the **total cost**, we will need to implement each controlled-unitary M_a^k in $O(n^2)$ and there are m such operations so $O(n^3)$ in total. Moreover, we have m Hadamard gates, which cost $O(n)$ and the QFT contributes $O(n^2)$. So, the quantum circuit will cost $O(n^3)$. Also, there are some classical computations that need to be performed such as computing the powers a^k in $\mathbb{Z}_N$ for $k = 2, 4, 8, ..., 2^{m-1}$, which are needed to create the controlled-unitary gates, as well as the Continued Fraction Algorithm that converts approximations of θ into fractions and both computations can be performed by Boolean circuits in $O(n^3)$. It is noted that these bounds can be improved using asymptotically fast algorithms but we used standard algorithms for basic arithmetic operations.

Integer Factorization

Now, we will see how we can reduce the Order Finding Problem to the Integer Factorization Problem. It is noted that this section is completely classical.

First, we will state the problem

Integer Factorization Problem:
Input: An integer $N \geq 2$.
Output: The prime factorization of N.

By this output, we mean a list of the prime factors of N and the powers to which they must be raised to obtain N by multiplication. For instance, the prime factors of 12 are 2 and 3, because $12 = 2^2 \cdot 3$. For that, we also use the *Fundamental Theorem of Arithmetic*, which says that there is only one prime factorization for each positive integer $N \geq 2$. The fastest known algorithm for factoring large integers is known as the *Number Field Sieve.* In order to understand the scale of this problem, if we use the Number Field Sieve algorithm to run the $RSA250$, we need thousand of CPU core-years, distributed across tens of thousands of machines around the world.

So, the **intuition** is that we want to factorize the number N and we can do this recursively. In particular, we will focus on the procedure of splitting N, which means finding any two integers $b, c \geq 2$ for which $N = b \cdot c$. We will use a Primality Testing Algorithm to see if N is a prime number, and if it is, then this is not possible but if it is not, we will try to split it. Once we do that, we can simply recurse on b and c until all of our factors are prime and we gain the prime factorization of N.

Now, regarding splitting:

- For even integers, we just split and the output is 2 and $\frac{N}{2}$.
- For perfect powers, $N = s^j$, for integers $s, j \geq 2$, we approximate the roots $N^{\frac{1}{2}}, N^{\frac{1}{3}}, N^{\frac{1}{4}}, ...$ and check nearby integers as suspects for s. This needs at most $\log(N)$ steps because at that point the root drops below 2 and will not reveal additional candidates.
- For odd N and not a prime power, Order Finding allows us to split N as follows:

  1. Input an odd, composite integer N that is not a prime power and iterate the next steps.
  2. Choose $a \in \{2, ..., N-1\}$ at random.
  3. Calculate $d = gcd(a, N)$.
  4. If $d > 1$, then output $b = d$ and $c = \frac{N}{d}$ and stop, else continue to the next step knowing that $a \in \mathbb{Z}_N^*$.
  5. Let r be the order of a modulo N by using the Order Finding Algorithm.
  6. If r is even, compute $x = a^{\frac{r}{2}} - 1(\mod N)$, then calculate $d = gcd(x, N)$ and finally, if $d > 1$, output $b = d$ and $c\frac{N}{d}$ and stop.
  7. If you reach at this step, the iteration has failed to find a factor of N because either the order of a modulo N is odd or the order of a modulo N is even and $gcd(a^{\frac{r}{2}} - 1, N) = 1$.

  Regarding the failure step, it can be proved that, for a random choice of a, with probability at least $\frac{1}{2}$ neither of these events happens. In particular, the probability is at most $2^{-(m-1)}$ for m being the number of distinct prime factors of N. For this to be true, we assume that N is not a prime number and also that is is odd.

Thus, if we repeat the process t times and choosing a at random each time, we will succeed in splitting N with probability at least $1 - \frac{1}{2^t}$. That is because if we choose a for which the order r of a modulo N is even, then we consider the numbers $a^{\frac{r}{2}} - 1(\mod N)$ and $a^{\frac{r}{2}} + 1(\mod N)$ and we get $\left(a^{\frac{r}{2}} - 1\right)\left(a^{\frac{r}{2}} + 1\right) = a^r - 1$. Also, we know that $a^r(\mod N) = 1$, by the definition of the order and that is another way of saying that N evenly divides $a^r - 1$. Thus, N evenly divides the number $\left(a^{\frac{r}{2}} - 1\right)\left(a^{\frac{r}{2}} + 1\right)$, i.e., that every prime factor of N must divide either $a^{\frac{r}{2}} - 1$ or $a^{\frac{r}{2}} + 1$ or both. Finally, for a random a, we are likely to have prime factors of N dividing both terms, which allows us to split N by computing the GCD.

7.2.5 RSA Protocol

The **RSA (Rivest-Shamir-Adleman) algorithm** (1977) is one of the first public key cryptosystems and it is used for secure data transmission because it provides the necessary operation to enable encryption, decryption, digital signatures and key exchange within a common framework.

We will use the standard scenario of two parties, Alice and Bob, who wish to communicate securely using Asymmetric Key Cryptography (AKC). The RSA algorithm involve key generation, key distribution, encryption and decryption.

In particular, the basic outline of the protocol is the following:

1. **Key generation**:

   Public and private keys are generated based on mathematical principles around prime numbers, where calculating them is easy, but the reverse is hard. We will refer to them as *public key* (e, n) and *private key* (d, n), where $n = p \cdot q$ $(p, q$ prime numbers) is common to both public and private key and is known as the modulus and also e is an integer that satisfies $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$ and d satisfies $d \cdot e \equiv 1(\mod \phi(n))$ (where $\phi(n) = (p-1)(q-1)$).

2. **Key distribution**:

   The public key (e, n) is made public to those who may wish to send a message and the private key (d, n) is kept secret.

3. **Encryption**:

   Alice wishes to send a message M to Bob and she uses Bob's public key (e, n) to encrypt the message into ciphertext C, $C \equiv M^e(\mod n)$.

4. **Decryption**:

   Bob receives the ciphertext C and uses his private key (d, n) to decrypt the message back into message M, $M \equiv C^d(\mod n)$.

An implementation of this can be seen below.

We can use RSA protocol to do symmetric key exchange, padding-based key exchange, key encapsulation mechanism (KEM) and also digital signatures.

Breaking the RSA Protocol

First, we should mention that the utility and security of RSA algorithm is based on two mathematical assumptions:

1. First is the fact that finding the *modular multiplicative inverse* d (given an integer a and modulus m, the remainder from ax/m is 1, for coprimes(numbers that have a GCD of 1, e.g. 3 and 13 because only 1 can divide both)) given that access only to (e, n) is computationally infeasible.

2. Second is the fact that in the RSA setting, the *modular exponentiation operation* (it is the remainder when an integer is raised to a power (exponent) and then divided by a number (modulus), e.g. $x^e \mod m$) behaves like a *one-way trapdoor function* (a one-way function where it is easy to compute an output from the inputs, but hard to do vice

versa). When used for encryption, it gives a ciphertext that is inscrutable (impossible to understand) and without access to the private key, inverting the operation to recover the plaintext from the ciphertext is infeasible. However, with access to the private key, which acts as a trapdoor, the ciphertext can be easily decrypted.

As far as **attacks** on the RSA algorithm are concerned, most of the time, their goal is to undermine the first assumption by recovering the private key number d via factorizing (finding out what numbers can be multiplied together to get the result) the modulus n (the remainder left over after dividing by the value specified). We will see that it is easy to recover d if one has access to either the prime factors p, q (prime numbers that are multiplied together to make n) of n or the totient $\phi(n)$ (the Euler's totient is a number(s) that are not greater than the given number and are relatively prime to it, having no factors in common). Remember that $p, q, \phi(n)$ are kept secret during key generation and after that they are discarded. However, a third party can recover this information using a classical or quantum computer and uncover the private key and use prime factorization, which is the key computational primitive necessary to break RSA.

- **Classical Computing and RSA**:

  For a classical setting, prime factorization of large integers is known to exhibit super-polynomial or sub-exponential scaling and the best-known classical algorithm for factoring integers larger than 10^{100} is the *General Number Field Sieve (GNFS)*. Furthermore, the largest integers factored on classical hardware right now are in the range of 829 bits or 250 decimal digits. However, given the exponential growth in classical computing power over the last several decades, 1024-bit RSA is no longer considered secure in the near term and is now deprecated and the newer 2048-bit RSA, with magnitude in the range of 10^{617} has replaced it. Although for now and the foreseeable future, factoring such integers might be infeasible on classical systems, the advent of quantum computer shows that this will not be the case.

- **Quantum Computing and RSA**:

  For a quantum setting, prime factorization can be computed using Shor's algorithm. This algorithm offers super-polynomial speedup relative to classical algorithms on a problem of great practical importance.

  In order to break the RSA, we will exploit the periodicity of the modular exponential function $f(x) = a^x(\mod n)$ and we will provide a quantum period-finding primitive that accomplishes efficient prime factorization of the modulus n. The outline of the steps that we need to follow is mentioned below:

  1. First, given the modulus n, which is published as part of the public key, choose a number a coprime to n, i.e. $gcd(a, n) = 1$. In fact, any number less than n that we pick at random is likely to be coprime to n, since $n = p \cdot q$ has exactly two prime factors (p, q).
  2. Next, find the exponent r such that $a^r \equiv 1(\mod n) \leftrightarrow a^r - 1 \equiv 0(\mod n)$. The existence of an exponent r so that the previous equivalence holds is guaranteed by the periodicity property of modular exponentiation.

3. Then,
   - if r is even, for some γ, $a^r - 1 \equiv 0(\mod n) \rightarrow \left(a^{\frac{r}{2}} - 1\right)\left(a^{\frac{r}{2}} + 1\right) = \gamma \cdot n$ and the left-hand side of the latter equality must contain p, q as two of its prime factors since the right-hand side does.
   - if r is odd, return to Step 1 and choose a different a.
4. After that, use the *extended Euclid algorithm* (improved GCD) for finding either $gcd\left(\left(a^{\frac{r}{2}} - 1\right), n\right)$ or $gcd\left(\left(a^{\frac{r}{2}} + 1\right), n\right)$. The computed GCD is very likely to identify one of the prime factors p or q and we can divide n with that prime factor to recover the other.
5. Finally, once p, q are known, use the steps from the original RSA algorithm for reconstructing for reconstructing the totient $\phi(n)$ and generating the private key number d as the modular inverse (given an integer a and modulus m, the remainder from ax/m is 1) of the known public key number e.

In the code provided below, we can see an implementation of this procedure.

On quantum computers this algorithm can accomplish poly-logarithmic scaling (an algorithm which scales similar to logarithmic scaling, e.g. Binary search) as favorable as $O\left((\log n)^2(\log\log n)\right)$ in terms of the modulus n, or polynomial scaling in terms of the number of bits needed to represent n. All in all, this is a super-polynomial speedup compared to the classical GNFS algorithm.

Finally, in terms of hardware, Shor's algorithm requires at least two sets of qubits: a register with n qubits to store the input integer (the number to be factored), and additional qubits for performing quantum operations (minimum, it typically requires around $2n$ qubits, where n is the number of bits in the integer we want to factor). The exact number of additional qubits depends on the specifics of the algorithm and the implementation, but it typically requires a polynomial number of qubits in relation to the input size. So, a few tens of thousands to several million qubits (he exact number depends on various factors and ongoing advancements in quantum hardware and error correction) will be necessary to break 2048-bit RSA using Shor's algorithm which might seem a lot for now but it is not inconceivable that quantum computers with several tens of thousands of qubits will become available over the next several years.

It is also noted, for completeness, that Diffie-Hellman and Digital Signature Algorithm (DSA) protocols can be broken if an attacker can solve an instance of the Discrete Logarithm Problem (DLP) because Shor's algorithm can solve DLP in polynomial time as well as, Elliptic-curve Diffie-Hellman (ECDH) and Elliptic Curve Digital Signature Algorithm (ECDSA). All of these algorithms have exposure to attack from quantum algorithms such as Shor's, since solutions can be developed to solve the mathematical problems which acted as a premise in their design. Thus, they will need to be replaced by quantum-safe asymmetric cryptosystems which are more resilient to attack from quantum

Implementation:

For the deeper understanding of the above analysis, there is an implementation in this **link**. For the development of this algorithm there is a simple implementation of the classical computation, a quick implementation of the quantum computation and some code about QFT & QPE in this **link**.

7.3 Toy Example - Application

Shor's algorithm was introduced by Peter Shor (1994) and not only proves that quantum computers can outperform classical computers but also that it will change our lives in the future. With this algorithm we can find the prime factors of large numbers. For instance, RSA relies on the idea that you cannot easily find the prime factor of a large number e.g., the factors of $N = 60$ are $N = 60 = 2^2 \times 3 \times 5$. But if someone gives us two numbers and we multiply them, it is very easy to verify the result, since $N = P \times Q$. Also, $n = len(N)$, so the complexity is $O(2^n)$. As a result, the encryption breaking process in the classical computing is exponential. For example, if you have 256 bits, it is 2^{256}. Thus, it takes a lot of time to find the prime factors of a particular long number. The RSA encryption relies on that. So, if someone wants to break the encryption without having the key, they must find the prime factors of a very large number and in classical computers, it can take thousands of years to calculate it and that is the reason that this encryption is safe for now, but it will change in the future if someone has a quantum computer with many qubits since they will be able to break it in less than a minute. Moreover, the Shor's algorithm can calculate it in a way that is not exponential and even if you increase the number of bits or the length of the number, it would not actually increase the amount of time that we need to break it at least exponentially. Hence, it would be very fast in quantum computers but it would not be very fast in classical computers and that is way it is safe today but not in the future.

A brief revision on **Modular Arithmetic**: $3 = 26 \mod 23$, $20 = 43 \mod 23$, $1 = 24 \mod 23$, $22 = -1 \mod 2346 + 18 \mod 23 = 0 + 18 \mod 23$, $46 \times 18 \mod 23 = 0 \times 18 \mod 23$.

A brief revision on **Greatest Common Divisor** (GCD): $gcd(15, 21) = 3$, $15 = 3 \times 5, 21 = 3 \times 7$.

Thus, in order to find the prime factors of $N = 21$ using the above, firstly, we write it in the following form: $x^2 = 1 \mod 21$. And we can see that x can be 1, -1, 8, -8, 13, 20, -20. And if we choose 8, we get: $8^2 - 1^2 = 0 \mod 21 \Rightarrow (8 - 1) \times (8 + 1) = 7 \times 9$. And now, if we try to find the gcd, we will get the prime factors: $gcd(21, 9) = 3$, $gcd(21, 7) = 7$. This works thanks to the **periodicity**. Hence, the **4 general rules** are:

1. It should not be ± 1 because it would give us 0: $x \neq \pm 1 \mod N$.
2. It should be equal to 1 in $\mod N$: $x^2 = 1 \mod N$.
3. We will finally get the factors with the gcd: $gcd(N, x + 1)$.

Example: We have $N = 21$, x is equal to a random number (here it is 2). Then, we are taking powers of 2 and we are writing them in modular arithmetic: $2^1 = 2 \mod 21$, $2^2 = 4 \mod 21$, $2^3 = 8 \mod 21$, $2^4 = 16 \mod 21$, $2^5 = 11 \mod 21$, $2^6 = 1 \mod 21$, $2^7 = 2 \mod 21$, $2^8 = 4 \mod 21$. We observe that it is a loop so, it is a periodic function with period equal to 6. Hence, we can write it as $x^6 = (x^3)^2 = 1 \mod 21$ and we found it because $2^3 = 8$. It seems that we got lucky, since we chose it randomly. In fact, it happens that we were going to get lucky 50% of the time. So, by taking any random number, with 50% probability we are going to find this solution. If the period was not an even number, we would not be able to divide it with 2 because it has to be an integer. In the case that it was not, then we would have to choose another random number and try to find its period and see it that gives us the solution.

4. The period number should be even.

The Solution:

- **Classical**:

  In a classical computer, we can try this with a small number but with a 256-bit number it would take thousands of years to calculate it.

- **Quantum**:

  The quantum solution uses the **Quantum Fourier Transform (QFT)**. The QFT is the quantum implementation of the Discrete Fourier Transform over the amplitudes of a wave-function. It is part of many quantum algorithms, most notably Shor's factoring algorithm and Quantum Phase Estimation (QPE).

For the *Quantum Solution*, we are going to transform the basis, meaning that we are going to take the vector from computational basis $\{|0\rangle, |1\rangle\}$ to the Fourier basis $\{|+\rangle, |-\rangle\}$.

1-Qubit QFT:

We can take this by applying the h gate but we have to control the angles. In the case of 1 qubit, the h is enough as shown below.

Proof that QFT works for 1 qubit: $N = 2^1$: $QFT|x\rangle = |\tilde{x}\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{\frac{2\pi ixy}{N}}|y\rangle = \frac{1}{\sqrt{2}}\sum_{y=0}^{2-1} e^{\frac{2\pi ixy}{2}}|y\rangle = \frac{1}{\sqrt{2}}e^{\frac{2\pi ix0}{2}}|0\rangle + \frac{1}{\sqrt{2}}e^{\frac{2\pi ix1}{2}}|1\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}e^{\pi ix}|1\rangle$.

- If $x = 0$: $\frac{1}{\sqrt{2}}|0\rangle + |1\rangle$ which is $|+\rangle$
- If $x = 1$: $\frac{1}{\sqrt{2}}|0\rangle - |1\rangle$ which is $|-\rangle$

2-Qubit QFT:

However, for 2 qubits, there are some problems with the previous method. So, we have the computational basis: $\{|00\rangle, |10\rangle, |01\rangle, |11\rangle\}$ and the general QFT formula is: $QFT|x\rangle =$

$|\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi ixy}{N}} |y\rangle$. Also, note that, $e^{\pi i} = -1$, $N = 2^n$, $n = \#qubits$. So, now, we have the previous formula but for $N = 2^n$: $QFT\,|x\rangle = |\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi ixy}{N}} |y\rangle$ and we try to substitute some notations as follows, $y = \begin{bmatrix} y_1, y_2, y_3, ..., y_n \end{bmatrix}$. What happens when we have (e.g., for 3 qubits there will be a $|7\rangle$): $|0\rangle, |1\rangle, |2\rangle, |3\rangle$? Well, it is actually the representation in the binary state, so: $|2\rangle = |10\rangle$. In general, we can calculate this decimal to binary transformation using this: $y = 2^{n-1} \times y_1 + 2^{n-2} \times y_2 + ... + 2^0 \times y_n = \sum_{k=0}^{n} y_k 2^{n-k}$. Thus, now we will use this y formula with $N = 2^n$ and we will get: $QFT\,|x\rangle = |\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi ixy}{N}} |y\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi ix \sum_{k=0}^{n} y_k 2^{-k}} |y_1, y_2, y_3, ..., y_n\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \prod_{k=1}^{n} e^{\frac{2\pi ixy_k}{2^k}} |y_1, y_2, y_3, ..., y_n\rangle = \frac{1}{\sqrt{N}}(|0\rangle + e^{\frac{2\pi ix}{2^1}} |1\rangle) \otimes (|0\rangle + e^{\frac{2\pi ix}{2^2}} |1\rangle) \otimes (|0\rangle + e^{\frac{2\pi ix}{2^3}} |1\rangle) \otimes ... \otimes (|0\rangle + e^{\frac{2\pi ix}{2^n}} |1\rangle)$ where, the 1st qubit is always in the $|0\rangle$ state and the 2nd qubit is $e^{\frac{2\pi ix}{2^3}} |1\rangle$ and they lead to a matrix: $\begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}$. Also, it is known that U-gate has a familiar form since $U_3(0, 0, \lambda) = U_1 = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix}$. So, if we apply this gate to the circuit on all qubits, starting from x_1 all the way to x_n, the result will be:

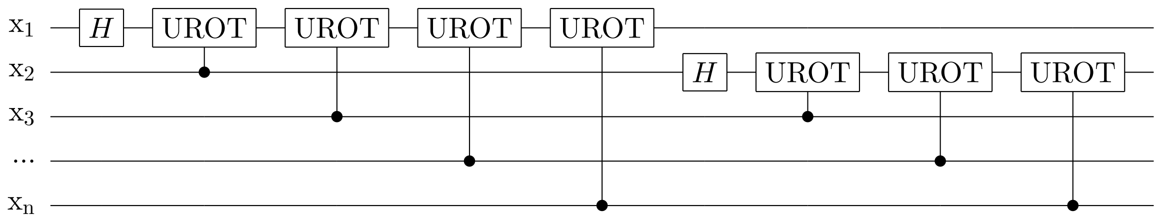


Figure 7.10: QFT

And we get $|x_1x_2x_3x_4x_5x_6...x_n\rangle = (|0\rangle + e^{\frac{2\pi ix_1}{2^1}} |1\rangle) \otimes |x_2x_3x_4x_5x_6...x_n\rangle = (|0\rangle + e^{\frac{2\pi ix_2}{2^2}} e^{\frac{2\pi ix_1}{2^1}} |1\rangle) \otimes |x_3x_4x_5x_6...x_n\rangle = (|0\rangle + e^{\frac{2\pi ix_3}{2^3}} e^{\frac{2\pi ix_2}{2^2}} e^{\frac{2\pi ix_1}{2^1}} |1\rangle) \otimes |x_4x_5x_6...x_n\rangle$ and we can see that it is exactly the reverse order of what we had before.

QPE:

First, we have a qubit $|\psi\rangle$ and we apply a U-gate: $U\,|\psi\rangle = e^{i\theta}\,|\psi\rangle$ and we already know that $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \rightarrow P(1) = P(0) = 50\%$. By applying a phase, we get: $e^{\frac{i\pi}{2}} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \rightarrow P(1) = \left| e^{\frac{i\pi}{2}} \frac{1}{\sqrt{2}} \right|^2 = P(0) = 50\%$. So, we can see that we end up with 50% as we have seen with the Z-gate. However, sometimes we have a **phase** that affects the outcome. The idea here is to find that θ that we can actually understand in what phase we are in when we do not know the state. That is why we use QPE.

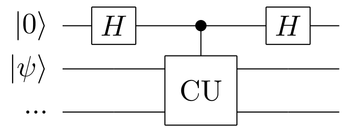


Figure 7.11: QPE

So, as the circuit reveals, firstly, we have $|0\rangle$ and $|\psi\rangle$ and we apply H in $|0\rangle$ and controlled U in $|\psi\rangle$ and then H in $|0\rangle$ again. Thus, $|0\rangle|\psi\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle|\psi\rangle) \xrightarrow{CU} \frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle e^{i\theta}|\psi\rangle) \xrightarrow{H} \frac{1}{\sqrt{2}}(\frac{(|0\rangle+|1\rangle)|\psi\rangle}{\sqrt{2}} + e^{i\theta}\frac{(|0\rangle-|1\rangle)|\psi\rangle}{\sqrt{2}}) = \frac{1}{2}(|0\rangle(1+e^{i\theta}) + |1\rangle(1-e^{i\theta}))|\psi\rangle$. Now, if we calculate the possibilities, we get: $P(1) = \left|(1-e^{i\theta}) \times \frac{1}{2}\right|^2$ and $P(0) = \left|(1+e^{i\theta}) \times \frac{1}{2}\right|^2$. In fact, if we actually rotate the measuring device or the state itself, we can change the probability for $\theta = 1$: $P(0) = 0.9999...$ and for $\theta = 10$: $P(0) = 0.9924....$ Thus, we can try to measure the result many times or we can use multiple qubits, in order to calculate θ more efficiently.

As in QFT, we apply H to all $|0\rangle$ and then the U-gates to $|\psi\rangle$.

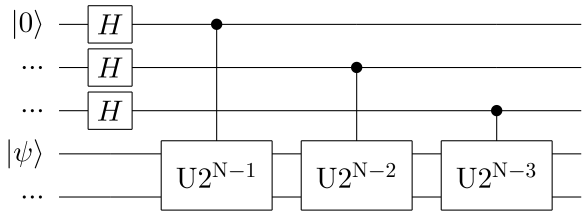


Figure 7.12: QPE 2

So, the result is very similar to QFT: $|0\rangle^{*n}|\psi\rangle \xrightarrow{H} (\frac{1}{\sqrt{2}})^n(|0\rangle + |1\rangle)^{*n}|\psi\rangle \xrightarrow{U} (\frac{1}{\sqrt{2}})^n(|0\rangle + e^{i\theta 2^{n-1}}|1\rangle) \otimes (|0\rangle + e^{i\theta 2^{n-2}}|1\rangle) \otimes ... \otimes (|0\rangle + e^{i\theta 2^0}|1\rangle)$. In fact, if we try to compare QFT and QPE, we get: $\frac{2\theta\pi}{2^n}$.

And now, if we try to inverse the QFT with QFT ad-joint (QFT^{+}) we will get:

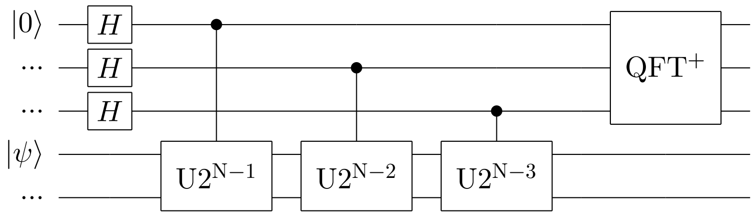


Figure 7.13: QFT^{+}

And we will end up with $|2^n\theta\rangle$ state and that is how we will find θ. Hence, the overall circuit can be seen in the figure.

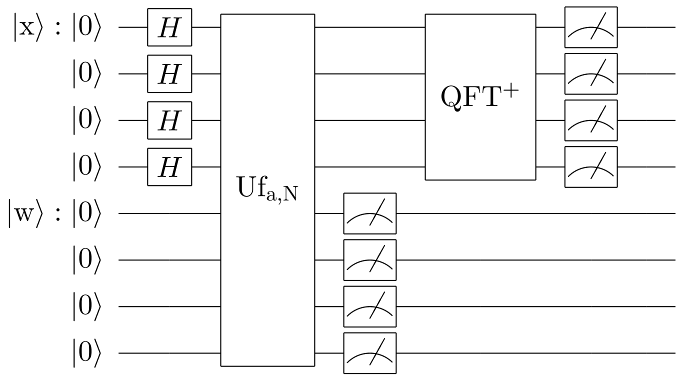


Figure 7.14: Overall circuit

Step-by-step analysis:

1. In $|x\rangle$ we have 4 qubits and in $|w\rangle$ we also have 4 qubits. We are trying to find the prime factors of number 15. In binary, 15 is 1111 and that is why we use 4 qubits. In the end, we will end up with $N = p \times q$. So, we start with: $|0\rangle^{*4}|0\rangle^{*4}$.

2. We apply the H and we get: $\frac{1}{4}(|0000\rangle + |0001\rangle + ... + |1111\rangle)|0\rangle^{*4}$.

3. We apply the U_f-gate, which will be the QPE that will do the modular exponentiation as: $f_{a,N}(x) = a^x \mod N$. Also, it is noted that this oracle will give us: $|x\rangle|w\rangle \rightarrow |x\rangle|w \oplus f_{a,N}(x)\rangle$. It can be reversible. It will change the state of the 2^{nd} qubit while keeping the 1^{st} in the superposition. So, we get: $\frac{1}{4}(|0000\rangle|0 \oplus (13^0 \mod 15)\rangle + |0001\rangle|0 \oplus (13^1 \mod 15)\rangle + ...) = \frac{1}{4}(|0000\rangle|(13^0 \mod 15)\rangle + |0001\rangle|(13^1 \mod 15)\rangle + ...) = ...$ So, the total state will be: $|x\rangle|w\rangle = \frac{1}{4}(|0\rangle|1\rangle + |1\rangle|13\rangle + |2\rangle|4\rangle + |3\rangle|7\rangle + |4\rangle|1\rangle + |5\rangle|13\rangle + |6\rangle|4\rangle + |7\rangle|7\rangle + |8\rangle|1\rangle + |9\rangle|13\rangle + |10\rangle|4\rangle + |11\rangle|7\rangle + |12\rangle|1\rangle + |13\rangle|13\rangle + |14\rangle|4\rangle + |15\rangle|7\rangle)$.

4. Then, we measure the $|w\rangle$ and if we assume that we measured $|w\rangle = 7$, the result will be: $|x\rangle = \frac{1}{4}(|3\rangle + |7\rangle + |11\rangle + |15\rangle)$. So, if we add them up together, we will get: $|x\rangle|w\rangle = \frac{1}{2}(|3\rangle + |7\rangle + |11\rangle + |15\rangle) \otimes |7\rangle$.

5. We apply the QFT^+: $QFT|x\rangle = |\tilde{x}\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{\frac{2\pi ixy}{N}}|y\rangle$ and $QFT^+|\tilde{x}\rangle = |x\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{\frac{-2\pi ixy}{N}}|y\rangle$. Moreover, we know $N = 2^4 = 16$ and we begin first with $|3\rangle$ and the rest will be calculated similarly, so: $QFT^+|3\rangle = \frac{1}{\sqrt{16}}\sum_{y=0}^{15} e^{\frac{-2\pi i3y}{16}}|y\rangle$. So, if we generalize it, we will get: $QFT^+|x\rangle = \frac{1}{8}\sum_{y=0}^{15}(e^{\frac{-3\pi y}{8}} + e^{\frac{-7\pi y}{8}} + e^{\frac{-11\pi y}{8}} + e^{\frac{-15\pi y}{8}})|y\rangle = \frac{1}{8}(4|0000\rangle + 4i|0100\rangle - 4|1000\rangle - 4i|1100\rangle) = \frac{1}{8}(4|0\rangle + 4i|4\rangle - 4|8\rangle - 4i|12\rangle)$. This result is an example of interference since all the rest are cancelling each other, leaving only the above values.

6. Finally, we measure the $|x\rangle$. Previously, we ended up with these 4 values $[0, 4, 8, 12]$, so the period r is 4 and a can be anything but in this case we chose 13. Therefore, $x^{\frac{r}{2}} = 13^{\frac{4}{2}} = 4 \mod 15$, $x + 1 = 5$, $x - 1 = 3$ $gcd(x + 1, N) = gcd(5, 15) = 5$, $gcd(x - 1, N) = gcd(3, 15) = 3$. And that is how we managed to find the prime factors.

   Proof for the connection between **Modular Exponentiation** $f_{a,N}(x) = a^x \mod N$ and QPE. In fact, for x: $x = x_1, x_2, x_3, ...x_n = 2^{n-1}x_1 + 2^{n-2}x_2 + ... + 2^0 x_n$ we get, $f_{a,N}(x) = a^x \mod N = a^{2^{n-1}x_1 + 2^{n-2}x_2 + ... + 2^0 x_n} \mod N = a^{2^{n-1}x_1} a^{2^{n-2}x_2} ... a^{2^0 x_n}$. Therefore, this is what we have been doing in order to find the modular powers that we discussed in the Shor's algorithm in the first place. This final result is exactly the same as the QPE formula that is shown below.

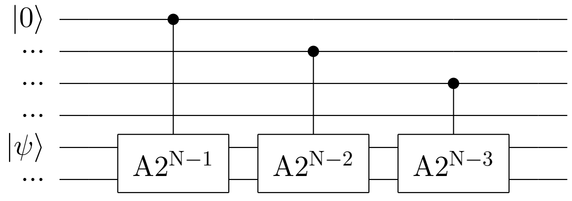


Figure 7.15: Proof

It should be noted that in the future, we may be able to run Shor with 256-bit numbers and break the RSA encryption efficiently in real time. Moreover, with Shor's algorithm we may will be able to break elliptic curve encryption as well. Quantum computers do not do brute-force but they tend to be much more efficient than the classical computers on some specific cases. For instance, when we use superposition and entanglement, we get very increased computational power but we can use it only for certain mechanisms. However, we do not know any algorithm that could break the AES encryption that is used in blockchain.

Chapter 8

Quantum Algorithms: Grover's Algorithm

8.1 Introduction

In this section, we have analyzed the Grover's Algorithm. Grover's algorithm provides a polynomial speed-up over the best-known classical algorithms for a wide class of important problems. We will also use this algorithm to solve the Max-Cut Problem.

8.2 Theoretical Analysis

8.2.1 The Search Problem

In order to understand the quantum search algorithm, we have to consider an example because such algorithms perform a generic search for a solution to a wide range of problems. So, for instance, for a large integer N, one can efficiently recognize whether an integer p is a non-trivial factor of N and thus one naive strategy for finding non-trivial factors of N is to simply search through the set $\{2, 3, 4, ..., \lfloor\sqrt{N}\rfloor\}$ until a factor is found. In fact, there are problems where one can efficiently recognize a good solution and wants to search through a list of potential solutions in order to find a good one and typically the number of potential solutions is exponential in the size of the problem instance and thus the naive algorithm is not efficient. In addition, classical search algorithms make almost no use of the structure of the problem, either because they rule out some impossible cases or to prioritize some more like ones, and hence, the overall complexity remains exponential. On the other hand, quantum searching is a useful tool for speeding up these sorts of generic searches through a space of potential solutions.

It is noted that having a means of recognizing a solution to a problem and knowing the set of possible solutions seems like we know the solution but, we cannot always efficiently find the solution (e.g. it is easy to recognize the factors of a number but finding them is hard). To put this problem in a more mathematical form, let us suppose that the solutions are expressible as binary strings of length n and let us also define a function $f : \{0, 1\}^n \to \{0, 1\}$ s.t. $f(x) = 1$ if x is the binary encoding of a "good" string (i.e. a solution to the search problem), and $f(x) = 0$ otherwise. It is important to know that if the function f is only provided as an oracle, then $\Omega(\sqrt{2^n})$ applications of the oracle are necessary in order to solve the Search Problem with high probability for any input, and thus, the quantum algorithms can provide at most a quadratic speed-up over classical exhaustive search.

The Search Problem:
Input: An oracle U_f for computing an unknown function $f : \{0,1\}^n \to \{0,1\}$.
Problem: Find an input $x \in \{0,1\}^n$ such that $f(x) = 1$.

In this analysis, we will examine functions with exactly one solution $x = w$ and we want our procedure to find the solution with probability at least $\frac{2}{3}$ for every such function f.

Now, we will examine what happens with the number of queries we are allowed to make:

- **No query**: The correct value $x = w$ will be outputted with probability $\frac{1}{2^n}$.
- **One query**: The best we can do is to guess a solution x_1 uniformly at random and then use the query to check if $f(x_1) = 1$ and if x_1 is correct, the output x_1, else, we guess x_2 uniformly at random from the set $\{0,1\}^n - \{x_1\}$ and output x_2. In the end, the correct value $x = w$ will be outputted with probability $\frac{2}{2^n}$.
- **Two queries**: The best we can do is to do the same as before and also use a second query to test if $f(x_2) = 1$ and if that is correct, then output x_2, else, we guess x_3 uniformly at random from $\{0,1\}^n - \{x_1, x_2\}$ and output x_3. In the end, the correct value $x = w$ will be outputted with probability $\frac{3}{2^n}$.
- k **queries**: The best we can do is the above procedure for $k < 2^n$ and in the end, the correct value $x = w$ will be outputted with probability $\frac{k+1}{2^n}$.

Now, let us consider a quantum version of the algorithm that makes a guess with no queries. Classically, this correct answer is given with probability $\frac{1}{2^n}$, so we know that the quantum version will achieve this with a probability amplitude of $\frac{1}{\sqrt{2^n}}$. The goal now is to find a quantum way to boost the amplitude by $\frac{1}{\sqrt{2^n}}$ after each query so that we could solve the search problem with only $O(\sqrt{2^n})$ queries. The way that we can make this happen is by using Grover's quantum algorithm.

8.2.2 Grover's algorithm - Introduction

On the one hand, a classical deterministic brute-force search takes $2^n - 1$ queries in the worst case, for the existence of exactly one solution and in general, any classical algorithm, that for any function f finds a solution with probability at least $\frac{2}{3}$ (this is arbitrary, as any constant strictly between $\frac{1}{2}$ and 1 would suffice), must take $\Omega(2^n)$ queries in the worst case. On the other hand, We know that Grover's algorithm performs the search quadratically faster than the classical way, meaning $O(\sqrt{2^n} = O(2^{\frac{n}{2}})$ queries. That is important if we take into consideration that it can give a quadratic speed-up in the solution of NP-complete problems, which account for many important hard problems in Computer Science.

First approach:

First, let us assume that we have a means of recognizing a solution and thus, we can have a quantum oracle U_f for f, without loss of generality: $U_f : |x\rangle |b\rangle \rightarrow |x\rangle |b \oplus f(x)\rangle$. Also, we assume that we set the target register $|b\rangle$ to $|0\rangle$, so, given a query value x encoded in the query register as $|x\rangle$, by making queries we get: $|x\rangle |0\rangle \mapsto_{U_f} |x\rangle |f(x)\rangle$. Then, if we measure the target qubit, we get the answer to the oracle query to f.

However, that is no better than just applying the oracle for f classically. In order to gain an advantage, we have to use quantum superposition.

Second approach:

First, we prepare the first register in a superposition of all possible query values and $N = 2^n$: $\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$. Then, we split the sum into two parts: for the first part, we sum over all x for which $f(x) = 0$. Those are the x that are not solutions to the search problem and we will call them "bad" and the set X_{bad} and for the second part we sum over all x for which $f(x) = 1$. Those are the x that are solutions to the search problem and we will call them "good" and the set X_{good}. Hence, if we assume that there is only one solution w, then $X_{good} = \{w\}$ and the states will be $|\psi_{good}\rangle = |w\rangle$, $|\psi_{bad}\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \in X_{bad}} |x\rangle$. Next, let us suppose that we prepare the target qubit U_f in the state $|0\rangle$ and the query register in a superposition of the form: $\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle = \frac{1}{\sqrt{N}} |w\rangle + \sqrt{\frac{N-1}{N}} |\psi_{bad}\rangle$. Therefore, the oracle U_f will give: $\frac{1}{\sqrt{N}} |w\rangle |1\rangle + \sqrt{\frac{N-1}{N}} |\psi_{bad}\rangle |0\rangle$. Thus, with probability $\frac{1}{N}$ a measurement of the target qubit will give $|1\rangle$ and the query qubits will be left in the "good" state $|w\rangle$.

It is noted that we use quantum superposition but not quantum interference so we can simulate this procedure with **classical randomness**, so it is the same as sampling an input x uniformly at random and computing $f(x)$. However, the quantum search algorithm is an iterative procedure that uses quantum interference to "push" the amplitude of the good state $|w\rangle$ before measuring the query register.

In Deutsch's algorithm, we saw that it we set the query register to some query index $|x\rangle$ and the target qubit to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ for the oracle U_f, we get: $|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \mapsto_{U_f} (-1)^{f(x)} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$.

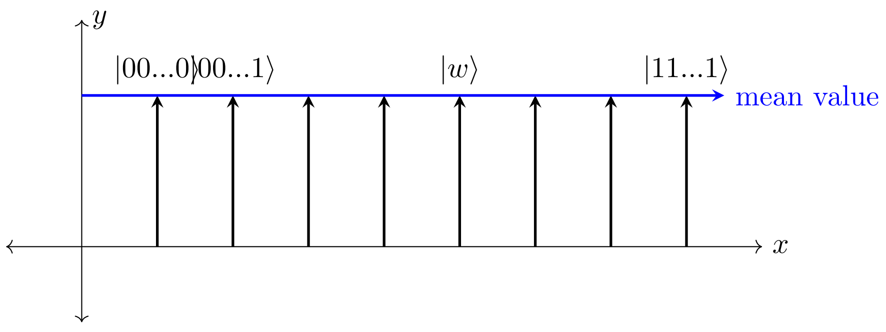
Final approach:

We will redefine U_f to be the n-qubit operator that performs $U_f : |x\rangle \mapsto (-1)^{f(x)} |x\rangle$. Thus, the effect on the first register is to encode the answer to the oracle query in a phase shift. Furthermore, we will define the n-qubit phase shift operator $U_{0^\perp}$ that applies a phase shift of -1 to all n-qubit states orthogonal to the state $|00...0\rangle$: $U_{0^\perp} : \begin{cases} |x\rangle \mapsto -|x\rangle, & x \neq 0 \\ |0\rangle \mapsto |0\rangle, & \text{otherwise} \end{cases}$. In addition, we will define the operator G that increments the amplitude of $|\psi_{good}\rangle = |w\rangle$ and it will be called **Grover iterate** or **quantum search iterate**: $G = HU_{0^\perp}HU_f$. And it can be implemented by applying the oracle U_f, then applying the n-qubit Hadamard gate H, after that, applying $U_{0^\perp}$ and finally, applying the n-qubit Hadamard gate H.

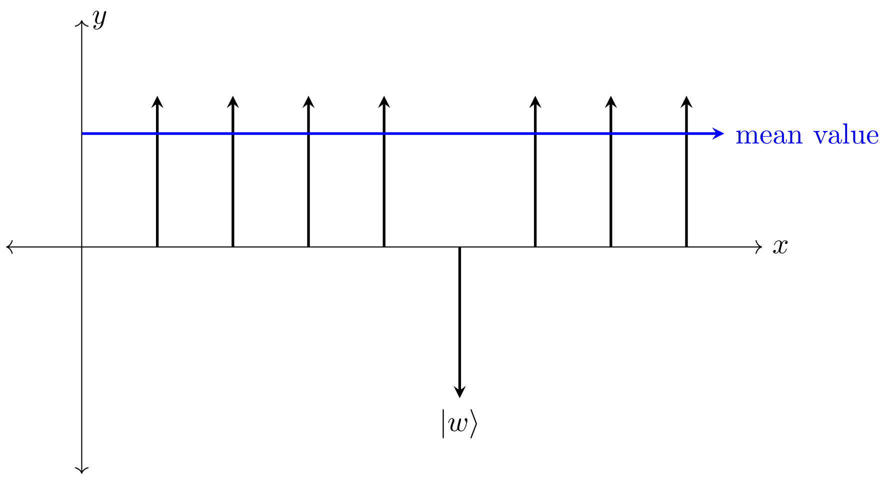
Proof that it actually does what it promises. Let $|\psi\rangle = H|00...0\rangle$, which has only real amplitudes and the mean value is $\frac{1}{\sqrt{N}}$. We apply $HU_{0^\perp}H$ and we get $HU_{0^\perp}H : |\psi\rangle \mapsto |\psi\rangle$. Then, let $V_\psi^\perp$ denote the vector space orthogonal to $|\psi\rangle$ and it is spanned by the state $H|x\rangle\,\forall x \neq 00...0$, and for all such states we get $HU_{0^\perp}H : H|x\rangle \mapsto -H|x\rangle$. Thus, the $HU_{0^\perp}H$ operator applies a phase shift of -1 to vectors in $V_\psi^\perp$ and thus $U_{\psi^\perp} = HU_{0^\perp}H$. Hence, $G = U_{\psi^\perp}U_f$.

Grover's algorithm Visualization:

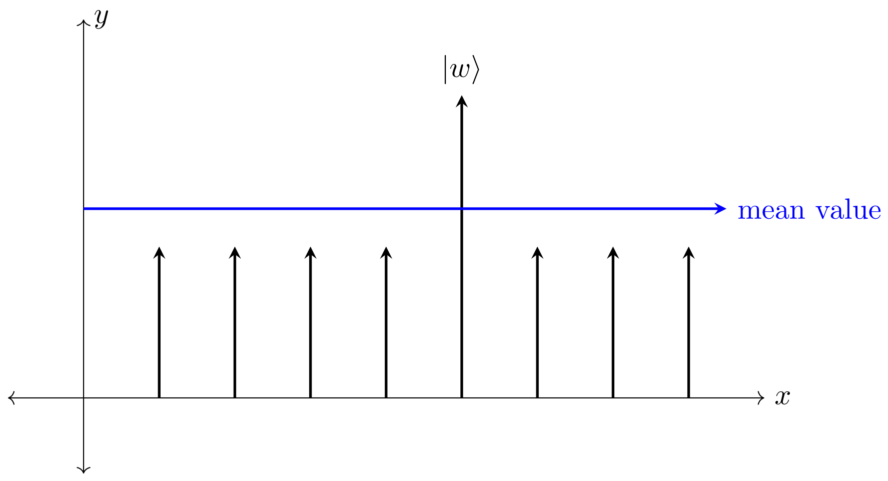
No complex phases where introduced, so the amplitudes will remain real and we can represent them as lines above (for positive) or below (for negative) an axis labelled by the N inputs.



After we apply U_f, the amplitude of $|w\rangle$ picks up a -1 phase shift and thus the mean value of the amplitudes shifts down.



The operator $U_{\psi^\perp}$ can be viewed as an inversion about the mean, which almost triples the size of the amplitude of $|w\rangle$ and pushes up the amplitudes of all the other basis states.



After another application of U_f, the amplitude of $|w\rangle$ becomes negative again and slightly pushes down the mean value of the amplitudes and the inversion about the mean operation adds roughly another $\frac{2}{\sqrt{N}}$ to the size of the amplitude of $|w\rangle$ and slightly pushes down the amplitudes of all the other basis states. Thus, we notice that roughly $\frac{\sqrt{N}}{2}$ iterations of the **Grover iterate** should boost the amplitude of $|w\rangle$ to be close to 1.

In order to understand the last part, we will analyze it separately. First, we write $|\psi\rangle$ in terms of $|w\rangle$ and $|\psi_{bad}\rangle$: $|\psi\rangle = H\,|00...0\rangle = \frac{1}{\sqrt{N}}\,|w\rangle + \sqrt{\frac{N-1}{N}}\,|\psi_{bad}\rangle$. By repeatedly applying U_f and $U_{\psi^\perp}$ the state of the system is left in the subspace spanned by $|w\rangle$ and $|\psi_{bad}\rangle$ and that is a 2-dimensional subspace, with two bases $\{|w\rangle\,, |\psi_{bad}\rangle\}$ and $\{|\psi\rangle\,, |\bar{\psi}\rangle\}$, of the N-dimensional state space. Where, $|\bar{\psi}\rangle$ is orthogonal to $|\psi\rangle$ and the mean value of the amplitudes is 0: $|\bar{\psi}\rangle = \sqrt{\frac{N-1}{N}}\,|w\rangle - \frac{1}{\sqrt{N}}\,|\psi_{bad}\rangle$. Also, for θ, s.t.: $\sin(\theta) = \frac{1}{\sqrt{N}} \quad \cos(\theta) = \sqrt{\frac{N-1}{N}}$, we get $|\psi\rangle = \sin(\theta)\,|w\rangle + \cos(\theta)\,|\psi_{bad}\rangle$, $|\bar{\psi}\rangle = \cos(\theta)\,|w\rangle - \sin(\theta)\,|\psi_{bad}\rangle$, $|w\rangle = \sin(\theta)\,|\psi\rangle + \cos(\theta)\,|\bar{\psi}\rangle$, $|\psi_{bad}\rangle = \cos(\theta)\,|\psi\rangle - \sin(\theta)\,|\bar{\psi}\rangle$.

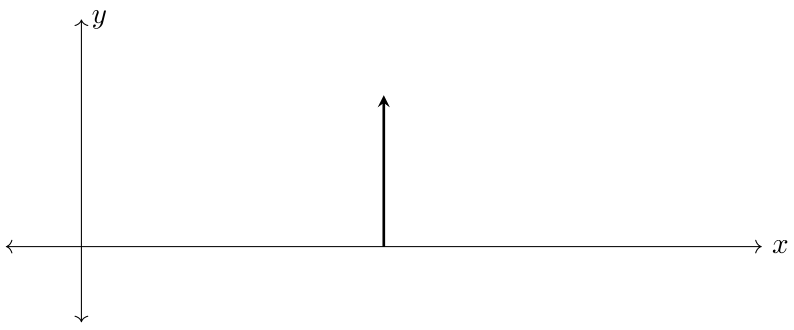


Figure 8.1: $|w\rangle$

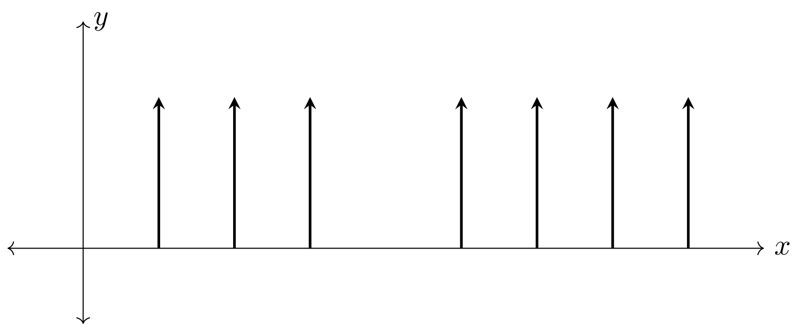


Figure 8.2: $|\psi_{bad}\rangle$

So, the $|\psi\rangle$ is equal to $\sin(\theta)$ times $|w\rangle$ plus $\cos(\theta)$ times $|\psi_{bad}\rangle$.

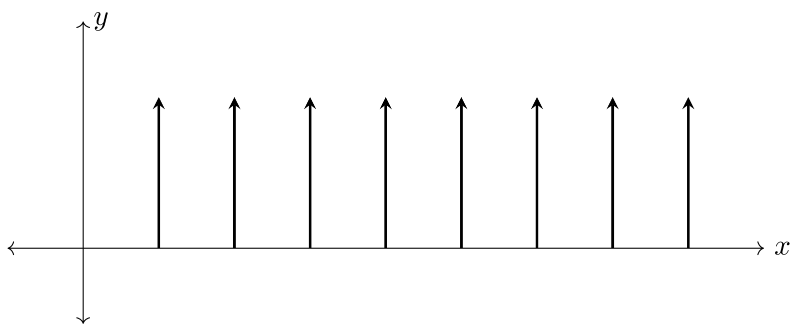


Figure 8.3: $|\psi\rangle$

And the $|\bar{\psi}\rangle$ is equal to $\cos(\theta)$ times $|w\rangle$ minus $\sin(\theta)$ times $|\psi_{bad}\rangle$.

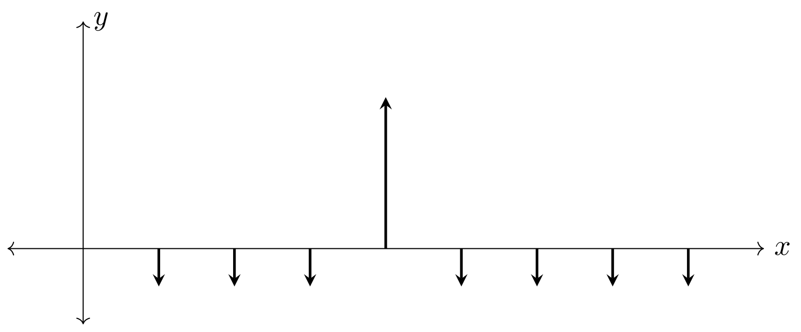


Figure 8.4: $|\bar{\psi}\rangle$

Hence, for the **Quantum Searching Algorithm**, we begin with the state $|\psi\rangle = \sin(\theta)|w\rangle + \cos(\theta)|\psi_{bad}\rangle$. Then, the operator U_f gives the state: $U_f|\psi\rangle = -\sin(\theta)|w\rangle + \cos(\theta)|\psi_{bad}\rangle = \cos(2\theta)|\psi\rangle - \sin(2\theta)|\bar{\psi}\rangle$. So, $U_f|\psi\rangle$ is equal to minus $\sin(\theta)$ times $|w\rangle$ plus $\cos(\theta)$ times $|\psi_{bad}\rangle$, which is also equal to $\cos(2\theta)$ times $|\psi\rangle$ minus $\sin(2\theta)$ times $|\bar{\psi}\rangle$.

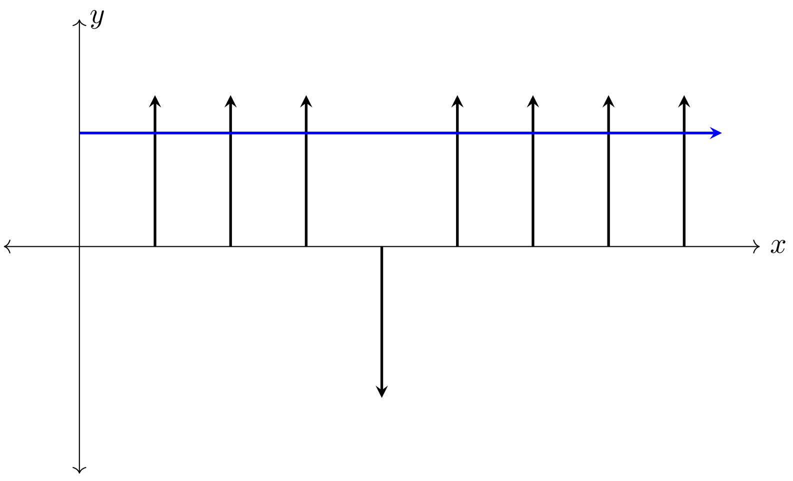


Figure 8.5: $U_f \left|\psi\right\rangle$

Next, the inversion about the mean, $U_{\psi^\perp}$, gives the state: $U_{\psi^\perp} U_f \left|\psi\right\rangle = \cos(2\theta)\left|\psi\right\rangle + \sin(2\theta)\left|\bar{\psi}\right\rangle = \sin(3\theta)\left|w\right\rangle + \cos(3\theta)\left|\psi_{bad}\right\rangle$. And $U_{\psi^\perp} U_f \left|\psi\right\rangle$ is equal to $\cos(2\theta)$ times $\left|\psi\right\rangle$ plus $\sin(2\theta)$ times $\left|\bar{\psi}\right\rangle$, which is also equal to $\sin(3\theta)$ times $\left|w\right\rangle$ plus $\cos(3\theta)$ times $\left|\psi_{bad}\right\rangle$.

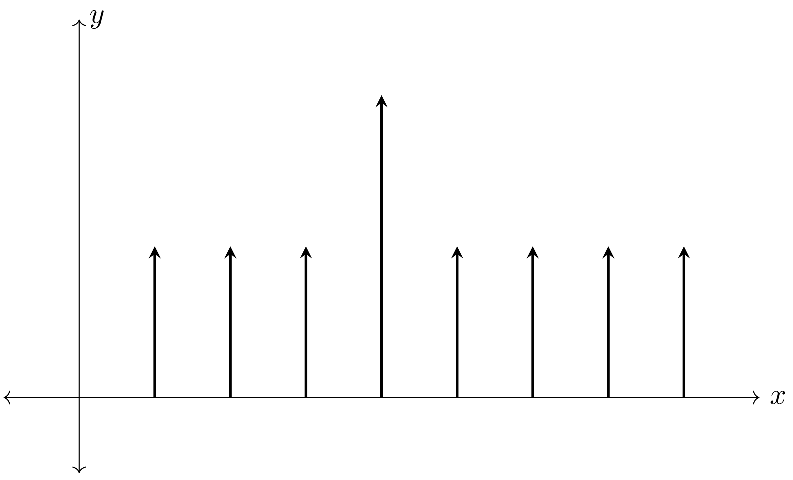


Figure 8.6: $U_{\psi^\perp} U_f \left|\psi\right\rangle$

After k iterations of the **Grover iterate** starting with $\left|\psi\right\rangle = H\left|00...0\right\rangle$, we are left with the state: $(U_{\psi^\perp} U_f)^k \left|\psi\right\rangle = \cos(2k\theta)\left|\psi\right\rangle + \sin(2k\theta)\left|\bar{\psi}\right\rangle = \sin((2k+1)\theta)\left|w\right\rangle + \cos((2k+1)\theta)\left|\psi_{bad}\right\rangle$. And $(U_{\psi^\perp} U_f)^k \left|\psi\right\rangle$ is equal to $\sin((2k+1)\theta)$ times $\left|w\right\rangle$ plus $\cos((2k+1)\theta)$ times $\left|\psi_{bad}\right\rangle$.

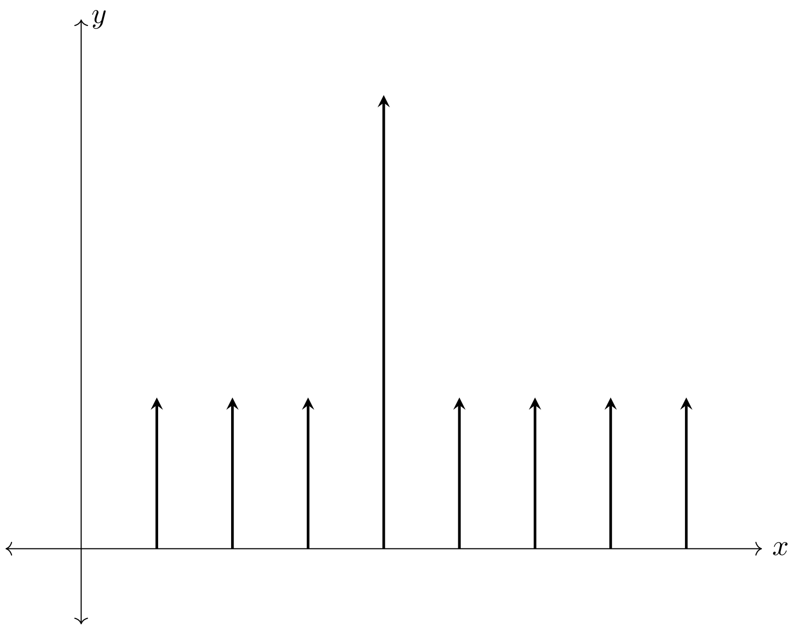


Figure 8.7: $(U_{\psi^\perp} U_f)^k |\psi\rangle$

In order to have a high probability of obtaining $|w\rangle$, we select k s.t. $\sin((2k+1)\theta) \approx 1$ meaning that $(2k+1)\theta \approx \frac{\pi}{2}$ hence $k \approx \frac{\pi}{4\theta} - \frac{1}{2} \approx \frac{\pi}{4}\sqrt{N}$ And if $\tilde{k}$ satisfies $(2\tilde{k}+1)\theta = \frac{\pi}{2}$ and $k = \lfloor \tilde{k} \rfloor$ and also $(2k+1)\theta = \frac{\pi}{2} + \epsilon, |\epsilon| \in O(\frac{1}{\sqrt{N}})$. Then $\sin\left(\frac{\pi}{2} + \epsilon\right) = \cos(\epsilon) \geq 1 - \frac{\epsilon^2}{2} \in 1 - O(\frac{1}{N})$.

Theorem 8.2.1. *Let f be a function with exactly one solution. Let $k = \lfloor \tilde{k} \rfloor$, for $\tilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$. Running the quantum search algorithm with k applications of the Grover iteration will find a solution to $f(x) = 1$ with probability at least $1 - O(\frac{1}{N})$.*

It is noted again that we work with the simplified definition of U_f (target qubit for U_f is omitted):

Grover's Quantum Search Algorithm:

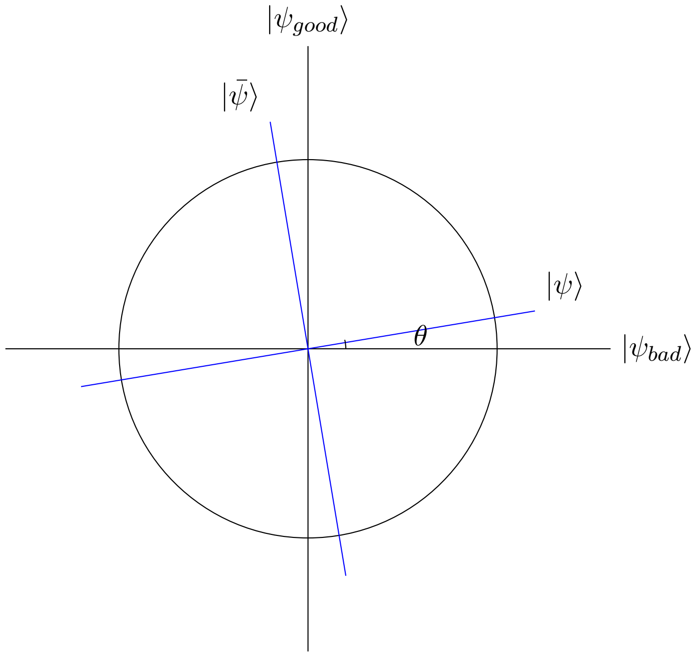
1. Begin with the n-qubit state $|00...0\rangle$ and apply the n-qubit Hadamard gate H to prepare the state $\frac{1}{\sqrt{N}} |\psi\rangle = \sum_{x=0}^{N-1} |x\rangle$.
2. Apply the *Grover iterate* G a total of $\lfloor \frac{\pi}{4} \frac{1}{\sqrt{N}} \rfloor$ times.
3. Measure the resulting state.

8.2.3 Amplitude Amplification

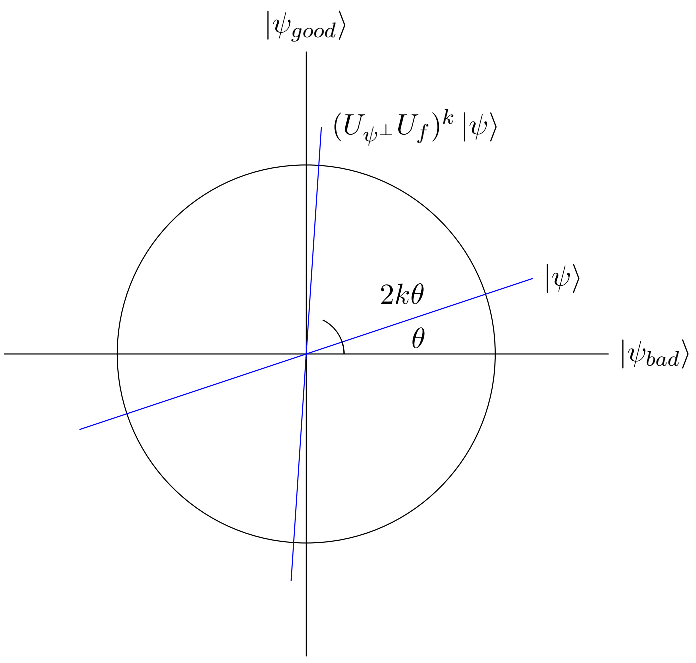
In the previous part, we described the case where we have only one solution but Grover's search algorithm can be generalized to apply to any algorithm A for "guessing" a solution. So, previously, we had $A = H^{\otimes n}$ and we guessed the solution by setting up a uniform superposition of all possible solutions, but now we want to generalize so:

First, we consider an A that starts with a generic input $|00...0\rangle$ and maps to some superposition of guesses $|\psi\rangle$ where we might have some "junk" information left in the workspace qubits: $|\psi\rangle \equiv A|00...0\rangle = \sum_x \alpha_x |x\rangle |junk(x)\rangle$ and $|\psi\rangle = \sum_{x\in X_{good}} \alpha_x |x\rangle |junk(x)\rangle + \sum_{x \in X_{bad}} \alpha_x |x\rangle |junk(x)\rangle$. It is noted that the probability of measuring a good state x and the one of a bad state x are correspondingly: $p_{good} = \sum_{x\in X_{good}} |\alpha_x|^2$, $p_{bad} = 1 - p_{good} = \sum_{x\in X_{bad}} |\alpha_x|^2$. And if $0 < p_{good} < 1$, then we can renormalize the "good" and "bad" components into: $|\psi_{good}\rangle = \sum_{x\in X_{good}} \frac{\alpha_x}{\sqrt{p_{good}}} |x\rangle |junk(x)\rangle$, $|\psi_{bad}\rangle = \sum_{x\in X_{bad}} \frac{\alpha_x}{\sqrt{p_{bad}}} |x\rangle |junk(x)\rangle$. While if $p_{good} = 1$ then no amplification is necessary and if $p_{good} = 0$, then amplification will not help since there is no good amplitude to amplify. So, $|\psi\rangle = \sqrt{p_{good}}\,|\psi_{good}\rangle + \sqrt{p_{bad}}\,|\psi_{bad}\rangle$ or $|\psi\rangle = \sin(\theta)|\psi_{good}\rangle + \cos(\theta)|\psi_{bad}\rangle$, where $\theta \in (0, \frac{\pi}{2})$ satisfies $\sin^2(\theta) = p_{good}$.

Next, we define a more general search iterate to be $Q = AU_{0^\perp}A^{-1}U_f$ which is equivalent to $U_{\psi^\perp}U_f$ and also $U_{\psi^\perp}|\psi\rangle = |\psi\rangle$ and $U_{\psi^\perp}|\phi\rangle = -|\phi\rangle \,\forall\, |\phi\rangle$ orthogonal to $|\psi\rangle$. So, the state before the application of Q is the superposition: $|\psi\rangle = \sin(\theta)|\psi_{good}\rangle + \cos(\theta)|\psi_{bad}\rangle$ and also $|\bar{\psi}\rangle$ which is orthogonal to $|\psi\rangle$ is: $|\bar{\psi}\rangle = \cos(\theta)|\psi_{good}\rangle - \sin(\theta)|\psi_{bad}\rangle$. Thus, we have 2 orthonormal bases for the same 2-dimensional subspace. As a next step, by starting at $|\psi\rangle$ and alternately apply U_f and $U_{\psi^\perp}$ we will end up with a 2-dimensional subspace over the real numbers spanned by $|\psi_{good}\rangle, |\psi_{bad}\rangle$ and since the amplitudes will be real numbers, we can draw the states on the unit circle:



So, U_f performs a reflection about the axis defined by $|\psi_{bad}\rangle$: $\sin(\theta)|\psi_{good}\rangle + \cos(\theta)|\psi_{bad}\rangle \mapsto_{U_f} -\sin(\theta)|\psi_{good}\rangle + \cos(\theta)|\psi_{bad}\rangle$. Then, $U_{\psi^\perp}$ performs a reflection about the axis defined by $|\psi\rangle$, in the basis $\{|\bar{\psi}\rangle, |\psi\rangle\}$: $U_f|\psi\rangle = -\sin(\theta)|\psi_{good}\rangle + \cos(\theta)|\psi_{bad}\rangle = \cos(2\theta)|\psi\rangle - \sin(2\theta)|\bar{\psi}\rangle$ so, $U_{\psi^\perp}U_f|\psi\rangle = U_{\psi^\perp}(-\sin(\theta)|\psi_{good}\rangle + \cos(\theta)|\psi_{bad}\rangle) = \cos(2\theta)|\psi\rangle + \sin(2\theta)|\bar{\psi}\rangle = \sin(3\theta)|\psi_{good}\rangle + \cos(3\theta)|\psi_{bad}\rangle$. Therefore, such reflections lead to a rotation through an angle 2θ in the 2-dimensional subspace and if we do this repeatedly for $Q = U_{\psi^\perp}U_f$ for k times, we get: $Q^k|\psi\rangle = \cos((2k+1)\theta)|\psi_{bad}\rangle + \sin((2k+1)\theta)|\psi_{good}\rangle$.



In conclusion, by applying Q an appropriate number of times, the state is ready and so, when we measure, an element of the subspace will yield spanned by $|\psi_{good}\rangle$ with high probability.

Lastly, we will examine the number of iterations we have to do (complexity) in order to achieve this amplitude amplification. So, in order to get a high probability of measuring a good value, the smallest $k > 0$ will be $(2k+1)\theta \approx \frac{\pi}{2}$, with $k \in \Omega(\frac{1}{\theta})$. We also know that for a small θ, it is true that $\sin(\theta) = \theta$. Here, $\sin(\theta) = \sqrt{p_{good}}$, so searching with amplitude amplification we will use $O\left(\sqrt{\frac{1}{p_{good}}}\right)$ queries to U_f. In contrast, if we assume that $p_{good} \geq \frac{1}{2^n}$, any classical algorithm would need $\Omega(\frac{1}{p_{good}})$ queries to U_f or guesses using the Algorithm A. It is noted that in order to apply this algorithm, we have to know ahead of time k for Q. For A, on the one hand, if it uniformly samples the input, the knowledge of the number of solutions to the search problem is required but, on the other hand, in a more general case, the knowledge of the probability with which A guesses a solution to $f(x) = 1$ is $\sin^2(\theta)$ and it is required. However, it is not always the case that we know in advance the number of solutions but we will see some approaches in the next sections.

8.2.4 Quantum Amplitude Estimation & Quantum Counting

In this section, we will focus on counting the number of solutions that exist. Suppose that we have a search space with N elements, indexed by $\{0,1\}$ and t of them are solutions to $f(x) = 1$ and we want to find t. This is called the **Counting Problem** and it is a special case of **Amplitude Estimation**, which estimates the amplitude with which an n-qubit circuit A maps $|00...0\rangle$ to the subspace of solutions to $f(x) = 1$. For that, suppose that X_{good} is the set of x that are solutions to the search problem and X_{bad} is the set of x that are not solutions to the problem. Also, $|\psi_{good}\rangle = \sum_{x \in X_{good}} \frac{\alpha_x}{\sqrt{p_{good}}} |x\rangle |junk(x)\rangle$, $|\psi_{bad}\rangle = \sum_{x \in X_{bad}} \frac{\alpha_x}{\sqrt{p_{bad}}} |x\rangle |junk(x)\rangle$.

Amplitude Estimation Problem:
Input: The operator A which $A|00...0\rangle = \sin(\theta)|\psi_{good}\rangle + \cos(\theta)|\psi_{bad}\rangle, 0 \leq \theta \leq \frac{\pi}{2}$ and the operator U_f that maps $|\psi_{good}\rangle \mapsto -|\psi_{good}\rangle$ and $|\psi_{bad}\rangle \mapsto |\psi_{bad}\rangle$.
Problem: Estimate $\sin(\theta)$.

And the **Counting Problem** occurs when A is such that $A|00...0\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} e^{i\phi_j}|j\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle$ for any $\phi_j \in [0, 2\pi)$. And, for $0 < t < N$, we get: $|\psi_{good}\rangle = \sum_{j\in X_{good}} \frac{1}{\sqrt{t}}|j\rangle$, $|\psi_{bad}\rangle = \sum_{j\in X_{bad}} \frac{1}{\sqrt{N-t}}|j\rangle$, $A|00...0\rangle = \sqrt{\frac{t}{N}}|\psi_{good}\rangle + \sqrt{\frac{N-t}{N}}|\psi_{bad}\rangle$. Therefore, we have $\sin^2(\theta) = \frac{t}{N}$ and thus, if we estimate $\sin^2(\theta)$ we can estimate t.

First, let us consider any n-qubit circuit A for which $A|00...0\rangle = \sum_{j=0}^{2^n-1} \alpha_j|j\rangle$ and $0 \leq \theta \leq \frac{\pi}{2}$ such that $\sum_{j\in X_{good}} |\alpha_j|^2 = \sin^2(\theta)$, $\sum_{j\in X_{bad}} |\alpha_j|^2 = \cos^2(\theta)$. For $0 < \sin(\theta) < 1$, we get: $A|00...0\rangle = \sin(\theta)|\psi_{good}\rangle + \cos(\theta)|\psi_{bad}\rangle$. Next, we know that Q is described by a rotation matrix: $\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$. So, the two independent eigenvectors with their eigenvalues are: $\begin{bmatrix} \frac{i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ with $e^{i2\theta}$ and $\begin{bmatrix} -\frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$ with $e^{-i2\theta}$. And if we express them in the $\{|\psi_{bad}\rangle, |\psi_{good}\rangle\}$ basis, this means that $|\psi_+\rangle = \frac{1}{\sqrt{2}}|\psi_{bad}\rangle + \frac{i}{\sqrt{2}}|\psi_{good}\rangle \quad |\psi_-\rangle = \frac{1}{\sqrt{2}}|\psi_{bad}\rangle - \frac{i}{\sqrt{2}}|\psi_{good}\rangle$ are the eigenvectors for Q with eigenvalues $e^{i2\theta}$ and $e^{-i2\theta}$, respectively. So, $|\psi\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle = e^{i\theta}\frac{1}{\sqrt{2}}|\psi_+\rangle + e^{-i\theta}\frac{1}{\sqrt{2}}|\psi_-\rangle$. Thus, $|\psi\rangle$ is an equally weighted superposition of eigenvectors for Q with eigenvalues $e^{i2\theta}$ and $e^{-i2\theta}$. In fact, the **Amplitude Estimation** algorithm is based on the application of eigenvalue estimation with the second register in the superposition $|\psi\rangle$ and returns an estimate of either 2θ or -2θ so that we can calculate $\sin^{(\theta)} = \sin^2(-\theta)$. Finally, the circuit outputs an integer $y \in \{0, 1, 2, ..., M-1\}$ where $M = 2^m, m \geq 1$ and the estimate of $p = \sin^2(\theta)$ is $\tilde{p} = \sin^2\left(\pi\frac{y}{M}\right)$.

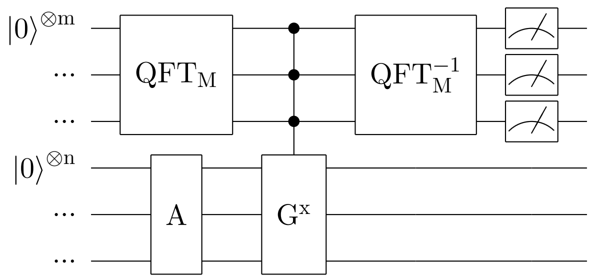


Figure 8.8: Quantum Amplitude Estimation Circuit

Now, if $A|00..0\rangle = |\psi_{good}\rangle$, which means $\theta = \frac{\pi}{2}$, then $QA|00..0\rangle = -A|00...0\rangle$. So, the eigenvalue estimation will estimate $-1 = e^{2\pi i\frac{1}{2}}$ and because M is even, we will get $y = \frac{M}{2}$ with certainty and hence, we will find the correct eigenvalue $-1 = e^{i\pi}$ with certainty. Therefore, the estimate $\tilde{p} = \sin^2(\pi\frac{1}{2}) = 1$ will be exactly correct with certainty. Otherwise, if $A|00..0\rangle = |\psi_{bad}\rangle$, which means $\theta = 0$, then $QA|00..0\rangle = |00...0\rangle$. So, the eigenvalue estimation will estimate 1 and because M is even, we will get $y = 0$ with certainty and hence, we will find the correct eigenvalue $1 = e^{i0}$ with certainty. Therefore, the estimate $\tilde{p} = \sin^2(0) = 0$ will be exactly correct with certainty.

Amplitude Estimation Algorithm:

1. Choose a precision parameter m and let $M = 2^m$.
2. Let $Q = A^{-1}U_{0^\perp}AU_f$.
3. Prepare an m-qubit control register and a second register containing the state $A\left|00...0\right\rangle$.
4. Apply the QFT_M to the first register.
5. Apply a controlled Q^x.
6. Apply the QFT_M^{-1} to the first register.
7. Measure first register to obtain a string representing some integer $y \in \{0, 1, 2, ..., M-1\}$.
8. Output $\sin^2(\pi\frac{y}{M})$.

Theorem 8.2.2. *For any positive integers $k, m, M = 2^m$, the amplitude estimation algorithm outputs $\tilde{p}, 0 \leq \tilde{p} \leq 1$, s.t. $|\tilde{p} - p| \leq 2\pi k\frac{\sqrt{p(1-p)}}{M} + k^2\frac{\pi^2}{M^2}$ with probability at least $\frac{8}{\pi^2}$ when $k = 1$ and with probability greater than $1 - \frac{1}{2(k-1)}$ for $k \geq 2$. If $p = 0$ then $\tilde{p} = 0$ with certainty and if $p = 1$ then $\tilde{p} = 1$ with certainty.*

Lastly, it is mentioned that there are some more algorithms for counting with error $O(\sqrt{t})$, counting with accuracy ϵ and exact counting and also the aspect of searching without knowing the success probability but we will not mention them here.

8.3 Further Analysis

It is known that Grover's algorithm is quadratically faster over classical algorithms, regarding unstructured search problems. This means that it requires a number of operations on the order of the square-root of the number of operations required to solve unstructured search classically.

8.3.1 Unstructured search

Let us assume that $f : \{0, 1\}^n \to \{0, 1\}$ is a function from binary string of length n to bits. We also claim that we can compute f efficiently, else it is arbitrary and we cannot rely on it having a special structure or specific implementation that suits our needs.

With Grover's algorithm, we will search for a string $x \in \{0, 1\}^n$ so that is satisfies $f(x) = 1$ and we will call strings like this as solutions to the searching problem (if there are multiple, then all of them are considered correct output and if there is non, then a correct answer requires that we report that there are no solutions).

Because of the fact that we cannot rely on f having a particular structure, we call this procedure an unstructured search problem and to put it simply, we are looking for "a needle in a haystack" and not a structure that is designed to facilitate searching. Intuitively, it is like having a extremely complicated Boolean circuit that computes f and we can run it easily on a selected input string if we choose but we not able to understand its inner processes.

Classically, we can iterate through all strings $x \in \{0,1\}^n$ and each time compute f and check whether or not it is a solution. So, for $N = 2^n$, there are N strings in $\{0,1\}^n$ and iterating through all of them requires N computations of f. Thus, with a deterministic algorithm, because of the fact that we have limitations and we can only compute f on chosen inputs, that is the best strategy we can follow in order to guarantee success. Also, with a probabilistic algorithm, we can randomly choose input strings to give to f in the hope of saving time but we still require $O(N)$ computations of f in order to have a high probability of success.

However, with a quantum computer, we can run Grover's algorithm and solve the unstructured search problem with high probability and require just $O(\sqrt{N})$ computations of f. Grover's algorithm is also iterative as, it computes f on superpositions of input strings and scatters these computations with other operations and in order to create interference patterns, leading with a high probability to a solution, if there is one, after $O(\sqrt{N})$ iterations.

We can run Grover's algorithm on any function f for which we have a Boolean circuit but, if we have such a circuit for computing f, we can transform it into a quantum circuit by using some number of workspace qubits that start and end the computation in the $|0\rangle$ state. It is mentioned that the computations that follow are done on the standard basis states and in general the actions of the operations are determined by linearity. Thus, by using the query model of computation, we will assume that we have access to the $f : \{0,1\}^n \to \{0,1\}$ through a query gate U_f defined as: $U_f(|a\rangle |x\rangle) = |a \oplus f(x)\rangle |x\rangle, \forall x \in \{0,1\}^n, a \in \{0,1\}$.

The Search Problem:
Input: A function $f : \{0,1\}^n \to \{0,1\}$.
Output: A string $x \in \{0,1\}^n$ satisfying $f(x) = 1$, or "no solution" if there is no such string x.

The Search problem is not a promise problem because f is arbitrary, but we can define a promise variant of it where we are guaranteed that there is exactly one solution as follows:

The Unique Search Problem:
Input: A function $f : \{0,1\}^n \to \{0,1\}$.
Promise: There is exactly one string $z \in \{0,1\}^n$ for which $f(z) = 1$, with $f(x) = 0, \forall x \neq z$.
Output: The string z.

8.3.2 Grover's algorithm

Grover's algorithm makes use of operations known as phase query gates and for a function f they are defined as: $Z_f |x\rangle = (-1)^{f(x)} |x\rangle, \forall x \in \{0,1\}^n$. The Z_f can be implemented using one query gate U_f as follows:

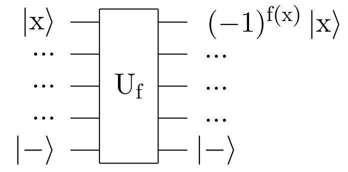


Figure 8.9: Z_f

As we can see, we can use the Phase Kick-back phenomenon and require that one qubit is initialized to $|-\rangle$ and that it remains the same after the implementation has completed so that it can be reused or discarded.

Next, we will need a phase query gate for the n-bit OR function which is defined for $x \in \{0,1\}^n$. The Z_{OR} operator can be implemented as a quantum circuit by beginning with a Boolean circuit for the OR function and then constructing the U_{OR} operation (i.e. a standard query gate for the n-bit OR function) and finally using Phase Kick-back get a Z_{OR} operation, that has no dependence on f. So: $OR(x) = \begin{cases} 0 & , \text{if } x = 0^n \\ 1 & , \text{if } x \neq 0^n \end{cases}$, $Z_{OR} |x\rangle = \begin{cases} |x\rangle & , \text{if } x = 0^n \\ -|x\rangle & , \text{if } x \neq 0^n \end{cases}$. Using the above operations we described, we can write Grover's algorithm. We will refer to a number t, which is the number of iterations it performs and also the number of queries to the f that are required and the choice of this number will be discussed later.

The Grover's algorithm:

1. Initialize an n qubit register Q to the all-zero state $|0^n\rangle$ and then apply a Hadamard operation to each qubit of Q.

2. Apply t times the unitary operation (called Grover operation) $G = H^{\otimes n} Z_{OR} H^{\otimes n} Z_f$ to the register Q.

3. Measure the qubits of Q with respect to standard basis measurements and output the resulting string.

It is noted that in the query model Z_f requires one query whereas Z_{OR} requires no queries and if we have a Boolean circuit for f and convert it to a quantum circuit for Z_f, then the quantum circuit will be larger and more complicated that one for Z_{OR}. So, the Grover operation circuit is this:

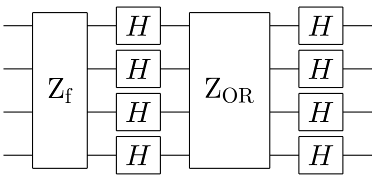


Figure 8.10: Grover operation

And for $t = 2$, a quantum circuit for the entire algorithm is this:

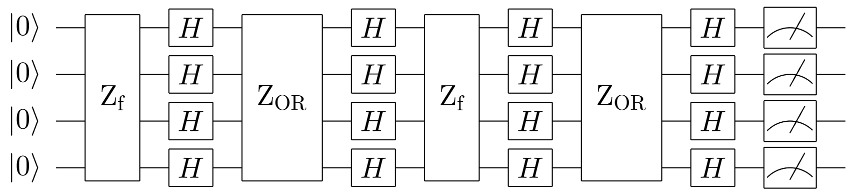


Figure 8.11: Grover quantum circuit for $t = 2$

Therefore, we can apply the Grover's algorithm to the Search Problem by choosing the number t, running Grover's algorithm on f using t to obtain a string $x \in \{0,1\}^n$, querying the f on the x to see it is a valid solution and if $f(x) = 1$, then we found a solution, so output x and stop, else, if $f(x) = 0$, then we either run the procedure again, possibly with a different t, or we output "no solution" and we stop.

Next, we will prove that by taking $t = O(\sqrt{N})$ with high probability we will obtain a solution to our Search Problem, if there is one.

8.3.3 Grover's algorithm analysis

First, we will calculate how the Grover operation G acts on certain states by a symbolic analysis. Let us assume that we have two sets of strings where A_0 contains the strings that are not solutions and A_1 contains all of the solution to our search problem and they satisfy $A_0 \cap A_1 = \varnothing$ and $A_0 \cup A_1 = \{0,1\}^n$, which is a bipartition of $\{0,1\}^n$ and: $A_0 = \{x \in \{0,1\}^n : f(x) = 0\}$, $A_1 = \{x \in \{0,1\}^n : f(x) = 1\}$. Also, we define two unit vectors representing uniform superpositions over the sets A_0, A_1 and each of these vectors is defined only when its corresponding set is nonempty: $|A_0\rangle = \frac{1}{\sqrt{|A_0|}} \sum_{x \in A_0} |x\rangle$, $|A_1\rangle = \frac{1}{\sqrt{|A_1|}} \sum_{x \in A10} |x\rangle$. Here, we will see the case where neither A_0 nor A_1 is empty and we will see the cases for $A_0 = \varnothing$ and $A_1 = \varnothing$ later. Moreover, we define $|u\rangle$ to be a uniform quantum state over all n-bit strings: $|u\rangle = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle = \sqrt{\frac{|A_0|}{N}} |A_0\rangle + \sqrt{\frac{|A_1|}{N}} |A_1\rangle$. And also, it represents the state of the register Q after the initialization in Step 1 of Grover's algorithm, so $|u\rangle = H^{\otimes n} |0^n\rangle$. So, before the iterations of G in Step 2, the state of Q is contained in the 2-dimensional vector space spanned by $|A_0\rangle, |A_1\rangle$ with real number coefficients and that will be the case after any number of iterations of G.

Next, we will focus on Grover operation $G = H^{\otimes n} Z_{OR} H^{\otimes n} Z_f$. We observe that if we replaced the f by a function g which will the composition of f with the NOT function, meaning the function we get by flipping the output bit of f: $g(x) = \neg f(x) = 1 \oplus f(x) = 1 - f(x) = \begin{cases} 1 & \text{, if } f(x) = 0 \\ 0 & \text{, if } f(x) = 1 \end{cases}$. Then, $Z_f = -Z_g$, and if we recall that $Z_f |x\rangle = (-1)^{f(x)} |x\rangle \, \forall x \in \{0,1\}^n$, then $(-1)^{g(x)} = (-1)^{1 \oplus f(x)} = -(-1)^{f(x)}$.

Thus, Grover's algorithm does not change behavior for g since it only need to be able to distinguish solutions and non-solutions to operate. Furthermore, we observe that $Z_f |A_0\rangle = |A_0\rangle$ and $Z_f |A_1\rangle - |A_1\rangle$. In addition, we have the operation $H^{\otimes n} Z_{OR} H^{\otimes n}$ where Z_{OR} is defined for every string $x \in \{0,1\}^n$ as: $Z_{OR} |x\rangle = \begin{cases} |x\rangle & \text{, if } x = 0^n \\ -|x\rangle & \text{, if } x \neq 0^n \end{cases} = 2 |0^n\rangle \langle 0^n| - \mathbb{I}$. Hence, $H^{\otimes n} Z_{OR} H^{\otimes n} = 2 H^{\otimes n} |0^n\rangle \langle 0^n| H^{\otimes n} - \mathbb{I} = 2 |u\rangle \langle u| - \mathbb{I}$. So, if we use the above and also $\langle u|A_0\rangle = \frac{|A_0|}{N} \quad \langle u|A_1\rangle = \frac{|A_1|}{N}$ we get

$$G |A_0\rangle = (2 |u\rangle \langle u| - \mathbb{I}) Z_f |A_0\rangle = (2 |u\rangle \langle u| - \mathbb{I}) |A_0\rangle = 2\sqrt{\frac{|A_0|}{N}} |u\rangle - |A_0\rangle =$$

$$= 2\sqrt{\frac{|A_0|}{N}} \left(\sqrt{\frac{|A_0|}{N}} |A_0\rangle + \sqrt{\frac{|A_1|}{N}} |A_1\rangle \right) - |A_0\rangle = - \left(1 - \frac{2|A_0|}{N} |A_0\rangle + \frac{2\sqrt{|A_0| \cdot |A_1|}}{N} |A_1\rangle \right) =$$

$$= \frac{|A_0| - |A_1|}{N} |A_0\rangle + \frac{2\sqrt{|A_0| \cdot |A_1|}}{N} |A_1\rangle$$

$$G |A_1\rangle = (2 |u\rangle \langle u| - \mathbb{I}) Z_f |A_1\rangle = -(2 |u\rangle \langle u| - \mathbb{I}) |A_1\rangle = -2\sqrt{\frac{|A_1|}{N}} |u\rangle + |A_1\rangle =$$

$$= -2\sqrt{\frac{|A_1|}{N}} \left(\sqrt{\frac{|A_0|}{N}} |A_0\rangle + \sqrt{\frac{|A_1|}{N}} |A_1\rangle \right) + |A_1\rangle = \left(1 - \frac{2|A_1|}{N} |A_1\rangle - \frac{2\sqrt{|A_1| \cdot |A_0|}}{N} |A_0\rangle \right) =$$

$$= \frac{|A_0| - |A_1|}{N} |A_1\rangle - \frac{2\sqrt{|A_1| \cdot |A_0|}}{N} |A_0\rangle$$

We are reminded that the state of Q before Step 2 is contained in the 2-dimensional space spanned by $|A_0\rangle, |A_1\rangle$. Also, we showed that G maps any vector in this space to another vector in the same space, so we can focus solely on this subspace. In fact, if we express G as a matrix M: $M = \begin{bmatrix} \frac{|A_0| - |A_1|}{N} & -\frac{2\sqrt{|A_1| \cdot |A_0|}}{N} \\ \frac{2\sqrt{|A_0| \cdot |A_1|}}{N} & \frac{|A_0| - |A_1|}{N} \end{bmatrix}$. Then, we can see that we can get M by squaring a simpler matrix: $\begin{bmatrix} \sqrt{\frac{A_0}{N}} & -\sqrt{\frac{A_1}{N}} \\ \sqrt{\frac{A_1}{N}} & \sqrt{\frac{A_0}{N}} \end{bmatrix}^2 = M$. And the left part of the equation is a rotation matrix, which we can write as: $\begin{bmatrix} \sqrt{\frac{A_0}{N}} & -\sqrt{\frac{A_1}{N}} \\ \sqrt{\frac{A_1}{N}} & \sqrt{\frac{A_0}{N}} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$ for $\theta = \sin^{-1}\left(\sqrt{\frac{A_1}{N}}\right)$. Thus, we can also write: $M = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}^2 = \begin{bmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \cos(2\theta) \end{bmatrix}$.

So, if we rotate the angle θ twice it is equal to rotating by an angle 2θ, so $\theta = \cos^{-1}\left(\sqrt{\frac{A_0}{N}}\right)$. And from trigonometry we remember that: $\cos(2\theta) = \cos^2(\theta) - \sin^2(\theta)$, $\sin(2\theta) = 2\sin(\theta)\cos(\theta)$. So, the state of register Q at the start of Step 2 is $|u\rangle = \cos(\theta)|A_0\rangle + \sin(\theta)|A_1\rangle$. And the effect of applying G to this state is to rotate it by an angle 2θ within the subspace and in general: $G^t|u\rangle = \cos((2t+1)\theta)|A_0\rangle + \sin((2t+1)\theta)|A_1\rangle$.

Finally, we will focus on the number of iterations. The goal is to find an element $X \in A_1$ and this can be achieved if we obtain the state $|A_1\rangle$ by measuring this state because we are guaranteed to get a measurement outcome $x \in A_1$. So, we should choose t so that $\langle A_1| G^t |u\rangle = \sin((2t+1)\theta)$ is as close to 1 as possible in absolute value in order to maximize the probability to obtain $x \in A_1$ from the measurement. For any angle $\theta \in (0, 2\pi)$, the value $\sin((2t+1)\theta)$ oscillates as t increase, without being necessarily periodic. The following illustration shows the values we obtain for different choices of θ and t.

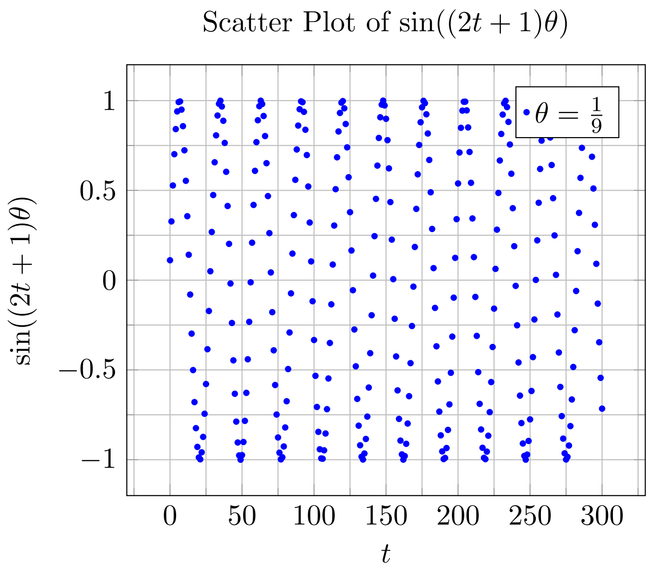


Figure 8.12: Scatter plot of $\sin((2t+1)\theta)$ for $\theta = \frac{1}{9}$ and t from 0 to 300

In fact, expect of making the probability of obtaining an element $x \in A_1$ from the measurement large, we would also like to choose t to be as small as possible, because t applications of the operation G requires t queries to the function f. So, we are aiming to make $\sin((2t+1)\theta)$ close to 1 in absolute value, so we choose t so that: $(2t+1)\theta \approx \frac{\pi}{2}$, $t \approx \frac{\pi}{4\theta} - \frac{1}{2}$. And t must be an integer, so we will not necessarily be able to hit this value exactly and that is why we will take the closest integer to this value, so $t = \lfloor \frac{\pi}{4\theta} \rfloor$. The closeness of this integer to the target value naturally affects the performance of the algorithm. By using $\theta = \sin^{-1}\left(\sqrt{\frac{A_1}{N}}\right)$ we can also see that our estimate $t \approx \frac{\pi}{4\theta} - \frac{1}{2}$ depends on the number of strings in A_1.

- **Case 1**: We will focus on the case where there is a single string x such that $f(x) = 1$ and that is called **Unique Search problem**. In this case: $\theta = \sin^{-1}\left(\sqrt{\frac{1}{N}}\right) \approx \sqrt{\frac{1}{N}}$. If we substitute $\theta = \frac{1}{\sqrt{N}}$ into $t = \lfloor \frac{\pi}{4\theta} \rfloor$ we get $t = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$. Also, if we remember that t is not only the number of times the operation G is performed, but also the number of queries to the function f required by the algorithm, we can see that we are on track to obtaining an algorithm that requires $O(\sqrt{N})$ queries. So, now we have to see how well this t works. For that, the probability that the final measurement results in the unique solution can be expressed explicitly as $p(N,1) = \sin^2((2t+1)\theta)$ where the first argument refers to the number of possible solutions and the second argument refers to the actual number of solutions. By running some experiments for N, we notice that these probabilities are not strictly increasing. Although, for $N = 4$ we get a solution with certainty. In general though: $p(N,1) \geq 1 - \frac{1}{N}, \forall N$. So, the probability of success goes to 1 in the limit as N becomes large and that is good. In fact, even a weak bound such as $p(N,1) \geq \frac{1}{2}$ establishes the utility of Grover's algorithm. Thus, for whatever measurement outcome x we obtain, we can always check to see if $f(x) = 1$ using a single query to f. If we fail to obtain the unique string x for which $f(x) = 1$ with probability at most $\frac{1}{2}$ by running the procedure one, then after m independent runs we will have failed to obtain this unique string x with probability at most 2^{-m}. So, using $O(m\sqrt{N})$ queries to f, we will obtain the unique solution x with probability at least $1 - 2^{-m}$. Finally, using a better bound $p(N,1) \geq 1 - \frac{1}{N}$ shows that the probability of finding $x \in A_1$, using this procedure is actually at least $1 - N^{-m}$.

- **Case 2**: Next, as the number of elements in A_1 varies, so does the angle θ, and that affects the algorithm's probability of success (**multiple solutions**). So, we will write $s = |A_1|$ to denote the number of solutions and $s \geq 1$. For instance, for $s = 4$ solutions, this means that $\theta = \sin^{-1}\left(\sqrt{\frac{4}{N}}\right)$ which is approximately double the angle that we had in unique search when N is large. In this case, if we selected t as in the unique solution setting, then: $t = \left\lfloor \frac{\pi}{4\sin^{-1}\left(\frac{1}{\sqrt{N}}\right)} \right\rfloor$. And that will be a disastrous choice because now the probability of success goes to 0 as N goes to infinity. This is happening because we are rotating twice as fast as we did when there was a unique solution, so we end up zooming past the target $|A_1\rangle$ and landing near $-|A_0\rangle$. Nevertheless, if we use the recommended choice of $t = \left\lfloor \frac{\pi}{4\theta} \right\rfloor$ for $\theta = \sin^{-1}\left(\sqrt{\frac{s}{N}}\right)$, then the performance will be better. In particular, this t will lead to success with high probability. In general, $p(N,s) \geq 1 - \frac{s}{N}$. The lower bound $1 - \frac{s}{N}$ on the probability of success is a little strange because it means that more solutions imply a worse lower bound but under the assumption that s is significantly smaller than N, we conclude that the probability of success is reasonably high and the fact that $p(N,s)$ is reasonably large implies the algorithm's usefulness. Also, it is true that $p(N,s) \geq \frac{s}{N}$ and this lower bound describes the probability that a string $x \in \{0,1\}^n$ selected uniformly at random is a solution, so Grover's algorithm always does at least as well as random guessing and it actually does random guessing when $t = 0$. Thus, for $t = \left\lfloor \frac{\pi}{4\theta} \right\rfloor$ and $\theta = \sin^{-1}\left(\sqrt{\frac{s}{N}}\right)$, we have that for every $a \in [0,1]$, it is the case that $\sin^{-1} a \geq a$, so: $\theta = \sin^{-1}\left(\sqrt{\frac{s}{N}}\right) \geq \sqrt{\frac{s}{N}}$. Hence, $t \leq \frac{\pi}{4\theta} \leq \frac{\pi}{4}\sqrt{\frac{N}{s}}$. So, this translates to a savings in the number of queries as s grows and that gives us $O\left(\sqrt{\frac{N}{s}}\right)$ queries.

- **Case 3**: Moreover, we will examine the case where there is an **unknown number of solutions** $s = |A_1|$. In this scenario, since we do not know s to inform our choice of t, we need a different approach. One way to go is to choose $t \in \left\{1, ..., \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor \right\}$ uniformly at random. By making such a choice for t, we find a solution, if there is one, with probability greater than 40%. It is logical since, the state of Q is being rotated a random number of times, and this will give us a vector for which the coefficient of $|A_1\rangle$ is large and if we repeat this procedure and check the outcome in the same way as described before, the probability to find a solution can be made almost certain (very close to 1). What is more, there is a refined method that finds a solution, when one exists, using $O\left(\sqrt{\frac{N}{s}}\right)$ queries, even when the number of solutions s is not known and it requires $O(\sqrt{N})$ queries to determine that there are no solutions when $s = 0$. The intuition behind this is that we select t uniformly at random from $\{1, ..., T\}$, starting from $T = 1$ and increasing it exponentially until a solution is found, where we terminate the process, or cap T so as not to waste queries when there is not a solution. In addition, to balance the rate of growth of T with the probability of success for each iteration we take $T \leftarrow \left\lceil \frac{5}{4}T \right\rceil$.

- **Case 4**: Lastly, we will refer to some **trivial cases**. We will see what happens when one of A_0, A_1 sets is empty. If every string $x \in \{0,1\}^n$ is a solution, then we will see a solution when we measure and when there are no solutions, we will not get one. However, if we run the mathematics for these trivial cases, we will see that the situation where one of A_0, A_1 is empty occurs when f is constant. In particular, A_1 is empty when $f(x) = 0, \forall x \in \{0,1\}^n$ and A_0 is empty when $f(x) = 1 \forall x \in \{0,1\}^n$, meaning that $Z_f |u\rangle = \pm |u\rangle$, so $G|u\rangle = (2|u\rangle\langle u| - \mathbb{I})Z_f|u\rangle = \pm(2|u\rangle\langle u| - \mathbb{I})|u\rangle = \pm|u\rangle$. Therefore, regardless of the number of iterations t we perform, the measurements will always reveal a uniform random string $x \in \{0,1\}^n$.

It is noted that within the query model, Grover's algorithm is **asymptotically optimal**, meaning that it is not possible to come up with a query algorithm for solving the *Search problem* or the *Unique Search problem*, that uses asymptotically less than $O(\sqrt{N}))$ queries in the worse case so even with this algorithm we have matched an already-known lower bound. However, Grover's algorithm is widely applicable and that is what we will discuss in the next section.

Implementation:

For the development of this algorithm there is an implementation and an application in this **link** and also an implementation using Qiskit for 2 and 3 qubits can be found in this **link**.

8.4 The Max-Cut Problem

8.4.1 Analysis for solving the Max-Cut Problem with Grover's Search algorithm

The Search problem

- **The algorithm**: Let us imagine that we have a database of N numbers and we want one element that is marked. Also, we have an oracle that knows which element is marked and we can make queries asking if the the element that we chose is the one that is marked. So, we send the element to the oracle and the oracle checks it and answers back to us and if the answer is "No", then we send another element and if the answer is "Yes" then we have found it and we stop. This algorithm requires approximately $\frac{N}{2}$ queries.

- **The classical Circuit**: Each record can be represented by a binary string $b_0...b_k$. We can start we 00...0 and by using NOT gates where necessary, we can apply our query to the oracle and it will give us an answer, 0 if it is not marked and 1 if the number is marked.

- **The Quantum Circuit - Intuition**: In order to achieve a similar calculation using a quantum computer we have to use qubits instead of bits and X-gate instead of NOT and add an extra qubit to store our answer and finally make a measurement. Thus, that is not correct because we do not get an answer and we still need to generate every possible number and check it at the end and that is why we have to use a quantum property called superposition.

  Using superposition, we can prepare all the combinations of our numbers, process them through our oracle and then one element, which will be our solution, will have the value of 1 in the answer qubit. However, again we did not gain anything useful because the probability that after the measurement we will get this state is $p - \frac{1}{N}$ and seems like a random query.

  Thus, in order to do better, we have to use the Hadamard gate as well.

Grover's algorithm brief revision

We discussed this thoroughly in the previous part, but we will briefly examine it here again. If we could draw a circuit, from left to right, we would begin first with our qubits $|0\rangle$ and our helper qubit $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Then, we would apply the Hadamard gate to the qubits, except the helper one. Next, we would would apply them to the oracle and the Grover iterate operator for k times and at the end we would make a measurement and hopefully we will get our answer.

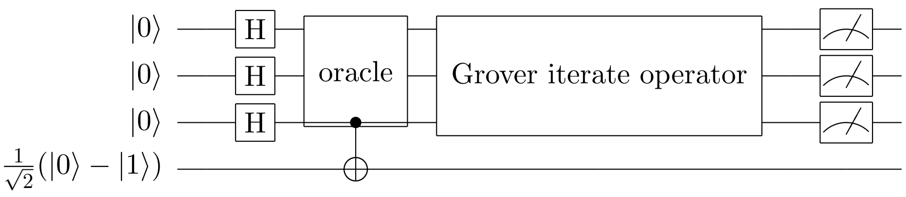
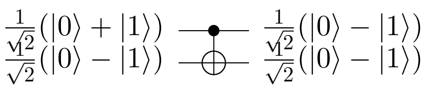


Figure 8.13: Grover algorithm circuit

And it turns out that for $k \approx \frac{\pi}{4}\sqrt{N}$, for N element database, we can find our solution with probability almost one.

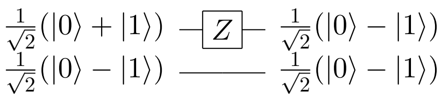
Step by step design

- Firstly, in order to get the helper qubit $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, we have to start with $|0\rangle$ and apply X-gate and then the H-gate.
- Next, the oracle applies NOT on the last qubit and it is how it marks that we are interested in this element. However, we do not measure this qubit and thus we get "phase kick-back". Remember that we are basically changing the local phase of the qubits.



$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

(a) Phase kick-back ($CNOT$)



$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
Z
$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

(b) Phase kick-back (Z)

- After that, we will see the gates of the Oracle and the Grover iterate operator. But first, some definitions:

  Definition 8.4.1 (Decomposition)**.** *A quantum circuit is decomposed if it consists only of single-qubit gates U and controlled-NOT gates.*

  Definition 8.4.2 (Toffoli gate: Controlled-Controlled-NOT)**.** *The Toffoli gate, $CCNOT$ gate, has 3-bit inputs and outputs and if the first two bits are both set to 1, it inverts the third bit, otherwise all bits stay the same.*

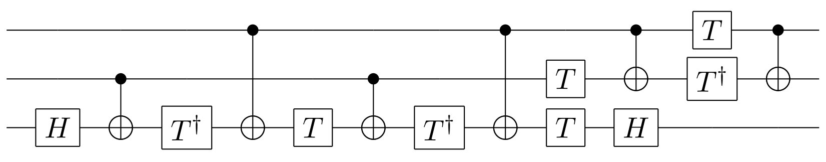


Figure 8.15: Extended Toffoli circuit

Theorem 8.4.3 (n-Toffoli gate)**.** *We can decompose n-Toffoli gate using $O(n)$ controlled-NOTs and $O(n)$ additional qubits, or $O(n^2)$ controlled-NOTs and no additional qubits*

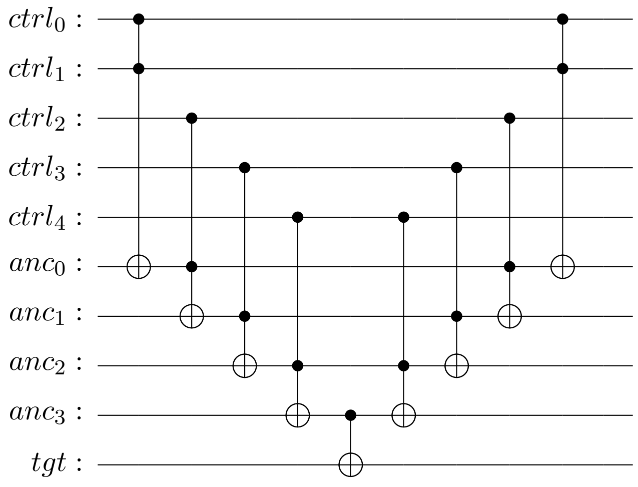


Figure 8.16: n-Toffoli gate

- Therefore, using this decomposition we can continue and design the Grover iterate operator. The circuit will be the following:

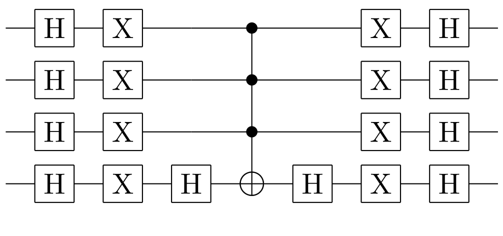


Figure 8.17: Grover iterate operator

- Finally, the implementation of the oracle depends on the problem we are trying to solve and thus, we cannot design a general formula. That is why, for now, we will examine a formula that is designed to solve the Maximum Cut Problem.

We will try to apply the Grover's algorithm to the known problem of Max-Cut and see if we can get the quadratic speedup that this algorithm offers. In order to achieve that, we will make some modifications to the initial Grover's algorithm so that it applies to our needs of the problem. So, the algorithm is the following:

The Grover's algorithm:

1. Let X be an n-qubit quantum register, i.e. a collection of n qubits to which we assign the name X. Let the starting state X be $|0^n\rangle$ and perform $H^{\otimes n}$ on X.
2. Apply to the register X the transformation G for k times, where: $G = -H^{\otimes n} Z_0 H^{\otimes n} Z_f$
3. Measure X and output the result: $Z_f |x\rangle = (-1)^{f(x)} |x\rangle$, $Z_0 |x\rangle = \mathbb{I} - 2|0\rangle\langle 0| = \begin{cases} -|x\rangle & \text{, if } x = 0^n \\ |x\rangle & \text{, if } x \neq 0^n \end{cases}$.

Now, we will focus on some of the steps for deeper understanding. So, regarding the **inversion around the mean**, first, we will see the diffusion operator: $|h\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$

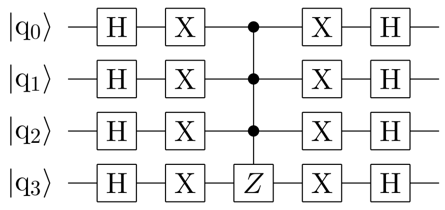
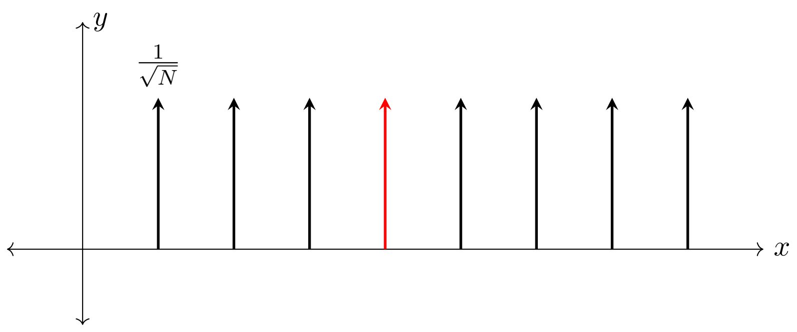
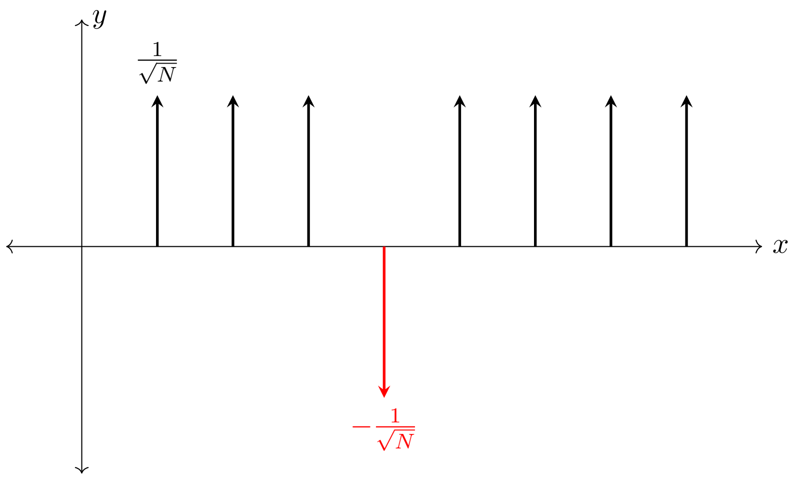


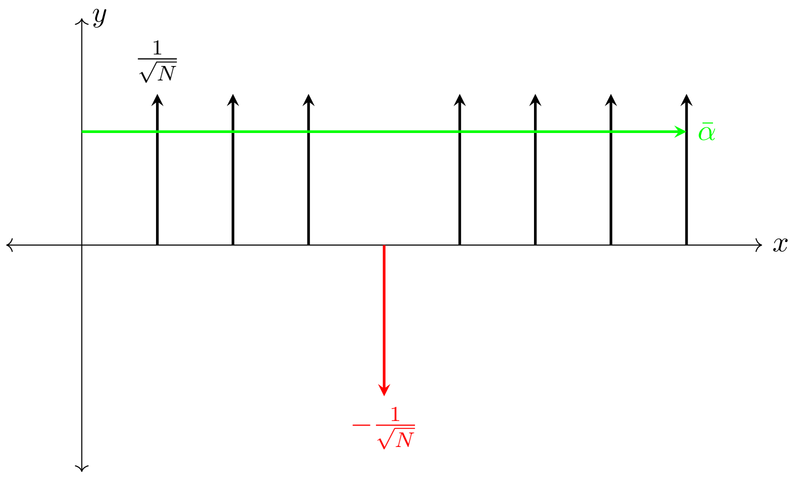
Figure 8.18: Grover diffusion operator

And also, the Grover's operator: $U = -H^{\otimes n} Z_0 H^{\otimes n} = 2|h\rangle\langle h| - \mathbb{I} = \frac{2}{N} \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix} - \mathbb{I}$, where $H^{\otimes n}(\mathbb{I} - 2|0\rangle\langle 0|)H^{\otimes n} = \mathbb{I} - 2|h\rangle\langle h|$ and $G = -H^{\otimes n} Z_0 H^{\otimes n} Z_f = U Z_f$. So, we can $U\left(\sum_x \alpha_x |x\rangle\right) = \sum_x \alpha_x U|x\rangle = \sum_x \alpha_x \left(\frac{2}{N} \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix} - \mathbb{I} \right) |x\rangle = \sum_x (2\mu - \alpha_x)|x\rangle$ where the mean is $\mu = \frac{1}{N}\sum_x \alpha_x$.

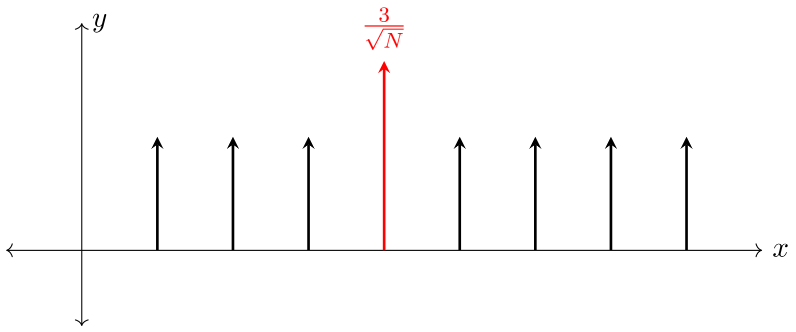
Visualization:







Finally, $2\mu - \alpha_x = 2\left(\frac{1}{\sqrt{N}}\right) - \left(-\frac{1}{\sqrt{N}}\right) = \frac{3}{\sqrt{N}}$.



And if we continue, we will get $\frac{5}{\sqrt{N}}$ and so on.

Graph Theory

We will now analyze briefly the mathematical background and the notation needed before continuing with the Max-Cut Problem. First, we will define an undirected graph:

Definition 8.4.4 (Undirected Graph)**.** *A pair $G = (V, E)$ where $V = \{v_1, ..., v_n\}$ are the vertices and E contains unordered pairs of vertices, i.e., edges $E = \{(v_i, v_j)|v_i, v_j \in V\}$*

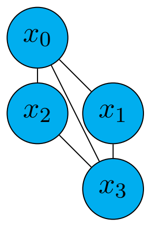


Figure 8.19: Graph example

Definition 8.4.5 (Cut)**.** *A partition of the vertices into two non-empty set S and $R = V - S$.*

For instance, for the Graph example we can have $S = \{0, 1\}$ and $R = \{2, 3\}$.

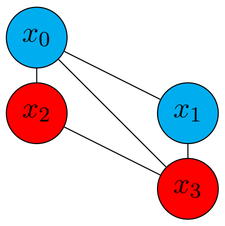


Figure 8.20: Graph example

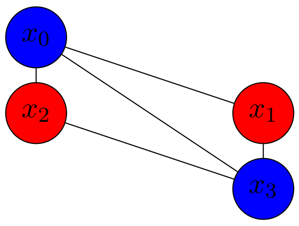
Definition 8.4.6 (Size of a cut)**.** *It is the number of edges between S and R.*

For instance, for the Graph example, we can have size equal to 3.

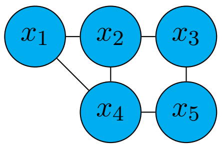
Definition 8.4.7 (Max-Cut)**.** *Let $G = (V, E)$. Find the partition $\{R, S\}$ of V such that the number of* ***cross-connecting*** *edges is maximized. In other words, find partitions that maximize the number of edges across S and R. Also, Max-Cut is NP-Complete.*

Examples:

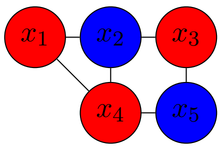
- **Example 1**: For the previous Graph example, we can have a size of cut equal to 4 and we can think of this partition as having colors:



- **Example 2**: Let us suppose we have:



We can identify a cut of size 5 when $R = \{1,3,4\}$ and if we map red to 0, then $f(1) = f(3) = f(4) = 0$ and $S = \{2,5\}$ and if we map blue to 1, then $f(2) = f(5) = 1$.



- **Example 3**: Suppose that we have a graph:

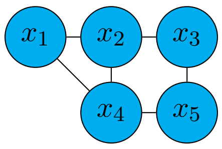


Figure 8.21: Graph example

We want to have two partitions of the vertices. We have colored them in red and green, and we are counting edges that connect vertices with different colors, as shown below:

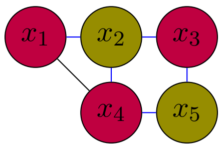


Figure 8.22: number of blue edges is 5

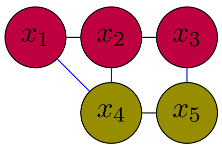


Figure 8.23: number of blue edges is 3

Thus, the **question** that occurs is how we can create a partition such that we have as much as possible edges, for a given graph, that connect vertices with different colors.

In order to answer this last question, let us say that we start with a partition, so we have two sets such that each node belongs to one of them and they define our colors. Thus, R $= \{1,3,4\}$ and S $= \{2,5\}$

The next step is to think of it as a function f: $f : \begin{cases} 1 \mapsto ■ \\ 2 \mapsto ■ \\ 3 \mapsto ■ \\ 4 \mapsto ■ \\ 5 \mapsto ■ \end{cases}$. Furthermore, since we only have two colors, we can use two bits instead, so $f : \begin{cases} 1 \mapsto 0 \\ 2 \mapsto 1 \\ 3 \mapsto 0 \\ 4 \mapsto 0 \\ 5 \mapsto 1 \end{cases}$. And thus, we can get rid of the colors. Also, since we are searching for a bit string which is actually a number, we can use it in our Grover's search.

Definition 8.4.8 (Bipartite Graphs)**.** *A graph G is bipartite if a set of vertices V can be partitioned/colored into two disjoint sets S and R, such that no edge exists between vertices in S and no edge exists between vertices in R.*

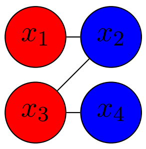


Figure 8.24: Bipartite Graph example

Actually, **bipartite graphs** admit two coloring, one color for each set. So, for that case, the Max-Cut problem is just the total number of edges, since all edges are cross-connecting. So, the size of the cut will be the number of edges. Thus, the new question we should ask is if the graph is bipartite or not. Up to this point, we have identified the Max-Cut problem as an optimization problem and we can convert it into an equivalent decision problem as long as we can accept a logarithmic overhead because of Binary search. So, instead of just asking to find the Max-Cut, the new decision problem will ask, given a G, find if there exists a cut of size at least k and then we can do a Binary search over the size of k in order to find the maximum size for k and that adds an overhead. The Max-Cut problem is known to be NP-Complete because it needs 2^n possible partitions/colorings (because we have two sets/colors) whereas, the generic quantum speedup offered via Grover's algorithm will give us $O(\sqrt{2^n})$. Finally, the Min-Cut problem belongs to P via the max-flow algorithm.

8.4.2 Generic Oracle Development

For this part, we have analyzed the construction of the oracle in two separate ways. In this section we will mention the first way and in the following section we will analyze the second way.

First Oracle Design

So, for the first way, before we start solving the Max-Cut for different instances, we will mention the circuit parts that we will use. The major problem is to figure out the correct way to represent information about the Max-Cut problem in our quantum circuit. It is worth mentioning that it might be easy to map the Traveling Salesman Problem (TSP) as well and in general, with Grover's algorithm we can solve problems like the Sudoku problem, the Triangle-finding problem, etc.

First, regarding the representation of vertices, as we discussed, we will use the following notation: for n vertices, we need n qubits and we define $|0\rangle \rightarrow$ colored blue and $|1\rangle \rightarrow$ colored red. So, for example:

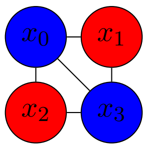


Figure 8.25: Graph example: $|0110\rangle$

Next, regarding the representation of edges, the procedure is called **edge checking**, we will answer the question of whether each end of an edge has vertices with different colors. For that purpose, we will use the following circuit where, $x \oplus y$ is the edge check and gives the answer to whether both vertices have different colors (if $x = y$, then $x \oplus y = 0$, else it is 1). So, for k edges, we need k qubits.

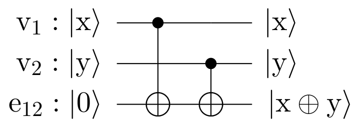
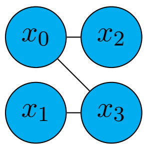

$\mathrm{v}_1 : |\mathrm{x}\rangle$
$\mathrm{v}_2 : |\mathrm{y}\rangle$
$\mathrm{e}_{12} : |0\rangle$
$|\mathrm{x}\rangle$
$|\mathrm{y}\rangle$
$|\mathrm{x} \oplus \mathrm{y}\rangle$

Figure 8.26: Edge Checking Quantum Circuit

Oracle Construction for Bipartite Graphs

So, initially, we will mention the special case of bipartite graphs, such as:



- First, we apply H because we want to consider the uniform superposition of all possible colorings in order to check later if that is a valid coloring or not given the three edges in the graph.

- So, we have 4 qubits for the 4 vertices and 3 qubits for the 3 edges and 1 qubit that will give the answer.
- Next, we are implementing the edge checking that we analyzed above and that is called U. If all these 3 qubits evaluate to 1 (if we have a bipartite, all edges will contribute to the cut).
- Then, we want to check so we apply this Toffoli gate and if all 3 edge qubits are 1 then the answer qubit will be 1.
- Then, we apply the Z gate and this adds a minus in the front. So, if we have identified a valid cut, we will be able to identify this minus sign.
- Finally, we apply the $U^\dagger$ to undo the work and make sure that the qubit are back to the state that they were before this transformation was applied.
- So, the oracle $Z_f \left|x\right\rangle = (-1)^{f(x)} \left|x\right\rangle$ just adds the minus sign in front of the cases for which $f(x) = 1$ and in order to guarantee that nothing else changes, we apply the inverse $U^\dagger$.
- Thus, the net result is a flip of sign for qubits representing correct colorings and this oracle can be used as an oracle for Grover.

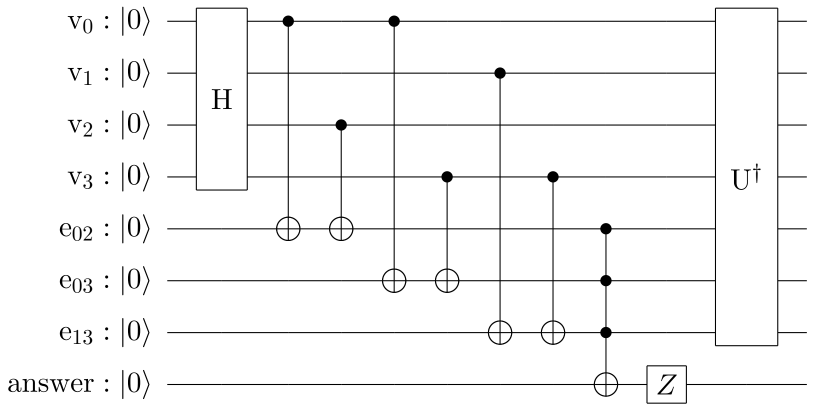


Figure 8.27: Oracle construction

Complexity: Our observation on applying Grover's algorithm for solving the Max-Cut Problem for the special case of finding bipartite graphs did not turn out to be better than the best classical algorithm that checks if a graph is bipartite. In particular, the classical algorithm accomplishes that in $O(n^2)$ and the quantum algorithm we developed will solve that problem in $O(\sqrt{2^n})$.

Oracle Construction for General Graphs:

Now, for the general case of an undirected graph that is not necessarily bipartite we apply the following steps:

- First, we apply the edge checking for each edge.
- Then, we sum the outputs of edge checking. For instance, let us imagine we have a graph with 3 edge checking qubits but this can be generalized to adding n qubits.

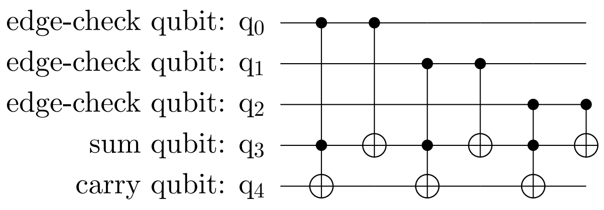


Figure 8.28: addition construction

- After that, we check whether the sum is greater than or equal to k and we store the result. In particular, the number checking procedure will look like a Binary search, since we will ask questions such as "Is $k \geq 4$?" If no, then "Is $k \geq 2$?" etc, else if yes, then "Is $k \geq 6$?" and if no, then "Is $k = 5$?" and if yes, then "Is $k = 7$?". And the circuit that we will need consists of the following parts:
  - "Is there $k \geq 100_2 = 4_{10}$?"

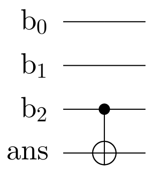


Figure 8.29: $k \geq 100$

  - **No**, then "Is there $k \geq 010_2 = 2_{10}$?" and so on...

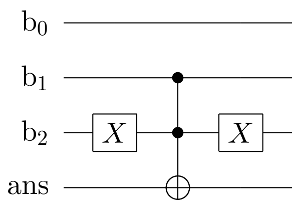


Figure 8.30: $k \geq 010$

  - **Yes**, then "Is there $k \geq 110_2 = 6_{10}$?"

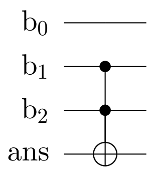


Figure 8.31: $k \geq 110$

- Finally, we apply the Z gate on the result.

Complexity: For each of these steps, we need to run a Grover's algorithm and that if-then case is like a Binary search so the depth of this tree will be logarithmic in the number of edges, so at most $\log(\#\text{ of edges})$. Just a reminder that the classical algorithm can check a graph in $O(2^n)$ and the quantum algorithm we developed will solve that problem in $O(\sqrt{2^n})$ so now we indeed have a quadratic speedup.

Second Oracle Design

Let us see an example of how we can design a method to check whether the vertices have different colors. To achieve this goal, we will use XOR operators. Also, we have 3 qubits, one for V_1, one for V_2 and another one $\rho_{1,2}$ for storing our answer and they have colors $|a\rangle, |b\rangle, |0\rangle$, correspondingly. So, the quantum circuit will be the following:

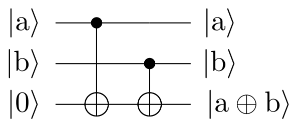


Figure 8.32: Quantum circuit

Explanation: If $|a\rangle$ is 1, then from the first $CNOT$ we get 1, and if $|a\rangle$ is 0, then we get 0, so the answer qubit will be $|a...\rangle$. After applying the second $CNOT$ gate, if $|b\rangle$ is 1, then we have to change the answer again, and if $|b\rangle$ is 0, then we do nothing, so the answer qubit will be $|a \oplus b\rangle$ (XOR gate). Thus, the helper qubit (ancilla qubit) will have the answer whether these two vertices have different colors.

So, now we are ready to start designing our circuit for the following graph instance.

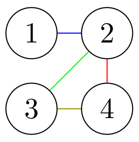


Figure 8.33: Graph

The quantum circuit will be the following:

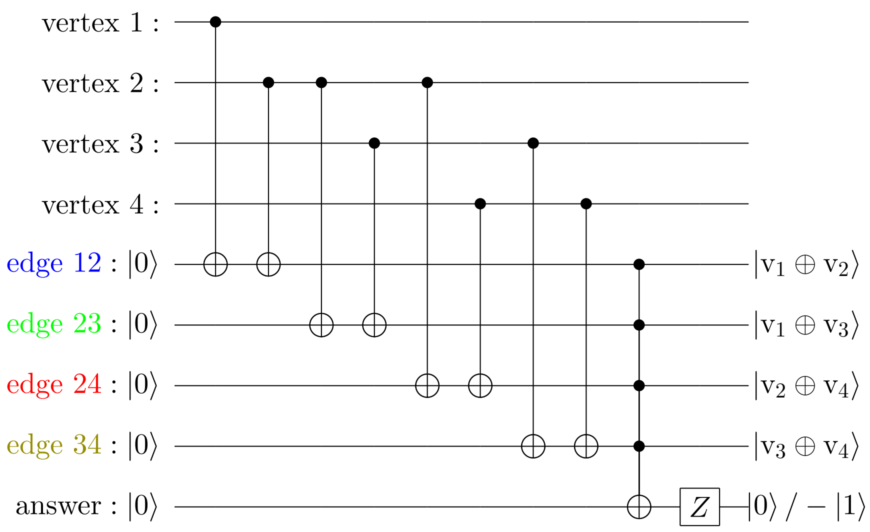


Figure 8.34: Quantum circuit for graph

And we can name the circuit that is left from Z as U. We need to change the representation so that "good" solutions have 1 at the end, hence, we mark the 1 using Z and then we undo the operations. So the complete quantum circuit will be:

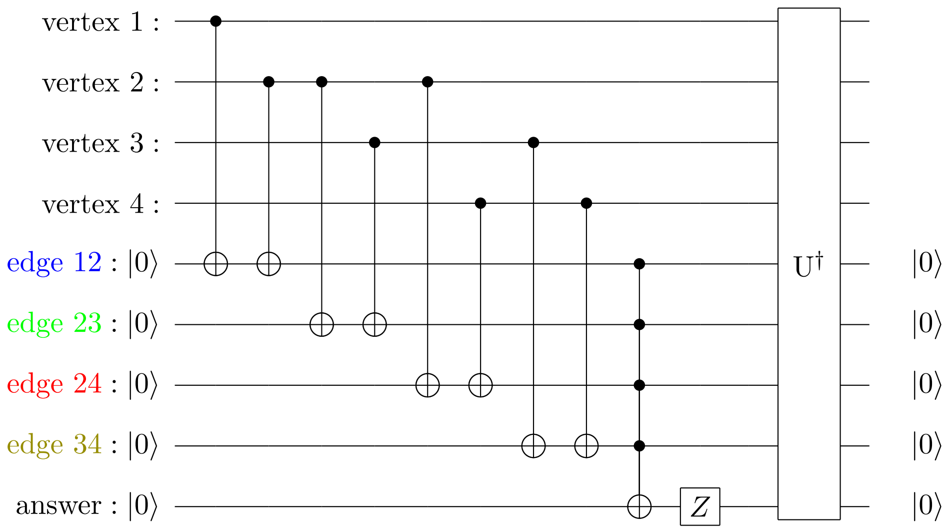


Figure 8.35: Complete Quantum circuit for graph

Explanation: The qubits represent our vertices and the helper qubit will give us the answer. For example, for the blue edge we are applying these two $CNOT$ gates and we store our answer in the first qubit. We repeat this procedure for each edge. So, for each edge we use a separate qubit. Next, we want to check whether we can find a coloring such that every edge connects vertices with different colors. Thus, we use a 4-Toffoli gate and e.g., if edge 12 and edge 23 and edge 24 and edge 34 is 1, then we apply the NOT. Hence, our answer will be 1 if all of the edges connects vertices with different colors, and 0 otherwise. Finally, we need to mark our element which is changing the phase for 1 and we need to undo all of the operations except the Z so that we can go back to our representation (in this way we transfer the minus/local phase to the top). All in all, we have an oracle which checks whether all edges connect vertices with different colors.

However, we are not done yet because this oracle addresses the problem of checking all edges and we do not want that. Luckily, there is another problem that is equivalent and it is called **Bipartiteness checking** which is checking if a graph is bipartite. Remember that a bipartite graph, also called a bigraph, is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent.

Our algorithm has a complexity of $O\left(\sqrt{2^{|V|}}\right)$ while the best classical algorithm has a complexity of $O(|E|) = O(|V|^2)$. So, for now, we have not achieved anything efficient yet. However, this is not the problem that we were trying to solve. In particular, we did not want to check whether all edges can connect vertices with different colors but how many edges can connect vertices with different colors. Therefore, we need a method for counting our edges.

So, the **new procedure** for the oracle that we need to follow is to keep $\lceil \log_2(|E|) \rceil$ qubits for storing the sum of edges (that connect vertices with different colors (the blue ones)) and set them to $|0\rangle$. Then, for each edge, we check the edge (as before) and it will be 0 or 1 and we add 0 or 1 to the qubits representing the sum. After that, we check the number and apply Z gate to mark our element. Finally, we undo everything except Z. So, the overall circuit will look like this:

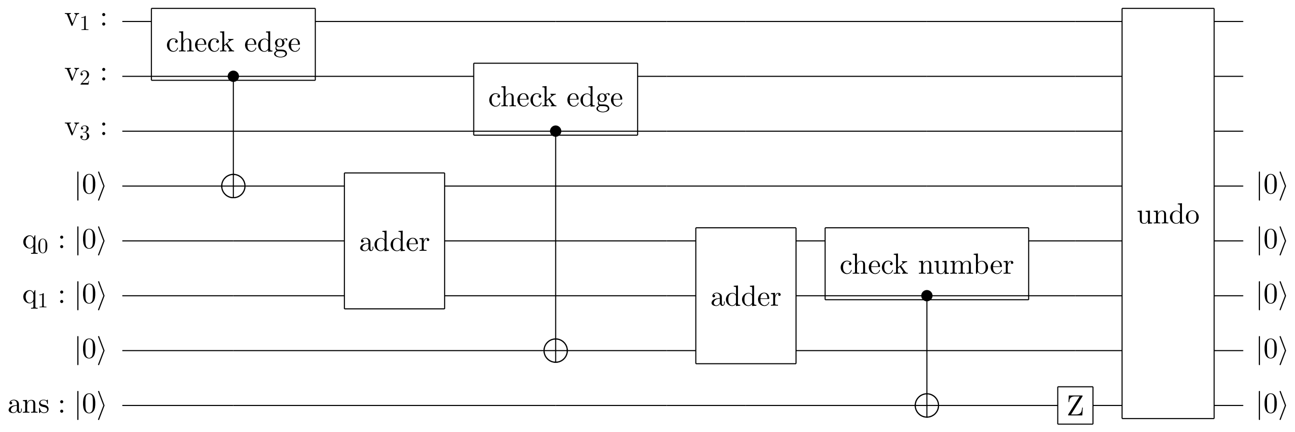


Figure 8.36: Oracle

Explanation: First, we need to use extra qubits for storing the number of edges that are blue and connect vertices with different colors. Then, for each edge, we need to apply the "check edge" and then we need to apply the "adder" so that we update the number that occurs from the edges. For example, if we suppose that one edge connects vertices with different colors, so we get 1 and to this number we apply 1 and if we get 0 then we apply 0. Unfortunately, we cannot check if it is correct because we would have to make a measurement and that is prohibited before our final measurement because it will change our computation. Then, after checking each edge, we need to check our number and apply a NOT on the last qubit for all the numbers that we are interested in. After that, we apply Z so that we mark our element. Finally, we apply "undo", to undo all these operations except the Z.

Now, we need to show what "adder" and "check number" gates are. For the "adder", the circuit is intuitive and it is provided below:

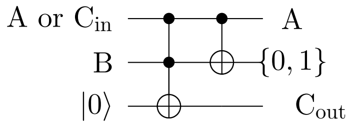


Figure 8.37: Adder B+1

For example,

$$\begin{array}{r}
11 \leftarrow \text{ bits } C_{out}/C_{in} \\
\mathbf{1} \leftarrow \text{ bit } A \\
101\mathbf{1} \leftarrow \text{ bits } B \\
------------ \\
1100 \leftarrow \text{ bits } \{0,1\}
\end{array}$$

Where, here, $A = 1$, $B = 1011$ and $B_0 = 1$ and $\{0, 1\}$ gives the answer. If we have $A = 1$ and $B_0 = 1$ then the $C_{in} = 1$ and it will be transformed into C_{out}. So, for a 3 digit number, it will look like this:

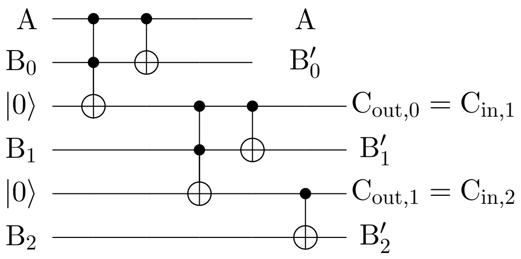


Figure 8.38: 3 digit Adder

Therefore, we created the "adder" circuit that we wanted. Note that for each digit/bit, we will need an extra qubit which will just store our C_{out} and C_{in} values.

Lastly, we need to create the ”check number”. For that, we just want to search over all partitions of the number of edges such that one is from the first set and the other is from the second set. And formally: arg $\max_{R,S} \#\{\rho = (v, w) : v \in R, w \in S\}$. However, we do not know either this partition or the number of these edges. Hence, it is difficult to know which number is interesting. A possible solution to this problem is by applying **Binary search**. Thus, if we assume that there are at most 7 edges, the cases with the circuit each time will be the same as we discussed in the first case.

It is noted that each case requires a separate Grover search algorithm run and it is at most $\lceil \log_2 (|E|) \rceil$ which is small even for a large number of edges.

Explanation: So, checking bits is simply done by applying special Toffoli gates. So, if we want to find $k \geq 4_{10} = 100_2$, we have to check whether there is a number $1\_\_$. If there is, then we have to check if $k \geq 6_{10} = 110_2$ and so we need a number $11\_$. In conclusion, we have all the tools that we need now to make the oracle.

Further Improvements:

Now, let us see from a computational perspective, what capabilities we have in terms of qubits and gates. So, *regarding qubits*, for a graph with n vertices and m edges we need n for the vertices, m for the edges, $\lceil \log_2 (m) \rceil$ for the counting edges and $m(\lceil \log_2 (m) \rceil - 1)$ and 1 for marking the element (e.g. for $n = 5$, $m = 10$, we need 46 qubits). And, *regarding gates*, for a graph with n vertices and m edges we need $2m$ $CNOT$s for each edge (for ”edge check” & reverse), 1 Toffoli and 1 $CNOT$ for each ”adder” for each edge & reverse, 1 $CNOT$ for ”check number” (only 1 bit checked) & reverse and 1 Z gate for marking the element (e.g. for $n = 5$, $m = 10$, we need 86 $CNOT$s and 72 1-qubit gates). Also, the oracle needs to be implemented $\sqrt{2^n}$ times. Hopefully, we can improve it!

First of all, regarding the ”adder”, we do not need the control on the last qubit because we can consider it as moving the answer from the previous to the last qubit to the last qubit. So, we saved 1 $CNOT$ and 1 qubit. Thus, the circuit will be:

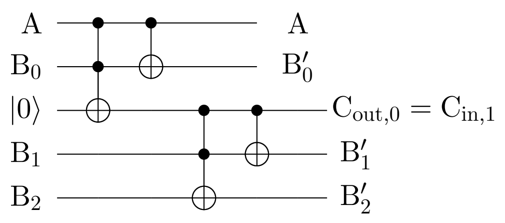
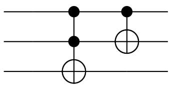
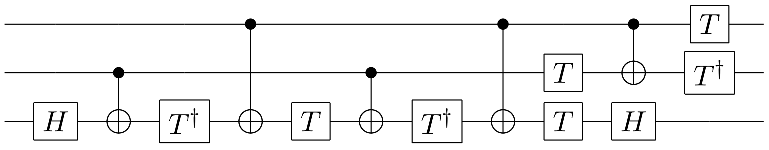


Figure 8.39: Improved 3 digit Adder

Secondly, we can skip some operations when adding numbers with the ”adder”. So, we do not have to use all adders. For instance, for the first edge we are making addition $00...0 + 1$, and we can save a lot of "adders". Thus, when the number is equal to $B_2B_1B_0 = 000_2$ or $B_2B_1B_0 = 001_2$, then the Toffoli gate from $|0\rangle B_1B_2$ is not useful, so we do not need the specific qubit for this addition because again we are just moving forward the state, hence we can remove some more $CNOT$s.

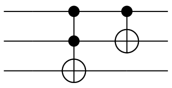
Thirdly, we can simplify our decomposition about addition by removing the last $CNOT$ in the decomposition because it cancels out, as it is shown below:

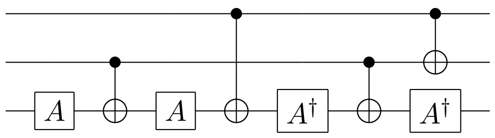

(a) Compact circuit



(b) Extended circuit

Lastly, we will create an alternative Toffoli gate because in fact we do not need to preserve our local phases. So, because of the fact that we apply an inverse at the end of the oracle, we can allow Toffoli to change the local phase ($|b_0b_1b_2\rangle \mapsto e^{i\phi_{b_0b_1b_2}} |c_0c_1c_2\rangle$), so we save another $CNOT$ gate and we can use it for each "adder". Hence,


(a) Compact circuit



(b) Extended circuit

Remarks: It is possible to make some other simple corrections in term of gates but we will not mention them. It is also possible to reduce the number of qubits significantly but it will cost us many additional gates. Grover's search algorithm requires $\frac{\pi}{4}\sqrt{\frac{N}{M}}$, where M is the number of marked solutions (the "good" solutions) but we can find it by using Quantum Counting.

Implementation:

An implementation of Grover's algorithm solving the Max-Cut Problem that we analyzed above can be found in this **link**.

8.4.3 Analysis for solving the Max-Cut Problem with The Ising Model & Quantum Annealing

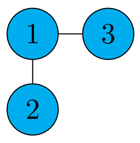
Max-Cut Problem for a Quantum Computer

In the previous sections, we discussed the **Maximum Cut / Max-Cut Problem** which states that we want to divide the vertices of a graph into two sets such that the number of edges with extremes in both sets is the maximum possible. It is proved that this problem is NP-hard (also APX-hard (the class APX (from "approximable") is the set of NP optimization problems that allow polynomial-time approximation algorithms) and thus, there is no classical polynomial-time approximation scheme ($PTAS$) that gets arbitrarily close to the solution (assuming $P \neq NP$)).

Now, we will see how we can apply this problem to a quantum setting by making the following customization. First, we identify each vertex i of the graph with a variable Z_i. Then, we assign value 1 to the vertices of one group and -1 to the others. Next, if E is the set of edges, the problem can be stated as $\min \sum_{(i,j)\in E} Z_i Z_j$ where, the products $Z_i Z_j$ occur from the edges, so if we have an edge that connects vertex i with vertex j, we add one of these terms (1, -1) in this sum.

This is equivalent to solving the Max-Cut Problem because if i and j are in different groups, then they are 1 and -1 or vice versa, hence, their product contributes -1 to the sum, else, if both vertices are in the same group, then they are both 1 or -1, hence, their product contributes 1 to the sum

Example: Let us assume the following graph that has three vertices (1, 2, 3) and two edges (one edge from 1 to 2 and one edge from 1 to 3):



Thus, if we use the previous formula, we need to minimize $H = Z_1 Z_2 + Z_1 Z_3$. Then, for this simple example, if we carefully observe and enumerate the possible *solutions*, it is easy to see the solution to the Max-Cut Problem, which is to put vertex 1 to one group and vertices 2, 3 to another group. Therefore, if we denote the first group with 0 and the second group with 1, and hence, the solution can be written as the *binary string* 011 (and/or 100 if we switch the name of the groups).

So, we managed to quantify the solution to this problem by using binary strings. Now, we are ready to transform the initial problem into something that we can approach with a quantum computer by taking into account that these can be the *states of a qubit register* and use the expression H as the *operators* that we will study in our quantum circuit.

Furthermore, from **Quantum Computing** if we remember that, the operation $\langle 0| Z |0\rangle$

gives: $\langle 0| Z |0\rangle = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} = 1$ and, the operation $\langle 1| Z |1\rangle$ gives: $\langle 1| Z |1\rangle = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix} = -1$, then, to get the value 1 or -1, corresponding to the assignment of the vertex, we will have the following instances: $\langle 01| Z_1Z_2 |01\rangle = (\langle 0| Z_1 |0\rangle)\cdot(\langle 1| Z_2 |1\rangle) = 1\cdot(-1) = -1$ and $\langle 101| Z_1Z_3 |101\rangle = (\langle 1| Z_1 |1\rangle)\cdot(\langle 1| Z_3 |1\rangle) = (-1)\cdot(-1) = 1$.

So, if we return to the Problem, we had the expression H and we identified a possible cut with $|011\rangle$ (vertex 1 in one set and vertices 2 and 3 in another) and now, using the above formulas, we are able to evaluate the cost for the solution will be $\langle 011| H |011\rangle = (\langle 011| (Z_1Z_2 + Z_1Z_3) |011\rangle) = \langle 011| Z_1Z_2 |011\rangle + \langle 011| Z_1Z_3 |011\rangle = -1 + (-1) = -2$ and for another cut $\langle 010| H |010\rangle = (\langle 010| (Z_1Z_2 + Z_1Z_3) |010\rangle) = \langle 010| Z_1Z_2 |010\rangle + \langle 010| Z_1Z_3 |010\rangle = -1 + 1 = 0$ which is not a minimum (we already knew that now, but this is going to help us for more complicated cases with many different vertices).

All in all, we modified the initial problem to a problem of obtaining a basis state (a solution) that makes the value H we calculate, the most possible minimum. This idea is not new, because it comes from another notion that Physicists study, called **Hamiltonians**.

Hamiltonian:

In Quantum Mechanics, the **Hamiltonian** of a system is an operator corresponding to the total energy of that system. A **Hamiltonian** is represented as a Hermitian matrix H (meaning $H = H^\dagger$). So, the *idea* here is that we are interested in finding a basis quantum state $|x\rangle$ such that the expression $\langle x| H |x\rangle$ is minimum. This is called expected energy or expected value of the state with this Hamiltonian and H is the cost function of the Max-Cut problem. This is a particular case of the very important problem of finding the **ground state** or **minimum energy state** of Hamiltonian. The expected energy of a state $|\psi\rangle$ is $\langle\psi| H |\psi\rangle$. Thus, here we have a connection to the possibility of using physical processes in order to solve this kind of problem. We are going to do this in a particular case called **the Ising Model**. This model is a generalization of this expression H.

The Ising Model:

Let us assume that we have n spins (up or down) that interact with their neighbours and we can associate these directions with 1 and -1. The Hamiltonian of such a system is given by: $H = \sum_{1\leq i<j\leq n} J_{ij}Z_iZ_j + \sum_{i=1}^{n} h_iZ_i$ where the coefficients J_{ij} and h_i are real numbers. Our goal is to find a value assignment (1 or -1) that minimizes the sum (the total energy). This problem is NP-hard as well, so it includes the Max-Cut Problem. A different formulation of these combinatorial problems related to this **Ising Model** is the **QUBO** Problem.

Quadratic Unconstrained Binary Optimization (QUBO):

In **QUBO** we substitute the values (1, -1) with the set (0,1). So, the goal is stated as $\min \sum_{1\leq i<j\leq n}^{n} w_{ij}x_ix_j$ where each x_i is a binary variable and coefficients w_{ij} are real numbers. We can transform it into the **Ising Model** by stating: $x_i = \frac{1-z_i}{2}$ and get back to **QUBO** with $z_i = 1 - 2x_i$.

Back to the Ising Model:

So, now, we have all the necessary tools and we want to find the ground state H. A natural approach is to apply H itself to reach the solution since this is a physical process. So, we can try to build some device that mimics this evolution of dynamics of this **Ising Model** and that is what we do when we consider **Adiabatic Quantum Computing**. The idea behind this is the **Adiabatic Theorem** which roughly says that, if we start in the *ground state* of a *initial Hamiltonian* (minimum energy state) and we apply a *time-dependent Hamiltonian* that acts **slowly** on this state, we will get a *new ground state* of the *final Hamiltonian.* In other words, if we do not inject energy to the system, it is not going to be able to jump from a ground state (minimum energy state) to an excited state (a more energetic state).

So, we start with the ground state of a simple *initial Hamiltonian* H_i and we evolve the system to the ground state of the problem *final Hamiltonian* H_f that encodes the problem that we want to solve. To achieve that, we apply the *time-dependent Hamiltonian* for time T as follows: $H(t) = \left(1 - \frac{t}{T}\right) H_i + \frac{t}{T} H_f$. Therefore, if we want to solve the **Ising Model** or a certain instance of the Max-Cut Problem, our H_f will be that instance of the problem that we want to solve and the H_i is going to be a simple Hamiltonian for a situation that we know or that we can easily prepare a state that is in the ground state (that has the minimum energy related to this Hamiltonian). Thus, at the end we will have a H_f that describes our problem and if we do this slowly enough, since we started in a ground state, we will end up in a ground state and this new ground state is going to be the solution to our problem because that Hamiltonian is the one that encoded our problem.

This model of computation is *equivalent* to the usual quantum circuit model we have been studying. Because of the fact that this process of implementing these different Hamiltonians might be **physically challenging** in some situations, we use **Quantum Annealing**. This is the basis of Quantum Computers used by **D-Wave Leap Platform**. The idea is that these Quantum Computers implement these kind of time-dependent Hamiltonians by taking the Hamiltonian H_f of the **Ising Model** because this allows us to encode interesting problems like the Max-Cut or other NP-hard problems. Again, in order to guarantee adiabaticity, T must grow as the inverse of the square of the spectral gap of $H(t)$ (difference between the first and the second energy levels).

Nevertheless, we have now transformed the initial computation of finding the minimum eigenvalue of H_f, to computing many more eigenvalues, because in order to compute this spectral gap, we need to find the ground states and the next excited state for all the values of H. Thus, this problem is even harder to compute. However, in practice, we use **Quantum Annealing**.

Quantum Annealing:

First, we take a simple Hamiltonian $H_i = -\sum_{j=1}^{n} X_j$ with ground state the following superposition that we know how to prepare, because we only need to use Hadamard gates $\sum_{x=0}^{2^n-1} |x\rangle$. Also, the final Hamiltonian H_f is an Ising Hamiltonian that encodes our problem. Regarding time T, we do not care about trying to compute the real time that we need for the whole process to be adiabatic but, we let the system evolve for time T by just trying different values of T. Then, we measure our candidate solution and we check what is the value that we obtained and hopefully, that will be a good approximation of the solution to the problem we are considering. If that is not the case, we repeat the process a number of times and keep the best solution.

We cannot prove that this always gives a good solution or always gives the best solution but it can be useful in some situations in practice. For instance, this has been used in many papers (e.g., [43], [36], [35], [56]).

Implementation:

An implementation of two cases of the **Max-Cut Problem** using the **Ising Model** can be found with the help of Qiskit and Leap Platform from D-Wave in this **link**.

Chapter 9

Conclusions & Future work

9.1 Conclusions

With the emergence of quantum computing, quantum cryptography has garnered escalating interest in recent years. In this study, we delve into fundamental quantum algorithms while also addressing their implementations and applications in pivotal domains such as RSA encryption and the Max-Cut problem within Graph Theory. By scrutinizing these foundational aspects, we aim to contribute to the ongoing discourse surrounding the potential of quantum computing in revolutionizing cryptographic paradigms and addressing graph-theoretic challenges with unparalleled efficacy.

9.2 Future Work

A further research direction would be to explore the use of quantum computing towards potential improvements of exact algorithms for problems that have recently seen significant progress, such as subset sum problems ([13], [4], [42], [1]), and to develop quantum-based methods of exploiting certain complexity-theoretic properties of classes of $\sharp P$-hard problems (see e.g. [5], [3], [6]). Exploring these avenues stands to offer profound insights into the advancement of quantum algorithms and their applicability to pertinent domains such as cryptography and graph theory.

Bibliography

[1] Giannis Alonistiotis, Antonis Antonopoulos, Nikolaos Melissinos, Aris Pagourtzis, Stavros Petsalakis, and Manolis Vasilakis. Approximating subset sum ratio via subset sum computations. In Cristina Bazgan and Henning Fernau, editors, *Combinatorial Algorithms - 33rd International Workshop, IWOCA 2022, Trier, Germany, June 7-9, 2022, Proceedings*, volume 13270 of *Lecture Notes in Computer Science*, pages 73–85. Springer, 2022.

[2] Andris Ambainis, Michele Mosca, Alain Tapp, and Ronald de Wolf. Private quantum channels. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*. IEEE Computer Society, 2000.

[3] Antonis Antonopoulos, Eleni Bakali, Aggeliki Chalki, Aris Pagourtzis, Petros Pantavos, and Stathis Zachos. Completeness, approximability and exponential time results for counting problems with easy decision version. *Theor. Comput. Sci.*, 915:55–73, 2022.

[4] Antonis Antonopoulos, Aris Pagourtzis, Stavros Petsalakis, and Manolis Vasilakis. Faster algorithms for k-subset sum and variations. *J. Comb. Optim.*, 45(1):24, 2023.

[5] Eleni Bakali, Aggeliki Chalki, Sotiris Kanellopoulos, Aris Pagourtzis, and Stathis Zachos. On the power of counting the total number of computation paths of nptms. *CoRR*, abs/2306.11614, 2023.

[6] Evangelos Bampas, Andreas-Nikolas Göbel, Aris Pagourtzis, and Aris Tentes. On the connection between interval size functions and path counting. *Comput. Complex.*, 26(2):421–467, 2017.

[7] Francisco Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241, 1982.

[8] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, nov 1995.

[9] J. S. Bell. On the einstein podolsky rosen paradox. *Physics Physique Fizika*, 1:195–200, Nov 1964.

[10] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM Journal on Computing*, 17(2):210–229, 1988.

[11] Chris Bernhardt. *Quantum Computing for Everyone*. The MIT Press, 03 2019.

[12] P. Oscar Boykin and Vwani Roychowdhury. Optimal encryption of quantum bits. *Phys. Rev. A*, 67:042317, Apr 2003.

[13] Karl Bringmann. Approximating subset sum ratio faster than subset sum. *CoRR*, abs/2310.07595, 2023.

[14] Nicolas Brunner, Daniel Cavalcanti, Stefano Pironio, Valerio Scarani, and Stephanie Wehner. Bell nonlocality. *Rev. Mod. Phys.*, 86:419–478, Apr 2014.

[15] Weng-Long Chang, Renata Wong, Wen-Yu Chung, Yu-Hao Chen, Ju-Chin Chen, and Athanasios V. Vasilakos. Quantum speedup for the maximum cut problem, 2023.

[16] Richard Cleve. An introduction to quantum complexity theory. In *Quantum Computation and Quantum Information Theory*, pages 103–127. WORLD SCIENTIFIC, jan 2001.

[17] Richard Cleve. Lecture notes in quantum information processing - quantum algorithms, October 2022.

[18] S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani. *Algorithms*. McGraw-Hill Higher Education, 2006.

[19] Charles Delorme and Svatopluk Poljak. Laplacian eigenvalues and the maximum cut problem. *Mathematical Programming*, 62:557–574, 1993.

[20] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. What is the computational value of finite-range tunneling? *Physical Review X*, 6(3), 2016.

[21] Jen Palmer Diana Franklin. Introduction to quantum computing for everyone, 2023.

[22] C. Easttom and an O'Reilly Media Company Safari. *Quantum Computing Fundamentals*. Addison-Wesley Professional, 2021.

[23] Marco Tomamichel et al. A monogamy-of-entanglement game with applications to device-independent quantum cryptography. *New Journal of Physics*, page 103002, oct 2013.

[24] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.

[25] Elias Fernandez-Combarro Alvarez. A practical introduction to quantum computing: from qubits to quantum machine learning and beyond. A practical introduction to quantum computing: from qubits to quantum machine learning and beyond, 2020.

[26] Christopher A. Fuchs and Jeroen van de Graaf. Cryptographic distinguishability measures for quantum mechanical states, 1998.

[27] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, page 47–63, New York, NY, USA, 1974. Association for Computing Machinery.

[28] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[29] Martin Grötschel and William R Pulleyblank. Weakly bipartite graphs and the max-cut problem. *Operations research letters*, 1(1):23–27, 1981.

[30] Lov K. Grover. A fast quantum mechanical algorithm for database search, 1996.

[31] C.W. Helstrom. *Quantum Detection and Estimation Theory.* Mathematics in Science and Engineering : a series of monographs and textbooks. Academic Press, 1976.

[32] B. Hensen, H. Bernien, A. Dréau, Andreas Reiserer, Norbert Kalb, M. Blok, J. Ruitenberg, R. Vermeulen, R. Schouten, Carlos Abellan, Waldimar Amaya, V. Pruneri, Morgan Mitchell, M. Markham, Daniel Twitchen, David Elkouss, Stephanie Wehner, Tim Taminiau, and R. Hanson. Loophole-free bell inequality violation using electron spins separated by 1.3 kilometres. *Nature*, 526, 10 2015.

[33] Takahiro Inagaki, Yoshitaka Haribara, Koji Igarashi, Tomohiro Sonobe, Shuhei Tamate, Toshimori Honjo, Alireza Marandi, Peter L McMahon, Takeshi Umeki, Koji Enbutsu, et al. A coherent ising machine for 2000-node optimization problems. *Science*, 354(6312):603–606, 2016.

[34] Siddharth Jain. Solving the traveling salesman problem on the d-wave quantum computer. *Frontiers in Physics*, 9:760783, 11 2021.

[35] Mark W. Johnson, Mohammad H. S. Amin, Gildert Suzanne, Trevor Lanting, Firas Hamze, Neil G. Dickson, Richard G. Harris, Andrew J. Berkley, J. Johansson, Paul I. Bunyk, E. M. Chapple, Colin Enderud, Jeremy P. Hilton, Kamran Karimi, Eric Ladizinsky, Nicolas Ladizinsky, Travis Oh, I. G. Perminov, Chris Rich, Murray C. Thom, Elena Tolkacheva, C. J. S. Truncik, Sergey Uchaikin, J. Wang, Blake A. Wilson, and Geordie Rose. Quantum annealing with manufactured spins. *Nature*, 473:194–198, 2011.

[36] Kirill P. Kalinin and Natalia G. Berloff. Complexity continuum within ising formulation of np problems, 2020.

[37] V. Kasirajan. *Fundamentals of Quantum Computing: Theory and Practice.* Springer International Publishing, 2021.

[38] Helmut G. Katzgraber, Firas Hamze, Zheng Zhu, Andrew J. Ochoa, and H. Munoz-Bauza. Seeking quantum speedup through spin glasses: The good, the bad, and the ugly. *Phys. Rev. X*, 5:031026, Sep 2015.

[39] P. Kaye, R. Laflamme, and M. Mosca. *An Introduction to Quantum Computing.* OUP Oxford, 2007.

[40] D.G. Kelly. *Introduction to Probability.* Macmillan Publishing Company, 1994.

[41] Robert Konig, Renato Renner, and Christian Schaffner. The operational meaning of min- and max-entropy. *IEEE Transactions on Information Theory*, 55(9):4337–4347, sep 2009.

[42] Nikolaos Melissinos, Aris Pagourtzis, and Theofilos Triommatis. Approximation schemes for subset-sums ratio problems. *Theor. Comput. Sci.*, 931:17–30, 2022.

[43] Alex Mott, Joshua Job, Jean-Roch Vlimant, Daniel Lidar, and Maria Spiropulu. Solving a higgs optimization problem with quantum annealing for machine learning. *Nature*, 550:375–379, 10 2017.

[44] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge 0University Press, 2010.

[45] Svatopluk Poljak and Franz Rendl. Solving the max-cut problem using eigenvalues. *Discrete Applied Mathematics*, 62(1-3):249–278, 1995.

[46] Xiaogang Qiang, Thomas Loke, Ashley Montanaro, Kanin Aungskunsiri, Xiaoqi Zhou, Jeremy L O'Brien, Jingbo B Wang, and Jonathan CF Matthews. Efficient quantum walk on a quantum processor. *Nature communications*, 7(1):11511, 2016.

[47] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, 2000.

[48] Jibran Rashid. WOMANIUM QUANTUM.

[49] E.G. Rieffel and W.H. Polak. *Quantum Computing: A Gentle Introduction*. Scientific and Engineering Computation. MIT Press, 2011.

[50] S.M. Ross. *A First Course in Probability*. Pearson Prentice Hall, 2010.

[51] Atıl Samancıoglu. The complete quantum computing course, 2021.

[52] Benjamin Schumacher and Michael Westmoreland. *Quantum Processes Systems, and Information*. Cambridge University Press, 2010.

[53] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.

[54] Thomas Vidick Stephanie Wehner. Quantum cryptography, 2019.

[55] Ben Toner. Monogamy of non-local quantum correlations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 465(2101):59–69, aug 2008.

[56] Sanroku Tsukamoto, Motomu Takatsu, Satoshi Matsubara, and Hirotaka Tamura. An accelerator architecture for combinatorial optimization problems. *Fujitsu Sci. Tech. J*, 53(5):8–13, 2017.

[57] Richard H. Warren. Solving the traveling salesman problem on a quantum annealer. *SN Applied Sciences*, 2(1):75, Dec 2019.

[58] Richard H. Warren. Solving combinatorial problems by two d-wave hybrid solvers: a case study of traveling salesman problems in the tsp library, 2021.

[59] John Watrous. IBM quantum learning, 2023.