

ΥΛΟΠΟΙΗΣΗ ΣΤΟΙΧΕΙΩΔΩΝ ΚΒΑΝΤΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ ΜΕ ΤΗΝ ΠΛΑΤΦΟΡΜΑ QISKIT

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΟΥ

Τουλουμτζή Νικήτα, Π2015023

Επιβλέπων Καθηγητής:
Ανδρόνικος Θεόδωρος



Σχολή Επιστήμης της Πληροφορίας & Πληροφορικής

Ιόνιο Πανεπιστήμιο
Τμήμα Πληροφορικής

Κέρκυρα, Ιούνιος 2022

ΥΛΟΠΟΙΗΣΗ ΣΤΟΙΧΕΙΩΔΩΝ ΚΒΑΝΤΙΚΩΝ ΑΛΓΟΡΙΘΜΩΝ ΜΕ ΤΗΝ ΠΛΑΤΦΟΡΜΑ QISKIT

Η πτυχιακή εργασία εγκρίθηκε από:

Θεόδωρος Ανδρόνικος, Αναπληρωτής Καθηγητής

Παναγιώτης Κουρουθανάσης, Αναπληρωτής
Πρόεδρος Τμήματος Πληροφορικής

Μιχαήλ Στεφανιδάκης, Αναπληρωτής Καθηγητής

Υπεύθυνη Δήλωση

Βεβαιώνω πως είμαι ο συγγραφέας της πτυχιακής εργασίας αυτής, η οποία προετοιμάστηκε από εμένα προσωπικά και εκπονήθηκε για τις απαιτήσεις πάνω στο πρόγραμμα σπουδών του Τμήματος Πληροφορικής του Ιονίου Πανεπιστημίου. Δηλώνω πως κάθε πηγή που χρησιμοποιήθηκε, όπως βιβλιογραφία ή δικτυογραφία, για την υποστήριξη της μελέτης και έρευνάς μου, ακολουθεί τα επιστημονικά πρότυπα, καθώς αναφέρονται ρητά οι αντίστοιχες παραπομπές, σε περίπτωση διεξαγωγής πληροφοριών από τα κείμενα αυτά.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω και να δηλώσω την ευγνωμοσύνη μου σε όλους τους/τις καθηγητές/τριες, που συνέβαλαν στην επιστημονική μου συγκρότηση σε όλα αυτά τα χρόνια φοίτησής μου στο Πανεπιστήμιο. Τέλος, θα ήθελα να ευχαριστήσω τον κ. Ανδρόνικο Θεόδωρο για τον χρόνο και καθοδήγηση που παρείχε για την εκπόνηση της πτυχιακής μου εργασίας.

Περίληψη

Η παρούσα εργασία αποσκοπεί στην υλοποίηση κβαντικών αλγορίθμων, που έπαιξαν καθοριστικό ρόλο στην εξέλιξη του τομέα της κβαντικής υπολογιστικής. Οι αλγόριθμοι αυτοί υλοποιούνται και εκτελούνται από τους διαθέσιμους κβαντικούς υπολογιστές που παρέχει η IBM μέσω της πλατφόρμας Qiskit, χρησιμοποιώντας τεχνολογίες, όπως τα Jupyter Notebooks και Python. Στο κεφάλαιο 1, γίνεται μια εισαγωγή της κβαντικής μηχανικής, αναφέροντας τις βασικές ιδιότητες που την διέπουν, ώστε να προχωρήσουμε στα επόμενα κεφάλαια, για τα πιο πρακτικά ζητήματα του τομέα της κβαντικής υπολογιστικής. Στο κεφάλαιο 2, παίρνουμε μια πρώτη ματιά στις βασικές έννοιες της κβαντικής υπολογιστικής, όπως οι τελεστές, το τανυστικό γινόμενο, καθώς και πολλές άλλες έννοιες. Στο κεφάλαιο 3, γίνεται αναφορά σε μια πληθώρα σημαντικών κβαντικών πυλών και τελεστών. Έπειτα, στο κεφάλαιο 4 εμβαθύνουμε στο τρόπο λειτουργίας και το μοντέλο κβαντικού προγραμματισμού στη πλατφόρμα Qiskit, καθώς και να εκτελούμε τον υπολογισμό πληροφοριών από έναν πραγματικό κβαντικό υπολογιστή. Στο κεφάλαιο 5, περιγράφεται η έννοια της κβαντικής τηλεμεταφοράς, πως λειτουργεί, την αντίστοιχη υλοποίησή της στο Qiskit, όπως και άλλες νέες ορολογίες που βοηθάνε στη σαφή κατανόηση των αλγορίθμων που θα ακολουθήσουν. Στο κεφάλαιο 6, καταλήγουμε στην υλοποίηση στοιχειωδών κβαντικών αλγορίθμων, όπως είναι ο αλγόριθμος του Shor και άλλοι.

Abstract

The present work aims to implement quantum algorithms, which have played a key role in the evolution of the field of quantum computing. These algorithms are implemented and executed by the available quantum computers provided by IBM through the Qiskit platform, using technologies such as Jupyter Notebooks and Python. In Chapter 1, an introduction to quantum mechanics is given, outlining the basic properties that govern it, so that we can move on to the next chapters on the most practical issues in the field of quantum computing. In Chapter 2, we take a first look at the basic concepts of quantum computing, such as operators, the tensor product and many more. In Chapter 3, a number of important quantum gates and operators are mentioned. Next, in Chapter 4, we delve into the operation and model of quantum programming on the Qiskit platform, as well as execute the calculation of information derived from a real quantum computer. Chapter 5 describes the concept of quantum teleportation, how it works, its corresponding implementation in Qiskit, as well as other new terminology that helps understanding of the algorithms that will follow in a clear way. In Chapter 6, we conclude with the implementation of fundamental quantum algorithms, such as the Shor's algorithm and others.

Περιεχόμενα

Εισαγωγή	1
Κεφάλαιο 1 - Αρχές κβαντικής μηχανικής	
1.1. Το πείραμα του Young	2
1.2. Κβαντική υπέρθεση & παρεμβολή	5
1.3. Κβαντική διεμπλοκή	6
Κεφάλαιο 2 - Βασικές έννοιες	
2.1. Εισαγωγή	8
2.2. Διανυσματικές καταστάσεις	9
2.3. Συμβολισμός Dirac	11
2.4. Εσωτερικό & Εξωτερικό γινόμενο	12
2.5. Τελεστές	13
2.6. Τανυστικό γινόμενο	14
Κεφάλαιο 3 - Κβαντικό μοντέλο προγραμματισμού	
3.1. Λογικές πύλες συμβατικών συστημάτων	16
3.2. Λογικές πύλες κβαντικών συστημάτων	19
3.2.1. Η κβαντική πύλη αδρανείας	20
3.2.2. Η κβαντική πύλη περιστροφής φάσης	21
3.2.3. Οι κβαντικές πύλες Pauli	22
3.2.4. Η κβαντική πύλη Hadamard	23
3.2.5. Η κβαντική πύλη CNOT	24
3.2.6. Η κβαντική πύλη CCNOT	26
3.3. Αναπαράσταση ενός κβαντικού κυκλώματος	27
Κεφάλαιο 4 - Η πλατφόρμα Qiskit	
4.1. Εισαγωγή στο Qiskit	29
4.2. Υλοποίηση ολοκληρωμένου κυκλώματος	32
4.3. Προσομοίωση & χρήση κβαντικού υπολογιστή	35
Κεφάλαιο 5 - Κβαντική τηλεμεταφορά	
5.1. Φάση και σφαίρα Bloch	41
5.2. Superdense coding	46
5.3. Ο αλγόριθμος της κβαντικής τηλεμεταφοράς	47
5.4. Υλοποίηση αλγορίθμου στο Qiskit	49
Κεφάλαιο 6 - Κβαντικοί αλγόριθμοι	
6.1. Ο αλγόριθμος Bernstein - Vazirani	51
6.1.1. Αυτοματοποίηση του αλγορίθμου	54
6.2. Ο αλγόριθμος του Deutsch	56
6.2.1. Υλοποίηση αλγορίθμου στο Qiskit	60

6.3. Ο αλγόριθμος του Grover	62
6.4. Ο αλγόριθμος του Shor	67
6.4.1. Ο κβαντικός μετασχηματισμός Fourier	71
6.4.2. Η κβαντική εκτίμηση φάσης	73
6.4.3. Υλοποίηση αλγορίθμου στο Qiskit	75
Βιβλιογραφία	80
Δικτυογραφία	82

Εισαγωγή

Η ιδέα ότι οι υπολογιστές θα μπορούσαν να αξιοποιήσουν την κβαντική μηχανική για να λύσουν συγκεκριμένα προβλήματα είναι σχετικά κάτι καινούργιο, σε σύγκριση με τους κλασσικούς υπολογιστές που γνωρίζουμε έως σήμερα. Αν και δεν υπήρξε αρκετό ενδιαφέρον στην κβαντική μηχανική, πλέον, παίζει έναν σημαντικό ρόλο σε πολλούς τομείς της βιομηχανίας, από την κατασκευή ολοκληρωμένων κυκλωμάτων έως και τη σχεδίαση ακτινών λέιζερ.

Χρονολογικά, το πρώτο κείμενο που συνδέει την κβαντική μηχανική με την κβαντική υπολογιστική ξεκίνησε από τον Paul Benioff^[1]. Συγκεκριμένα, στο κείμενό του περιέγραφε ένα αντίστοιχο κβαντικό μοντέλο μιας Turing μηχανής. Μια Turing μηχανή θεωρείται ένα μαθηματικό μοντέλο υπολογισμού που περιγράφει μια αφηρημένη μηχανή, η οποία χειρίζεται σύμβολα σε μια λωρίδα ταινίας, σύμφωνα με έναν πίνακα κανόνων. Το κβαντικό μοντέλο του Benioff ήταν αντιστρέψιμο και δεν σπαταλούσε ενέργεια^[2]. Έτσι, ενώ εκείνη την εποχή υπήρχαν έρευνες που διαφωνούσαν με τη δημιουργία ενός αντιστρέψιμου κβαντικού μοντέλου, ο Benioff ήταν ο πρώτος που παρουσίασε ότι είναι κάτι θεωρητικά εφικτό. Με τη δουλειά του Benioff, καθώς και άλλων ερευνητών, όπως οι David Deutsch, Richard Feynman και Peter Shor, εισήχθη ο τομέας της κβαντικής υπολογιστικής.

Συνοπτικά, το 1982, ο Richard Feynman δημοσίευσε την έρευνά του, όπου πρότεινε την ιδέα χρήσης ενός κβαντικού υπολογιστή για προσομοίωση των κανόνων της κβαντικής φυσικής, καθώς περιγράφει τις δυσκολίες που εμπλέκονται για να προσομοιωθούν, χρησιμοποιώντας κλασσικά συστήματα. Έτσι, καταλήγει πως θα ήταν μια καλύτερη εναλλακτική η χρήση των κβαντικών υπολογιστών^[3]. Έπειτα, το 1985, ο David Deutsch παρουσίασε τη δική του έρευνα, με το δικό του μοντέλο ενός καθολικού κβαντικού υπολογιστή, ο οποίος είναι ικανός να προσομοιώνει τέλεια κάθε πεπερασμένο κλασσικό σύστημα^[4]. Έτσι, ενώ οι κβαντικοί υπολογιστές βρίσκονταν σε θεωρητικό επίπεδο, αυτό δεν σταμάτησε τους ερευνητές να σχεδιάζουν αλγόριθμους που θα μπορούσαν να εκτελεστούν σε κβαντικό υπολογιστή, για τη λύση ορισμένων προβλημάτων. Ένα από τα διασημότερα παραδείγματα είναι ο αλγόριθμος του Shor, ο οποίος και αναπτύχθηκε το 1994^[5]. Αυτός είναι ένας αλγόριθμος κβαντικού υπολογιστή για την εύρεση των πρώτων παραγόντων ενός ακέραιου αριθμού. Εκτός από τον αλγόριθμο του Shor, υπήρξαν και άλλοι καθοριστικοί αλγόριθμοι, όπως ο αλγόριθμος του Grover, του Deutsch και πολλοί άλλοι.

Μερικοί από τους πιο γνωστούς αλγόριθμους αποτελούν σημείο μελέτης της εργασίας, καθώς έπαιξαν καθοριστικό ρόλο στην ανάπτυξη της κβαντικής υπολογιστικής. Όμως, για να εμβαθύνουμε σε ένα τόσο περίπλοκο και ενδιαφέρον θέμα, είναι αναγκαίο να κατανοηθεί, πρώτα, η ραχοκοκκαλιά του ερευνητικού αυτού τομέα, δηλαδή οι βασικές έννοιες και ιδιότητες της κβαντικής μηχανικής και υπολογιστικής, ώστε να καταλήξουμε στην ανάπτυξη των ζητούμενων αλγορίθμων.

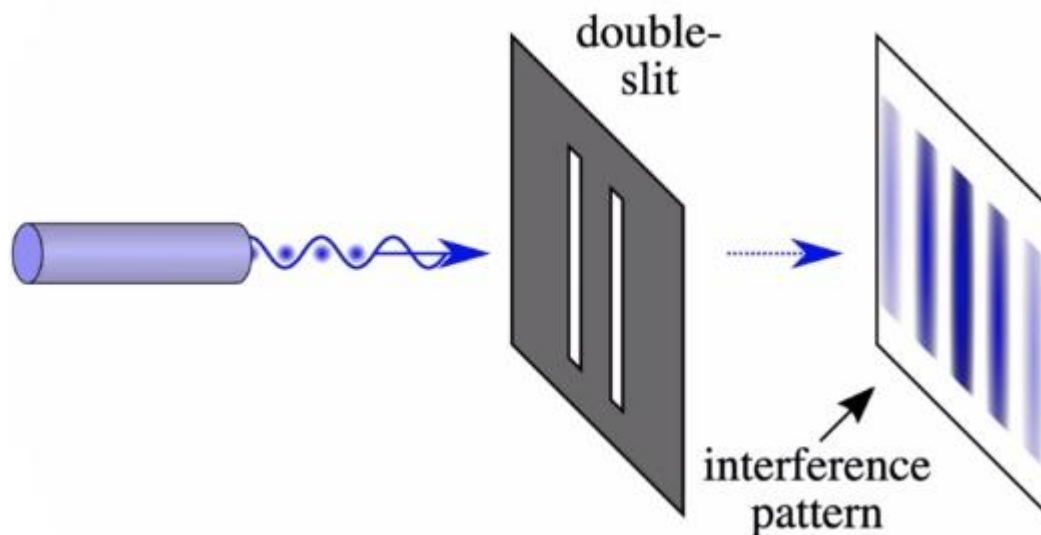
Κεφάλαιο 1

Αρχές κβαντικής μηχανικής

Ξεκινάμε, εισάγοντας μερικές θεμελιώδεις έννοιες από τη κβαντική μηχανική, προτού εμβαθύνουμε στον τομέα της κβαντικής υπολογιστικής και την υλοποίηση των κβαντικών αλγορίθμων.

1.1. Το πείραμα του Young

Το πείραμα του Young ή αλλιώς πείραμα των δύο σχισμών επιδεικνύει πως μικρά σωματίδια, όπως ηλεκτρόνια ή φωτόνια, μπορούν να εκδηλώσουν συμπεριφορά σωματιδίου ή και κύματος. Το πείραμα διεξήχθη από τον Άγγλο φυσικό Thomas Young στις αρχές της δεκαετίας του 1800. Στο πείραμα αυτό, σωματίδια ελαχίστου μεγέθους ύλης ή φωτός εκπέμπονται από μια πηγή, τα οποία διαπερνούν από δύο διατεταγμένες παράλληλες σχισμές πέφτοντας, τελικά, σε μια οθόνη, όπου και αφήνουν το ίχνος τους ως σωματίδια ^[6].

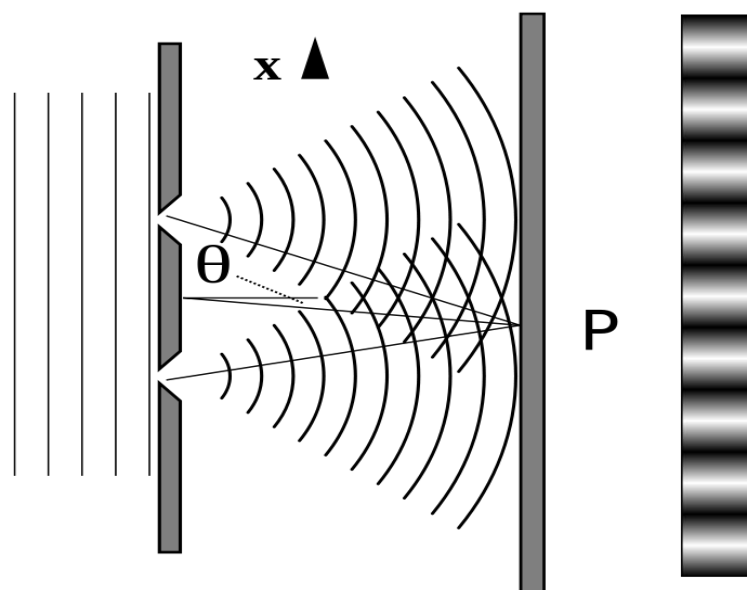


Εικόνα 1.1.1: Επίδειξη πειράματος του Young [Πηγή: Wikipedia]

Ο λόγος που αναφέρεται και είναι πολύ σημαντικό αυτό το πείραμα είναι η προσπάθεια εξήγησης ορισμένων φαινομένων που συμβαίνουν κατά τη διεξαγωγή του, καθώς με αυτό καταφέρνουμε να εξηγήσουμε δύο από τις τρεις βασικές έννοιες

της κβαντικής μηχανικής, την κβαντική υπέρθεση (quantum superposition) και την κβαντική παρεμβολή (quantum interference).

Αρχικά, όταν το πείραμα αυτό διεξάγεται με τη διάταξη μιας σχισμής, το φως διαπερνά τη σχισμή, αποτυπώνοντας στην οθόνη το σχηματισμό του φαινομένου της περίθλασης. Αντιθέτως, όταν περνάει από τη κάρτα με τις δύο σχισμές, πέρα από το φαινόμενο της περίθλασης, παρουσιάζεται μαζί και το φαινόμενο της συμβολής, που πρόκειται για κυματική συμπεριφορά. Με λίγα λόγια, από τη στιγμή που το κύμα περάσει τις σχισμές, θα δημιουργηθεί και μια νέα πηγή κύματος. Όταν αυτά τα δύο κύματα συναντηθούν, τότε στο σημείο που η κορυφή του ενός κύματος θα συμπίπτει με την κορυφή του άλλου, θα ενισχύονται μεταξύ τους, ενώ αντιθέτως εκεί που η κορυφή του ενός θα συμπίπτει με το αντίστοιχο χαμηλότερο σημείο του άλλου, θα εξουδετερώνονται. Έτσι, καταλήγουμε και στην Εικόνα 1.1.2 του φαινομένου της συμβολής (wave interference), με πιο φωτεινά και σκοτεινά πλαίσια στην οθόνη, που θεωρείται και το μοτίβο παρεμβολής. Όμως, ενώ αποδεικνύεται πως τα σωματίδια έχουν κυματική συμπεριφορά, αυτό δεν ισχύει στην πραγματικότητα, μιας

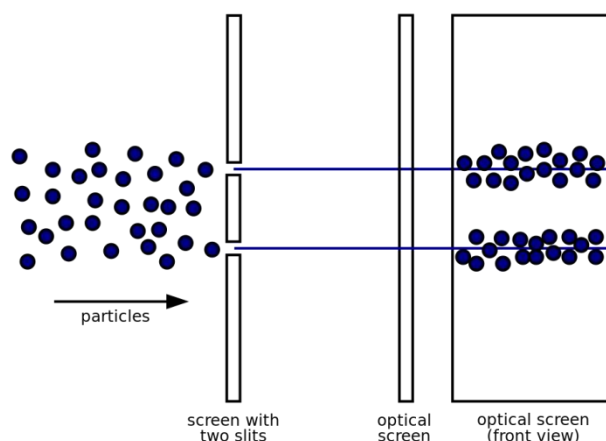


Εικόνα 1.1.2: Αναπαράσταση φαινομένου συμβολής [Πηγή: Wikipedia]

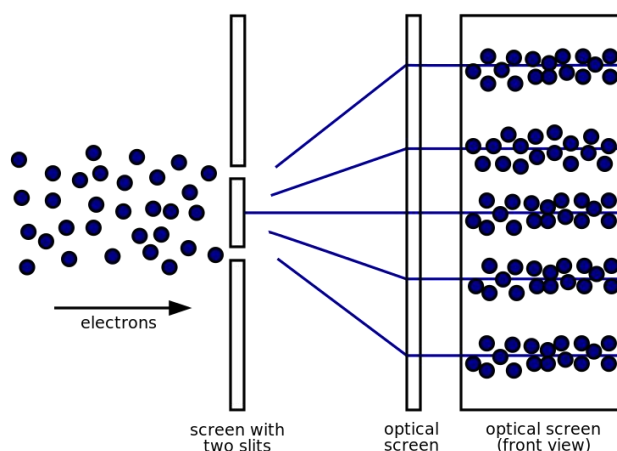
και ξεκινάνε τα παράδοξα του πειράματος. Οι ερευνητές αποφασίζουν να επαναλάβουν το πείραμα, αλλά αυτή τη στιγμή να διέρχεται μόνο ένα σωματίδιο τη φορά από τη διάταξη των δύο γραμμών. Η πρόθεση είναι πως με ένα σωματίδιο τη φορά δεν θα είναι δυνατή η αλληλεπίδραση ή σύγκρουση των κυμάτων, σχηματίζοντας το φαινόμενο της συμβολής, αλλά θα πρέπει να εμφανιστεί στην οθόνη το αποτύπωμα των δύο σχισμών, δηλαδή δύο γραμμές παράλληλες μεταξύ τους. Καταλήγουμε, λοιπόν, στο παράδοξο πως όσο αποτυπώνονται τα ίχνη των σωματιδίων στην οθόνη, εμφανίζεται πάλι το φαινόμενο της συμβολής! Η μόνη

λογική εξήγηση είναι πως το σωματίδιο φαίνεται να περνάει και από τις δύο σχισμές, κάνοντας συμβολή με τον εαυτό του. Αυτή τη στιγμή, καταλαβαίνουμε πως το σωματίδιο αυτό, βρίσκεται ταυτόχρονα είτε στην αριστερή σχισμή ή στην δεξιά, όπου οδηγούμαστε στην έννοια της υπέρθεσης. Υπέρθεση δεν σημαίνει πως έχουμε πολλαπλά σωματίδια σε πολλαπλά σημεία, αλλά στην πραγματικότητα είναι οι πιθανότητες, ώστε να βρίσκεται παντού το σωματίδιο αυτό.

Στην συνέχεια, το πείραμα επαναλήφθηκε με την προϋπόθεση να πραγματοποιείται μέτρηση του σωματιδίου αυτού, για να κατανοηθεί από ποια σχισμή περνά τελικά το σωματίδιο. Έτσι, τοποθετήθηκε μια συσκευή μέτρησης-παρατήρησης στη κάρτα των δύο σχισμών, με αποτέλεσμα να καταγραφεί σε ποια σχισμή θα διαπεράσει το σωματίδιο. Εδώ φτάνουμε στο επόμενο παράδοξο του πειράματος, πως πλέον, το σωματίδιο συμπεριφέρεται ως σωματίδιο, μιας και παύει να σχηματίζεται το φαινόμενο της συμβολής! Το επόμενο συμπέρασμα είναι πως με την παρατήρηση αλλοιώνεται και το αποτέλεσμα.



Εικόνα 1.1.3: Σωματίδια που περνούν από τη διάταξη δύο σχισμών [Πηγή: Wikipedia]



Εικόνα 1.1.4: Ηλεκτρόνια που περνούν από τη διάταξη δύο σχισμών [Πηγή: Wikipedia]

Πλέον, από αυτό το πείραμα κατανοούμε τις δύο βασικές ιδιότητες της κβαντικής μηχανικής. Αρχικά, το σωματίδιο πριν την μέτρησή του, μπορεί να βρίσκεται παντού, δηλαδή οι πιθανότητες να βρεθεί σε οποιοδήποτε μέρος, το οποίο στην κβαντική μηχανική αποκαλείται **κβαντική υπέρθεση (quantum superposition)**. Τέλος, το σωματίδιο αυτό μετά από μέτρηση ή όπως στο πείραμα στη περίπτωση παρατήρησης, φτάνει τελικά σε κάποιο προορισμό και δημιουργεί ένα μοτίβο παρεμβολής, μετά από σύγκρουση άλλων σωματιδίων, όπως είδαμε με τις ιδιότητες της κυματικής συμπεριφοράς. Το αποτύπωμα αυτό ονομάζεται **κβαντική παρεμβολή (quantum interference)**. Αυτές οι δύο έννοιες περιγράφονται αμέσως μετά και θα γίνουν περισσότερο αντιληπτές, όσο προχωράμε στα επόμενα κεφάλαια.

1.2. Κβαντική υπέρθεση & παρεμβολή

Από το πείραμα του Young προκύπτουν δύο από τις βασικές ιδιότητες της κβαντικής μηχανικής και θα χρειαστεί να αναλύσουμε περαιτέρω τις έννοιες αυτές, καθώς και τι σχέσεις αποτυπώνονται.

Η κβαντική υπέρθεση είναι η ικανότητα ενός κβαντικού σωματιδίου (ηλεκτρόνιο, φωτόνιο) να μπορεί να καταχωρεί πολλαπλές καταστάσεις σε οποιαδήποτε χρονική περίοδο. Αυτό δεν σημαίνει πως έχουμε πολλαπλά σωματίδια, αλλά πιθανότητες, ώστε το σωματίδιο αυτό να μπορεί να βρεθεί σε πολλαπλά μέρη ταυτόχρονα. Για παράδειγμα, ένα φωτόνιο φέρει την ιδιότητα της πόλωσης, που σημαίνει ο προορισμός της ταλάντωσής του στο χώρο. Η πόλωση του φωτονίου μπορεί να είναι είτε κάθετη ή οριζόντια. Κάθε πόλωσή του μπορεί να περιγραφεί ως διανύσματα κατάστασης, με τη κάθετη πόλωση να αναπαρίσταται ως $|\uparrow\rangle$ και την οριζόντια πόλωση ως $|\rightarrow\rangle$ ^[6]. Ενώ και οι δύο καταστάσεις έχουν 90° διαφορά, λόγω της υπέρθεσης, το σωματίδιο μπορεί να βρίσκεται είτε κάθετα ή οριζόντια με τις ίδιες πιθανότητες, σύμφωνα με την παρακάτω σχέση:

$$|\psi\rangle = \frac{1}{\sqrt{2}} |\uparrow\rangle + \frac{1}{\sqrt{2}} |\rightarrow\rangle$$

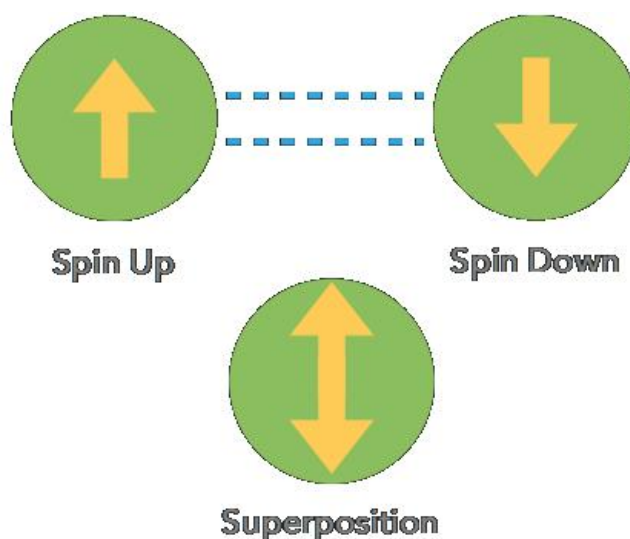
Αυτό σημαίνει πως, όταν ένα κβαντικό σωματίδιο εκθέσει το φαινόμενο της κατάρρευσης κυματοσυνάρτησης, η υπέρθεση θα καταρρεύσει σε ένα από τα σημεία που ορίζει, όταν το σωματίδιο μετρηθεί. Άρα, το σωματίδιο έχει την ίδια πιθανότητα να καταλήξει είτε στην κάθετη $|\uparrow\rangle$ ή οριζόντια $|\rightarrow\rangle$ κατάσταση. Έτσι και η κβαντική παρεμβολή είναι το αποτέλεσμα του κβαντικού σωματιδίου από τη στιγμή που θα μετρηθεί.

1.3. Κβαντική διεμπλοκή

Η τρίτη βασική ιδιότητα της κβαντικής μηχανικής, ονομάζεται **κβαντική διεμπλοκή (quantum entanglement)**. Όταν θέλουμε να γνωρίζουμε τη τοποθεσία από ένα αντικείμενο μέσα στον χώρο, μπορούμε να χρησιμοποιήσουμε διάφορες μετρήσεις. Για παράδειγμα, μπορούμε να χρησιμοποιήσουμε το γεωγραφικό πλάτος και μήκος ή και τους άξονες X, Y και Z στον τρισδιάστατο χώρο. Μπορούμε να φέρουμε ως παράδειγμα τα ηλεκτρόνια, μιας και φέρουν την ίδια αρχή, επίσης. Μπορούμε να ορίσουμε τις θέσεις ή καταστάσεις τους με κάποιου είδους μετρήσεις ή συντεταγμένες. Γι' αυτό, υπάρχουν τέσσερις διαφορετικές μετρήσεις ή τιμές, όταν προσπαθούμε να βρούμε τη κατάσταση ενός ηλεκτρονίου^[7]. Αυτές είναι οι εξής:

- Ο κύριος κβαντικός αριθμός (n) (principal)
- Ο κβαντικός αριθμός της τροχιακής στροφορμής (orbital angular momentum)
- Ο μαγνητικός κβαντικός αριθμός (magnetic)
- Ο κβαντικός αριθμός του περιστροφικής κίνησης (spin)

Η μέτρηση που μας απασχολεί είναι ο κβαντικός αριθμός του spin. Ενώ τα ηλεκτρόνια δεν στροβιλίζουν, στην πραγματικότητα, όταν αναφερόμαστε στο spin ενός ηλεκτρονίου εννοούμε τη θέση του, η οποία μπορεί να έχει τιμές είτε πάνω (spin up) ή κάτω (spin down). Φυσικά, όταν το μετρήσουμε μπορούμε να γνωρίζουμε το spin του, αλλά πριν απ' αυτό δεν μπορούμε να το καθορίσουμε, μιας και βρίσκεται σε κατάσταση υπέρθεσης. Επίσης, θα πρέπει να γνωρίζουμε πως χάνουμε τη πληροφορία, όταν το ηλεκτρόνιο βρίσκεται σε κατάσταση υπέρθεσης και ενώ, αν για παράδειγμα, εμφανιστεί σε spin up κατάσταση, τότε στην επόμενη μέτρηση θα



Εικόνα 1.3.1: Καταστάσεις ηλεκτρονίου

ξαναεμφανιστεί στην ίδια κατάσταση, όπως και παρομοίως θα συμβεί αν εμφανιστεί στην κατάσταση spin down. Αυτό που το καθιστά ενδιαφέρον, είναι πως αυτές τις καταστάσεις μπορούμε να τις αναπαριστούμε με τις τιμές 0 και 1, όπως συμβαίνει στα κλασσικά συστήματα, στο τέλος της μέτρησης.

0 → spin down

1 → spin up

Όταν δύο ηλεκτρόνια αλληλεπιδρούν μεταξύ τους, μπαίνουν σε κατάσταση διεμπλοκής (entanglement). Είναι αλληλοεξαρτώμενα μεταξύ τους, που σημαίνει ότι τα spins τους έχουν κάποια συσχέτιση μεταξύ τους. Αν μετρήσουμε ένα ηλεκτρόνιο και βρούμε πως βρίσκεται σε κατάσταση spin up, τότε το άλλο αυτομάτως θα είναι spin down και δεν υπάρχει κάποιος περιορισμός στις αποστάσεις τους. Έτσι, χρησιμοποιούμε αυτό το φαινόμενο, ώστε να μπλέξουμε τα qubits, που θεωρείται η αντίστοιχη μονάδα μέτρησης της πληροφορίας στην κβαντική υπολογιστική, όπως χρησιμοποιούμε τα bits στα κλασσικά συστήματα, ώστε να συσχετίζονται μεταξύ τους. Με αυτό το τρόπο, μόλις ανακαλύψουμε τη τιμή ενός qubit, θα γνωρίζουμε αμέσως την τιμή του άλλου.

Γι' αυτό και στο επόμενο κεφάλαιο θα αναφερθούν περισσότερο τα ζητήματα που απασχολούν την κβαντική υπολογιστική και έννοιες που χρειάζεται να γνωρίζουμε για να προχωρήσουμε στο τελικό στάδιο σχεδιασμού των κβαντικών αλγορίθμων.

Κεφάλαιο 2

Βασικές έννοιες κβαντικής υπολογιστικής

Στο κεφάλαιο αυτό, θα κάνουμε μια εισαγωγή στην κβαντική υπολογιστική, κατανοώντας τις qubit καταστάσεις, τους συμβολισμούς και τα απαραίτητα μαθηματικά που χρησιμοποιούνται στον κλάδο, ώστε να προχωρήσουμε, στη συνέχεια με τα επόμενα κεφάλαια στην υλοποίηση κβαντικών κυκλωμάτων.

2.1. Εισαγωγή

Σήμερα μία από τις πιο δυναμικές τάσεις της επιστήμης της Πληροφορικής είναι αυτή που προεβλέπει τη χρήση μη συμβατικών υπολογιστικών μεθόδων. Η σημαντικότερη ίσως μη συμβατική τάση είναι αυτή που αφορά κβαντικούς υπολογιστές, καθώς ενδέχεται να προσφέρουν πολλά πλεονεκτήματα συγκριτικά με τους κλασικούς υπολογιστές. Μία εξίσου σημαντική τάση αφορά τον μοριακό και γενικότερο τον βιο-εμπνευσμένο υπολογισμό. Επίσης, πολλοί κλασικοί στην ουσία αλγόριθμοι, έχουν εμπνευστεί από τεχνικές του μη συμβατικού υπολογισμού. Ειδικότερα έχουν αναπτυχθεί πολλά υποσχόμενες μεθυστικές μέθοδοι για προβλήματα βελτιστοποίησης εμπνευσμένες από τον κβαντικό υπολογισμό [8] [9].

Όσο αφορά καθαρά κβαντικούς υπολογιστές, οι εμπορικά διαθέσιμοι αυτή τη στιγμή ακολουθούν 2 διαφορετικά μοντέλα κβαντικού υπολογισμού: το μοντέλο της κβαντικής ανόπτησης και το κυκλωματικό μοντέλο. Με χρήση της κβαντικής ανόπτησης είναι δυνατό να επιλυθούν πολύ δύσκολα προβλήματα βελτιστοποίησης, όπως για παράδειγμα το Πρόβλημα του Πλανόδιου Πωλητή (Traveling Salesman Problem) [10][11]. Το κυκλωματικό μοντέλο βρίσκει πληθώρα εφαρμογών, σε μία ευρεία γκάμα που ξεκινά από καινοτόμους κβαντικούς αλγόριθμους που υπερέχουν των αντίστοιχων κλασικών, περιλαμβάνει την ασφάλεια και την κβαντική κρυπτογραφία [12], και φτάνει σε ένα νέο πεδίο, την κβαντική θεωρία παιγνίων [13][14] [15]. Κλείνοντας αυτή τη σύντομη ανασκόπηση, αξίζει να αναφερθεί ότι και ο μοριακός και βιο-εμπνευσμένος υπολογισμός έχει να επιδείξει σημαντικά επιτεύγματα. Η αναγωγή βιολογικών διαδικασιών και συστημάτων ως μοντέλα για την εκτέλεση υπολογισμού ανοίγει νέους ορίζοντες στην επιστήμη της Πληροφορικής [16] [17] [18].

Συγκεκριμένα, ο τομέας της κβαντικής υπολογιστικής αξιοποιεί τις κβαντομηχανικές ιδιότητες που αναφέρθηκαν στο προηγούμενο κεφάλαιο, όπως η υπέρθεση και διεμπλοκή [19]. Σε σύγκριση με το παραδοσιακό ψηφιακό υπολογισμό,

ο κβαντικός υπολογισμός προσφέρει τη δυνατότητα να μειώσει δραματικά την εκτέλεση χρόνου και κατανάλωσης ενέργειας. Εκτός αυτών, σημαντικό παράγοντα στην γρήγορη εξέλιξη του τομέα έπαιξαν ρόλο και η σταθερή πρόοδος στο κλάδο της νανοβιομηχανίας, όπως και η επιβράδυνση των παραδοσιακών νόμων κλιμάκωσης του υλικού (hardware), γνωστό και ως νόμος του Moore. Αυτό σημαίνει ότι, όσο το μέγεθος των τρανζίστορς συρρικνώνεται τόσο περισσότερη θερμότητα παράγεται, όπου προκαλούνται και περισσότερες ηλεκτρικές διαρροές, κάτι που σημαίνει και μικρότερη απόδοση των κλασσικών υπολογιστών. Όλα τα βασικά στοιχεία και κυρίως, η μαθηματική βάση του κβαντικού υπολογισμού, για τους περισσότερους κβαντικούς αλγορίθμους και το κβαντικό μοντέλο προγραμματισμού έχουν δημοσιευθεί πριν από δεκαετίες, δηλαδή από τη δεκαετία του '90 και έπειτα. Αν και ενδιαφέρον αφοσιώνει μια σχετικά μικρή κοινότητα, εκτιμάται πως είναι η κατάλληλη ώρα για την υλοποίηση κβαντικών αλγορίθμων, οι οποίες και θα είναι προσβάσιμες σε ένα ευρύ φάσμα ερευνητών και προγραμματιστών, όπως και σε υπόλοιπους τομείς ^[19]. Ο κβαντικός προγραμματισμός διαφέρει από τον παραδοσιακό. Επίσης, ο κλάδος αυτός κυριαρχείται από τη Φυσική με αλγεβρικούς συμβολισμούς, κάτι που μπορεί να εμφανίζει ορισμένα εμπόδια σε επιστήμονες υπολογιστών, καθώς και άλλους πιο μαθηματικά καταρτισμένους επιστήμονες. Γι' αυτό και σε αυτό το κεφάλαιο, θα παρουσιαστούν οι βασικές έννοιες και συμβολισμοί ή μαθηματικές σχέσεις που χρειάζονται, ώστε να καταφέρουμε να συμβαδίσουμε με τις μετέπειτα ορολογίες, ειδικότερα όσο η παρούσα εργασία φτάνει στο τέλος της.

2.2. Διανυσματικές καταστάσεις

Μέχρι στιγμής, μπορούμε να κατανοήσουμε τη συμπεριφορά ενός σωματιδίου, όπως ένα ηλεκτρόνιο ή φωτόνιο, με τις αναπαραστάσεις spin up και spin down καταστάσεων, καθώς και τι τιμές θα μπορούσαν να εκχωρηθούν για την κάθε περίπτωση, όπως το 0 και 1, που συμβαίνει με τα κλασσικά συστήματα. Η στοιχειώδης μονάδα μέτρησης της πληροφορίας στην κβαντική υπολογιστική, που χρησιμοποιείται για να περιγράψουμε ένα δισδιάστατο κβαντικό σύστημα, ονομάζεται **qubit** ή **κβαντικό bit**. Το qubit αναπαριστάται ως ένας πίνακας (matrix) με 2×1 διαστάσεις, υπό τη μορφή:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \text{ όπου } \alpha, \beta \in \mathbb{C}$$

Οι τιμές αυτές ονομάζονται και ως πλάτη πιθανότητας, όπου και χαρακτηρίζονται ως τα μέτρα κλίσης ενός συστήματος προς μια κατάσταση. Μπορούμε να χρησιμοποιήσουμε αυτούς τους πίνακες, ώστε να διεξάγουμε

ορισμένες πράξεις, όπως να θέσουμε κάποια qubits σε κατάσταση υπέρθεσης ή διεμπλοκής, χρησιμοποιώντας κβαντικές πύλες, κτλπ. Για παράδειγμα, ο πίνακας

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

δείχνει πως η κλίση α στην πρώτη γραμμή δείχνει προς την κατάσταση 1, ενώ η κλίση β στην τελευταία γραμμή προς τη κατάσταση 0. Αυτό σημαίνει πως η κλίση α αναπαριστά τη πιθανότητα να βρίσκεται σε spin down κατάσταση, ενώ η κλίση β σε spin up κατάσταση, οπότε έχουμε 100% πιθανότητα το qubit να βρίσκεται σε spin down κατάσταση. Άρα, έχουμε:

$$P(\downarrow) = 1$$

$$P(\uparrow) = 0$$

Γιατί συμβαίνει αυτό; Μιας και οι τιμές στον πίνακα είναι μιγαδικοί, μπορούμε να αποδείξουμε το εξής:

$$\alpha = \chi + \psi i \Rightarrow$$

$$|\alpha|^2 = \chi^2 + \psi^2 \Rightarrow$$

$$|\chi|^2 + |\psi|^2 = 1$$

Με αυτό το τρόπο, το σύστημά μας είναι κανονικοποιημένο (normalized), μιας και οι πιθανότητες είναι ίσες με 1 και έτσι, ορίζουμε τη κβαντική κατάσταση ενός qubit. Έτσι, καταλαβαίνουμε πως η κβαντική κατάσταση του qubit θεωρείται ένα μοναδιαίο διάνυσμα, δηλαδή το μέτρο του διανύσματος να ισούται με 1, σε ένα μιγαδικό διανυσματικό χώρο δύο διαστάσεων. Φυσικά, δεν χρειάζεται να βρίσκεται μόνο στις καταστάσεις 0 ή 1. Ο παρακάτω πίνακας αναπαριστά τη κατάσταση υπέρθεσης ενός qubit, όπου:

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \rightarrow \left(\frac{1}{\sqrt{2}}\right)^2 + \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2} + \frac{1}{2} = 1$$

2.3. Συμβολισμός Dirac

Για να καταφέρουμε να αναπαραστήσουμε πιο εύκολα και απλά τις κβαντικές καταστάσεις, επινοήθηκε ο **συμβολισμός Dirac** ή γνωστός ως **bra-ket συμβολισμός**. Ο συμβολισμός αυτός προήλθε από τον Paul Dirac, που πρόκειται για έναν ακόμη μεγάλο φυσικό στον χώρο της κβαντικής μηχανικής, όπου συμβόλισε τις διανυσματικές καταστάσεις των κβαντικών συστημάτων με μισές αγκύλες, δηλαδή με τα σύμβολα: $\langle$ | και $| \rangle$. Το πρώτο φέρει την ονομασία **bra**, ενώ το δεύτερο την ονομασία **ket**, εμπνέοντας από τη λέξη bracket. Τα διανύσματα κατάστασης $|\psi\rangle$, όπου ψ ένας τυχαίος αριθμός, ονομάζονται διανύσματα ket, ενώ αντιθέτως με το συμβολισμό $\langle\psi|$, ονομάζονται διανύσματα bra. Για παράδειγμα τα ket του 1 και 0, συμβολίζονται ως:

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Για να δούμε, πόσο χρήσιμος είναι ο συμβολισμός αυτός, μπορούμε να πάρουμε ως παράδειγμα τον παρακάτω πίνακα υπέρθεσης, ο οποίος και θα φέρει την εξής σχέση:

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$$

Ισοδύναμα το παραπάνω παράδειγμα μπορεί να γραφεί και ως:

$$|\psi\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \alpha|0\rangle + \beta|1\rangle \quad \text{όπου } \alpha, \beta = \frac{1}{\sqrt{2}}$$

Τέλος, για κάθε ket αντιστοιχεί και ένα bra. Αν γνωρίζουμε τη τιμή του ket, τότε αυτόματα βρίσκουμε και τη τιμή του bra. Αν η τιμή του ket είναι:

$$|\psi\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \alpha|0\rangle + \beta|1\rangle$$

τότε, η τιμή του bra θα είναι:

$$\langle\psi| = \alpha^* [1 \quad 0] + \beta^* [0 \quad 1] = \alpha\langle 0| + \beta\langle 1| = [\alpha^* \quad \beta^*]$$

όπου α^* θεωρείται ο συζυγής μιγαδικός του α και, β^* ο αντίστοιχος συζυγής μιγαδικός του β . Μιγαδικός συζυγής του αριθμού $(\alpha + \beta i)$ είναι ο $(\alpha - \beta i)$. Άρα, ο πίνακας που

προκύπτει στο διάνυσμα $\langle\psi|$ προκύπτει από τον πίνακα που αντιστοιχεί στο διάνυσμα $|\psi\rangle$, μετατρέποντας τη στήλη σε γραμμή και αντικαθιστώντας τα στοιχεία με τα μιγαδικά συζυγή τους.

2.4. Εσωτερικό & Εξωτερικό γινόμενο

Ο συμβολισμός για το εσωτερικό γινόμενο δύο καταστάσεων, έστω $|\chi\rangle$ και $|\psi\rangle$, γράφεται ως $\langle\chi|\psi\rangle$. Αυτό μεταφράζεται ως το γινόμενο του bra της πρώτης κατάστασης με το ket της δεύτερης. Σε μορφή πινάκων, αυτό φαίνεται ως:

$$\langle\chi|\psi\rangle = [\gamma^* \quad \delta^*] \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \gamma^* \alpha + \delta^* \beta$$

Μπορούμε να πάρουμε ως παράδειγμα το συνδυασμό των ket των $|0\rangle$ και $|1\rangle$:

$$\langle 0|0\rangle = [1 \quad 0] \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 1$$

$$\langle 0|1\rangle = [1 \quad 0] \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0$$

$$\langle 1|0\rangle = [0 \quad 1] \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 0$$

$$\langle 1|1\rangle = [0 \quad 1] \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1$$

Στις περιπτώσεις που το εσωτερικό γινόμενο ισούται με 0, τότε λέμε πως οι δύο καταστάσεις ονομάζονται ορθογώνιες καταστάσεις.

Αντίστοιχα, το εξωτερικό γινόμενο δύο καταστάσεων, έστω $|\chi\rangle$ και $|\psi\rangle$, γράφεται ως $|\chi\rangle\langle\psi|$. Αυτό μεταφράζεται ως το γινόμενο του ket της πρώτης κατάστασης με το bra της δεύτερης. Σε μορφή πινάκων φαίνεται ως:

$$|\chi\rangle\langle\psi| = \begin{bmatrix} \gamma \\ \delta \end{bmatrix} [\alpha^* \quad \beta^*] = \begin{bmatrix} \gamma\alpha^* & \gamma\beta^* \\ \delta\alpha^* & \delta\beta^* \end{bmatrix}$$

2.5. Τελεστές

Στους συμβατικούς υπολογιστές πραγματοποιείται χειραγώγηση του ρεύματος για την επεξεργασία των δεδομένων, με τη βοήθεια ηλεκτρονικών κυκλωμάτων και λογικών πυλών. Ενώ οι λογικές πύλες αποτελούνται από πολλά τρανζίστορς, όπου και με τη τάση του ρεύματος μπορούμε να σχηματίσουμε πληροφορία, αυτό στη περίπτωση των κβαντικών συστημάτων δεν ισχύει. Σε ένα κβαντικό υπολογιστή, οι κβαντικές πύλες δεν έχουν φυσική υπόσταση και, στην ουσία, ορίζονται από τελεστές δράσης ή αλλαγής στις καταστάσεις των qubits. Η πληροφορία δεν μπορεί να ταξιδέψει, παρά να μείνει αποθηκευμένη σε κάποιο κβαντικό σύστημα, ενώ παράλληλα, οι τελεστές να μπορούν να αλληλεπιδρούν ο ένας με τον άλλον ή να δράσουν, όπως αναφέρθηκε, προκαλώντας την αλλαγή κατάστασης του κβαντικού συστήματος, που εννοείται και η πληροφορία. Αυτές οι αλλαγές στις καταστάσεις των κβαντικών συστημάτων, ονομάζονται **τελεστές** και αναπαρίστανται ως πίνακες. Ο συμβολισμός του τελεστή για έναν πίνακα A , είναι το $\hat{A}$. Θα μπορούσαμε να πραγματοποιήσουμε πολλές δράσεις σε ένα κβαντικό σύστημα, όπως από το να μην το επηρεάσουμε καθόλου ή να στρέψουμε την κατάσταση και πολλά άλλα ακόμη. Αν θέλαμε, για παράδειγμα, να μην επηρεάσουμε την κατάσταση και να μην κάνουμε τίποτα πάνω στην κατάσταση του qubit, ο τελεστής $\hat{I}$ θα περιγραφόταν από τον μοναδιαίο πίνακα I :

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Αν επιθυμούσαμε να στρέψουμε τη κατάσταση με τον τελεστή $\hat{A}$, τότε και ο αντίστοιχος πίνακας A θα είναι:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Με βάση τα παραδείγματα, ο τελεστής $\hat{A}$, σε αυτή την περίπτωση, έχει στρέψει τα διανύσματα κατάστασης προς την αντίθετη κατάσταση, όπως δηλαδή το $|0\rangle$ γίνεται $|1\rangle$ και το $|1\rangle$ γίνεται $|0\rangle$. Αντίστοιχα, όταν πολλαπλασιάζουμε με τον μοναδιαίο $\hat{I}$, βλέπουμε πως έχει μηδενική επιρροή, δίνοντας το ίδιο αποτέλεσμα. Κάπως έτσι, θα μπορούσαν να υπάρχουν διαφορετικές δράσεις επάνω στις καταστάσεις και να προκαλούν, κάλλιστα, νέα υπέρθεση των καταστάσεων αυτών.

2.6. Τανυστικό γινόμενο

Η κατάσταση των κβαντικών συστημάτων αντιπροσωπεύεται από ένα διάνυσμα στον **χώρο Hilbert**. Χώρο Hilbert εννοούμε ένα διανυσματικό χώρο πάνω στο πεδίο των μιγαδικών αριθμών και, γι' αυτό θεωρείται ένας μιγαδικός διανυσματικός χώρος. Τι σημαίνει, όμως, αυτό στη πραγματικότητα; Στη περίπτωση των κβαντικών συστημάτων, όταν επιθυμούμε να αλλάξουμε την κατάσταση με μια δράση από το $|\chi\rangle$ στο $|\psi\rangle$, τότε θα πρέπει να είμαστε ικανοί να αντιστρέψουμε την κατάσταση, επιστρέφοντας στην αρχική κατάσταση, δηλαδή πίσω στο $|\chi\rangle$. Για παράδειγμα, αν πολλαπλασιάσουμε με έναν πίνακα την κατάσταση $|\chi\rangle$, θα πάρουμε την κατάσταση $|\psi\rangle$. Βέβαια, σε αυτή τη περίπτωση, θα πρέπει, όταν πολλαπλασιάζουμε την κατάσταση $|\psi\rangle$ με τον αντίστροφο αυτού του πίνακα, θα πρέπει να επιστρέψουμε στην κατάσταση $|\chi\rangle$. Επίσης, οι μιγαδικοί αυτοί πίνακες που έχουν την δυνατότητα να περιστρέφουν τα διανύσματα κατάστασης, ονομάζονται ορθομοναδιαίοι.

Βέβαια, για να κατασκευαστεί ένας χώρος Hilbert που θα είναι χρήσιμο για πιο σύνθετα και περίπλοκα συστήματα, χρειαζόμαστε το μαθηματικό εργαλείο, που ονομάζεται **τανυστικό γινόμενο (tensor product)** που, μάλιστα, αποτελεί θεμελιώδης λειτουργία στη κατασκευή κβαντικών συστημάτων. Συγκεκριμένα, το τανυστικό γινόμενο είναι χρήσιμο για την ένωση δύο ή παραπάνω χώρων Hilbert, διανυσμάτων ή και τελεστών. Ένα παράδειγμα είναι ένα κβαντικό σύστημα δύο σωματιδίων και, συνεπώς, δύο χώρων Hilbert, έστω H_1 και H_2 , όπου και ο καθένας φέρουν $n \times m$ διαστάσεις. Το τανυστικό γινόμενο των δύο αυτών χώρων, δημιουργεί έναν νέο χώρο H με διαστάσεις $n \times m$ και φαίνεται με την παρακάτω σχέση:

$$H = H_1 \otimes H_2$$

Έστω, ότι οι καταστάσεις των σωματιδίων είναι:

$$|\chi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \text{ και } |\psi\rangle = \begin{bmatrix} \gamma \\ \delta \end{bmatrix}$$

τότε,

$$|\chi\rangle \otimes |\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \otimes \begin{bmatrix} \gamma \\ \delta \end{bmatrix} = \begin{bmatrix} \alpha\gamma \\ \alpha\delta \\ \beta\gamma \\ \beta\delta \end{bmatrix} = |\chi\psi\rangle$$

Πιο πρακτικά, μπορούμε να δούμε τους υπολογισμούς δύο qubits σε περιπτώσεις που φέρουν τις τιμές 0 και 1, αντίστοιχα:

$$|0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |00\rangle$$

$$|0\rangle \otimes |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = |01\rangle$$

$$|1\rangle \otimes |0\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle$$

$$|1\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle$$

Κεφάλαιο 3

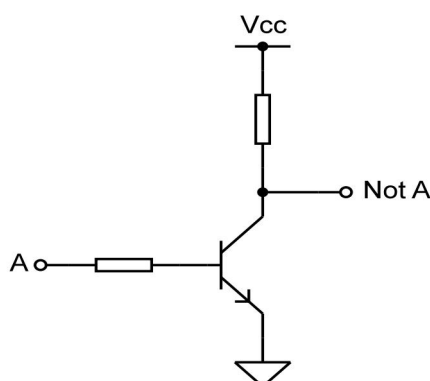
Κβαντικό μοντέλο προγραμματισμού

Στο κεφάλαιο αυτό περιγράφονται οι έννοιες των συμβατικών και κβαντικών πυλών, όπου η δεύτερη κατηγορία θεωρούνται τελεστές του χώρου Hilbert. Παρουσιάζονται οι διάφορες κατηγορίες των κβαντικών πυλών που δρουν είτε σε ένα qubit ή σε κβαντικούς καταχωρητές με δύο ή τρία qubits. Τέλος, παίρνουμε μια πρώτη ματιά στην υλοποίηση ενός κβαντικού κυκλώματος.

3.1. Λογικές πύλες συμβατικών συστημάτων

Ακόμα και με δύο καταστάσεις του ρεύματος (ενεργό & ανενεργό), μπορούμε να αναπαραστήσουμε σημαντικές πληροφορίες. Την αναπαράσταση αυτή, την ονομάζουμε δυαδική (binary) – που κυριολεκτικά σημαίνει “με 2 καταστάσεις”. Υπήρξαν και άλλες τέτοιες αναπαραστάσεις, όπως η τριαδική (ternary) και η πενταδική (quinary), αλλά δεν χρησιμοποιούνται λόγω παρεμβολών και αυτό θα γινόταν χειρότερο, όταν υπάρχουν πολλά τρανζίστορς που αλλάζουν καταστάσεις εκατομμύρια φορές το δευτερόλεπτο. Ένας άλλος λόγος που επικράτησε η δυαδική λογική, ήταν τα μαθηματικά. Τα μαθηματικά λειτουργούν με τον ίδιο τρόπο, υποθέτοντας μια κατάσταση, για να καταλήξουμε σε ένα αποτέλεσμα, που είναι true ή false. Ο τομέας των μαθηματικών αυτών, ονομάζεται δυαδική άλγεβρα (boolean algebra). Ο George Boole, από τον οποίον πήρε και το όνομά της, ήταν αυτοδίδακτος Άγγλος μαθηματικός, από το 1800. Η προσέγγιση του Boole επέτρεψε την έννοια της αλήθειας να αποδεικνύεται συστηματικά και φορμαλιστικά, μέσα από εξισώσεις, τις οποίες και εισήγαγε στο πρώτο του βιβλίο “The Mathematical Analysis of Logic” το 1847^[20]. Στην Άλγεβρα, οι μεταβλητές είναι, κυρίως, αριθμοί και οι λειτουργίες της αποτελούνται από πολλαπλασιασμούς, διαιρέσεις, κτλπ. Στην Δυαδική Άλγεβρα, οι μεταβλητές που χρησιμοποιούνται είναι οι τιμές true & false, ενώ οι λειτουργίες της βασίζονται από λογική. Υπάρχουν 3 βασικές λειτουργίες στην δυαδική άλγεβρα: NOT, AND, OR, αλλά θα αναλυθούν οι πρώτες δύο, μιας και είναι αρκετές για να προχωρήσουμε στα τελευταία κεφάλαια. Αξίζει να αναφερθεί, πως όλα τα operations, τα οποία αναφέρονται παρακάτω δημιουργούνται με τρανζίστορς, οπότε στο πρώτο παράδειγμα της NOT θα οπτικοποιηθεί η διαφορά αναπαράστασής της σε μια ενιαία συμβατική πύλη ^[21].

Το NOT παίρνει μια δυαδική τιμή (true/false) και την αναιρεί. Για να το



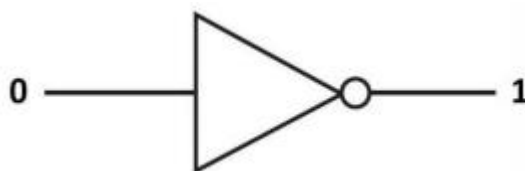
Εικόνα 3.1.1: Εσωτερική ματιά σε μια πύλη NOT [Πηγή: Wikipedia]

δημιουργήσουμε, χρειαζόμαστε ένα σύρμα ελέγχου ως είσοδο και ένα ηλεκτρόδιο ως έξοδο. Όταν το σύρμα ελέγχου δέχεται ρεύμα, δηλαδή είναι ενεργό, τότε και η έξοδος θα είναι ενεργή, όπως θα ίσχυε και το αντίστροφο, να ήταν όλα ανενεργά. Όμως, για να καταλήξουμε να αναιρέσουμε την τελική τιμή της εξόδου, θα πρέπει να κάνουμε κάποιες μετατροπές – η διαδικασία μετατροπής μιας πύλης, αναπαρίσταται από έναν **λογικό πίνακα (logic table)** ή **πίνακα αλήθειας (truth table)**. Ο αντίστοιχος πίνακας μιας NOT πύλης παρουσιάζεται ως εξής:

<i>Bit Εισόδου</i>	<i>Bit Εξόδου</i>
0	1
1	0

Πίνακας 3.1.1: Πίνακας αλήθειας πύλης NOT

Κατανοώντας τα συστατικά μέρη μιας NOT πύλης, μπορούμε να ανέβουμε ένα επίπεδο αναπαράστασης, όπως κάνουν και οι μηχανικοί, δίνοντας σχήματα-σύμβολα στις πύλες αυτές. Το αντίστοιχο σύμβολο της NOT φαίνεται στην Εικόνα 3.1.2. Οι μηχανικοί τα αναπαριστούν έτσι, γιατί μας επιτρέπουν να χτίσουμε ακόμη μεγαλύτερα συστατικά μέρη.



Εικόνα 3.1.2: Συμβολισμός της πύλης NOT

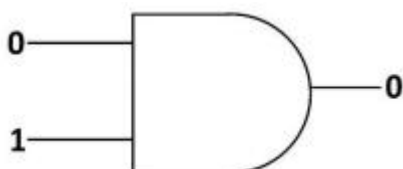
Φυσικά, τα bits, μπορούμε να τα αναπαραστήσουμε και ως διανύσματα στήλης, συνεπώς και οι λογικές πύλες μπορούν να περιγραφούν με την μορφή πινάκων. Για παράδειγμα, η πύλη NOT μπορεί να αναπαρασταθεί ως ένας πίνακας με διαστάσεις 2×2 , ως εξής:

$$\text{NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

όπου με το παραπάνω παράδειγμα θεωρούμε πως το bit εισόδου φέρει την τιμή 1, ώστε και η τιμή που θα πάρουμε, έπειτα από εφαρμογή της πύλης NOT, να είναι το 0. Αυτό επαληθεύεται ως:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 * 0 + 1 * 1 \\ 1 * 0 + 0 * 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Αντίστοιχα, η λογική πύλη AND παίρνει ως είσοδο δύο bits και εξάγει ένα αποτέλεσμα.



Εικόνα 3.1.3: Αναπαράσταση της πύλης AND

Ο πίνακας αλήθειας της πύλης φαίνεται ως:

<i>Bits Εισόδου</i>		<i>Bits Εξόδου</i>
0	0	0
0	1	0
1	0	0
1	1	1

Πίνακας 3.1.2: Πίνακας αλήθειας πύλης AND

Παράλληλα, η αναπαράσταση της πύλης AND με τη μορφή πίνακα θα είναι:

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ο πίνακας αυτός προκύπτει, αν γνωρίζουμε τα εξής:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |00\rangle$$

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = |01\rangle$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle$$

Στην ουσία, κάθε γραμμή για κάθε bit αναπαριστά τις εξόδους του συστήματος — στη πρώτη γραμμή βλέπουμε τις δύο εξόδους 11 και 10 και στην δεύτερη τις 00 και 01. Αντίστοιχα, κάθε στήλη σημαίνει τις εισόδους του συστήματος για κάθε bit. Αν για παράδειγμα, τα bits εισόδου που έχουμε είναι 1 και 0, αντίστοιχα, τότε σημαίνει πως έχουμε την κατάσταση 10 και, συνεπώς, υλοποιώντας την πύλη AND, επαληθεύουμε το εξής:

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 * 0 + 1 * 0 + 1 * 1 + 0 * 0 \\ 0 * 0 + 0 * 0 + 0 * 1 + 1 * 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Αυτές οι λογικές πύλες βοηθάνε στην καλύτερη κατανόηση των κβαντικών πυλών, μετέπειτα, όπου θα τις περιγράψουμε και, πάλι, με τη μορφή πινάκων.

3.2. Λογικές πύλες κβαντικών συστημάτων

Γνωρίζουμε πως, οι συμβατικοί υπολογιστές θεωρούνται φυσικά συστήματα, κάτι που δεν συμβαίνει στη περίπτωση των κβαντικών συστημάτων. Οι κβαντικές πύλες αντιπροσωπεύουν τις δράσεις που ασκούνται στα qubits ^[21]. Επίσης, οι δράσεις αυτές αντιπροσωπεύονται από τους τελεστές του χώρου Hilbert. Συνεπώς, οι καταστάσεις των qubits θεωρούνται διανύσματα στο χώρο Hilbert, όπως και οι κβαντικές πύλες θεωρούνται τελεστές, αλλάζοντας την κατάσταση των qubits. Στην

ουσία, οι κβαντικές πύλες θα περιστρέψουν τα διανύσματα κατάστασης των qubits, χωρίς βέβαια, να αλλάξουν το μήκος τους. Φυσικά, αυτό δεν σημαίνει πως είναι κανόνας, δηλαδή ότι κάθε τελεστής στο χώρο Hilbert να είναι και κβαντική πύλη. Αυτό εννοείται, γιατί οι κβαντικές πύλες θα πρέπει να μην μεταβάλλουν το μήκος του διανύσματος κατάστασης, καθώς και να μην επηρεάζουν τις τιμές των εσωτερικών γινομένων αυτών των διανυσμάτων κατάστασης. Γι'αυτό, οι κβαντικές πύλες αναπαριστώνται ως **μιγαδικοί μοναδιαίοι πίνακες (unitary matrices)** ^[22]. Ένας μιγαδικός τετραγωνικός πίνακας U θεωρείται μοναδιαίος, όταν ο ανάστροφος συζυγής (conjugate transpose), ο οποίος συμβολίζεται ως $U^\dagger$, ισούται με τον αντίστροφό του (inverse). Αυτό φαίνεται από τη παρακάτω σχέση:

$$UU^\dagger = U^\dagger U = I, \text{ όπου } I \text{ μοναδιαίος πίνακας}$$

Αυτό το βλέπουμε και από το παρακάτω παράδειγμα, όπου:

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 1 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

Θα προχωρήσουμε, λοιπόν, στις κβαντικές πύλες, κατανοώντας τα αποτελέσματα των δράσεών τους.

3.2.1. Η κβαντική πύλη αδρανείας

Η πύλη αδρανείας (identity gate ή Pauli I gate) είναι και η πιο απλή από τις υπόλοιπες πύλες. Στην ουσία, αφήνει αμετάβλητη την κατάσταση του qubit. Ο πίνακας αλήθειας είναι ο εξής:

<i>Qubit Εισόδου</i>	<i>Qubit Εξόδου</i>
$ 0\rangle$	$ 0\rangle$
$ 1\rangle$	$ 1\rangle$
$ q\rangle$	$ q\rangle$

Πίνακας 3.2.1: Πίνακας αλήθειας πύλης αδρανείας ^[21]

Σε κάθε πίνακα των κβαντικών πυλών που θα περιγραφούν και στην συνέχεια, θα παρουσιάζεται στην πρώτη στήλη η κατάσταση πριν εφαρμοστεί η αντίστοιχη πύλη και στην δεύτερη η κατάσταση μετά.

Βέβαια, πρέπει να γνωρίζουμε πως κάθε qubit δεν περνάει μέσα από τη πύλη, αλλά τα συμβολίζουμε σύμφωνα με τις αντίστοιχες καταστάσεις τους, δηλαδή τις τιμές πριν και μετά τη δράση.

Έπειτα, ο πίνακας με τον τελεστή της πύλης αυτής αποδεικνύεται, όπως παρακάτω:

$$\begin{aligned}
 \sigma_I &= I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
 &= |0\rangle\langle 0| + |1\rangle\langle 1| \\
 &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
 \end{aligned}$$

οπότε, ισχύει:

$$I |\psi\rangle = |\psi\rangle$$

3.2.2. Η κβαντική πύλη περιστροφής φάσης

Η πύλη περιστροφής φάσης (rotation gate), συμβολίζεται με Φ ή R_φ^Z και είναι, επίσης, απλή πύλη, αφού αλλάζει μονάχα τη φάση ενός qubit. Ο πίνακας αλήθειας φαίνεται παρακάτω:

<i>Qubit Εισόδου</i>	<i>Qubit Εξόδου</i>
$ 0\rangle$	$ 0\rangle$
$ 1\rangle$	$e^{i\varphi} 1\rangle$
$\alpha 0\rangle + \beta 1\rangle$	$\alpha 0\rangle + e^{i\varphi}\beta 1\rangle$

Πίνακας 3.2.2: Πίνακας αλήθειας πύλης περιστροφής φάσης ^[21]

Έπειτα, ο πίνακας με τον τελεστή της πύλης αυτής επαληθεύεται ως:

$$\Phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix}$$

Όπως βλέπουμε στον πίνακα αλήθειας, αν έχουμε ένα qubit σε μια τυχαία κατάσταση, τότε:

$$|q_{\varepsilon\xi}\rangle = \Phi |q_{\varepsilon\sigma}\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ e^{i\varphi}\beta \end{bmatrix}$$

3.2.3. Οι κβαντικές πύλες Pauli

Οι κβαντικές πύλες Pauli είναι μια πολύ σημαντική κατηγορία κβαντικών πυλών, οι οποίες δρουν μόνο σε ένα qubit και επιτρέπουν την περιστροφή των διανυσμάτων κατάστασης. Οι πύλες αυτές είναι τρεις, βάσει των τριών αξόνων στον τρισδιάστατο χώρο και συμβολίζονται ως X, Y, Z. Ο πίνακας αλήθειας φαίνεται παρακάτω:

<i>Qubit Εισόδου</i>	<i>Qubit Εξόδου</i>	
$\alpha 0\rangle + \beta 1\rangle$	$\beta 0\rangle + \alpha 1\rangle$	X
$\alpha 0\rangle + \beta 1\rangle$	$-i\beta 0\rangle + i\alpha 1\rangle$	Y
$\alpha 0\rangle + \beta 1\rangle$	$\alpha 0\rangle - \beta 1\rangle$	Z

Πίνακας 3.2.3: Πίνακας αλήθειας των τριών πυλών Pauli ^[21]

Έπειτα, οι πίνακες με τους τελεστές και των τριών πυλών φαίνονται ως εξής:

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

Όπως βλέπουμε από τον πίνακα αλήθειας, αν έχουμε ένα qubit σε μια τυχαία κατάσταση, τότε οι δράσεις και των τριών πυλών αποδεικνύονται ως:

$$X |q_{\varepsilon\sigma}\rangle = X (\alpha|0\rangle + \beta|1\rangle) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \beta|0\rangle + \alpha|1\rangle = |q_{\varepsilon\xi}\rangle$$

$$Y |q_{\varepsilon\sigma}\rangle = X (\alpha|0\rangle + \beta|1\rangle) = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -i\beta \\ i\alpha \end{bmatrix} = -i\beta|0\rangle + i\alpha|1\rangle = |q_{\varepsilon\xi}\rangle$$

$$Z |q_{\varepsilon\sigma}\rangle = X (\alpha|0\rangle + \beta|1\rangle) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ -\beta \end{bmatrix} = \alpha|0\rangle - \beta|1\rangle = |q_{\varepsilon\xi}\rangle$$

3.2.4. Η κβαντική πύλη Hadamard

Η πύλη Hadamard θεωρείται μια από τις σημαντικότερες πύλες, επειδή μπορεί να δράσει στα qubits, ενώ βρίσκονται σε μία από τις δύο βασικές καταστάσεις και να τα φέρει σε κατάσταση υπέρθεσης. Αυτό σημαίνει πως το qubit θα έχει ίση πιθανότητα, μόλις το μετρήσουμε, να εμφανίσει τη βασική κατάσταση 0 ή και 1, οπότε και οι πιθανότητες είναι ίσες με 50% εμφάνισης ενός από των δύο τιμών. Ο πίνακας αλήθειας είναι ο εξής:

<i>Bit Εισόδου</i>	<i>Bit Εξόδου</i>
$ 0\rangle$	$\frac{1}{\sqrt{2}} (0\rangle + 1\rangle)$
$ 1\rangle$	$\frac{1}{\sqrt{2}} (0\rangle - 1\rangle)$
$\frac{1}{\sqrt{2}} (0\rangle + 1\rangle)$	$ 0\rangle$
$\frac{1}{\sqrt{2}} (0\rangle - 1\rangle)$	$ 1\rangle$

Πίνακας 3.2.4: Πίνακας αλήθειας πύλης Hadamard ^[21]

Έπειτα, ο πίνακας με τον τελεστή της πύλης φαίνεται ως εξής:

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

Όπως βλέπουμε από τον πίνακα αλήθειας, αν έχουμε ένα qubit είτε στη βασική κατάσταση 0 ή 1, τότε η δράση της πύλης αποδεικνύεται ως:

$$H |0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$$

$$H |1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Μπορούμε, επίσης, να αποδείξουμε πως είναι και αντιστρέψιμη με το qubit να βρίσκεται στη κατάσταση υπέρθεσης:

$$H \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$$

$$H \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

3.2.5. Η κβαντική πύλη CNOT

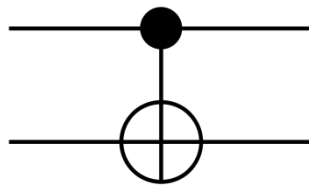
Η κβαντική πύλη ελεγχόμενου ΟΧΙ (Controlled-NOT ή CNOT) δέχεται ως εισόδους δύο qubits και εξάγει δύο αποτελέσματα. Κάθε qubit έχει μια ονομασία, με το πρώτο να ονομάζεται control qubit, ενώ το δεύτερο qubit εισόδου να ονομάζεται target qubit. Στην ουσία, η CNOT θα αφήσει τη κατάσταση του control qubit αναλλοίωτη και θα αντιστρέψει την κατάσταση του target qubit, μονάχα, όταν το control qubit βρίσκεται στην κατάσταση $|1\rangle$. Οπότε, έχουμε τις περιπτώσεις, όπου η πύλη θα αλλάξει την κατάσταση του target qubit, όταν η κατάσταση του control qubit είναι $|1\rangle$, ενώ θα αφήσει την κατάσταση του target qubit αμετάβλητη, όταν η κατάσταση του control qubit είναι $|0\rangle$. Ο πίνακας αλήθειας παρουσιάζεται παρακάτω, όπου με c (control) εννοείται το qubit ελέγχου και με t (target) το qubit στόχου.

$ c_{\text{εισ}} t_{\text{εισ}}\rangle$	$ c_{\text{εξ}} t_{\text{εξ}}\rangle$
$ 00\rangle$	$ 00\rangle$
$ 01\rangle$	$ 01\rangle$
$ 10\rangle$	$ 11\rangle$
$ 11\rangle$	$ 10\rangle$

Πίνακας 3.2.5: Πίνακας αλήθειας πύλης CNOT ^[21]

Έπειτα, ο πίνακας με τον τελεστή και το σύμβολο της πύλης φαίνεται ως εξής:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



Εικόνα 3.2.1: Συμβολισμός πύλης CNOT

Έστω ότι παίρνουμε για παράδειγμα ως είσοδο $|10\rangle$, τότε θα έχουμε:

$$\text{CNOT} |10\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |11\rangle$$

δηλαδή παρατηρούμε πως η κατάσταση του target qubit έχει αλλάξει από τη κατάσταση $|0\rangle$ σε $|1\rangle$, μιας και η κατάσταση του control qubit είναι $|1\rangle$. Σε αντίθετη περίπτωση:

$$\text{CNOT} |01\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = |01\rangle$$

Μιας και το control qubit βρίσκεται στη κατάσταση $|0\rangle$, τότε και η κατάσταση του control qubit δεν αλλάζει.

3.2.6. Η κβαντική πύλη CCNOT

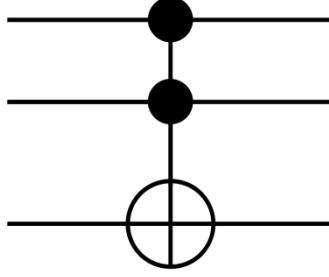
Η κβαντική πύλη διπλά ελεγχόμενου ΟΧΙ (Controlled-Controlled-NOT ή CCNOT ή Toffoli) δέχεται ως εισόδους τρία qubits και εξάγει τρία αποτελέσματα. Όπως και με τη CNOT, κάθε qubit έχει παρόμοια ονομασία, όπου τα δύο πρώτα ονομάζονται control qubits, ενώ το τελευταίο qubit ονομάζεται target qubit. Στην ουσία, η CCNOT θα αλλάξει την κατάσταση του target qubit, όταν και τα δύο control qubits βρίσκονται στην κατάσταση $|1\rangle$, ενώ δεν αλλάζει την κατάσταση του target qubit σε κάθε άλλη περίπτωση. Ο πίνακας αλήθειας φαίνεται παρακάτω, όπου με c (control) εννοούνται τα qubits ελέγχου και με t (target) το qubit στόχου.

$ c_{2,εισ} c_{1,εισ} t_{εισ}\rangle$	$ c_{2,εξ} c_{1,εξ} t_{εξ}\rangle$
$ 000\rangle$	$ 000\rangle$
$ 001\rangle$	$ 001\rangle$
$ 010\rangle$	$ 010\rangle$
$ 011\rangle$	$ 011\rangle$
$ 100\rangle$	$ 100\rangle$
$ 101\rangle$	$ 110\rangle$
$ 110\rangle$	$ 101\rangle$
$ 111\rangle$	$ 111\rangle$

Πίνακας 3.2.6: Πίνακας αλήθειας πύλης CCNOT ^[21]

Έπειτα, ο πίνακας με τον τελεστή και το σύμβολο της πύλης φαίνεται ως εξής:

$$CCNOT = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Εικόνα 3.2.2: Συμβολισμός πύλης CCNOT

Έστω ότι παίρνουμε για παράδειγμα ως είσοδο $|101\rangle$, τότε θα έχουμε:

$$\text{CCNOT } |101\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |101\rangle$$

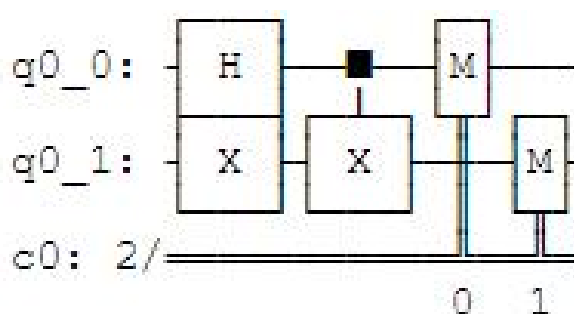
Σε αντίθετη περίπτωση με τιμή εισόδου $|111\rangle$:

$$\text{CCNOT } |101\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |110\rangle$$

3.3. Αναπαράσταση ενός κβαντικού κυκλώματος

Για να χρησιμοποιήσουμε και να τοποθετήσουμε τις διάφορες κβαντικές πύλες, όπως περιγράφηκαν προηγουμένως, δημιουργούμε μια ενιαία σύνδεση μεταξύ τους. Η σύνδεση αυτή μπορεί να ονομαστεί ως ένα **κβαντικό κύκλωμα**. Για να ορίσουμε ένα κβαντικό κύκλωμα, θα πρέπει πρώτα να υπάρχει ένα σύνολο από qubits εισόδου, τα οποία θα βρίσκονται στην βασική κατάσταση $|0\rangle$. Όπως γνωρίζουμε και

από τις συμβατικές πύλες, θεωρούμε πως υπάρχει ένα καλώδιο για κάθε qubit, το οποίο και θα διαχέει την αντίστοιχη πληροφορία. Έπειτα, προσθέτουμε τις κβαντικές πύλες που επιθυμούμε σε οποιοδήποτε qubit, εκτελώντας έτσι, μοναδιαίους μετασχηματισμούς πάνω σε αυτά. Στη συνέχεια, θα πρέπει να εκτελούμε ορισμένες μετρήσεις, που θα πρέπει να είναι μία ή περισσότερες, ώστε να εξάγουμε και τα αποτελέσματα που βγάζουν τα qubits. Ένα παράδειγμα κβαντικού κυκλώματος φαίνεται στην παρακάτω εικόνα:



Εικόνα 3.3.1: Παράδειγμα ενός κβαντικού κυκλώματος με 2 qubits

Πιο συγκεκριμένα, στο παράδειγμα αυτό, εφαρμόζουμε πρώτα στο πρώτο qubit μία πύλη Hadamard (H), ενώ στο δεύτερο qubit εφαρμόζουμε μια πύλη Pauli X. Στη συνέχεια, εισάγουμε μια πύλη CNOT, όπου μεταξύ του πρώτου qubit με του δεύτερου, το πρώτο λειτουργεί ως ένα control qubit και το δεύτερο ως target qubit. Τέλος, προχωράμε στις μετρήσεις και των δύο qubit για να προσδιορίσουμε τα αποτελέσματα. Ο στόχος του παραπάνω κυκλώματος είναι για την αναπαράσταση των καθ'αυτών κβαντικών κυκλωμάτων και για να κατανοήσουμε πως θα εφαρμόζουμε τις κβαντικές πύλες για να εξάγουμε την επιθυμητή πληροφορία. Το συγκεκριμένο κύκλωμα δεν εξάγει και δεν υλοποιεί κάτι χρήσιμο. Γι'αυτό, στο επόμενο κεφάλαιο, θα ξεκινήσουμε σε πιο πρακτικά ζητήματα, κάνοντας μια περιήγηση στην πλατφόρμα Qiskit, υλοποιώντας τα πρώτα μας κβαντικά κυκλώματα και, στην συνέχεια, στην σχεδίαση των κβαντικών αλγορίθμων.

Κεφάλαιο 4

Η πλατφόρμα Qiskit

Στο κεφάλαιο αυτό, θα γίνει εισαγωγή της πλατφόρμας Qiskit, για το σκοπό ανάπτυξής του, τη δομή του και τι τεχνολογίες χρησιμοποιούνται. Επίσης, θα υλοποιήσουμε το πρώτο μας κβαντικό κύκλωμα στη πλατφόρμα και, στη συνέχεια, θα το εκτελέσουμε σε πραγματικό κβαντικό υπολογιστή, ώστε να δούμε τι διαφορές υπάρχουν από αυτόν σε σύγκριση ενός προσομοιωτή, σαν ένα συμβατικό σύστημα.

4.1. Εισαγωγή στο Qiskit

Το Qiskit είναι ένα κιτ ανάπτυξης λογισμικού ανοιχτού κώδικα (SDK) για την ανάπτυξη κυκλωμάτων και αλγορίθμων σε κβαντικούς υπολογιστές. Παρέχει εργαλεία για τη δημιουργία και το χειρισμό κβαντικών προγραμμάτων και την εκτέλεσή τους σε πρωτότυπες κβαντικές συσκευές στη πλατφόρμα IBM Quantum Experience ή σε προσομοιωτές σε τοπικό υπολογιστή. Το Qiskit ιδρύθηκε από την IBM Research για να επιτρέψει την ανάπτυξη λογισμικού για την υπηρεσία κβαντικού υπολογιστικού νέφους, IBM Quantum Experience. Μάλιστα, ένα μεγάλο πλεονέκτημα, είναι πως δίνεται η δυνατότητα συνεισφοράς σε εξωτερικούς υποστηρικτές, καθώς και ακαδημαϊκά ιδρύματα. Το Qiskit βασίζεται στη χρήση της γλώσσας προγραμματισμού Python. Οι εκδόσεις για το Swift και τη JavaScript διερευνήθηκαν αρχικά, αν και η ανάπτυξη για αυτές τις εκδόσεις έχει σταματήσει. Προτείνεται στους χρήστες η εγκατάσταση του Anaconda, που πρόκειται για μια διανομή της Python και χρησιμοποιείται, κυρίως, για επιστημονικές εφαρμογές. Παρέχονται παραδείγματα από jupyter notebooks με τον πηγαίο κώδικα πίσω από τις επιστημονικές μελέτες που χρησιμοποιούνται στο Qiskit, καθώς και ένα σύνολο ασκήσεων για να βοηθήσουν τους ανθρώπους να μάθουν τα βασικά του κβαντικού προγραμματισμού. Στην ουσία, τα Jupyter Notebooks, που περιλαμβάνεται στο Anaconda, προτείνεται για την αλληλεπίδραση με το Qiskit, όταν επιθυμούμε να εκτελέσουμε κώδικα. Πρόκειται για μία εφαρμογή, η οποία επιτρέπει την επεξεργασία και την εκτέλεση αρχείων σε notebook format μέσω ενός φυλλομετρητή ιστού. Τα αρχεία που χρησιμοποιούνται μπορούν να περιλαμβάνουν κώδικα γραμμένα σε γλώσσα Python, διάφορες οπτικοποιήσεις δεδομένων, καθώς και κείμενο σε μορφή markdown, όπως παράγραφοι, εξισώσεις, προσθήκη

υπερσυνδέσμων, κτλπ. Τέλος, παρέχεται και ένα εγχειρίδιο ανοιχτού κώδικα που βασίζεται στο Qiskit, το οποίο ακόμη πολύ χρήσιμο για μαθήματα κβαντικών αλγορίθμων σε πανεπιστημιακό επίπεδο ή συμπλήρωμα μαθήματος κβαντικού υπολογισμού.

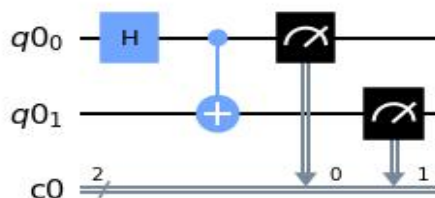
Συγκεκριμένα, το Qiskit αποτελείται από στοιχεία που συνεργάζονται για να επιτρέψουν την ενεργοποίηση της κβαντικής υπολογιστικής, που σύμφωνα με το Qiskit καλούνται ως API References. Ο κεντρικός στόχος του Qiskit είναι να δημιουργήσει μια στοίβα λογισμικού που θα διευκολύνει οποιονδήποτε που θέλει να χρησιμοποιήσει κβαντικούς υπολογιστές, ανεξάρτητα από το επίπεδο δεξιοτήτων ή την περιοχή ενδιαφέροντός του. Το Qiskit επιτρέπει στους χρήστες να σχεδιάζουν εύκολα πειράματα και εφαρμογές και να τα εκτελούν σε πραγματικούς κβαντικούς υπολογιστές ή και κλασικούς προσομοιωτές. Επίσης, παρέχει τη δυνατότητα ανάπτυξης κβαντικού λογισμικού τόσο σε επίπεδο κώδικα μηχανής του OpenQASM όσο και σε αφηρημένα επίπεδα κατάλληλο για τελικούς χρήστες χωρίς τεχνογνωσία στο κλάδο των κβαντικών υπολογιστών. Αυτή η λειτουργία παρέχεται από τα ακόλουθα διακριτά στοιχεία, που είναι το Qiskit Terra και Aer.

• Qiskit Terra:

Το στοιχείο Terra είναι το θεμέλιο πάνω στο οποίο είναι χτισμένο το υπόλοιπο Qiskit και σηματοδοτείται από τη λέξη Γη στα ελληνικά. Το Qiskit Terra παρέχει εργαλεία για τη δημιουργία κβαντικών κυκλωμάτων στο επίπεδο ή κοντά στο επίπεδο του κώδικα της κβαντικής μηχανής. Επιτρέπει στις διαδικασίες που εκτελούνται σε κβαντικό υλικό να κατασκευαστούν ρητά με όρους κβαντικών πυλών. Παρέχει, επίσης, εργαλεία που επιτρέπουν τη βελτιστοποίηση των κβαντικών κυκλωμάτων για μια συγκεκριμένη συσκευή, καθώς και τη διαχείριση παρτίδων εργασιών και την εκτέλεσή τους σε κβαντικές συσκευές και προσομοιωτές απομακρυσμένης πρόσβασης. Παρακάτω στην Εικόνα 4.1.1, φαίνεται ένα απλό παράδειγμα Qiskit Terra. Σε αυτό, δημιουργείται ένα κβαντικό κύκλωμα για δύο qubits, το οποίο αποτελείται από τις κβαντικές πύλες που απαιτούνται για τη δημιουργία της κατάστασης διεμπλοκής αυτών. Το κβαντικό κύκλωμα, στη συνέχεια, εκτελείται είτε σε προσομοιωτή, με τον οποίο εξάγουμε ένα bit πληροφορίας για κάθε qubit ή σε πραγματικό κβαντικό μηχάνημα. Αυτό το παράδειγμα θα αποτελέσει και το πρώτο μας κβαντικό κύκλωμα, όπου θα περιγραφούν πιο λεπτομερέστερα τα βήματα και ο τρόπος δημιουργίας αυτού.

```
In [24]: circuit.h(qr[0])
circuit.cx(qr[0], qr[1])
circuit.measure(qr, cr)
circuit.draw(output='mpl')
```

Out[24]:



Εικόνα 4.1.1: Αναπαράσταση δημιουργίας κβαντικού κυκλώματος

• Qiskit Aer:

Το στοιχείο Aer παρέχει κβαντικούς προσομοιωτές υπολογιστών υψηλής απόδοσης με ρεαλιστικά μοντέλα θορύβου. Βραχυπρόθεσμα, η ανάπτυξη κβαντικού λογισμικού θα εξαρτηθεί σε μεγάλο βαθμό από την προσομοίωση μικρών κβαντικών συσκευών. Για το Qiskit, αυτό παρέχεται από το στοιχείο Aer. Αυτό παρέχει προσομοιωτές που φιλοξενούνται τοπικά στη συσκευή του χρήστη, καθώς και πόρους υπερυπολογιστών που διατίθενται μέσω του νέφους (cloud). Οι προσομοιωτές μπορούν, επίσης, να προσομοιώσουν τα αποτελέσματα του θορύβου για απλά και εξελιγμένα μοντέλα θορύβου. Αυτό σημαίνει, πως συνεχίζοντας με το προηγούμενο παράδειγμα, αφού δημιουργηθεί το κβαντικό κύκλωμα, μπορεί έπειτα να εκτελεστεί σε ένα backend (είτε κβαντικό υλικό ή προσομοιωτή). Μέσα απ' αυτό θα δούμε και άλλες έννοιες, όπως τους providers, τα jobs και τα αποτελέσματα που βγάζουμε με αυτά.

• Qiskit Ignis & Aqua:

Πλέον αυτά τα στοιχεία έχουν καταργηθεί από τη πλατφόρμα, ενώ χρησιμοποιούνταν για συγκεκριμένους τομείς στην κβαντική υπολογιστική, όπως μείωση θορύβου κβαντικών υπολογιστών ή για τη βοήθεια και συσχέτιση με κλάδους, όπως η Φυσική & Χημεία, η μηχανική εκμάθηση, χρηματοοικονομική κτλπ. Πλέον, υπάρχουν διαφορετικά APIs για, σχεδόν, κάθε τομέα, όπως το Qiskit Machine Learning, Qiskit Nature, Qiskit Finance κ.α.

4.2. Υλοποίηση ενός ολοκληρωμένου κυκλώματος

Η βασική ροή εργασιών που χρησιμοποιεί το Qiskit χωρίζεται σε τέσσερα στάδια:

- **Build:** Σχεδιάζουμε ένα κβαντικό κύκλωμα που αντιπροσωπεύει το πρόβλημα που εξετάζουμε.
- **Compile:** Μεταγλώττιση κυκλωμάτων για μια συγκεκριμένη κβαντική υπηρεσία, π.χ. ένα κβαντικό σύστημα ή κλασσικός προσομοιωτής.
- **Run:** Εκτελούμε τα μεταγλωττισμένα κυκλώματα στις καθορισμένες κβαντικές υπηρεσίες. Αυτές οι υπηρεσίες μπορεί να βασίζονται είτε στο νέφος ή τοπικά, ψάχνοντας για διάφορα backends.
- **Analyze:** Υπολογίζουμε συνοπτικά στατιστικά στοιχεία και οπτικοποιούμε τα αποτελέσματα των πειραμάτων.

Ακολουθεί ένα παράδειγμα υλοποίησης και μέτρησης της κβαντικής κατάστασης διεμπλοκής δύο qubits, προκειμένου να παρουσιαστούν τα βασικά στοιχεία προγραμματισμού του Qiskit. Η ροή εργασίας που ακολουθούμε είναι η εξής:

- Εισαγωγή πακέτων
- Αρχικοποίηση μεταβλητών
- Προσθήκη πύλων
- Οπτικοποίηση κυκλώματος
- Προσομοίωση του πειράματος
- Οπτικοποίηση των αποτελεσμάτων

Τα βασικά στοιχεία που χρειάζονται για το πρόγραμμά μας εισάγονται ως εξής:

```
In [2]: import numpy as np
from qiskit import QuantumCircuit
from qiskit.providers.aer import QasmSimulator
from qiskit.visualization import plot_histogram
```

Εικόνα 4.2.1 (α): Εισαγωγή πακέτων στο Qiskit

Βέβαια, μπορούμε και πιο απλούστερο τρόπο να εισάγουμε πακέτα ως εξής:

```
In [2]: from qiskit import *
```

Εικόνα 4.2.1 (β): Εισαγωγή πακέτων στο Qiskit

Πιο αναλυτικά, οι εισαγωγές αυτές σημαίνουν:

- **QuantumCircuit**: μπορεί να θεωρηθεί ως οι οδηγίες του κβαντικού συστήματος. Κρατάει όλες τις κβαντικές λειτουργίες μας.
- **QasmSimulator**: είναι ο προσομοιωτής κυκλώματος υψηλής απόδοσης Aer.
- **plot_histogram**: δημιουργεί ιστογράμματα.

Στην συνέχεια, εξετάζουμε την επόμενη γραμμή κώδικα

```
In [21]: # circuit = QuantumCircuit(qr, cr)
circuit = QuantumCircuit(2, 2)
```

Εικόνα 4.2.2: Αρχικοποίηση μεταβλητών στο Qiskit

Το κύκλωμά μας μπορεί να χωριστεί σε 2 καταχωρητές. Ο ένας είναι ο κβαντικός και ο άλλος ο κλασσικός καταχωρητής. Στο κβαντικό αντιστοιχούμε τα qubits και εκεί καταχωρούμε τις κβαντικές πύλες που θα χρησιμοποιήσουμε. Αντίστοιχα, στο κλασσικό αντιστοιχούμε τα bits, όπου με αυτό το τρόπο θα αποθηκεύσουμε και το αποτέλεσμα από τις μετρήσεις μας. Η παραπάνω εντολή στην Εικόνα 4.2.2 μεταφράζεται και, όπως φαίνεται στην Εικόνα 4.2.3:

```
In [3]: qr = QuantumRegister(2)
```

```
In [4]: cr = ClassicalRegister(2)
```

```
In [21]: circuit = QuantumCircuit(qr, cr)
```

Εικόνα 4.2.3: Εναλλακτική αρχικοποίηση κυκλώματος στο Qiskit

Στην μεταβλητή qr αρχικοποιούνται δύο qubits στην βασική κατάσταση $|0\rangle$, ενώ αντίστοιχα, στην μεταβλητή cr αρχικοποιούνται δύο bits, όπου και αποθηκεύονται τα αποτελέσματα των μετρήσεων για κάθε qubit, αντίστοιχα. Τέλος, στην μεταβλητή circuit δημιουργείται το κβαντικό κύκλωμα με τα περιεχόμενα του κβαντικού καταχωρητή και του κλασσικού. Έτσι και, στην Εικόνα 4.2.2 φαίνεται η απλοποίηση αρχικοποίησης με σύγκριση την Εικόνα 4.2.3, όπου οι αριθμοί της παρένθεσης υποδηλώνουν απλά το μέγεθος των καταχωρητών.

Μετά την αρχικοποίηση, συνέχεια έχει, η προσθήκη των πυλών (λειτουργίες) για να χειριστούμε τους καταχωρητές του κυκλώματός μας. Εξετάζουμε, λοιπόν, τις ακόλουθες δύο γραμμές κώδικα στην Εικόνα 4.2.6:

```
In [24]: circuit.h(qr[0])
          circuit.cx(qr[0], qr[1])
```

Εικόνα 4.2.4: Υλοποίηση των κβαντικών πυλών

Ο παραπάνω κώδικας ισχύει για τις ακόλουθες πύλες:

- **circuit.h(qr[0])** ή και **circuit.h(0)**: Μια πύλη Hadamard στο qubit 0, η οποία τη βάζει σε κατάσταση υπέρθεσης.
- **circuit.cx(qr[0], qr[1])** ή και **circuit.cx(0, 1)**: Μια λειτουργία ελεγχόμενου-ΟΧΙ στο qubit ελέγχου 0 και στο qubit στόχου 1, που βάζει τα qubits σε μια διεμπλεκόμενη κατάσταση (entangled state).

Μόλις υλοποιήσουμε τις επιθυμητές κβαντικές πύλες ή λειτουργίες, μπορούμε να μετρήσουμε τα qubits με τη παρακάτω εντολή:

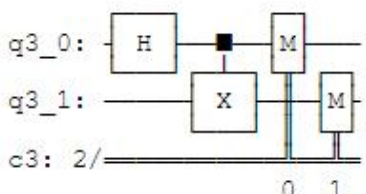
```
In [24]: circuit.measure(qr, cr)
```

Εικόνα 4.2.5: Μέτρηση κυκλώματος

Έτσι, εάν περάσουμε ολόκληρους τους κβαντικούς και τους κλασικούς καταχωρητές για μέτρηση, το αποτέλεσμα της μέτρησης του n qubit θα αποθηκευτεί στο αντίστοιχο n κλασικό bit.

Έπειτα, μπορούμε να οπτικοποιήσουμε το κύκλωμά μας. Μπορούμε να χρησιμοποιήσουμε το `qiskit.circuit.QuantumCircuit.draw()` για να προβάλουμε το κύκλωμα που έχουμε σχεδιάσει με τις διάφορες μορφές που χρησιμοποιούνται σε πολλά σχολικά βιβλία και ερευνητικά άρθρα, όπως στην παρακάτω εικόνα:

```
In [36]: circuit.draw()
Out[36]:
```

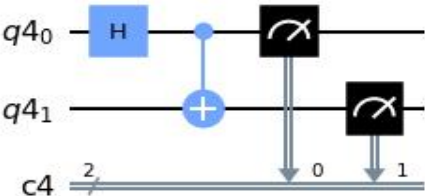


Εικόνα 4.2.6: Χρήση της draw συνάρτησης

Βέβαια για την καλύτερη αναπαράσταση των κυκλωμάτων μας θα χρησιμοποιούμε το εξής format:

```
In [39]: circuit.h(qr[0])
circuit.cx(qr[0], qr[1])
circuit.measure(qr, cr)
circuit.draw(output='mpl')
```

Out[39]:



Εικόνα 4.2.7: Χρήση ενός matplotlib πακέτου

Για να λειτουργήσει ισώς χρειαστεί τις παρακάτω εντολές:

```
In [44]: %matplotlib inline
```

```
In [45]: pip install pylatexenc
```

Εικόνα 4.2.8: Εγκατάσταση πακέτων

Πλέον, έχουμε ακολουθήσει τα περισσότερα βήματα της συνολικής ροής εργασίας και στο επόμενο υποκεφάλαιο θα μελετήσουμε πως μπορούμε να προσομοιώσουμε το κύκλωμά μας σε πραγματικό κβαντικό υπολογιστή που παρέχει η IBM, καθώς και να αναλύσουμε τα αποτελέσματα που βγαίνουν απ' αυτό.

4.3. Προσομοίωση & χρήση κβαντικού υπολογιστή

Πλέον, μένει να τελειοποιήσουμε το κύκλωμά μας, εκτελώντας το σε προσομοιωτή και, έπειτα σε ένα κβαντικό υπολογιστή. Προηγουμένως υλοποιήσαμε ένα κύκλωμα με δύο qubits και 2 κλασσικά bits. Πρώτα θα ξεκινήσουμε, εκτελώντας το κύκλωμα σε προσομοίωση και, έπειτα σε πραγματικό κβαντικό υπολογιστή. Για να εκτελέσουμε τον προσομοιωτή, θα πρέπει να θυμηθούμε το module Aer που αναφέραμε σε προηγούμενο υποκεφάλαιο και να εκτελέσουμε την παρακάτω εντολή:

```
In [23]: simulator = Aer.get_backend('qasm_simulator')
```

Εικόνα 4.3.1: Εκτέλεση προσομοιωτή στο Qiskit

Βέβαια, πως μπορούμε να γνωρίζουμε σε ποιο προσομοιωτή να τρέξουμε το κύκλωμα; Υπάρχει ένας τρόπος μέσα από το Quantum Composer της IBM Quantum Experience, όπου όλα λειτουργούν με drag-and-drop λογική, όπου και υπάρχει λίστα των διαθέσιμων προσομοιωτών, αλλά στην παρούσα φάση θα παρουσιάσουμε ένα πρακτικό τρόπο μέσα από το Qiskit, σε ένα διαφορετικό jupyter notebook, για να μας εμφανίζει την λίστα διαθέσιμων προσομοιωτών, χωρίς να χρησιμοποιούμε διαφορετικές πλατφόρμες τη φορά. Όλα αυτά θα αναλυθούν στη συνέχεια, μόλις είμαστε σε θέση να οπτικοποιήσουμε τα αποτελέσματα, σε μορφή γραφήματος, του κυκλώματός μας.

Χρησιμοποιήσουμε την execute συνάρτηση που παρέχει το Qiskit, όπως φαίνεται παρακάτω, για να εκτελέσουμε το κύκλωμα με το προσομοιωτή που μόλις δημιουργήσαμε. Φυσικά, αποθηκεύουμε τα αποτελέσματα σε μια μεταβλητή για να τα επιστρέψουμε.

```
In [24]: result = execute(circuit, backend = simulator).result()
```

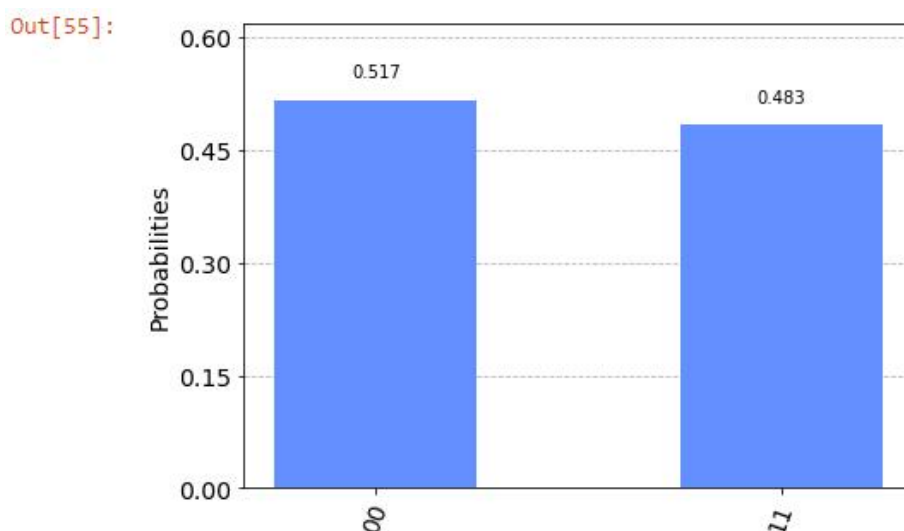
Εικόνα 4.3.2: Εκτέλεση κυκλώματος σε προσομοιωτή

Το επόμενο βήμα είναι να οπτικοποιήσουμε τα αποτελέσματα. Για να το κάνουμε αυτό, χρησιμοποιούμε τη βιβλιοθήκη plot_histogram, όπως φαίνεται παρακάτω και σαν έξοδο θα μας εμφανίσει το γράφημά μας σε μορφή bar chart:

```
In [25]: from qiskit.tools.visualization import plot_histogram
```

```
In [26]: plot_histogram(result.get_counts(circuit))
```

Εικόνα 4.3.3: Οπτικοποίηση αποτελεσμάτων



Εικόνα 4.3.4: Αποτελέσματα κυκλώματος σε κατάσταση διεμπλοκής

Έτσι, βλέπουμε πως τα δύο qubits βρίσκονται σε κατάσταση διεμπλοκής και είναι αλληλοεξαρτώμενα. Αυτό προκύπτει, μιας και υπάρχει μισή πιθανότητα και τα δύο qubits είτε να βρίσκονται στις καταστάσεις $|00\rangle$ ή αντίστοιχα $|11\rangle$. Φυσικά, μέχρι στιγμής, δεν γνωρίζουμε πως θα εκμεταλλευτούμε αυτό το είδος πληροφορίας ή υπολογισμό, αλλά αυτό θα μελετηθεί στα επόμενα κεφάλαια των κβαντικών αλγορίθμων.

Αφού, γνωρίζουμε πως να τρέξουμε ένα κύκλωμα σε προσομοιωτή, σειρά έχει να κατανοήσουμε πως βρίσκουμε τις ιδιότητες και πληροφορίες των κβαντικών υπολογιστών, ώστε να εκτελέσουμε το κύκλωμά μας στον καταλληλότερο. Μπορεί να είναι αρκετά λεπτομερές το κομμάτι αυτό, αλλά θα είναι αρκετά πρακτικό σε μετέπειτα κυκλώματα, όπου θα μπορούμε να χρησιμοποιούμε το νέο jupyter notebook αυτό, για κάθε νέο μας κύκλωμα. Κατά την εγγραφή μας στη πλατφόρμα της IBM Quantum Experience, μας παρέχεται ένα API token, το οποίο και θα χρησιμοποιήσουμε για να συνδέσουμε το τοπικό μηχανήμα μας με το αντίστοιχο κβαντικό υπολογιστή που επιλέξουμε, ώστε να έχουμε τη δυνατότητα να εκτελούμε τους υπολογισμούς των κυκλωμάτων μας σε αυτούς. Το μόνο που χρειάζεται να κάνουμε, παρουσιάζεται στην παρακάτω εικόνα (δημιουργήθηκε ένα αρχείο .txt για λόγους εμπιστευτικότητας του token):

```
In [2]: from qiskit import *

In [3]: IBMQ.save_account(open("ibmapi.txt","r").read())
configrc.store_credentials:WARNING:2022-05-18 22:21:51,178: Credentials already present. Set overwrite=True to overwrite.

In [4]: IBMQ.load_account()

Out[4]: <AccountProvider for IBMQ(hub='ibm-q', group='open', project='main')>
```

Εικόνα 4.3.5: Χρήση του API token της IBM

Αν και, αυτή τη στιγμή δεν θα προχωρήσουμε με την παρακάτω εντολή, δεδομένου ότι και μας εμφανίζει απλά προσομοιωτές, μπορούμε να δούμε απλά ενδεικτικά τον τύπο προσομοιωτών που παρουσιάζονται, όπως για παράδειγμα τον StatevectorSimulator, που μας εμφανίζει τα αποτελέσματα σαν διανύσματα κατάστασης με τη μορφή πινάκων.

```
In [6]: Aer.backends()

Out[6]: [AerSimulator('aer_simulator'),
AerSimulator('aer_simulator_statevector'),
AerSimulator('aer_simulator_density_matrix'),
AerSimulator('aer_simulator_stabilizer'),
AerSimulator('aer_simulator_matrix_product_state'),
AerSimulator('aer_simulator_extended_stabilizer'),
AerSimulator('aer_simulator_unitary'),
AerSimulator('aer_simulator_superop'),
QasmSimulator('qasm_simulator'),
StatevectorSimulator('statevector_simulator'),
UnitarySimulator('unitary_simulator'),
PulseSimulator('pulse_simulator')]
```

Εικόνα 4.3.6: Λίστα προσομοιωτών

Για να αναζητήσουμε τους πραγματικούς κβαντικούς υπολογιστές που βρίσκονται διαθέσιμοι, εκτελούμε την παρακάτω εντολή:

```
In [8]: provider = IBMQ.get_provider("ibm-q")
```

```
In [9]: provider.backends()
```

```
Out[9]: [<IBMQSimulator('ibmq_qasm_simulator') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_armonk') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_santiago') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_bogota') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_lima') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_belem') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_quito') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQSimulator('simulator_statevector') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQSimulator('simulator_mps') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQSimulator('simulator_extended_stabilizer') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQSimulator('simulator_stabilizer') from IBMQ(hub='ibm-q', group='open', project='main')>,
<IBMQBackend('ibmq_manila') from IBMQ(hub='ibm-q', group='open', project='main')>]
```

Εικόνα 4.3.7: Λίστα πραγματικών κβαντικών υπολογιστών

Φυσικά, βλέπουμε τον qasm simulator, καθώς και διάφορους άλλους, όπως santiago, bogota, lima, κτλπ. Πως επιλέγουμε κάποιον απ'αυτούς, όμως και, ποιον απ'όλους; Ποιος είναι ο καταλληλότερος; Αρχικά, έχουμε έναν καλό τρόπο σύγκρισης για να επιλέξουμε τον κατάλληλο στην περίπτωση μας και ανάλογα τη χρονική περίοδο που θέλουμε να εκτελέσουμε το κύκλωμά μας. Βλέπουμε ποσα διαθέσιμα qubits έχει ο κάθε υπολογιστής, καθώς και πόσα εκτελέσιμα jobs υπάρχουν, που αυτό σημαίνει θεωρητικά, πόσοι χρησιμοποιούν τον κβαντικό υπολογιστή που αναζητούμε, δηλαδή μια ουρά αναμονής. Μπορούμε εύκολα να δημιουργήσουμε ένα for loop για κάθε backend στη λίστα αυτή, όπου backend θεωρείται ο κάθε κβαντικός υπολογιστής, ώστε να τυπώσουμε τις ιδιότητες κάθε backend με το όνομα κάθε κβαντικού υπολογιστή, τα jobs του και τα διαθέσιμα qubits σε ομοιόμορφη διάταξη. Το block κώδικα παρουσιάζεται στην παρακάτω εικόνα:

```
In [10]: provider = IBMQ.get_provider("ibm-q")

for backend in provider.backends():
    try:
        qubit_count = len(backend.properties().qubits)
    except:
        qubit_count = "simulated"
    print(f"{backend.name()} : {backend.status().pending_jobs} & {qubit_count} qubits ")

ibmq_qasm_simulator : 0 & simulated qubits
ibmq_armonk : 0 & 1 qubits
ibmq_santiago : 1 & 5 qubits
ibmq_bogota : 1 & 5 qubits
ibmq_lima : 1 & 5 qubits
ibmq_belem : 0 & 5 qubits
ibmq_quito : 0 & 5 qubits
simulator_statevector : 0 & simulated qubits
simulator_mps : 0 & simulated qubits
simulator_extended_stabilizer : 0 & simulated qubits
simulator_stabilizer : 0 & simulated qubits
ibmq_manila : 5 & 5 qubits
```

Εικόνα 4.3.8: Εμφάνιση λίστας κβαντικών υπολογιστών με ομοιόμορφο τρόπο

Αναλυτικά, χρησιμοποιούμε το `try statement` για τα `error handlings` και σε αυτό θέλουμε να μας εμφανίζεται το πλήθος των qubits που έχει ο κάθε κβαντικός υπολογιστής. Αν δεν υπάρχουν, τότε στο `except statement` θα μας εμφανίζει πως αυτός ο υπολογιστής δεν λειτουργεί πλέον σαν κβαντικός, αλλά αν τον χρησιμοποιήσουμε, θα μας επιστρέψει μια απλή προσομοίωση των αποτελεσμάτων μας. Τέλος, απλώς τυπώνουμε τις ιδιότητες των υπολογιστών αυτών, συμπεριλαμβάνοντας το όνομα του υπολογιστή, τα jobs, κτλπ. Με τη παρούσα λίστα που εμφανίζεται στην παραπάνω εικόνα (Εικόνα 4.3.8), θα πρέπει να επιλέξουμε έναν κβαντικό υπολογιστή που δεν θεωρείται `simulated` και όσο μικρότερος ο αριθμός των jobs, τόσο πιο γρήγορα θα πάρουμε την πρόσβαση σε αυτόν για να εκτελεστούν τα αποτελέσματά μας. Βέβαια, σε αυτή τη περίπτωση οι περισσότεροι δεν είναι κατειλημμένοι και θα μπορούσαμε να επιλέξουμε τους περισσότερους απ'αυτούς, όπως ο `santiago`, `bogota`, `lima`, κτλπ.

Τώρα, αφού μπορούμε να αναζητήσουμε το κατάλληλο κβαντικό υπολογιστή, θα εκτελέσουμε το κύκλωμα που δημιουργήσαμε προηγουμένως. Η επιλογή είναι τυχαία μιας και δεν υπάρχουν `pending jobs`. Έτσι, εκτελούμε τον κώδικα στο ίδιο `jupyter notebook` με το κύκλωμά μας, όπως παρακάτω:

```
In [8]: IBMQ.load_account()

Out[8]: <AccountProvider for IBMQ(hub='ibm-q', group='open', project='main')>

In [9]: provider = IBMQ.get_provider('ibm-q')

In [10]: qcomp = provider.get_backend('ibmq_lima')
```

Εικόνα 4.3.9: Επιλογή κβαντικού υπολογιστή

Φυσικά, ακόμη και να επιλέξουμε το λιγότερο απασχολημένο backend, είναι πιθανό και πάλι να χρειαστεί αρκετός χρόνος για να εκτελεστεί το κύκλωμά μας. Γι'αυτό, θα χρειαστεί να γνωρίζουμε την ουρά αναμονής μας και το κάνουμε με τις παρακάτω εντολές:

```
In [11]: job = execute(circuit, backend=qcomp)

In [12]: from qiskit.tools.monitor import job_monitor

In [13]: job_monitor(job)

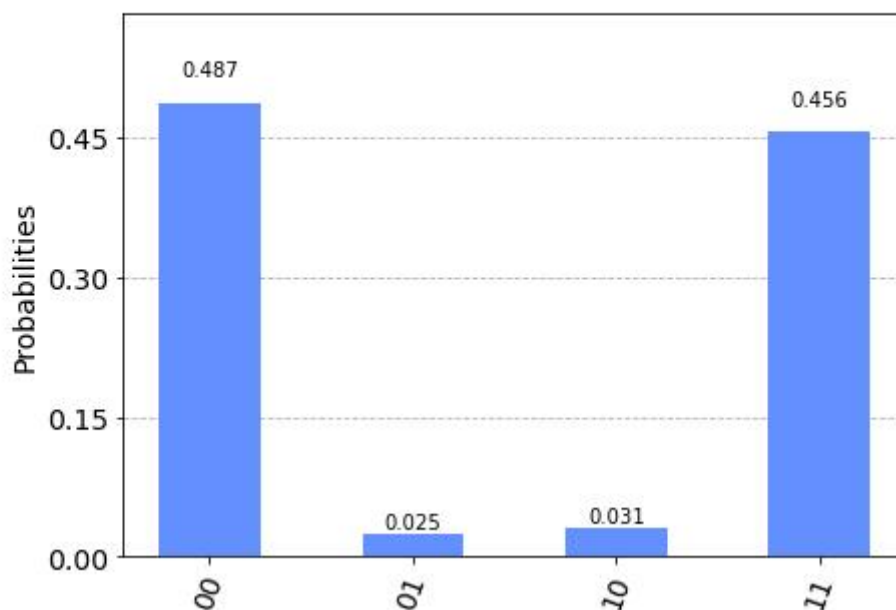
Job Status: job has successfully run
```

Εικόνα 4.3.10: Ενημέρωση ουράς αναμονής

Αφού εκτελεστεί το κύκλωμά μας, μπορούμε να οπτικοποιήσουμε τα αποτελέσματά μας, όπως στην παρακάτω εικόνα. Αυτό που βλέπουμε είναι ότι οι κβαντικοί υπολογιστές δεν είναι τέλειοι, μιας και δεν βλέπουμε αποκλειστικά τις καταστάσεις 00 και 11, όπως στην προσομοίωση πριν. Αυτό οφείλεται στους λεγόμενους κβαντικούς θορύβους, όπου οι κβαντικοί υπολογιστές μπορεί να εμφανίσουν σημάδια θερμότητας ή σκόνης κτλπ και, συνεπώς τα qubits είναι πιθανό να χάνουν τις ιδιότητές τους. Και πάλι, όμως, παρατηρούμε ότι τα αποτελέσματα κλίνουν στις καταστάσεις 00 και 11. Γι' αυτό, είναι σημαντικό να γνωρίζουμε πως ένας κβαντικός υπολογιστής δεν θα μας δώσει τέλεια αποτελέσματα και θα εμφανίζει κάτι που συνήθως δεν αναμένουμε, αλλά αυτή τη στιγμή μπορούν να αγνοηθούν πλήρως χωρίς να επηρεάσουν την ανάλυσή μας.

```
In [14]: result = job.result()  
         plot_histogram(result.get_counts(circuit))
```

Out[14]:



Εικόνα 4.3.11: Αποτελέσματα κατάστασης διεμπλοκής σε κβαντικό υπολογιστή

Κεφάλαιο 5

Κβαντική Τηλεμεταφορά

Στο κεφάλαιο αυτό θα περιγραφεί η έννοια της κβαντικής τηλεμεταφοράς, υλοποιώντας το αντίστοιχο κβαντικό κύκλωμα και στο Qiskit. Θα εξηγηθούν βασικές έννοιες, όπως η φάση και η σφαίρα Bloch για την ανάλυση των διανυσμάτων κατάστασης, όπως και τι σημαίνει superdense coding. Έχοντας κατανοήσει τα προαπαιτούμενα των προηγούμενων κεφαλαίων, είναι στιγμή να προχωρήσουμε σε αυτό το σημαντικό αλγόριθμο της κβαντικής τηλεμεταφοράς.

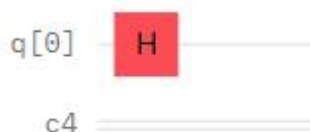
5.1. Φάση και σφαίρα Bloch

Για να κατανοήσουμε την έννοια της φάσης (phase), θα επικεντρωθούμε στην κβαντική πύλη Pauli Z. Η πύλη αυτή θα μας οδηγήσει στην έννοια αυτή και θα μας βοηθήσει να τη καταλάβουμε με τα διάφορα παραδείγματα που θα ακολουθήσουν. Μελετήσαμε στο κεφάλαιο 3.2.3, όπου περιγράφονται οι κβαντικές πύλες Pauli και είδαμε τον τελεστή και πίνακα αλήθειας αυτής, αλλά δεν την χρησιμοποιήσαμε ακόμη.

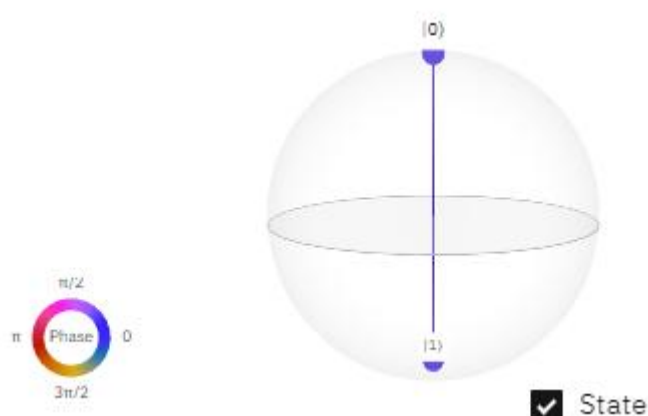
Η πύλη αυτή συσχετίζεται με το phase flip, όπου ισχύει:

$$\begin{aligned}|0\rangle &\rightarrow |0\rangle \\ |1\rangle &\rightarrow -|1\rangle\end{aligned}$$

Μπορούμε να το ελέγξουμε αμέσως μέσα από τη πλατφόρμα της IBM, στο IBM Composer. Αν προσθέσουμε μια Hadamard πύλη, γνωρίζουμε πως θα φέρει το qubit σε κατάσταση υπέρθεσης, όπως φαίνεται στην Εικόνα 5.1.2:



Εικόνα 5.1.1: Σύμβολο Hadamard πύλης στο IBM Composer

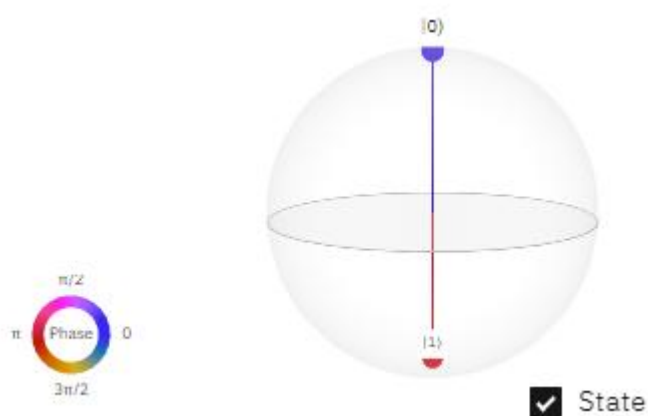


Εικόνα 5.1.2: Q-Sphere

Παρατηρούμε πως το qubit μπορεί να βρεθεί στη κατάσταση $|0\rangle$ ή $|1\rangle$, όπως και το χρώμα που βρίσκεται στο μπλε προσδιορίζει τη γωνία $\varphi = 0$. Αν προσθέσουμε τη πύλη Z, θα δούμε τα αποτελέσματα στην Εικόνα 5.1.4.



Εικόνα 5.1.3: Συνδυασμός H και Z πυλών

Εικόνα 5.1.4: Αλλαγή φάσης σε π

Η φάση αλλάζει από το 0 σε π , όπου υπάρχει και εμφανής αλλαγή του χρώματος από μπλε σε κόκκινο. Αυτό μπορεί να μην φαίνεται χρήσιμο, μιας και οι πιθανότητες δεν αλλάζουν για την κατάσταση του qubit. Ακόμη, παραμένει στις ίδιες πιθανότητες να βρεθεί σε κατάσταση $|0\rangle$ ή $|1\rangle$. Στην πραγματικότητα, η πύλη αυτή είναι πολύ σημαντική και, για αυτό το λόγο σε ένα νέο jupyter notebook, θα ξεκινήσουμε την

πρώτη εισαγωγή στη σφαίρα Bloch, όπου θα μας εμφανίσει στον τρισδιάστατο χώρο την θέση και τη φορά του qubit, με μερικά παραδείγματα.

Θα ξεκινήσουμε, εισάγοντας τις απαραίτητες βιβλιοθήκες, για να μπορούμε, επίσης, να εμφανίσουμε τα απαραίτητα γραφήματα, όπως φαίνεται στην Εικόνα 5.1.5:

```
In [1]: from qiskit import *
        from qiskit.tools.visualization import plot_bloch_multivector
        from qiskit.visualization import plot_histogram
        %matplotlib inline
        import math
```

Εικόνα 5.1.5: Εισαγωγή απαραίτητων βιβλιοθηκών για αναπαράσταση διανύσματος

Στην συνέχεια, αποθηκεύουμε σε αντίστοιχες μεταβλητές τους δύο προσομοιωτές που θα χρειαστούμε, γιατί θα παρουσιαστούν ενδεικτικά παραδείγματα με δυο qubits, οι οποίοι έχουν μάλιστα αναφερθεί. Στόχος είναι να συγκριθούν τα δύο αυτά qubits με ενδεικτικά παραδείγματα, παρατηρώντας τη φάση γωνίας τους στο τρισδιάστατο χώρο. Γι'αυτό θα χρειαστεί να δημιουργήσουμε μια συνάρτηση που θα παίρνει σαν είσοδο το κύκλωμά μας. Στην ουσία, θα παίρνουμε τα αποτελέσματα του statevector και του qasm simulator για n qubits (με τη βοήθεια της circuit.num_qubits ιδιότητας) Βέβαια, εμείς θα χρησιμοποιούμε δύο qubits στην παρούσα φάση, αλλά δεν χρειάζεται να είναι hardcoded. Τέλος, δημιουργούμε το κύκλωμά μας και το εμφανίζουμε.

```
In [2]: qasm_simulator = Aer.get_backend('qasm_simulator')
        statevector_simulator = Aer.get_backend('statevector_simulator')

In [3]: def run_on_simulators(circuit):
        statevec_job = execute(circuit, backend=statevector_simulator)
        result = statevec_job.result()
        statevec = result.get_statevector()

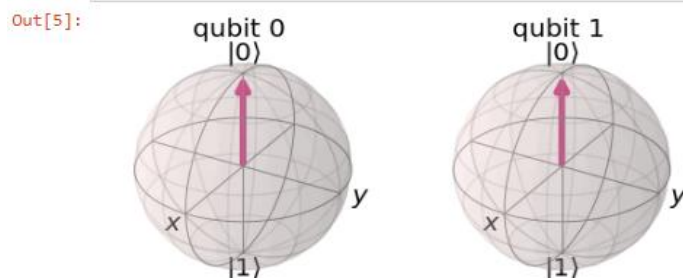
        num_qubits = circuit.num_qubits
        circuit.measure([i for i in range(num_qubits)], [i for i in range(num_qubits)])

        qasm_job = execute(circuit, backend=qasm_simulator, shots=1024).result()
        counts = qasm_job.get_counts()

        return statevec, counts

In [4]: circuit = QuantumCircuit(2,2)
        statevec, counts = run_on_simulators(circuit)

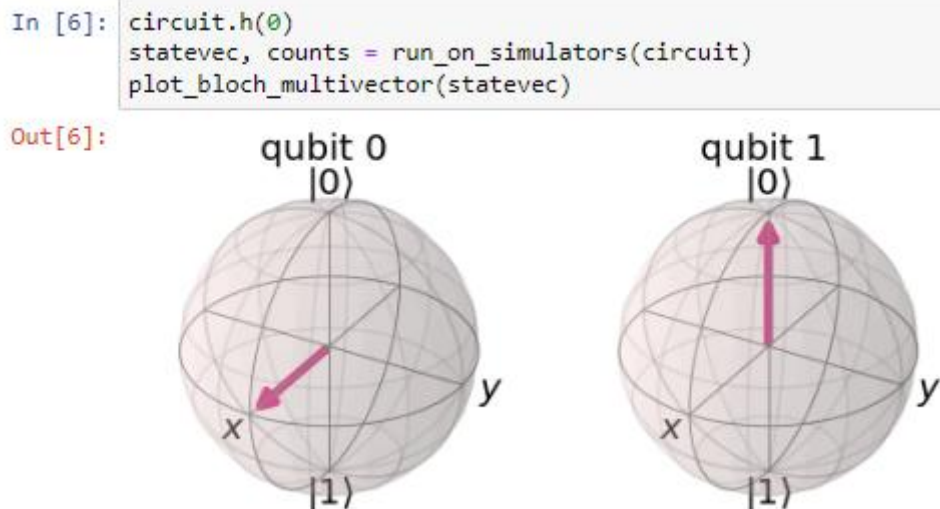
In [5]: plot_bloch_multivector(statevec)
```



Εικόνα 5.1.6: Αναπαράσταση της σφαίρας Bloch με 2 qubits

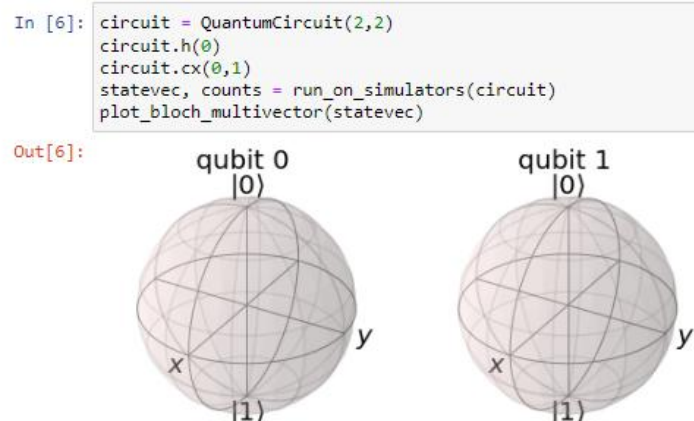
Πλέον, μπορούμε να δούμε την αναπαράσταση με τα δύο qubits, όπου βρίσκονται στη κατάσταση $|0\rangle$, μιας και δεν αλλάξαμε κάτι πάνω τους.

Ας ξεκινήσουμε με ένα παράδειγμα, όπου προσθέτουμε μια Hadamard πύλη στο qubit 0, χωρίς να δράσουμε στο qubit 1, όπως φαίνεται στην Εικόνα 5.1.7:



Εικόνα 5.1.7: Αναπαράσταση κατάστασης υπέρθεσης σε σφαίρα Bloch

Μόλις προσθέτουμε την πύλη, παρατηρούμε πως αλλάζει θέση προς τον άξονα x. Αυτό δεν συνέβη στην Q-Sphere που είδαμε προηγουμένως, το οποίο απλά μας εμφάνιζε τις πιθανότητες να εμφανίσει $|0\rangle$ ή $|1\rangle$ τιμή. Σε αυτή τη περίπτωση, βλέπουμε το πραγματικό τρόπο λειτουργίας ενός qubit με μαθηματικό ή γεωμετρικό τρόπο. Αντίθετα, το qubit 1 δεν δέχτηκε κάποια αλλαγή, οπότε θα είναι πάντα στη κατάσταση $|0\rangle$. Βέβαια, αν προσθέσουμε μια CNOT πύλη, ώστε να οδηγηθούμε σε κατάσταση διεμπλοκής, τότε η σφαίρα Bloch δεν είναι η κατάλληλη για την αναπαράσταση της πληροφορίας.



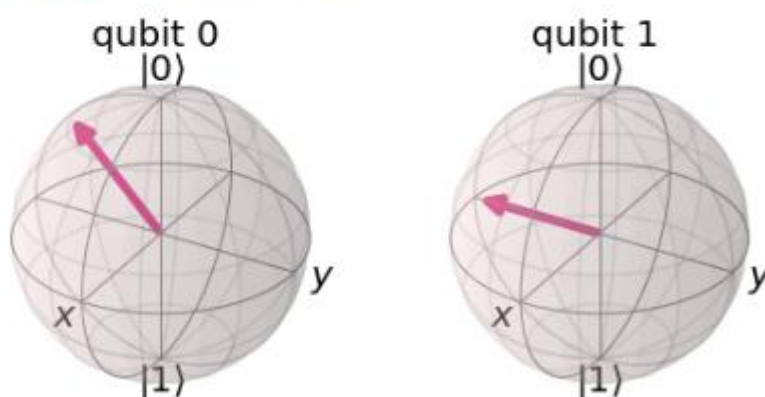
Εικόνα 5.1.8: Κατάσταση διεμπλοκής στη σφαίρα Bloch

Σε τέτοιες περιπτώσεις είναι χρήσιμες οι αναπαραστάσεις είτε του Q-Sphere ή ενός histogram. Αυτό είναι λογικό, μιας και δεν αντιστοιχεί με την πραγματικότητα, γιατί συμβαίνει μόνο στον μικροσκοπικό κόσμο των σωματιδίων.

Πλέον, είναι καλή ευκαιρία να δούμε παραδείγματα με τη μετατόπιση φάσης στη σφαίρα Bloch. Παρακάτω, επισυνάπτονται ορισμένα παραδείγματα:

```
In [9]: circuit = QuantumCircuit(2,2)
circuit.rx(math.pi/4, 0)
circuit.rx(math.pi / 2, 1)
statevec, counts = run_on_simulators(circuit)
plot_bloch_multivector(statevec)
```

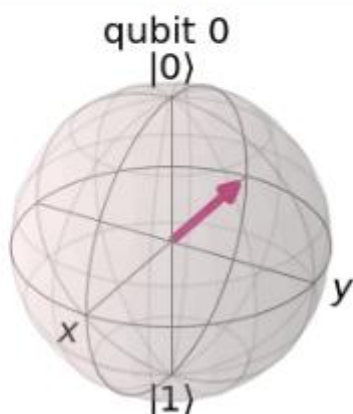
Out[9]:



Εικόνα 5.1.9: Διαφορετική τιμή γωνίας για μετατόπιση φάσης των 2 qubits

```
In [12]: circuit = QuantumCircuit(1,1)
circuit.h(0)
circuit.z(0)
statevec, counts = run_on_simulators(circuit)
plot_bloch_multivector(statevec)
```

Out[12]:



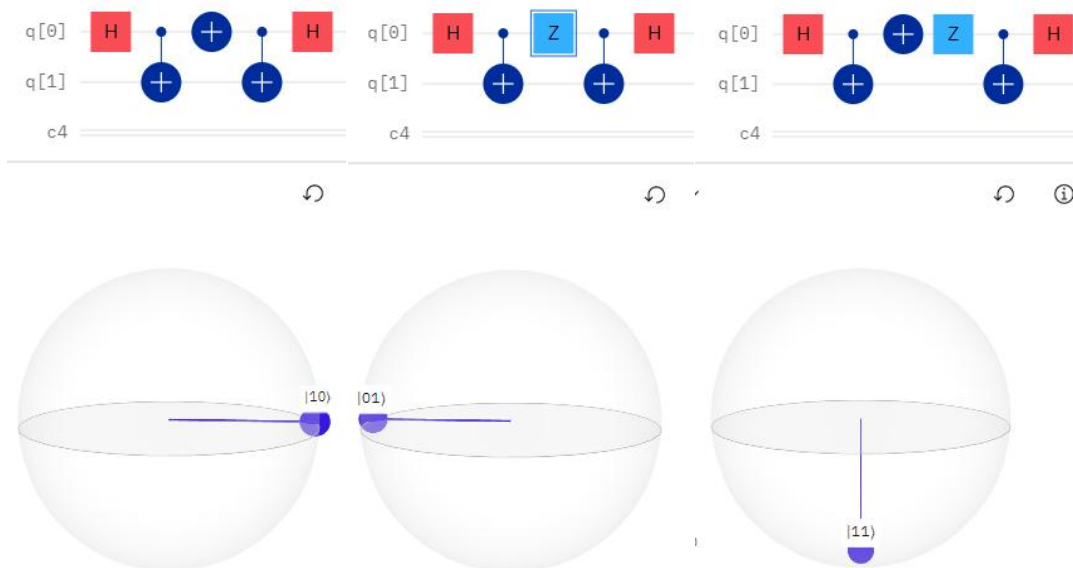
Εικόνα 5.1.10: Αναπαράσταση προσθήκης H και Z πυλών στη σφαίρα Bloch

5.2. Superdense coding

Superdense coding θεωρείται η αποθήκευση δύο bits πληροφορίας σε ένα qubit. Βέβαια, αυτή η δήλωση δεν είναι ακριβώς σωστή. Θεωρητικά μπορούμε να φέρουμε ένα qubit σε κατάσταση υπέρθεσης ή διεμπλοκής με ένα άλλο, αλλά πάντα όταν πρόκειται να το μετρήσουμε, θα αποθηκεύουμε μία μονάχα πληροφορία (0 ή 1), όπως και στα κλασσικά συστήματα. Αυτό που κάνουμε είναι να εκμεταλλευτούμε το πλεονέκτημα της κατάστασης διεμπλοκής, ώστε όταν αλλάζουμε τη τιμή του ενός qubit να χειραγωγούμε και το αποτέλεσμα, χωρίς να επηρεάσουμε το άλλο ^[6].

Όπως και στη βιβλιογραφία της Κρυπτογραφίας, θα αναφερθούμε στην Alice, Eve και Bob. Η Eve θελει να δώσει δύο qubits, ένα στην Alice και ένα στον Bob. Το πρώτο πράγμα που κάνει είναι να φέρει σε κατάσταση διεμπλοκής τα δύο qubits, χρησιμοποιώντας τη Hadamard και CNOT πύλες. Τελικά, μόλις το ένα qubit φτάσει στην Alice θα εφαρμόσει τη δικιά της πύλη για να δράσει στο qubit. Θα εφαρμόσει μια πύλη της επιλογής της, ώστε να επηρεάσει το αποτέλεσμα και να το στείλει στον Bob με τη νέα του κατάσταση. Ο Bob με τη σειρά του, θα λάβει και τα δύο qubits, ένα από την Eve και ένα από την Alice και θα αντιστρέψει το αποτέλεσμα για να το φέρει στην αρχική τους θέση, εφαρμόζοντας μια CNOT και μια Hadamard πύλη. Βέβαια, ανάλογα και με τη πύλη που θα εφαρμόσει η Alice, το αποτέλεσμα θα αλλάξει, απλά χειραγωγώντας το ένα qubit που της δόθηκε από την Eve.

Μπορούμε να δούμε μερικά παραδείγματα από το IBM Composer, με κάποιες πύλες που μπορεί να χρησιμοποιεί η Alice για να αλλάξει και το αποτέλεσμα, όπως φαίνεται στις παρακάτω εικόνες:



Εικόνα 5.2.1: Παραδείγματα Superdense coding

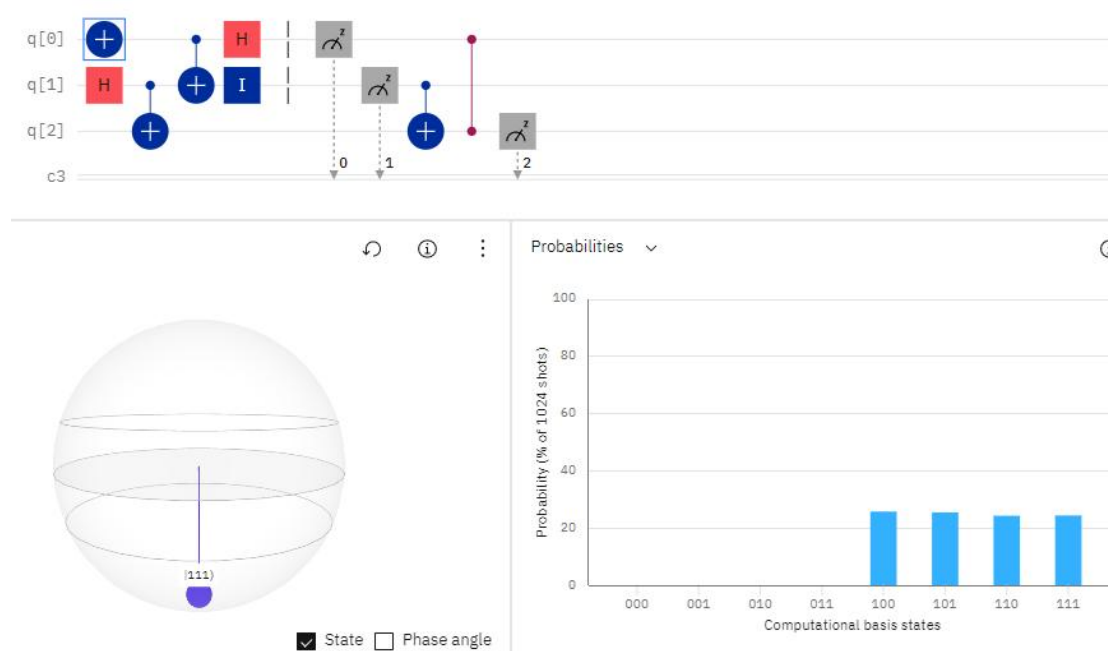
5.3. Ο αλγόριθμος της κβαντικής τηλεμεταφοράς

Η κβαντική τηλεμεταφορά αφορά τον μετασχηματισμό της κατάστασης ενός qubit σε ένα άλλο. Είναι πολύ χρήσιμο να κατανοούμε τον αλγόριθμο αυτό, για τον εξής λόγο. Στα κλασσικά συστήματα μπορούμε να κλωνοποιήσουμε μια πληροφορία, αφού φέρει μονάχα τις δυαδικές τιμές. Όμως, στη περίπτωση των qubits, δεν είναι πάντα εύκολο αυτό, διότι δεν γνωρίζουμε την κατάσταση των qubits πριν την μέτρηση (no clone theorem) και, αφού το μετρήσουμε θα χάσουμε την πληροφορία πριν, δηλαδή όσο ήταν σε κατάσταση υπέρθεσης.

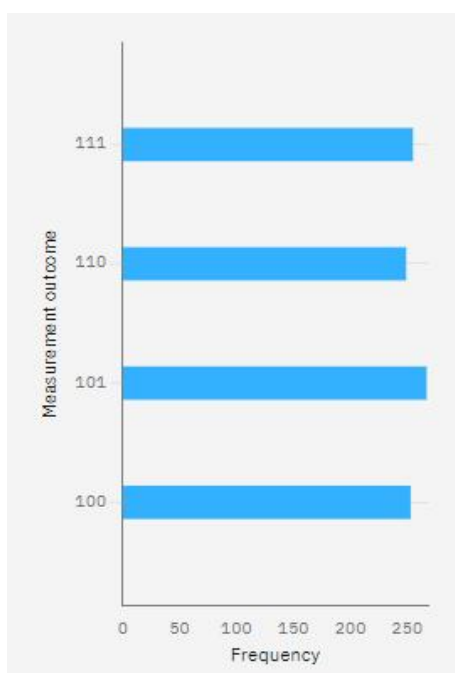
Στη περίπτωση αυτή, η Alice έχει στην κατοχή της ένα qubit σε τυχαία κατάσταση $q_0 = |\psi\rangle$ και ο Bob δύο qubits. Στόχος είναι να μεταφέρουμε τη πληροφορία του qubit που βρίσκεται σε τυχαία κατάσταση $|\psi\rangle$, δηλαδή αυτό που έχει στην κατοχή της η Alice στο δεύτερο qubit που κατέχει ο Bob, q_2 . Το ένα από τα δύο qubits του Bob θα παίζει υποστηρικτικό ρόλο, ως q_1 , για να καταφέρουμε να μεταφέρουμε την πληροφορία στο άλλο qubit. Αρχικά, ο Bob, έχοντας στη κατοχή του δύο qubits, τα φέρει σε κατάσταση διεμπλοκής, εφαρμόζοντας Hadamard και CNOT πύλες. Στη συνέχεια, δίνει το υποστηρικτικό qubit q_1 στην Alice, ώστε με τη σειρά της να εφαρμόσει τις επιθυμητές πύλες της πάνω στο q_0 και q_1 , με στόχο να φέρει τη κατάσταση του q_0 στο q_2 του Bob. Έτσι, η Alice εφαρμόζει μια CNOT πύλη, ώστε να την βοηθήσει να κλωνοποιήσει το q_0 στο q_2 , μετασχηματίζοντας την συνολική κατάσταση των q_0 και q_1 και, στη συνέχεια εφαρμόζει τη Hadamard πύλη. Τέλος, προχωράει στις μετρήσεις και των δύο qubits που κατέχει, q_0 και q_1 . Γιατί, όμως, η Alice προχώρησε στην εφαρμογή αυτών των πυλών; Ο λόγος είναι, διότι μπορεί να φέρει τα qubits σε όλες τις πιθανές καταστάσεις που υπάρχουν, δηλαδή $|00\rangle$, $|01\rangle$, $|10\rangle$ και $|11\rangle$. Πλέον, ανάλογα τα αποτελέσματα των μετρήσεων της Alice, ο Bob θα μπορεί να εφαρμόσει συγκεκριμένες πύλες για να φέρει τα αντίστοιχα αποτελέσματα και, στη συνέχεια, να φέρει το q_2 στη κατάσταση $|\psi\rangle$, όπως ήταν το q_0 . Μαθηματικά, η επεξήγηση αυτή, επαληθεύεται ως εξής:

$$\begin{aligned}
 |\psi\rangle &= \alpha|0\rangle + \beta|1\rangle \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
 &= \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle) \\
 &= \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|101\rangle + \beta|110\rangle) \\
 &= \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \alpha|100\rangle + \alpha|111\rangle + \beta|001\rangle + \beta|010\rangle + \beta|101\rangle + \beta|110\rangle)
 \end{aligned}$$

Μπορούμε να μελετήσουμε, επίσης, και την επίδειξη του αλγορίθμου στο IBM Composer στις Εικόνες 5.3.1 και 5.3.2. Προαιρετικά, εφαρμόζεται μια πύλη X στο q_0 για να βγάζουμε, συνέχεια, το αποτέλεσμα $|1\rangle$ σε αυτό. Επίσης, εφαρμόζουμε πριν την μέτρηση του q_2 , την CZ πύλη που βοηθάει στον μετασχηματισμό και βλέπουμε στα αποτελέσματα πως η τιμή του q_2 δεν αλλάζει, παραμένοντας πάντα στη κατάσταση $|1\rangle$.



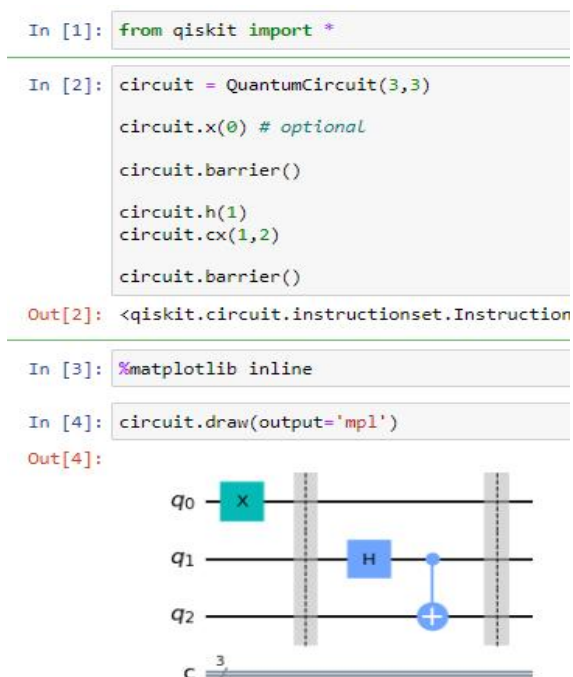
Εικόνα 5.3.1: Κύκλωμα κβαντικής τηλεμεταφοράς



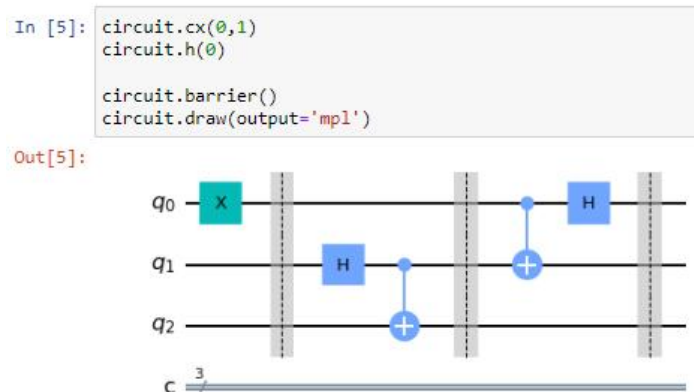
Εικόνα 5.3.2: Αποτελέσματα σε προσομοιωτή

5.4. Υλοποίηση αλγορίθμου στο Qiskit

Πλέον, μπορούμε να υλοποιήσουμε τον αλγόριθμο της κβαντικής τηλεμεταφοράς στο Qiskit. Στην ουσία, επαναλάβουμε την ίδια διαδικασία για την κατασκευή του κυκλώματος και οπτικοποίηση των αποτελεσμάτων. Όλη η διαδικασία παρέχεται σε όλες τις παρακάτω εικόνες, από την Εικόνα 5.4.1 μέχρι και την 5.4.4.



Εικόνα 5.4.1: Εφαρμογή πυλών από μεριάς Bob για κβαντική τηλεμεταφορά

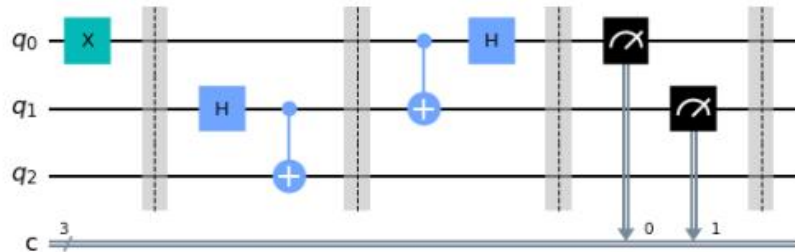


Εικόνα 5.4.2: Εφαρμογή πυλών από μεριάς Alice για κβαντική τηλεμεταφορά

```
In [6]: circuit.measure([0, 1], [0, 1])
```

```
circuit.barrier()
circuit.draw(output='mpl')
```

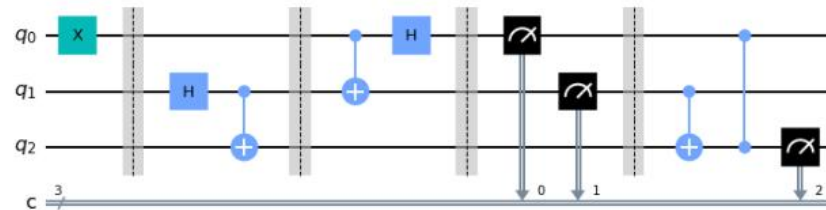
Out[6]:



Εικόνα 5.4.3: Μέτρηση των q_0 και q_1 από μεριάς Alice για κβαντική τηλεμεταφορά

```
In [7]: circuit.cx(1, 2)
circuit.cz(0, 2)
circuit.measure([2], [2])
circuit.draw(output='mpl')
```

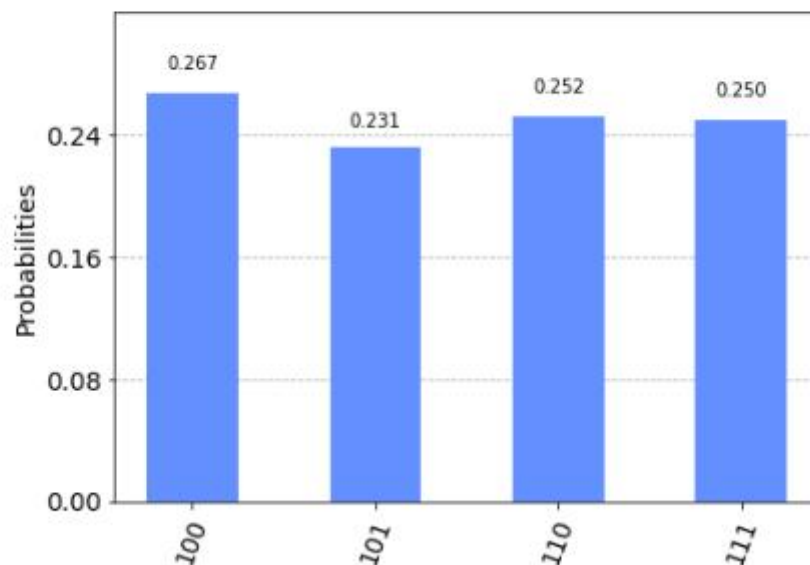
Out[7]:



Εικόνα 5.4.4: Προσθήκη CNOT, CZ από τον Bob για κβαντική τηλεμεταφορά από το q_0 στο q_2

```
In [8]: simulator = Aer.get_backend('qasm_simulator')
result = execute(circuit, backend=simulator, shots=1024).result()
from qiskit.visualization import plot_histogram
plot_histogram(result.get_counts(circuit))
```

Out[8]:



Εικόνα 5.4.5: Αποτελέσματα κβαντικής τηλεμεταφοράς

Κεφάλαιο 6

Κβαντικοί αλγόριθμοι

Στο κεφάλαιο αυτό περιγράφονται ορισμένοι στοιχειώδεις κβαντικοί αλγόριθμοι, όπου έπαιξαν καθοριστικό ρόλο στην εξέλιξη της κβαντικής υπολογιστικής. Το κεφάλαιο θα αποτελείται από τους αλγορίθμους του Bernstein-Vazirani, του Deutsch, του Grover και του Shor, αναλύοντας την λογική πίσω από τον καθένα, καθώς και την υλοποίηση αυτών στο περιβάλλον Qiskit.

6.1. Ο αλγόριθμος Bernstein-Vazirani

Θα δούμε αρκετούς αλγορίθμους και θα τους εντάξουμε σε δύο κατηγορίες. Θα δούμε κάποιους αλγορίθμους που, πραγματικά, θα αλλάξουν τον κόσμο και θα χρησιμοποιούνται σε πραγματικά προβλήματα. Με την τελειοποίηση των κβαντικών υπολογιστών (error free) και την εφαρμογή αυτών των αλγορίθμων είναι εφικτό πως θα αλλάξουν τον τρόπο ζωής μας. Από την άλλη, θα δούμε άλλους αλγορίθμους που θα αποδεικνύουν πως οι κβαντικοί υπολογιστές υπερέχουν από τους κλασσικούς υπολογιστές (κβαντική υπεροχή). Αυτό σημαίνει πως οι κβαντικοί υπολογιστές δεν θα υπερέχουν πάντα των κλασσικών, αλλά θα εξαρτάται από το πρόβλημα. Αν το πρόβλημα μπορεί να λυθεί αποδοτικά από την κατάσταση υπέρθεσης ή διεμπλοκής, τότε θα ήταν εύκολο να χρησιμοποιήσουμε έναν κβαντικό υπολογιστή για να το διεκπεραιώσουμε. Παρ'όλ'αυτά, δεν αφορούν όλα τα προβλήματα τις καταστάσεις αυτές. Δεν υπάρχουν κβαντικοί αλγόριθμοι για κάθε πρόβλημα. Οι κλασσικοί υπολογιστές είναι αποδοτικότεροι σε άλλα προβλήματα, όπως για παράδειγμα σε περιπτώσεις επίλυσης ενός προβλήματος με την τεχνική ωμής βίας (brute force). Οι κβαντικοί υπολογιστές, απλά, εκμεταλλεύονται τις καταστάσεις της υπέρθεσης, διεμπλοκής και παρεμβολής για να λύσουν συγκεκριμένα προβλήματα.

Ο αλγόριθμος Bernstein-Vazirani είναι ένας από τους αλγορίθμους που μπορεί να μας παρουσιάσει πως οι κβαντικοί υπολογιστές υπερέχουν σε μια τέτοια κατάσταση, η οποία θα εξηγηθεί αμέσως. Έστω, λοιπόν, ότι προσπαθούμε να βρούμε έναν αριθμό σε δυαδική μορφή, πχ 1011001. Με τον αλγόριθμο αυτόν, θα προσπαθήσουμε να μαντέψουμε τον αριθμό αυτόν. Για να βρούμε τον αριθμό αυτόν, θα χρειαζόταν 2^7 προσπάθειες, μιας και ο αριθμός αποτελείται από 7 ψηφία, κάτι που ονομάζεται γραμμική αναζήτηση (linear search). Στη περίπτωση των κλασσικών υπολογιστών, θα χρειαστούν μόλις 4 προσπάθειες για να μαντέψει τον αριθμό,

χρησιμοποιώντας την δυαδική αναζήτηση (binary search). Όμως, αυτό δεν είναι αρκετό, αφού ο αλγόριθμος Bernstein-Vazirani μπορεί να μαντέψει τον αριθμό με μία μόνο προσπάθεια! Έτσι, θα ήταν μια πολύ καλή επίδειξη του αλγορίθμου στο Qiskit. Ακολουθούμε τη τυπική ροή εργασίας για την κατασκευή του κυκλώματός μας, εισάγοντας πρώτα τις απαραίτητες βιβλιοθήκες για την αναπαράστασή του. Επίσης, θα αποθηκεύσουμε έναν αριθμό σε μορφή συμβολοσειράς (φυσικά μπορούμε να ζητάμε τον αριθμό από τον χρήστη ή και να μετατρέψουμε την συμβολοσειρά σε integer αριθμό, αλλά στη περίπτωση μας δεν μας απασχολεί). Έπειτα, δημιουργούμε το κύκλωμά μας που θα περιέχει 7+1 qubits και 7 κλασσικά bits, όπου το ένα παραπάνω qubit θα έχει υποστηρικτικό ρόλο.

```
In [1]: from qiskit import *
        %matplotlib inline
        from qiskit.tools.visualization import plot_histogram

In [3]: secretNumber = '1000101'

In [4]: circuit = QuantumCircuit(8,7)
```

Εικόνα 6.1.1: Δημιουργία του κυκλώματος Bernstein-Vazirani χειροκίνητα

Αρχικά, θα υλοποιήσουμε τον αλγόριθμο χειροκίνητα, με την έννοια ότι σύμφωνα με τον αριθμό που θέλουμε ως απάντηση, θα δημιουργήσουμε αντίστοιχα και το κύκλωμα. Με αυτό το τρόπο, πρώτα, θα μπορέσουμε να κατανοήσουμε πως λειτουργεί ο αλγόριθμος και αν δουλεύει τελικά. Στο επόμενο υποκεφάλαιο, θα αυτοματοποιήσουμε τον αλγόριθμο, ώστε να βρίσκει μόνος του τον μυστικό αριθμό. Ο αλγόριθμος εφαρμόζει σε κάθε qubit τις Hadamard πύλες, ενώ στο βοηθητικό qubit εφαρμόζεται πρώτα μια X πύλη και, έπειτα μια Hadamard. Με χειροκίνητο τρόπο εφαρμόζουμε CNOT πύλες στα qubits που πρέπει να φέρουν την κατάσταση 1 και, τέλος εφαρμόζουμε και πάλι σε όλα τα qubits απο μία Hadamard πύλη και οδηγούμαστε στην μέτρησή τους, πέρα του βοηθητικού.

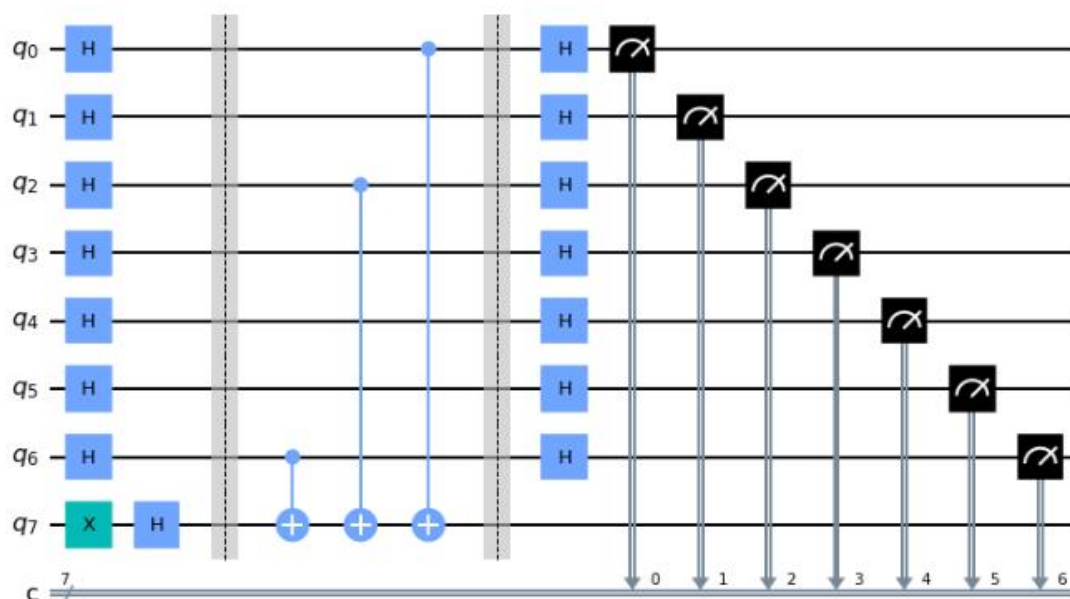
```
In [5]: circuit.h([0,1,2,3,4,5,6])
        circuit.x(7)
        circuit.h(7)

        circuit.barrier()
        circuit.cx(6,7)
        circuit.cx(2,7)
        circuit.cx(0,7)
        circuit.barrier()

        circuit.h([0,1,2,3,4,5,6])
        circuit.measure([0,1,2,3,4,5,6],[0,1,2,3,4,5,6])

        circuit.draw(output='mpl')
```

Εικόνα 6.1.2: Αλγόριθμος Bernstein-Vazirani με χειροκίνητο τρόπο



Εικόνα 6.1.3: Οπτικοποίηση αλγορίθμου Bernstein-Vazirani χειροκίνητα

```
In [5]: simulator = Aer.get_backend('qasm_simulator')
result = execute(circuit, backend=simulator, shots=1).result()
counts = result.get_counts()
print(counts)

{'1000101': 1}
```

Εικόνα 6.1.4: Αποτέλεσμα αλγορίθμου Bernstein-Vazirani χειροκίνητα

Βλέπουμε το αποτέλεσμα πως εμφανίζεται ο επιθυμητός αριθμός με μόνο μια προσπάθεια ($\text{shots} = 1$). Πως, όμως, συμβαίνει αυτό; Προφανώς, εφαρμόζουμε Hadamard και CNOT πύλες για να φέρουμε ίσες πιθανότητες το κάθε qubit να φέρει 0 ή 1, αλλά ποιο ρόλο παίζει το q_7 σαν βοηθητικό; Αν εφαρμόσουμε μονάχα Hadamard στο q_7 δεν είναι καλή ιδέα, μιας και θα μας φέρνει μονάχα αποτελέσματα του μηδενός. Αν δεν εφαρμόσουμε καμία πύλη στο q_7 , τότε θα έχουμε ίσες πιθανότητες να μας εμφανίσει ή μόνο μηδενικά αποτελέσματα στα qubits ή το σωστό αποτέλεσμα, δηλαδή τον αριθμό μας, μιας και εφαρμόζουμε Hadamard πύλες στα qubits. Εφαρμόζοντας X και Hadamard πύλες στο q_7 αποκλείουμε, στην ουσία, το ένα αποτέλεσμα με τα μηδενικά και οδηγούμαστε στο επιθυμητό. Φυσικά, αυτός ήταν ένας χειροκίνητος τρόπος υλοποίησης του αλγορίθμου, αλλά θα προχωρήσουμε στην αυτοματοποίηση αυτού στη συνέχεια.

6.1.1. Αυτοματοποίηση του αλγορίθμου

Κατανοώντας την ροή του αλγορίθμου, σειρά έχει να τον αυτοματοποιήσουμε. Πλέον, μπορούμε να δημιουργήσουμε ένα κύκλωμα που θα προσαρμόζεται από την τιμή του μυστικού αριθμού.

```
In [1]: from qiskit import *
        %matplotlib inline
        from qiskit.tools.visualization import plot_histogram

In [3]: secretNumber = '10101010100'

In [4]: circuit = QuantumCircuit(len(secretNumber)+1, len(secretNumber))
```

Εικόνα 6.1.5: Δημιουργία του κυκλώματος Bernstein-Vazirani αυτοματοποιημένα

Το κύκλωμα θα εφαρμόζει τις αντίστοιχες κβαντικές πύλες ανάλογα το μήκος του μυστικού αριθμού, γι' αυτό και για να δημιουργήσουμε μια σειρά Hadamard πυλών, θα χρησιμοποιήσουμε τη συνάρτηση `range()` με είσοδο το μήκος του μυστικού αριθμού. Μετά ελέγχουμε τα bits αν είναι 1, ώστε να εφαρμοστούν και οι αντίστοιχες CNOT πύλες, αλλά θα πρέπει να ελέγχονται αντίστροφα. Γι' αυτό, χρησιμοποιούμε τη συνάρτηση `enumerate(reversed(secretNumber))`. Η υλοποίηση του κυκλώματος φαίνεται στην Εικόνα 6.1.6, όπως και η οπτικοποίηση στην Εικόνα 6.1.7, παρακάτω.

```
In [5]: circuit.h(range(len(secretNumber)))
        circuit.x(len(secretNumber))
        circuit.h(len(secretNumber))

        circuit.barrier()

        for index, one in enumerate(reversed(secretNumber)):
            print(f"index{index} is {one}")
            if one == "1":
                circuit.cx(index, len(secretNumber))

        circuit.barrier()

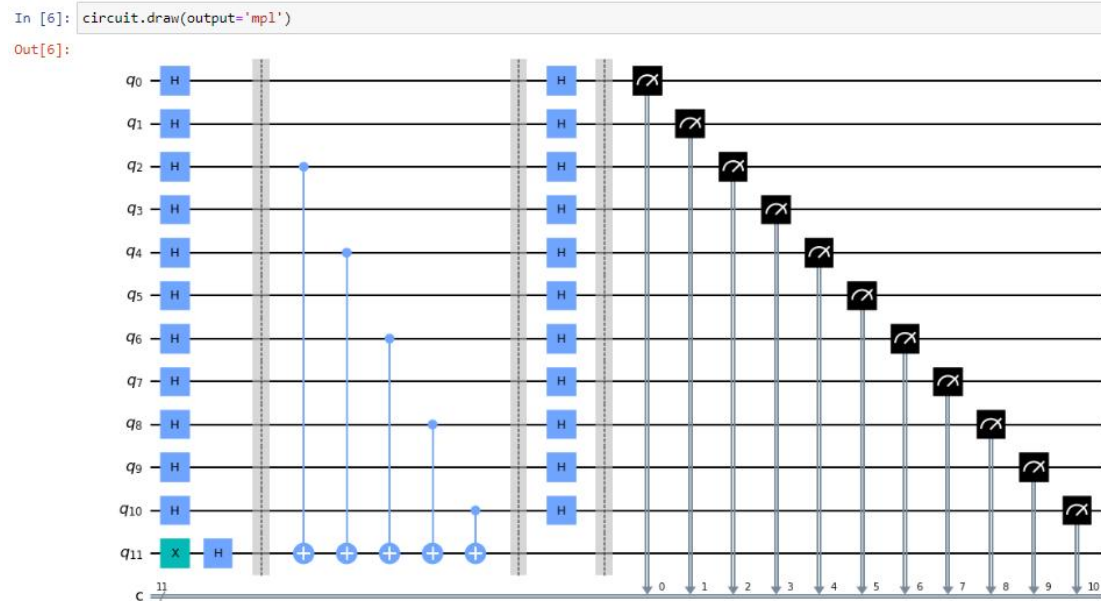
        circuit.h(range(len(secretNumber)))

        circuit.barrier()

        circuit.measure(range(len(secretNumber)), range(len(secretNumber)))

index0 is 0
index1 is 0
index2 is 1
index3 is 0
index4 is 1
index5 is 0
index6 is 1
index7 is 0
index8 is 1
index9 is 0
index10 is 1
```

Εικόνα 6.1.6: Αλγόριθμος Bernstein-Vazirani με αυτοματοποιημένο τρόπο



Εικόνα 6.1.7: Οπτικοποίηση αλγορίθμου Bernstein-Vazirani αυτοματοποιημένα

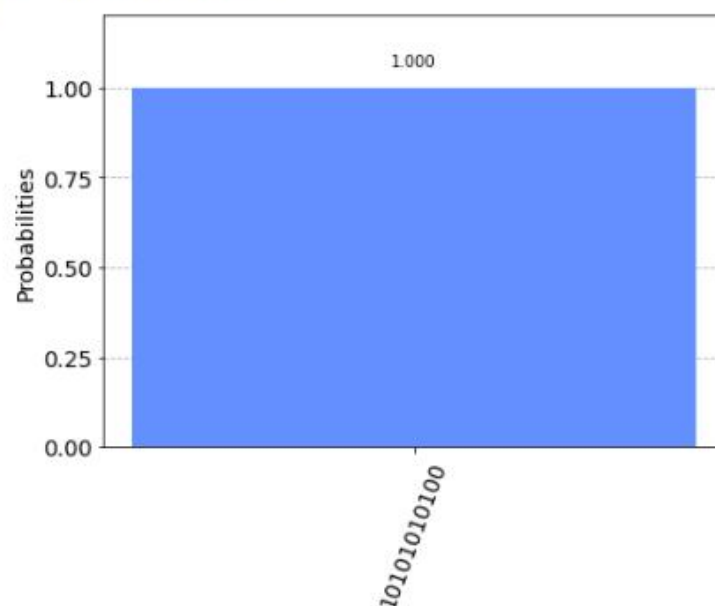
Έτσι, βλέπουμε και τα αποτελέσματα του ολοκληρωμένου αλγορίθμου:

```
In [7]: simulator = Aer.get_backend('qasm_simulator')
result = execute(circuit, backend=simulator, shots=1).result()
counts = result.get_counts()
print(counts)

{'10101010100': 1}
```

```
In [8]: plot_histogram([counts])
```

Out[8]:



Εικόνα 6.1.8: Αποτέλεσμα αλγορίθμου Bernstein-Vazirani αυτοματοποιημένα

6.2. Ο αλγόριθμος του Deutsch

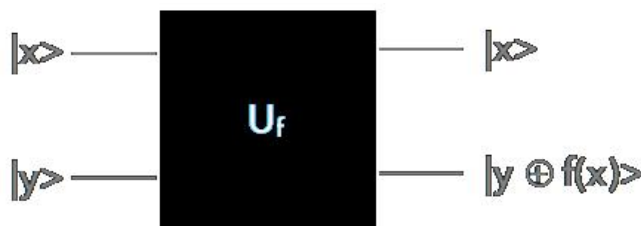
Όπως και με τον αλγόριθμο των Bernstein-Vazirani, έτσι και ο αλγόριθμος του Deutsch επιδεικνύει την κβαντική υπεροχή έναντι των κλαστικών συστημάτων. Το πρόβλημα στον αλγόριθμο αυτόν είναι ότι έχουμε μια συνάρτηση f , η οποία δεν γνωρίζουμε πως λειτουργεί ή τι κάνει (oracle function), αλλά ξέρουμε ότι παίρνει τιμές 0 ή 1 σαν είσοδο. Για το αποτέλεσμα έχουμε τέσσερις πιθανότητες, 00 ή 01 ή 10 ή 11. Όταν, έχουμε το αποτέλεσμα 00 ή 11, τότε η συνάρτηση ονομάζεται συνεχής συνάρτηση (constant), ενώ όταν δέχεται μια από τις υπόλοιπες τιμές, ονομάζεται ισορροπημένη συνάρτηση (balanced). Αυτό σημαίνει ότι το αποτέλεσμα δεν εξαρτάται από την είσοδο, στη περίπτωση της συνεχούς συνάρτησης, όπου είτε φέρουμε σαν είσοδο το 0 ή το 1, το αποτέλεσμα θα είναι ίδιο, όπως επαληθεύεται και με τη παρακάτω σχέση:

$$f: \{0, 1\} \rightarrow \{0, 1\}$$

$$f(0) = f(1)$$

Αντίθετα, το αποτέλεσμα εξαρτάται από την είσοδο, στη περίπτωση της ισορροπημένης συνάρτησης. Γι' αυτό, θέλουμε να προσδιορίσουμε πότε πρόκειται για συνεχής ή ισορροπημένη συνάρτηση. Σε ένα κλαστικό σύστημα για το μάθουμε αυτό, απλά δίνουμε και τις δύο πιθανές τιμές 0 και 1 στην συνάρτηση και ανάλογα το αποτέλεσμα, μπορούμε να προσδιορίσουμε την συνάρτηση. Όμως, αυτό δεν είναι και πάλι αρκετό, μιας και ο αλγόριθμος του Deutsch μπορεί να μάθει τη συνάρτηση αυτή με μόνο μια προσπάθεια. Έτσι, ο αλγόριθμος αυτός αποδεικνύει και πάλι την κβαντική υπεροχή με τα κλασικά συστήματα ^[21].

Πριν προχωρήσουμε στην υλοποίηση του αλγορίθμου είναι σημαντικό να κατανοήσουμε ορισμένες σχέσεις που θα μας βοηθήσουν και στους επόμενους αλγορίθμους. Πως θα μοιάζει, λοιπόν, το κύκλωμά μας; Θα χρειαστούμε δύο qubits εισόδου, ένα που θα το μετρήσουμε, έστω $|x\rangle$ και το δεύτερο, έστω $|\psi\rangle$ που θα το χρειαστούμε ως βοηθητικό qubit. Όπως αναφέρθηκε, δεν μας ενδιαφέρει τι κάνει ακριβώς η συνάρτησή μας, αλλά προσπαθούμε να την προσδιορίσουμε. Δέχοντας ως είσοδο αυτά τα δύο qubits, το αποτέλεσμα για το qubit εισόδου $|x\rangle$, θα είναι $|x\rangle$, ενώ για το $|\psi\rangle$ θα είναι $|\psi \oplus f(x)\rangle$, δηλαδή το XOR αποτελέσμά τους.



Εικόνα 6.2.1: Αναπαράσταση εισόδων & εξόδων συνάρτησης

Θα πρέπει να θυμόμαστε πως αυτό πρέπει να είναι αντιστρέψιμο. Αυτό σημαίνει πως αν εφαρμόσουμε ακόμη μια φορά την U_f στο κύκλωμά μας θα αντιστρέψουμε το αποτέλεσμα, επιστρέφοντας στις αρχικές τιμές των $|x\rangle$ και $|\psi\rangle$. Μαθηματικά, επαληθεύεται ως εξής:

$$|x\rangle|\psi\rangle \xrightarrow{U_f} |x\rangle|\psi \oplus f(x)\rangle \xrightarrow{U_f} |x\rangle|\psi \oplus f(x)\rangle \oplus f(x)\rangle \rightarrow |x\rangle|\psi \oplus 0\rangle \rightarrow |x\rangle|\psi\rangle$$

Στην ουσία, αφού επαναλάβουμε την U_f θα βγει το αποτέλεσμα 0, όταν η $f(x)$ επαληθεύεται με XOR με τον εαυτό της και, συνεπώς φτάνουμε στην αρχική μας κατάσταση. Μπορούμε, επίσης, να αναπαραστήσουμε την U_f με οποιαδήποτε κβαντική πύλη θέλουμε να εφαρμόσουμε, πχ CNOT, CZ, κτλπ.

Για να γίνει καλύτερα αντιληπτό, είναι αρκετά χρήσιμο να γνωρίζουμε την λάθος προσέγγιση και γιατί είναι λάθος. Ας υποθέσουμε ότι θέλουμε να εφαρμόσουμε μια Hadamard πύλη στο qubit που θέλουμε να μετρήσουμε, στην περίπτωση μας στο $|x\rangle$. Παράλληλα, δεν εφαρμόζουμε καμία πύλη στο $|\psi\rangle$, το οποίο και θα βρίσκεται στην κατάσταση $|0\rangle$, οπότε και εφαρμόζουμε την U_f και μετράμε το $|x\rangle$, ώστε να μας επιστρέψει πάλι την τιμή $|x\rangle$. Να σημειωθεί πως, αφού εφαρμοστεί η Hadamard πύλη, το $|x\rangle$ θα βρίσκεται στη κατάσταση $|+\rangle$, μιας και βρίσκεται σε κατάσταση υπέρθεσης. Έτσι για να βρούμε την συνολική κατάσταση των δύο qubits, αυτό επαληθεύεται ως εξής:

$$\begin{aligned} & U_f (|+\rangle \otimes |0\rangle) \\ &= U_f \left(\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \otimes |0\rangle \right) \\ &= \left(\frac{1}{\sqrt{2}} U_f |0\rangle \right) \otimes |0\rangle + \left(\frac{1}{\sqrt{2}} U_f |1\rangle \right) \otimes |0\rangle \\ &= \left(\frac{1}{\sqrt{2}} |0\rangle f(0) \right) \oplus |0\rangle + \left(\frac{1}{\sqrt{2}} |1\rangle f(1) \right) \oplus |0\rangle \end{aligned}$$

Το πρόβλημα είναι πως καταλήγουμε με 50% πιθανότητες να εμφανιστούν μια από τις τιμές $|0\rangle f(0)$ και $|1\rangle f(1)$. Όμως, δεν θέλουμε να λύσουμε το πρόβλημα με δύο προσπάθειες, αλλά με μία. Έτσι, δεν γίνεται να προσθέσουμε μονάχα μια Hadamard πύλη, αλλά θα χρειαστεί και άλλες. Επειδή, ο αλγόριθμος μοιάζει αρκετά με τον αλγόριθμο Bernstein-Vazirani, εφαρμόζουμε μια X και Hadamard πύλη στο βοηθητικό qubit, μια επιπλέον Hadamard στο $|x\rangle$ για να αντιστρέψουμε το αποτέλεσμα και, τέλος να το μετρήσουμε. Πιο συγκεκριμένα, φέρνουμε σε κατάσταση υπέρθεσης το $|x\rangle$, οπότε αποκτά φάση και μετατρέπεται σε $|+\rangle$. Αντίστοιχα, εφαρμόζοντας τις πύλες X και Hadamard στο $|\psi\rangle$, η φάση του αλλάζει και μετατρέπεται σε $|-\rangle$. Συνεπώς, μαθηματικά αυτό επαληθεύεται ως:

$$|+\rangle|-\rangle = \frac{1}{2}(|0\rangle + |1\rangle \otimes |0\rangle - |1\rangle) = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

Αν εφαρμόσουμε την συνάρτηση U_f , θα έχουμε:

$$\rightarrow \frac{1}{2}(|0\rangle|0 \oplus f(0)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|0 \oplus f(1)\rangle - |1\rangle|1 \oplus f(1)\rangle)$$

οπότε στις περιπτώσεις που η συνάρτηση πραγματοποιείται XOR με τον εαυτό της, έχουμε:

$$\rightarrow \frac{1}{2}(|0\rangle|f(0)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|f(1)\rangle - |1\rangle|1 \oplus f(1)\rangle)$$

Σε αυτή τη περίπτωση πρέπει να κάνουμε ορισμένες υποθέσεις. Αν υποθέσουμε πως ισχύει $f(0) = f(1)$, ώστε η συνάρτηση να είναι συνεχής, τότε:

$$\rightarrow \frac{1}{2}(|0\rangle|f(0)\rangle - |0\rangle|1 \oplus f(0)\rangle + |1\rangle|f(0)\rangle - |1\rangle|1 \oplus f(0)\rangle)$$

$$\rightarrow \frac{1}{2}((|0\rangle + |1\rangle)|f(0)\rangle + (|0\rangle - |1\rangle)|1 \oplus f(0)\rangle)$$

$$\rightarrow \frac{1}{2}((|0\rangle + |1\rangle)(|f(0)\rangle - |1 \oplus f(0)\rangle))$$

Αυτό που ξέρουμε είναι πως το $\frac{1}{2}(|0\rangle + |1\rangle)$ ισούται με $|+\rangle$. Άρα,

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

Τι θα συμβεί, λοιπόν, αν εφαρμόσουμε μια Hadamard πύλη και μετρήσουμε την κατάσταση $|+\rangle$; Μόλις το μετρήσουμε θα έχουμε την κατάσταση $|0\rangle$, ενώ παράλληλα, δεν μας ενδιαφέρει η κατάσταση του άλλου qubit, δηλαδή:

$$H(|+\rangle) = |0\rangle$$

Τι συμβαίνει, όμως, στην υπόθεση ότι είναι ισορροπημένη συνάρτηση; Αυτό σημαίνει πως πρέπει να ισχύει ότι $f(0) \neq f(1)$ ή αλλιώς $f(0) = f(1) \oplus 1$. Δηλαδή,

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

$$\rightarrow \frac{1}{2} (|0\rangle \otimes (|f(0)\rangle - |f(1)\rangle) - |1\rangle \otimes (|f(0)\rangle - |f(1)\rangle))$$

$$\rightarrow \frac{1}{2} ((|0\rangle - |1\rangle) \otimes (|f(0)\rangle - |f(1)\rangle))$$

Αυτό που ξέρουμε είναι πως το $\frac{1}{2}(|0\rangle - |1\rangle)$ ισούται με $|-\rangle$. Οπότε,

$$\rightarrow \frac{1}{\sqrt{2}} (|-\rangle \otimes (|f(0)\rangle - |f(1)\rangle))$$

Άρα,

$$H(|-\rangle) = |1\rangle$$

Έτσι, καταλαβαίνουμε αμέσως το είδος της συνάρτησης μετά από τη μέτρηση. Στη συνέχεια, θα δούμε τον κώδικα του αλγορίθμου.



Εικόνα 6.2.2: Αναπαράσταση αλγορίθμου Deutsch

6.2.1. Υλοποίηση αλγορίθμου στο Qiskit

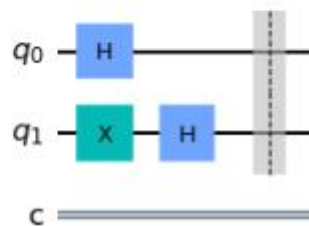
Κατανοώντας τον τρόπο λειτουργίας του αλγορίθμου, πλέον, θα είναι αρκετά εύκολο να τον υλοποιήσουμε στο Qiskit. Γι' αυτό, επισυνάπτονται οι παρακάτω εικόνες από το Qiskit.

```
In [1]: from qiskit import *
        from qiskit.tools.visualization import plot_histogram
        %matplotlib inline
```

```
In [2]: circuit = QuantumCircuit(2,1)
```

```
In [3]: circuit.h(0)
        circuit.x(1)
        circuit.h(1)
        circuit.barrier()
        circuit.draw(output='mpl')
```

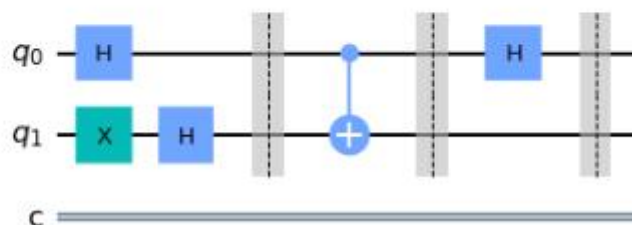
Out[3]:



Εικόνα 6.2.3: Πρώτο βήμα υλοποίησης αλγορίθμου Deutsch

```
In [4]: circuit.cx(0,1)
        circuit.barrier()
        circuit.h(0)
        circuit.barrier()
        circuit.draw(output='mpl')
```

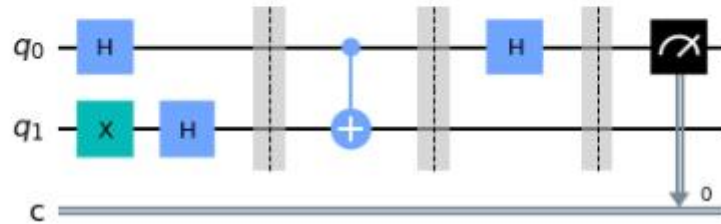
Out[4]:



Εικόνα 6.2.4: Δεύτερο βήμα υλοποίησης αλγορίθμου Deutsch

```
In [5]: circuit.measure(0,0)
Out[5]: <qiskit.circuit.instructionset.InstructionSet at 0x7fb3a83c1c70>

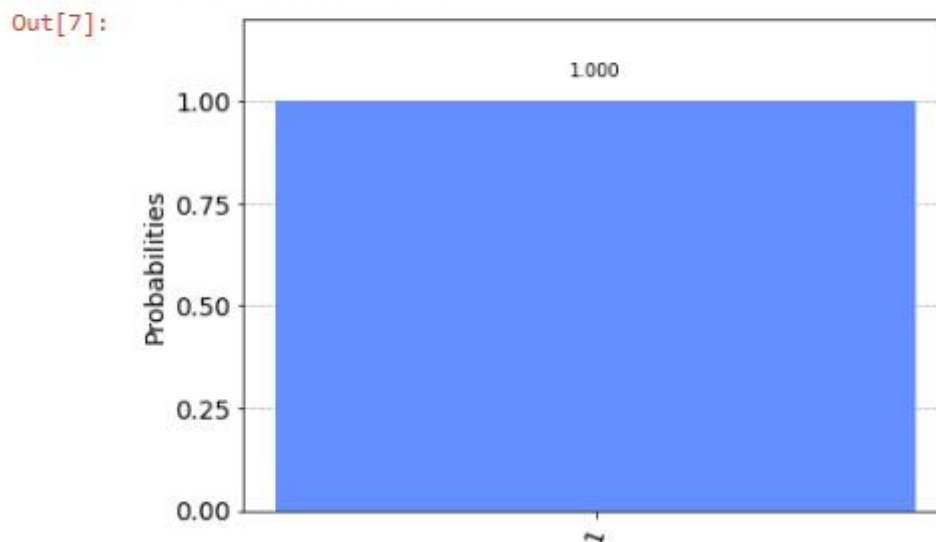
In [6]: circuit.draw(output='mpl')
Out[6]:
```



Εικόνα 6.2.5: Μέτρηση του qubit στον αλγόριθμο Deutsch

```
In [7]: backend = Aer.get_backend('qasm_simulator')
result = execute(circuit, backend=backend, shots=1024).result()
counts = result.get_counts(circuit)

plot_histogram([counts])
```



Εικόνα 6.2.6: Αποτελέσματα προσομοιωτή αλγορίθμου Deutsch

```

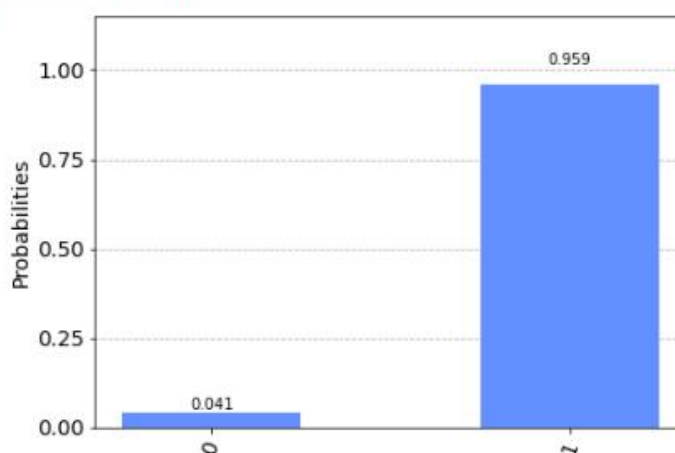
In [12]: quantum_computer = provider.get_backend('ibmq_manila')

In [13]: quantum_result = execute(circuit, backend=quantum_computer, shots=1024).result()

In [14]: quantum_counts = quantum_result.get_counts(circuit)
          plot_histogram([quantum_counts])

Out[14]:

```



Εικόνα 6.2.7: Αποτελέσματα κβαντικού υπολογιστή αλγορίθμου Deutsch

6.3. Ο αλγόριθμος του Grover

Πλέον, μπορούμε να εμβαθύνουμε σε αλγορίθμους που είναι καθοριστικοί στον τομέα της κβαντικής υπολογιστικής και έχουν την δυνατότητα να αλλάξουν τον κόσμο. Ο αλγόριθμος του Grover θεωρείται ένας αλγόριθμος αναζήτησης. Αν για παράδειγμα, έχουμε μια λίστα με 10 καταχωρήσεις και θέλουμε να βρούμε έναν αριθμό μέσα σε αυτή, τότε με μια γραμμική αναζήτηση της λίστας αυτής θα έχουμε $O(n)$ στη χειρότερη περίπτωση και $O(1)$ στη καλύτερη. Ένα παράδειγμα στο Qiskit φαίνεται στην Εικόνα 6.3.1. Έστω, λοιπόν, δημιουργούμε μια λίστα με 10 καταχωρήσεις και επιθυμούμε να βρούμε τον αριθμό 8. Δημιουργούμε μια oracle συνάρτηση, όπου όπως εξηγήθηκε και σε προηγούμενο κεφάλαιο, πρόκειται για μια συνάρτηση που δεν ξέρουμε πως λειτουργεί και είναι κάτι που μας απασχολεί. Αυτή μπορεί να ονομαστεί και ως black box. Έτσι, στην συνάρτηση αυτή δημιουργούμε ένα if statement, όπου ελέγχουμε αν βρέθηκε ο ζητούμενος αριθμός μέσα στη λίστα. Έπειτα, σε μια for loop εμφανίζουμε που βρίσκεται ο αριθμός αυτός, όπως και πόσες προσπάθειες χρειάστηκε για να βρεθεί.

Σε αντίθετη περίπτωση, ο αλγόριθμος του Grover υπόσχεται πως το στοιχείο αυτό μπορεί να βρεθεί, στη χειρότερη περίπτωση, σε $O(\sqrt{n})$. Αυτό είναι μια μεγάλη απόκλιση, ειδικά σε μεγάλες βάσεις δεδομένων, αλλά ακόμη και σε μικρές λίστες, όπου μπορούμε να βρούμε το επιθυμητό στοιχείο και με μία προσπάθεια.

```

In [1]: from qiskit import *
        from qiskit.tools.visualization import plot_histogram
        %matplotlib inline

In [3]: myList = [5,4,6,9,1,2,3,7,8,0]

In [4]: def oracle(number):
        winningNumber = 8
        if number == winningNumber:
            response = True
        else:
            response = False
        return response

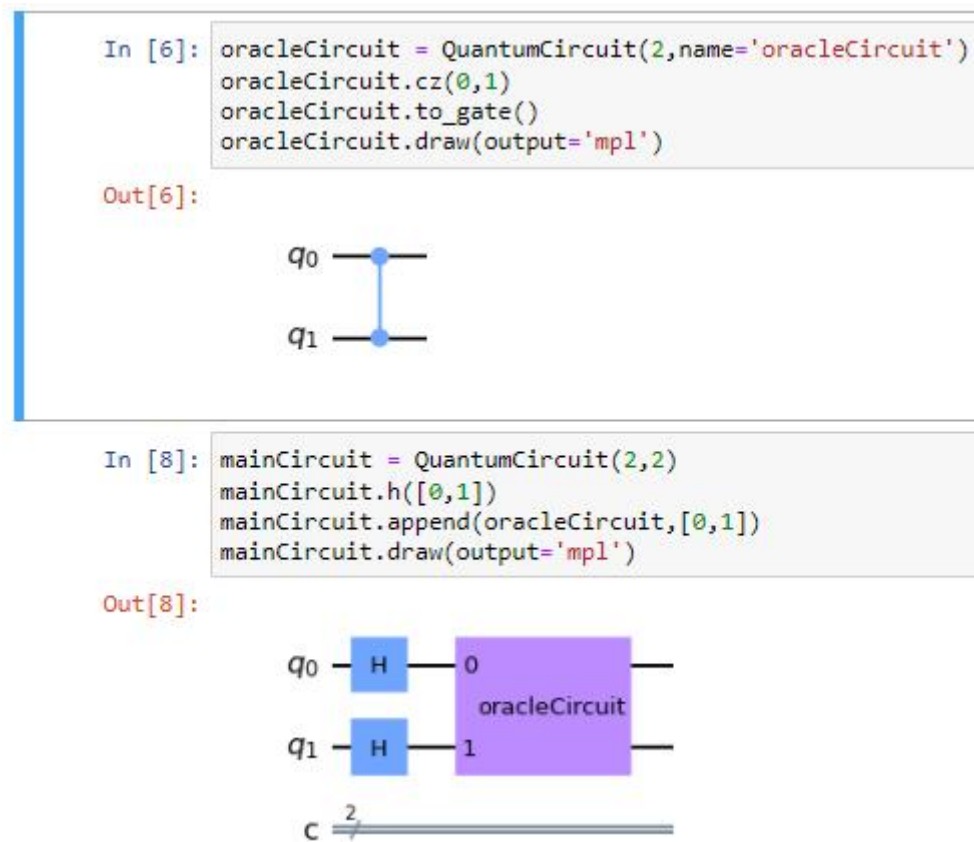
In [5]: for index, number in enumerate(myList):
        if oracle(number) is True:
            print(f"winning number index: {index}")
            print(f"execution count: {index+1}")
            break

winning number index: 8
execution count: 9

```

Εικόνα 6.3.1: Κλασσικός τρόπος γραμμικής αναζήτησης

Τώρα, ας ξεκινήσουμε με την υλοποίηση του αλγορίθμου του Grover. Το παράδειγμά μας θα υλοποιηθεί με δύο qubits, οπότε θα πρέπει να ψάξουμε στις καταστάσεις $|00\rangle$, $|01\rangle$, $|10\rangle$ ή $|11\rangle$. Έστω ότι θέλουμε να αναζητήσουμε τη κατάσταση $|11\rangle$. Στην περίπτωση μας, δεν θα υλοποιήσουμε μόνο μια συνάρτηση oracle, όπως προηγουμένως, αλλά θα χρειαστούμε και μια καινούργια, που ονομάζεται reflection συνάρτηση. Ο αλγόριθμος έχει τρία βήματα. Αρχικά, εφαρμόζουμε Hadamard πύλη και στα δύο qubits και, έπειτα μια CZ πύλη, η οποία θα έχει το ρόλο της oracle συνάρτησης. Στην ουσία, όταν προσθέτουμε τις Hadamard πύλες, φέρουμε τα qubits σε κατάσταση υπέρθεσης, επιτρέποντας όλες τις πιθανές τιμές και με τη CZ και στα δύο qubits θα μετασχηματίσει τη κατάσταση $|11\rangle$ στη γωνία π . Στην Εικόνα 6.3.2 έχουμε τον τρόπο υλοποίησης του κυκλώματος στο Qiskit, καθώς και μια ευκαιρία να χτίσουμε το δικό μας κύκλωμα από αυτό. Μιας και η CZ θεωρείται η oracle συνάρτησή μας, μπορούμε να την ορίσουμε σαν δικό μας κύκλωμα και να την εισάγουμε στο βασικό. Από τη στιγμή, που δεν χρειαζόμαστε κλασσικά bits στο κύκλωμά μας, αφού δεν θα τα μετρήσουμε είναι μια καλή ευκαιρία να δούμε πως μπορούμε να δημιουργήσουμε τα δικά μας κυκλώματα και να τα εισάγουμε σε άλλα, υλοποιώντας πιο περίπλοκα συστήματα με αυτό το τρόπο.



Εικόνα 6.3.2: Υλοποίηση oracle συνάρτησης

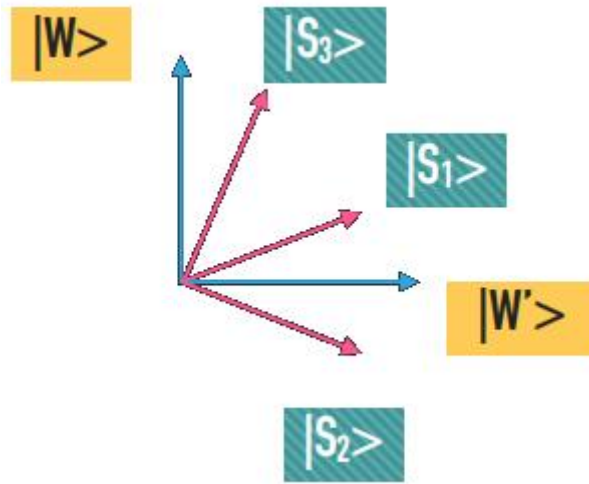
Για να προχωρήσουμε στο επόμενο βήμα υλοποίησης της reflection συνάρτησης, θα πρέπει να κατανοήσουμε πως δουλεύει και γιατί τη χρειαζόμαστε. Ο στόχος μας είναι να βρούμε 100% πιθανότητα στη κατάσταση $|11\rangle$. Έστω, η επιθυμητή κατάσταση συμβολίζεται με W , με S_1 συμβολίζεται η κατάσταση υπέρθεσης και με S_2 η μετατόπιση φάσης (phase shift) στην κατάσταση $-|11\rangle$. Γνωρίζουμε, ήδη, πως:

$$|W\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle$$

$$|S_1\rangle = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$|S_2\rangle = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

Αυτό που πρέπει να κάνουμε είναι να πάμε από τη κατάσταση $|S_1\rangle$ στη κατάσταση $|S_2\rangle$ με μία τιμή φ . Μπορούμε να βρούμε ένα διάνυσμα, έστω $|W'\rangle$ που βρίσκεται σε ορθογώνια θέση από το $|W\rangle$. Με αυτό το τρόπο, χρησιμοποιούμε τη γωνία φ , ώστε να πάμε στη κατάσταση $|S_2\rangle$. Γεωμετρικά, παρουσιάζεται στην παρακάτω εικόνα.



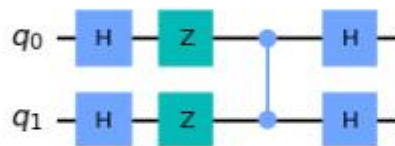
Εικόνα 6.3.3: Γεωμετρική αναπαράσταση εντοπισμού μιας κατάστασης

Με αυτό το τρόπο, εκμεταλεύοντας την γωνία φ , μεταξύ της $|S_1\rangle$ και $|W'\rangle$ μπορούμε να εφαρμόσουμε και πάλι την γωνία φ για να καταλήξουμε στην $|S_2\rangle$. Τέλος, για να τα φτάσουμε στην $|S_3\rangle$ εφαρμόζουμε δύο φορές την γωνία φ από τη κατάσταση $|S_2\rangle$. Τώρα, γιατί όμως το θέλουμε αυτό; Γιατί, εφαρμόζοντας δύο φορές την γωνία φ , θα βρισκόμαστε πιο κοντά στην επιθυμητή κατάσταση $|11\rangle$. Οπότε, η reflection συνάρτηση θα εφαρμόζει την γωνία αυτή, ώστε να φτάσει πιο κοντά στον ζητούμενο αριθμό-κατάσταση. Η υπόθεση του Grover ήταν πως αν το κάνουμε αυτό επανειλημμένα, θα φτάνει όλο και πιο κοντά στην κατάσταση που ζητάμε σε κάθε βήμα. Έτσι, προσπαθεί να μεγιστοποιήσει την πιθανότητα να βρει την κατάσταση αυτή, εφαρμόζοντας αυτή τη μετατόπιση.

Πλέον, μπορούμε να προχωρήσουμε στην υλοποίηση της reflection συνάρτησης και, έπειτα στην ολοκλήρωση του αλγορίθμου. Οι πύλες που χρειάζονται για την κατασκευή της συνάρτησης παρέχονται στην Εικόνα 6.3.4, όπου δημιουργείται, όπως και η oracle, σαν ξεχωριστό κύκλωμα και την εισάγουμε στο βασικό μας, δημιουργώντας κάτι ενιαίο. Τέλος, στην Εικόνα 6.3.5, βλέπουμε τα επιθυμητά μας αποτελέσματα.

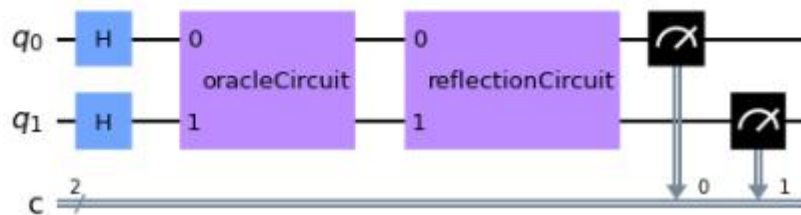
```
In [9]: reflectionCircuit = QuantumCircuit(2,name="reflectionCircuit")
reflectionCircuit.h([0,1])
reflectionCircuit.z([0,1])
reflectionCircuit.cz(0,1)
reflectionCircuit.h([0,1])
reflectionCircuit.to_gate()
reflectionCircuit.draw(output='mpl')
```

Out[9]:



```
In [10]: mainCircuit.append(reflectionCircuit,[0,1])
mainCircuit.measure([0,1],[0,1])
mainCircuit.draw(output='mpl')
```

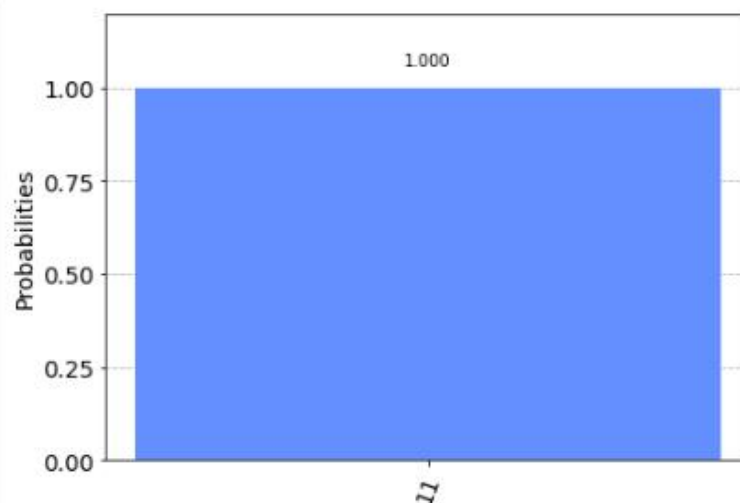
Out[10]:



Εικόνα 6.3.4: Υλοποίηση reflection συνάρτησης

```
In [14]: backend = Aer.get_backend('qasm_simulator')
result = execute(mainCircuit,backend=backend,shots=1).result()
counts = result.get_counts(mainCircuit)
plot_histogram([counts])
```

Out[14]:



Εικόνα 6.3.5: Αποτελέσματα αλγορίθμου Grover

6.4. Ο αλγόριθμος του Shor

Ο αλγόριθμος του Shor θεωρείται από τους σημαντικότερους αλγορίθμους που αναπτύχθηκαν, δεδομένου ότι μπορεί να αποδείξει και την κβαντική υπεροχή έναντι των κλασσικών υπολογιστών και μπορεί να αλλάξει τον τρόπο ζωής στα επόμενα χρόνια, όσο προχωράει και η ανάπτυξη της κβαντικών υπολογιστών. Εν ολίγοις, ο αλγόριθμος είναι ικανός να βρίσκει το γινόμενο πρώτων παραγόντων μεγάλων ακέραιων αριθμών. Αυτό είναι μεγάλη επανάσταση στον κλάδο, διότι γνωστοί αλγόριθμοι, όπως ο RSA, βασίζονται στην δυσκολία εύρεσης των πρώτων παραγόντων σε μεγάλους αριθμούς για την προστασία μας στο Διαδίκτυο. Αυτή τη στιγμή, ο αλγόριθμος του Shor μπορεί να εκτελεστεί σε μικρούς αριθμούς, μιας και οι κβαντικοί υπολογιστές παρέχουν λίγα qubits. Όμως, όσο αναπτύσσεται η κβαντική ισχύς και αυξάνονται τα qubits, φτάνουμε κοντά στην επίλυση μεγάλων αριθμών που υπόσχεται ο αλγόριθμος. Όμως, αυτό μπορεί να μεταφραστεί και ως ένα πρόβλημα. Αρκετοί ερευνητές ασχολούνται πάνω στον τομέα της μετακβαντικής υπολογιστικής (post-quantum) για να υλοποιήσουν αλγορίθμους που θα παρέχουν προστασία έναντι του αλγορίθμου, αφού ο RSA θα μπορεί να αποκρυπτογραφηθεί σε λίγα λεπτά. Το νούμερο είναι εξωφρενικό, γιατί πρόκειται για εκθετική αύξηση στις 2^n προσπάθειες, ώστε ένας κλασσικός υπολογιστής να καταφέρει να “σπάσει” την κρυπτογράφηση. Ο αλγόριθμος του Shor δεν χρειάζεται εκθετική αύξηση προσπαθειών. Γι’ αυτό, πριν προχωρήσουμε στην υλοποίηση του αλγορίθμου θα πρέπει να γνωρίζουμε κάποιες βασικές έννοιες για να καταλήξουμε και στην υλοποίηση, αρχικά, του κλασσικού και, στη συνέχεια, του κβαντικού αλγορίθμου.

Η πρώτη μαθηματική έννοια που πρέπει να γνωρίζουμε είναι ο Μέγιστος Κοινός Διαιρέτης ή αλλιώς $\text{MKΔ}(\alpha, \beta)$. Για παράδειγμα:

$$\text{MKΔ}(15, 21) = 3, \text{ αφού}$$

$$15 = 3 \cdot 5$$

$$21 = 3 \cdot 7$$

Έχοντας τον MKΔ στο μυαλό μας, θα πρέπει να βρούμε τους πρώτους παράγοντες ενός μεγάλου ακέραιου αριθμού. Αν θέλουμε να τους βρούμε, θα πρέπει να ακολουθήσουμε μια μαθηματική σχέση για να το καταφέρουμε. Γι’ αυτό, ας πάρουμε ως παράδειγμα έναν μικρό αριθμό, έστω $N = 21$, προσπαθώντας να βρούμε τους πρώτους παράγοντες του αριθμού με μια μαθηματική φόρμουλα. Αφού αναζητούμε τον αριθμό 21, η σχέση θα επαληθευτεί ως:

$$x^2 = 1(\text{mod } 21)$$

Πως, όμως, θα βρούμε το x ; Θα μπορούσε να έχει πολλές τιμές, όπως 1, -1, 8, -8, 13, 20 ή -20. Μπορούμε να διαλέξουμε ένα τυχαίο αριθμό, πχ το 8 και να προχωρήσουμε τη παραπάνω σχέση μας. Αυτή θα φαίνεται ως εξής:

$$8^2 - 1^2 = 0 \pmod{21}$$

Το αριστερό σκέλος της ισότητας, μπορεί να μεταφραστεί και ως:

$$(8 - 1) \cdot (8 + 1)$$

Αν εφαρμόσουμε το ΜΚΔ στο 21 και στις δύο αυτές παρενθέσεις, θα έχουμε:

$$\text{ΜΚΔ}(21, 8+1) = 3$$

$$\text{ΜΚΔ}(21, 8-1) = 7$$

Μόλις βρήκαμε τους πρώτους παράγοντες του 21! Πως καταλήξαμε, όμως, ως εδώ και γιατί λειτουργεί; Αρχικά, η μαθηματική σχέση που ακολουθήσαμε στην πραγματικότητα είναι:

$$x^2 = 1 \pmod{N}$$

$$\text{ΜΚΔ}(N, x+1)$$

Έστω πως συνεχίζουμε με το ίδιο παράδειγμα και αναζητούμε το γινόμενο πρώτων παραγόντων του 21, άρα $N = 21$. Έστω, πως διαλέγουμε και έναν τυχαίο ακέραιο αριθμό, πχ τον αριθμό 2. Αν πάρουμε τον αριθμό αυτόν και τον αυξάνουμε συνεχώς κατά 1 την δύναμή του, θα έχουμε:

$$2^1 = 2 = 2 \pmod{21}$$

$$2^2 = 4 = 4 \pmod{21}$$

$$2^3 = 8 = 8 \pmod{21}$$

$$2^4 = 16 = 16 \pmod{21}$$

$$2^5 = 32 = 11 \pmod{21}$$

$$2^6 = 64 = 1 \pmod{21}$$

$$2^7 = 128 = 2 \pmod{21}$$

$$2^8 = 256 = 4 \pmod{21}$$

...

...

Παρατηρούμε μια αύξηση από το 2 έως το 16 και μετά μια μείωση μέχρι τον αριθμό 1 και αυτό συνεχίζει να επαναλαμβάνεται. Μπορούμε, λοιπόν, να πούμε πως πρόκειται για μια περιοδική συνάρτηση και η περίοδος εδώ είναι ο αριθμός 6, δηλαδή ο αριθμός της δύναμης του τυχαίου αριθμού που μαντέψαμε. Με λίγα λόγια, κάθε 6 επαναλήψεις η συνάρτηση επαναλαμβάνεται. Από τη στιγμή που αναζητούμε τον αριθμό του x^2 , μπορούμε να επαληθεύσουμε πως:

$$x^6 = (x^3)^2 = 1(\text{mod } 21)$$

Όμως, μαντέψαμε πως:

$$x = 2 \rightarrow x^3 = 8$$

Με αυτό το τρόπο βρήκαμε τον αριθμό 8 και συγκρίναμε τον ΜΚΔ του 21 με αυτόν τον αριθμό στην αρχή, βρίσκοντας τελικά το γινόμενο των πρώτων παραγόντων του 21. Όμως, μήπως αυτό οφείλεται στην τύχη; Στην πραγματικότητα, ναι. Θα πρέπει η ο αριθμός της περιόδου να είναι άρτιος αριθμός, όπως βρήκαμε στην περίπτωση μας τον αριθμό 6 από τη περιοδική συνάρτηση και, συνεπώς, απλοποιήσαμε την δύναμη του x^6 . Βέβαια, οι πιθανότητες για να βρούμε τον κατάλληλο αριθμό περιόδου είναι 50%.

Επίσης, αξίζει να παρουσιαστεί ο αλγόριθμος με κλασσικό τρόπο και στο Qiskit. Στις παρακάτω εικόνες, επισυνάπτονται τα βήματα υλοποίησης του αλγορίθμου, σύμφωνα με όσα ειπώθηκαν προηγουμένως.

```
In [1]: import math
import matplotlib.pyplot as plt
import numpy as np

In [2]: N = 21
a = 2

In [3]: math.gcd(a, N)

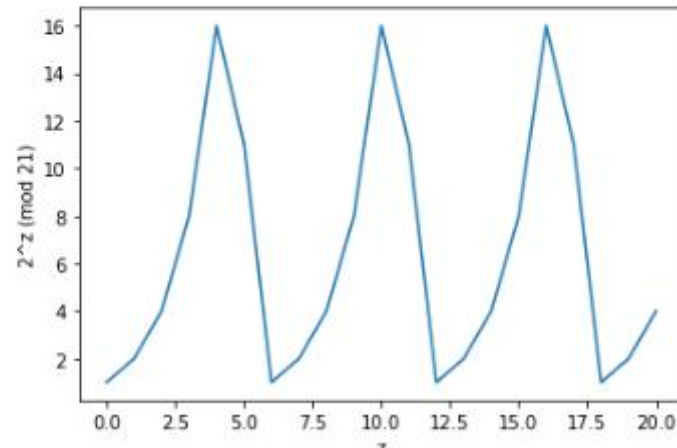
Out[3]: 1
```

Εικόνα 6.4.1: Αρχικοποιήσεις και εύρεση ΜΚΔ δύο αριθμών

Στην Εικόνα 6.4.1 εισάγουμε τις απαραίτητες βιβλιοθήκες και αρχικοποιούμε τον αριθμό N που αναζητάμε και έναν τυχαίο αριθμό a . Σαν παράδειγμα, με την εντολή `math.gcd(a, N)` μπορούμε να βρούμε τον ΜΚΔ των δύο αυτών αριθμών. Στη συνέχεια, στην Εικόνα 6.4.2, παίρνουμε τον τυχαίο αριθμό a υψωμένο στον αριθμό z με κάθε αριθμό της λίστας N , επιστρέφοντας το modulo, ώστε να οπτικοποιήσουμε την συνάρτηση περιοδικότητας και να βρούμε την περίοδο.

```
In [4]: z = list(range(N))
        y = [a**z0 % N for z0 in z]

        mpl.plot(z,y)
        mpl.xlabel('z')
        mpl.ylabel(f'{a}^z (mod {N})')
        mpl.show()
```



Εικόνα 6.4.2: Οπτικοποίηση περιοδικής συνάρτησης

Τέλος, στην Εικόνα 6.4.3 βρίσκουμε τον αριθμό περιόδου με το παρακάτω notation, βρίσκοντας τον αριθμό 6 και μέσα από έναν if statement, ελέγχουμε αν ο αριθμός αυτός είναι άρτιος, ώστε να προχωρήσουμε στην εύρεση του γινομένου των πρώτων παραγόντων με τα $x+1$ και $x-1$, αλλιώς ειδοποιούμε πως η περίοδος είναι περιττός αριθμός.

```
In [5]: r = z[y[1:].index(1)+1]
        print(r)
```

6

```
In [6]: if r % 2 == 0:
        x = (a**(r/2.)) % N
        print(f'x: {x}')
        if ((x+1) % N) != 0:
            print(math.gcd((int(x)+1),N), math.gcd((int(x)-1),N))
        else:
            print('x+1 is 0 (mod N)')
    else:
        print('r is odd')
```

x: 8.0

3 7

Εικόνα 6.4.3: Εύρεση ΜΚΔ του αριθμού $N = 21$

Ο αλγόριθμος στην κλασσική του μορφή φαίνεται απλός. Για να υλοποιήσουμε τον κβαντικό αλγόριθμο, βέβαια, θα χρειαστεί να εξηγήσουμε μερικές σημαντικές έννοιες που ακολουθεί ο αλγόριθμος. Στα επόμενα, λοιπόν, κεφάλαια θα αναλυθούν οι έννοιες αυτές, καταλήγοντας και στην υλοποίηση του κβαντικού αλγορίθμου του Shor στο Qiskit.

6.4.1. Ο κβαντικός μετασχηματισμός Fourier

Ο κβαντικός μετασχηματισμός Fourier ή εν συντομία QFT είναι η κβαντική υλοποίηση του διακριτού μετασχηματισμού Fourier στα πλάτη μιας κυματοσυνάρτησης. Αποτελεί μέρος πολλών κβαντικών αλγορίθμων, με πιο αξιοσημείωτο τον αλγόριθμο παραγοντοποίησης του Shor και την εκτίμηση κβαντικής φάσης, που θα μελετήσουμε στο επόμενο υποκεφάλαιο. Ο κβαντικός μετασχηματισμός Fourier (QFT) μετασχηματίζεται μεταξύ δύο βάσεων, της υπολογιστικής βάσης και της βάσης Fourier. Μετασχηματίζει, δηλαδή, τις καταστάσεις $|0\rangle$ και $|1\rangle$ που αποτελούν την υπολογιστική βάση στις καταστάσεις $|+\rangle$ και $|-\rangle$ και αποτελούν την βάση Fourier.

Σε περιπτώσεις υπολογισμού ενός qubit, η Hadamard πύλη φέρνει την υπολογιστική βάση στην βάση Fourier, δεδομένου ότι φέρνει τα διανύσματα κατάστασης σε υπέρθεση. Φυσικά, δεν θα βρισκόταν ακριβώς στο ίδιο διάνυσμα κατάστασης, αλλά θα είχε επίσης μια γωνία ή κάποια φάση. Αντίστοιχα, τι συμβαίνει, σε περιπτώσεις που έχουμε δύο qubits; Στα δύο qubits, ακολουθούμε μια συγκεκριμένη μαθηματική σχέση:

$$\text{QFT}|x\rangle = |\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{\psi=0}^{N-1} e^{\frac{2\pi i x \psi}{N}} |\psi\rangle \quad (1)$$

όπου $N = 2^n$, με n ο αριθμός των qubits

Γι' αυτό, ξεκινάμε με το τι συμβαίνει, όταν εφαρμόζουμε τη Hadamard πύλη σε ένα qubit. Θα καταλήξουμε στις εξής καταστάσεις:

$$\begin{aligned} (1) \xrightarrow{N=2^1} \frac{1}{\sqrt{2}} \sum_{\psi=0}^{2-1} e^{\frac{2\pi i x \psi}{2}} |\psi\rangle &= \frac{1}{\sqrt{2}} e^{\frac{2\pi i x 0}{2}} |0\rangle + \frac{1}{\sqrt{2}} e^{\frac{2\pi i x 1}{2}} |1\rangle \\ &= \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} e^{2\pi i x} |1\rangle \quad (2) \end{aligned}$$

Τώρα, αν υποθέσουμε πως το $x = 0$, τότε

$$(2) \xrightarrow{e^{\pi i} = -1} \frac{1}{\sqrt{2}} |0\rangle + |1\rangle = |+\rangle$$

Αλλιώς, αν το $x = 1$:

$$(2) \xrightarrow{e^{\pi i} = -1} \frac{1}{\sqrt{2}} |0\rangle - |1\rangle = |-\rangle$$

Άρα, βλέπουμε πως μετασχηματίζει τις υπολογιστικές καταστάσεις σε καταστάσεις Fourier ενός qubit.

Στις περιπτώσεις με δύο ή παραπάνω qubits και, ενώ γνωρίζουμε πως το ψ μπορεί να είναι οποιοσδήποτε αριθμός, ανάλογα την είσοδο των qubits που θα ορίσουμε, εφαρμόζουμε τον παρακάτω τύπο:

$$2^{n-1} \psi_1 + 2^{n-2} \psi_2 + \dots + 2^0 \psi_n$$

ώστε να καταλήξουμε στην σχέση:

$$|\psi\rangle = \sum_{k=0}^n \psi_k 2^{n-k}$$

Ορίζοντας με αυτό το τρόπο το ψ και γνωρίζουμε την βασική φόρμουλα, μπορούμε να αντικαταστήσουμε το ψ με τη τιμή του. Έτσι, θα έχουμε:

$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}} \sum_{\psi=0}^{N-1} \prod_{k=1}^n e^{\frac{2\pi i x \psi_k}{2^k}}$$

όπου ο αντίστοιχος πίνακας της παραπάνω σχέσης είναι:

$$\begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}$$

Πως, όμως, θα χρησιμοποιήσουμε αυτή τη πληροφορία; Με αυτό το τρόπο μπορούμε να εφαρμόσουμε μια επιπλέον κβαντική πύλη Controlled-ROT που θα μετασχηματίζει τα qubits σε καταστάσεις Fourier. Για παράδειγμα, στο πρώτο qubit του κυκλώματός μας θα εφαρμόζουμε μια Hadamard πύλη για να το φέρουμε σε κατάσταση Fourier και, έπειτα μια πύλη CROT, για κάθε n qubit που θέλουμε να μετασχηματιστούν, επίσης. Το πρώτο qubit θα παίζει το ρόλο του control qubit, ενώ

τα υπόλοιπα n qubits θα είναι τα target qubits. Έτσι, θα μπορούμε να χτίσουμε ένα ενιαίο κύκλωμα, που θα χρειαστεί στην μετέπειτα υλοποίηση του κβαντικού αλγορίθμου του Shor. Έτσι, στο επόμενο υποκεφάλαιο θα αναφερθεί ο όρος της κβαντικής εκτίμησης φάσης (quantum phase estimation), όπου θα φτάνουμε όλο και πιο κοντά στον στόχο μας.

6.4.2. Η κβαντική εκτίμηση φάσης

Έστω, η παρακάτω σχέση:

$$U |\psi\rangle = e^{i\theta} |\psi\rangle$$

Προσπαθούμε να εφαρμόσουμε μια τιμή φάσης στο τυχαίας κατάστασης qubit, προσπαθώντας να βρούμε τη φάση του διανύσματος κατάστασης. Δεν γνωρίζουμε ποια θα είναι η φάση, αλλά η φόρμουλα της κβαντικής εκτίμησης φάσης θα μας βοηθήσει σε αυτό. Αρχικά, γνωρίζουμε το εξής:

$$\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \rightarrow \begin{cases} P(1) = 50\% \\ P(0) = 50\% \end{cases}$$

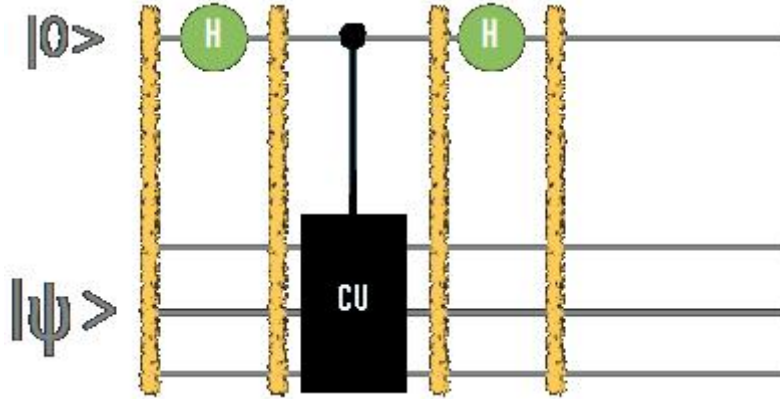
Αυτή η κατάσταση υπέρθεσης θα μπορούσε, επίσης, να φέρει και μια τιμή φάσης, όπως να βρίσκεται σε κάποια κατάσταση $|+\rangle$ ή $|-\rangle$. Σε αυτή τη περίπτωση:

$$e^{\frac{\pi i}{2}} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \rightarrow \begin{cases} P(1) = |e^{\frac{\pi i}{2}} \frac{1}{\sqrt{2}}|^2 = 50\% \\ P(0) = 50\% \end{cases}$$

Αντιλαμβανόμαστε πως εφαρμόζοντας μια τιμή φάσης, δεν αλλάζει το αποτέλεσμα. Ας προχωρήσουμε, λοιπόν, στην μαθηματική σχέση της κβαντικής εκτίμησης φάσης. Θεωρώντας πως έχουμε δύο qubits, όπου το ένα βρίσκεται στην κατάσταση $|0\rangle$, ενώ το άλλο στην κατάσταση $|\psi\rangle$, θα μελετήσουμε την αναπαράσταση του κυκλώματος στην Εικόνα 6.4.4.

Με την εφαρμογή Hadamard πύλης, γνωρίζουμε πως θα φαίνονται οι καταστάσεις, όπως παρακάτω:

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



Εικόνα 6.4.4: Πρώτο βήμα εκτίμησης φάσης

Έπειτα, με την πύλη CU θα εφαρμόσουμε μια τιμή φάσης. Η πύλη αυτή, επίσης, αλλάζει το ρόλο των qubits, όπου το control qubit παίρνει την θέση του target qubit και το αντίστροφο. Έτσι, θα ισχύει:

$$\frac{1}{\sqrt{2}} (|0\rangle|\psi\rangle + |1\rangle e^{i\theta}|\psi\rangle)$$

Με την τελευταία εφαρμογή άλλης Hadamard πύλης στο $|0\rangle$, θα έχουμε:

$$\frac{1}{2} (|0\rangle(1 + e^{i\theta}) + |1\rangle(1 - e^{i\theta})|\psi\rangle)$$

Αυτό μας βοηθάει στο εξής. Αν υπολογίσουμε τις πιθανότητες να βρίσκεται στην κατάσταση $|1\rangle$ ή $|0\rangle$, καταλαβαίνουμε πως αντιστρέφοντας την κατάσταση στη μέτρηση, μειώνουμε και την αντίστοιχη πιθανότητα. Αν το $\theta = 1$, τότε η πιθανότητα να είναι $|0\rangle$ θα είναι πολύ κοντά στο 100%, αλλά όχι ακριβώς. Αν αλλάξουμε τη τιμή φάσης, αυξάνοντας έτσι, την τιμή της θ , τότε η πιθανότητα να έχουμε $|0\rangle$ θα μειωθεί παραπάνω, αυξάνοντας έτσι και την πιθανότητα να βρεθεί στη κατάσταση $|1\rangle$. Με τον ίδιο τρόπο μπορούμε να το υλοποιήσουμε σε πολλαπλά qubits, δηλαδή από τη κατάσταση $|0\rangle$ θα έχουμε καταστάσεις $|0\rangle^n$ όπου αν για παράδειγμα $n=3$, τότε θα έχουμε 3 qubits, κ.ο.κ. Αυτό σημαίνει πως μπορούμε να χρησιμοποιήσουμε διαδοχικά τις CU πύλες για να επαναλάβουμε αυτή την περιστροφή αρκετές φορές μέχρι να έχουμε κωδικοποιήσει τη φάση θ ως αριθμό μεταξύ του 0 και 2^n στη βάση Fourier. Έπειτα, μπορούμε να εφαρμόσουμε μια QFT[†] στο κύκλωμά μας για να γυρίσουμε στην υπολογιστική φάση, δηλαδή είτε $|0\rangle$ ή $|1\rangle$. Η υλοποίηση του κυκλώματος μπορεί να γίνει και περισσότερο κατανοητή στο επόμενο και τελευταίο υποκεφάλαιο, παρουσιάζοντας και από μια σειρά εικόνων την τελική υλοποίηση του κβαντικού αλγορίθμου του Shor.

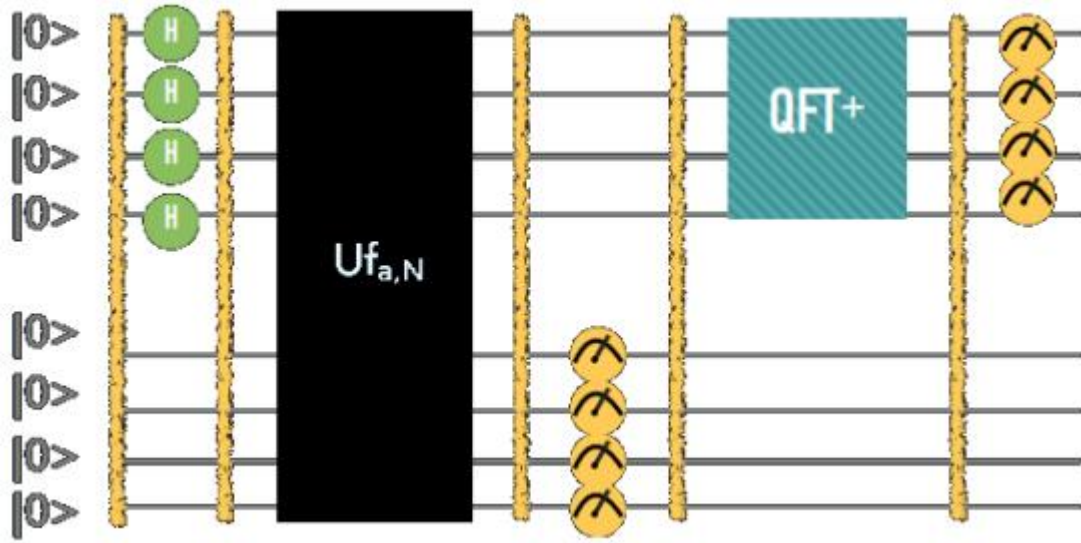
6.4.3. Υλοποίηση αλγορίθμου στο Qiskit

Είναι, πλέον, στιγμή να εφαρμόσουμε όλες τις πληροφορίες προηγουμένως και να προχωρήσουμε στην ανάπτυξη του κώδικα του αλγορίθμου, με τη μορφή ενός παραδείγματος.

Έστω πως προσπαθούμε να βρούμε το γινόμενο των πρώτων παραγόντων του αριθμού $N = 15$, με τις καταστάσεις $|\chi\rangle = |0\rangle^4$ για την πρώτη τετράδα qubits και $|\psi\rangle = |0\rangle^4$ για τα υπόλοιπα, οπότε συνολικά 8 qubits. Επιλέγουμε έναν ακέραιο q , έτσι ώστε $N < q < 2N$, έστω $q = 16$ και έναν τυχαίο ακέραιο x , έτσι ώστε $\text{MKΔ}(x, N) = 1$. Έστω $x = 13$. Η μετατροπή του, δεκαδικής μορφής, αριθμού 15 είναι 1111 σε δυαδική. Στόχος μας είναι να βρούμε τις τιμές της σχέσης:

$$N = pq$$

Το κύκλωμα που πρόκειται να υλοποιήσουμε, αναπαριστάται στην Εικόνα 6.4.5, παρακάτω.



Εικόνα 6.4.5: Αναπαράσταση κβαντικού αλγορίθμου του Shor

Ξεκινώντας, εφαρμόζουμε τις πύλες Hadamard στα τέσσερα πρώτα qubits, έστω $|\chi\rangle = |0\rangle^4$. Η μαθηματική σχέση φαίνεται ως εξής:

$$\frac{1}{4} (|0000\rangle + |0001\rangle + \dots + |1111\rangle) |\chi\rangle$$

Έπειτα, εφαρμόζοντας την U_f πύλη από τον μετασχηματισμό $f(a) = x^a \bmod N$, θα έχουμε:

$$\begin{aligned} & \frac{1}{4} (|0000\rangle |0 \oplus (13^0 \bmod 15)\rangle + |0001\rangle |0 \oplus (13^1 \bmod 15)\rangle + \dots) \\ & = \frac{1}{4} (|0000\rangle |(13 \bmod 15)\rangle + |0001\rangle |(13 \bmod 15)\rangle + \dots) \end{aligned}$$

Αν μετατρέψουμε τις δυαδικές μορφές σε δεκαδικές, η παραπάνω σχέση μας εμφανίζει και την περιοδικότητα της συνάρτησης.

$$\begin{aligned} & \frac{1}{4} (|0\rangle|1\rangle + |1\rangle|13\rangle + |2\rangle|4\rangle + |3\rangle|7\rangle + |4\rangle|1\rangle + |5\rangle|13\rangle + |6\rangle|4\rangle + |7\rangle|7\rangle + \\ & |8\rangle|1\rangle + |9\rangle|13\rangle + |10\rangle|4\rangle + |11\rangle|7\rangle + |12\rangle|1\rangle + |13\rangle|13\rangle + |14\rangle|4\rangle + |15\rangle|7\rangle) \end{aligned}$$

Παρατηρούμε ότι το $|\psi\rangle$ μπορεί να φέρει τις τιμές 1, 13, 4 ή 7. Αυτή είναι και η συνολική κατάσταση στο βήμα αυτό.

Πλέον, μπορούμε να υποθέσουμε πως η μέτρηση του $|\psi\rangle = 7$. Τότε,

$$\begin{aligned} |\chi\rangle &= \frac{1}{4} (|3\rangle + |7\rangle + |11\rangle + |15\rangle) \\ \rightarrow |\chi\rangle|\psi\rangle &= \frac{1}{2} (|3\rangle + |7\rangle + |11\rangle + |15\rangle) \otimes |7\rangle \end{aligned}$$

Προχωράμε στην εφαρμογή της $\text{QFT}^\dagger$. Γνωρίζοντας, μάλιστα, τον τύπο του QFT από το αντίστοιχο προηγούμενο κεφάλαιο, μετράμε εύκολα και τον συζυγή του. Έστω, πως παίρνουμε άλλο τυχαίο αριθμό $|\chi\rangle = 3$, τότε:

$$\begin{aligned} \text{QFT}^\dagger|\tilde{x}\rangle &= |\chi\rangle = \frac{1}{\sqrt{q}} \sum_{\psi=0}^{q-1} e^{\frac{-2\pi i x \psi}{q}} |\psi\rangle \\ \xrightarrow{N=2^4} \text{QFT}|3\rangle &= \frac{1}{\sqrt{16}} \sum_{\psi=0}^{15} e^{\frac{-2\pi i 3 \psi}{16}} |\psi\rangle \\ &= \frac{1}{8} \sum_{\psi=0}^{15} \left(e^{\frac{-3\pi i}{8\psi}} + e^{\frac{-7\pi i}{8\psi}} + e^{\frac{-11\pi i}{8\psi}} + e^{\frac{-15\pi i}{8\psi}} \right) |\psi\rangle \\ &= \frac{1}{8} (4|0000\rangle + 4i|0100\rangle - 4|1000\rangle - 4i|1100\rangle) \end{aligned}$$

όπου $|0000\rangle = |0\rangle$, $|0100\rangle = |4\rangle$, $|1000\rangle = |8\rangle$ και $|1100\rangle = |12\rangle$.

Πλέον, γνωρίζουμε πως η περίοδος r ισούται με 4, οπότε μπορούμε να επαληθεύσουμε ότι:

$$x^{\frac{r}{2}} = 13^{\frac{4}{2}} = 4(\text{mod } 15)$$

Άρα και καταλήγουμε στις μετρήσεις, πως:

$$\text{MK}\Delta(13^{\frac{4}{2}} + 1, N) = \text{MK}\Delta(168, 15) = 3$$

$$\text{MK}\Delta(13^{\frac{4}{2}} - 1, N) = \text{MK}\Delta(170, 15) = 5$$

Έτσι, λοιπόν, λειτουργεί και ο κβαντικός αλγόριθμος του Shor. Παρακάτω, επισυνάπτονται οι αντίστοιχες υλοποιήσεις των κυκλωμάτων στο Qiskit.

```
In [1]: from qiskit import *
import numpy as np
from qiskit.tools.visualization import plot_histogram

In [2]: def c_amod15(a, power):
    """Controlled multiplication by a mod 15"""
    if a not in [2,7,8,11,13]:
        raise ValueError("'a' must be 2,7,8,11 or 13")
    U = QuantumCircuit(4)
    for iteration in range(power):
        if a in [2,13]:
            U.swap(0,1)
            U.swap(1,2)
            U.swap(2,3)
        if a in [7,8]:
            U.swap(2,3)
            U.swap(1,2)
            U.swap(0,1)
        if a == 11:
            U.swap(1,3)
            U.swap(0,2)
        if a in [7,11,13]:
            for q in range(4):
                U.x(q)
    U = U.to_gate()
    U.name = "%i^%i mod 15" % (a, power)
    c_U = U.control()
    return c_U
```

Εικόνα 6.4.6: Υλοποίηση συνάρτησης modulo

```

In [3]: n_count = 8
        a = 7

In [4]: def qft_dagger(n):
        qc = QuantumCircuit(n)
        for qubit in range(n//2):
            qc.swap(qubit, n-qubit-1)
        for j in range(n):
            for m in range(j):
                qc.cu1(-np.pi/float(2**(j-m)), m, j)
            qc.h(j)
        qc.name = "QFT Dagger"
        return qc

In [5]: # Create QuantumCircuit with n_count counting qubits
        # plus 4 qubits for U to act on
        qc = QuantumCircuit(n_count + 4, n_count)

        # Initialize counting qubits
        # in state |+>
        for q in range(n_count):
            qc.h(q)

        # And auxiliary register in state |1>
        qc.x(3+n_count)

        # Do controlled-U operations
        for q in range(n_count):
            qc.append(c_amod15(a, 2**q),
                    [q] + [i+n_count for i in range(4)])

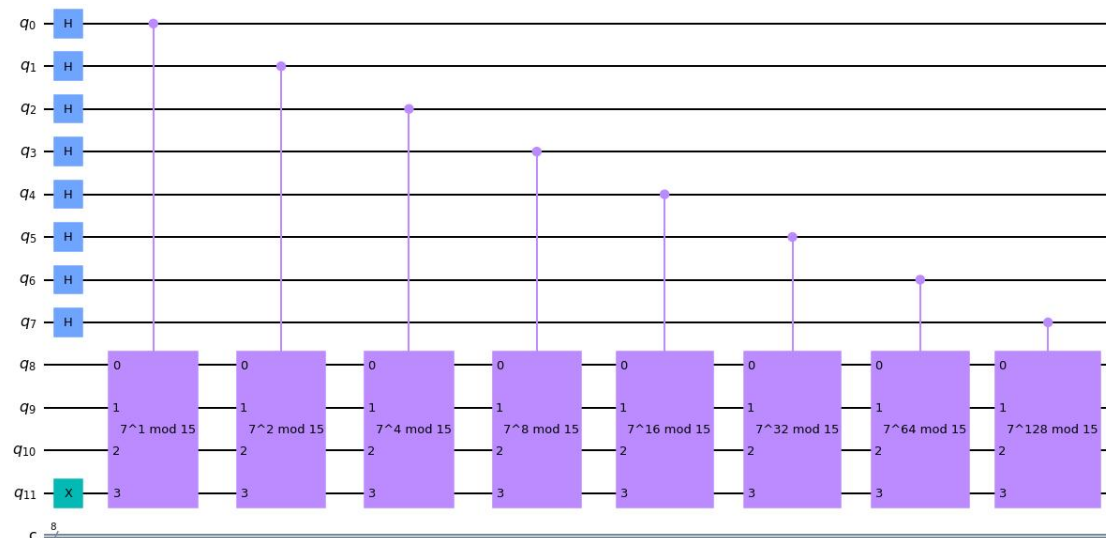
        # Do inverse-QFT
        qc.append(qft_dagger(n_count), range(n_count))

        # Measure circuit
        qc.measure(range(n_count), range(n_count))

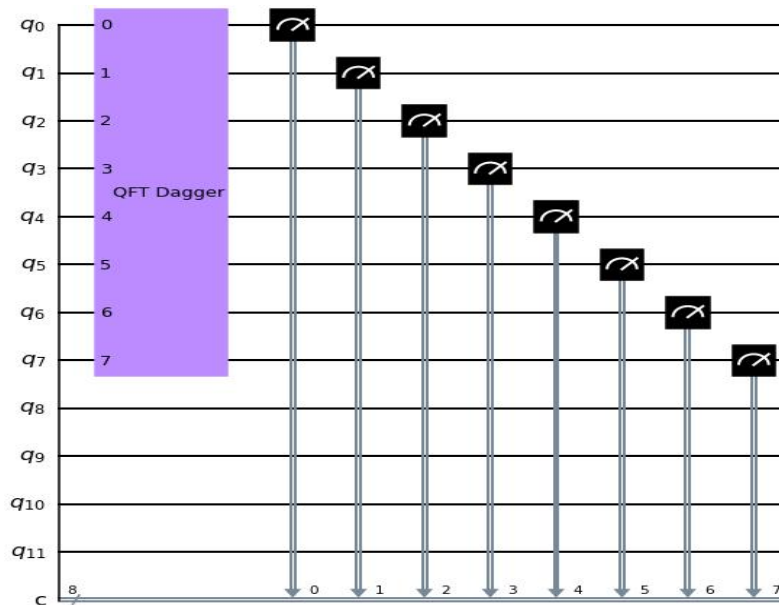
        qc.draw(output='mpl')

```

Εικόνα 6.4.7: Υλοποίηση κβαντικού αλγορίθμου του Shor

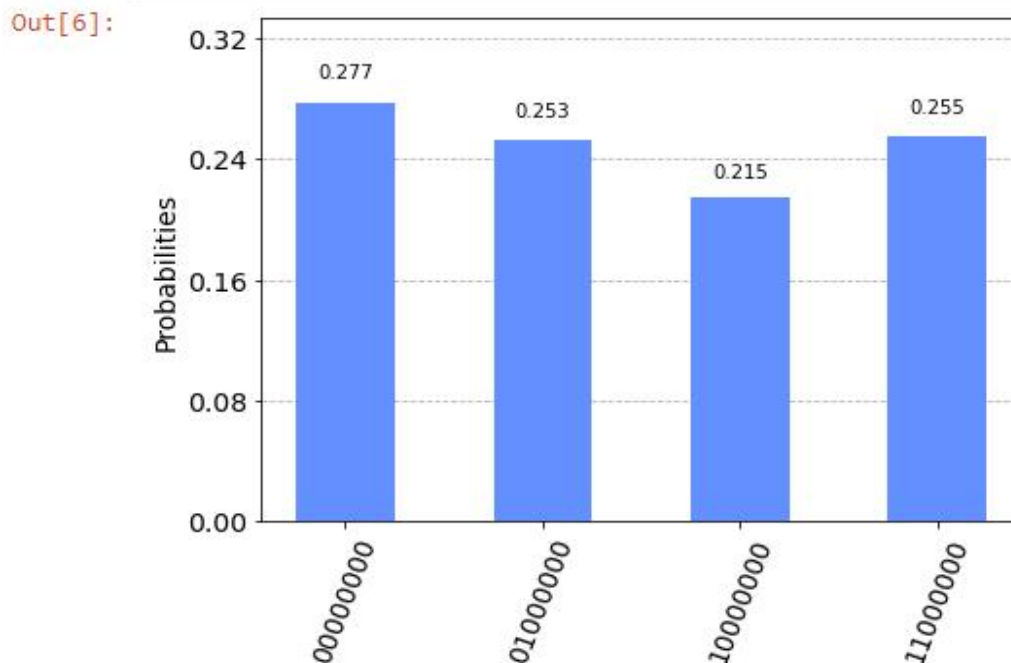


Εικόνα 6.4.8 (a): Αναπαράσταση κυκλώματος αλγορίθμου του Shor



Εικόνα 6.4.8 (β): Αναπαράσταση κυκλώματος αλγορίθμου του Shor

```
In [6]: backend = Aer.get_backend('qasm_simulator')
results = execute(qc, backend, shots=1024).result()
counts = results.get_counts()
plot_histogram([counts])
```



```
In [7]: # r = 4 => a^r/2 - 1 = 7^2 - 1 = 48 (gcd(15,48) = 3)
# r = 4 => a^r/2 + 1 = 7^2 + 1 = 50 (gcd(15,50) = 5)
```

Εικόνα 6.4.9: Αποτελέσματα παραδείγματος κβαντικού αλγορίθμου του Shor

Βιβλιογραφία

- [1] Paul Benioff (1980) *"The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines"*, Journal of Statistical Physics, 22, 563
- [2] Paul Benioff (1982) *"Quantum Mechanical Models of Turing Machines That Dissipate No Energy"*, Physical Review Letters, 48, 1581
- [3] Feynman, Richard (1982) *"Simulating Physics with Computers"*.
- [4] Deutsch, David (1985). *"Quantum theory, the Church-Turing principle and the universal quantum computer"*. Proceedings of the Royal Society A. 400 (1818): 97–117
- [5] Shor, P.W. (1994). *"Algorithms for quantum computation: discrete logarithms and factoring"*, Proceedings 35th Annual Symposium on Foundations of Computer Science. IEEE Comput. Soc. Press: 124–134.
- [6] Macauley Coggins (2021) *"Introduction to Quantum Computing with Qiskit"* Scarborough Quantum Computing Ltd
- [7] Κωνσταντίνος Σιμσερίδης (2015) *"Καταστάσεις της ύλης"*, Τμήμα Φυσικής, ΕΚΠΑ, Αθήνα
- [8] Christos Papalitsas, Panayiotis Karakostas, Theodore Andronikos, *"A Performance Study of the Impact of Different Perturbation Methods on the Efficiency of GVNS for Solving TSP"*, (MDPI - Open Access Publishing), Applied System Innovation 2019, 2(4), 31. DOI: 10.3390/asi2040031.
- [9] Christos Papalitsas, Theodore Andronikos, *"Unconventional GVNS for Solving the Garbage Collection Problem with Time Windows"*, (MDPI - Open Access Publishing), Technologies 2019, 7(3), 61. DOI: 10.3390/technologies7030061.
- [10] Christos Papalitsas, Theodore Andronikos, Konstantinos Giannakis, Georgia Theocharopoulou, Sofia Fanarioti, *"A QUBO Model for the Traveling Salesman Problem with Time Windows"*, (MDPI - Open Access Publishing), Algorithms 2019, 12(11), 224. DOI: 10.3390/a12110224.
- [11] Evangelos Stogiannos, Christos Papalitsas, Theodore Andronikos, *"Experimental Analysis of Quantum Annealers and Hybrid Solvers Using Benchmark Optimization Problems"*, Mathematics (MDPI - Open Access Publishing), Mathematics 2022, 10(8), 1294. DOI: 10.3390/math10081294.

- **[12]** Michael Ampatzis, Theodore Andronikos, "*QKD Based on Symmetric Entangled Bernstein-Vazirani*", Entropy (MDPI - Open Access Publishing), Entropy 2021, 23(7), 870. DOI: 10.3390/e23070870.
- **[13]** Theodore Andronikos, Alla Sirokofskich, "*The Connection between the PQ Penny Flip Game and the Dihedral Groups*", Mathematics (MDPI - Open Access Publishing), Mathematics 2021, 9(10), 1115. DOI: 10.3390/math9101115.
- **[14]** Theodore Andronikos, Michail Stefanidakis, "*A Two-Party Quantum Parliament*", (MDPI - Open Access Publishing), Algorithms 2022, 15(2), 62. DOI: 10.3390/a15020062.
- **[15]** Theodore Andronikos, Alla Sirokofskich, Kalliopi Kastampolidou, Magdalini Varvouzou, Konstantinos Giannakis, Alexandros Singh, "*Finite Automata Capturing Winning Sequences for All Possible Variants of the PQ Penny Flip Game*", Mathematics, Special Issue Mathematical Game Theory, (MDPI - Open Access Publishing), Mathematics 2018, 6(2), 20. DOI: 10.3390/math6020020.
- **[16]** Konstantinos Giannakis, Theodore Andronikos, "*Membrane automata for modeling biomolecular processes*", Natural Computing, Springer, Volume 16, Issue 1, pages 151-163, March 2017. DOI: 10.1007/s11047-015-9518-1.
- **[17]** Konstantinos Giannakis, Georgia Theocharopoulou, Christos Papalitsas, Theodore Andronikos, Panayiotis Vlamos, "*Associating ω -automata to Path Queries on Webs of Linked Data*", Engineering Applications of Artificial Intelligence, Elsevier, Volume 51, pages 115-123, May 2016. DOI: 10.1016/j.engappai.2016.01.013.
- **[18]** Dimitris Kostadimas, Kalliopi Kastampolidou, Theodore Andronikos, "*Correlation of biological and computer viruses through evolutionary game theory*", IEEE Proceedings of the 16th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP), 4-5 November, Corfu, Greece, 2021. DOI: 10.1109/SMAP53521.2021.9610778.
- **[19]** Noson S. Yanofsky and Mirco A. Mannucci (2008) "*Quantum Computing for Computer Scientists*", Cambridge University Press
- **[20]** George Boole (1847) "*The Mathematical Analysis of Logic, Being an Essay towards a Calculus of Deductive Reasoning*" Archived 11 May 2016 at the Wayback Machine (London, England: Macmillan, Barclay, & Macmillan,).
- **[21]** Ιωάννης Καραφυλλίδης (2015) "*Κβαντική Υπολογιστική*" Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Δ.Π.Θ.
- **[22]** C.P. Williams (2011) "*Explorations in Quantum Computing*", Springer-Verlag London Limited 2011, p. 717, Chapter 2

Δικτυογραφία

- <https://commons.wikimedia.org/wiki/File:Double-slit.svg>
 - <https://commons.wikimedia.org/wiki/File:Ebohr1.svg>
 - https://commons.wikimedia.org/wiki/File:Two-Slit_Experiment_Particles.svg
 - https://commons.wikimedia.org/wiki/File:Two-Slit_Experiment_Electrons.svg
 - https://commons.wikimedia.org/wiki/File:RTL_NOT_Gate.svg
-
- <https://qiskit.org/>
 - <https://quantum-computing.ibm.com/composer/docs/idx/>
 - <https://quantum-computing.ibm.com/composer/docs/idx/guide/>
 - <https://qiskit.org/textbook/ch-states/introduction.html>
 - <https://qiskit.org/documentation/tutorials.html>