QUANTUM RANDOM WALK

Thesis
Submitted in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

PHYSICS

by

SHWETA DADHWAL
(206054)

PHYSICS DEPARTMENT
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE -575025
June, 2022

DECLARATION

I, hereby declare that the report of P.G Project Work entitled "Quantum Random Walk" which is being submitted to the National Institute of Technology Karnataka, Surathkal, in partial fulfilment of the requirements for the award of the Degree of Master of Science in the Department of Physics, is a bonafide report of the work carried out by me. The material contained in this report has not been submitted to any University or Institution for the award of any degree.

NITK SURATHKAL
June 2022

SHWETA DADHWAL
206PH024

CERTIFICATE

This is to certify that the P.G. Project Work Report entitled **"Quantum Random Walks"** submitted by **SHWETA DADHWAL, (Roll Number: 206PH024)** as the record of the work carried out by him, is accepted as the **P.G. Project Work Report Submission** in partial fulfilment of the requirements for the award of the degree of **EMaster of Science** in the **Department of Physics**.

External Guide	**Internal Guide**
Dr. Saikat Dutta	**Dr Deepak Vaid**
(Name and signature with date and seal)	(Name and signature with date and seal)

Chairman- DPGC
(Signature with date and seal)

ACKNOWLEDGEMENT

I would like to express my sincere thanks to my project advisor Dr Deepak Vaid, Department of Physics, NITK, Surathkal for his guidance, instruction and for the valuable project opportunity that he provided for me. I would like to thank our project coordinator Dr Kartick Tarafdar, Department of Physics, NITK Surathkal for his kind support. Specially, I would like to thank my parents for their support and for having faith in me.

Place: NITK SURATHKAL
Date: June 2022

SHWETA DADHWAL
206PH024

Abstract

Quantum walks (QWs) are becoming widespread in physics. They were first introduced as quantum versions of random classical processes and quantum cellular automata , and have since been extensively studied, particularly in the context of quantum information science. Here we will start with some basics of Quantum Computation and see how to construct, run some of the basic Quantum Algorithms and submit them to run on IBM's Quantum computing simulator. IBM provides a software called Quantum Information software kit (QISKIT) that provides interface between quantum computer and users with the help of which one compute any program source code on real quantum computer. And then we will see what a Quantum random walk is and discover some fascinating features of QRW and how it differs from Classical Random Walk. And study different models of Quantum Random Walks and their implementation on 1D, 2D lattices and graphs.

Contents

Acknowledgements **i**

Abstract **iv**

1 Introduction **1**

2 Basics of Quantum Computation **3**

2.1 Qubits 3

2.2 **Multiple particle Quantum States** 4

2.3 Quantum Gates 4

2.3.1 Single Qubit gates 6

2.3.2 Multi qubit gates 7

2.4 Entanglement 9

2.4.1 Bell States 9

2.5 Teleportation 11

3 Deutsch Josza Algorithm **13**

4 Grover's algorithm **17**

5 Quantum Fourier Transform **23**

6 Quantum Phase Estimation **27**

7 Classical Random Walk in 1-D **29**

7.1 Implementation of CRW in 1D 32

8 Quantum Random walk **34**

8.1 Introduction 34

8.2 Coined Quantum Random walk 38

8.2.1 Implementation on IBM computer 41

8.3 Staggered Quantum Walk 44

8.3.1 SQW on 1D lattice 45

8.3.2 Implementation on IBM Computers 50

8.3.3 Quantum Walk on 2-D square lattice 52

8.3.4 Decomposition of evolution operator 53
8.3.5 Implementation on IBM computers 54
8.4 Szegedy's Quantum Walks 56

9 Future Scope 59

10 Conclusion 60

A Teleportation circuit in QISKIT 62

B Deutsh Josza Algorithm in QISKIT 65

C Steps to write Grover's Algorithm 68

D Classical random walk in QISKIT 70

E Coined Quantum Random walk in 1D in QISKIT 71

F 1D Staggered Quantum walk in QISKIT 74

G 2D Staggered Quantum walk in QISKIT 77

H Link to Source code: 81

1 Introduction

Random walk is a process by which a particles moves in random direction in mathematical space with certain probabilities assigned by us. A random walk is one of the most extensively used random processes in science and engineering for simulation and approximation.
It was first proposed by Y. Aharonov et al. in 1993[2]. They demonstrated that, due to the nature of quantum characteristics, the average length of a walking path on a line can be greater than that of a classical random walk.Here we will discover the new features of Quantum Random Walk and plot the probability distribution graphs to verify the results. As a result, the quantum random walk can be utilised to build a variety of additional quantum algorithms. A quantum computer uses discrete registers, which means its state space is a vast but finite Hilbert space. We'll be able to map the quantum walk to a computation by such a machine by discretizing it. So a Quantum computer can efficiently simulate the quantum random walk and use it to carry out specific computational tasks. Here we'll want to use some of the 'strange' consequences of the walk afforded by quantum mechanics to boost our computing capacity. Furthermore, it can solve graph issues, which can be used to formulate many real-world problems. For this many types of Quantum Random walk models are being proposed like Staggered Quantum Walk(SQW), Szegedy's model for quantum walk which perform Quantum walk irrespective of the coin state. Quantum algorithms, on the other hand, are only effective in reality if they can be implemented efficiently on a quantum computer.
Aside from the fascinating new physics which we'll discover in the study of quantum random walks, we'll want to use some of the 'strange' consequences of the walk afforded by quantum mechanics to boost our computing capacity. The concept is that a quantum computer can efficiently implement (simulate) the quantum random walk and use it to carry out specific computational tasks.In this way, we hope to use quantum random walk features to find more efficient algorithms on a quantum computer.

This report is organized in nine sections. section 1 discusses a general Introduction of the report. section 2 describes the basics of Quantum computation and in the same section we introduce the mathematical preliminaries for working with QISKIT such as subsection 2.1 Qubits which are the building blocks of Quantum computer,

subsection 2.2 how to define multi particle quantum state, subsection 2.3 Quantum gates which are used to make quantum circuits in quantum computer and then we defined entanglement subsection 2.4 and the formation of bell sates. Using all these building blocks, at last we introduced quantum teleportation circuitsubsection 2.5. In further sections we discussed some of the Quantum algorithms.In section 3 we discussed Deutsch Josza Algorithm. In section 4 we discussed Grover's algorithm. In section 5 we discussed Quantum fourier Transform. In section 6 we Quantum Phase Estimation. In section 7 we introduced the most basic Classical random walk in 1D. And then in section 8 we discussed Quantum Random Walks and subsection 8.1 discovered some strange features of Quantum random walks and how they differ from classical random walk. Then we studied about different types of Quantum random walks. In subsection 8.2 we studied coined quantum random walk in 1D and it's implementation on IBM computers in subsubsection 8.2.1.In subsection 8.3 we introduced Staggered quantum random walk and its basic definitions. In further subsubsection 8.3.1 we studied SQW in 1D and it's implementation on IBM computers in subsubsection 8.3.2. In subsubsection 8.3.3 we studied SQW in 1D and it's implementation on IBM computers in subsubsection 8.3.5. In subsection 8.4 we introduced Szegedy's Quantum Walk. In section 9 we discussed about the future scope of Quantum random walks. And at last we concluded this report in section 10.
Appendix A includes code for Quantum teleportation circuit in QISKIT. Appendix B includes python code for Deutsch Josza Algorithm in QISKIT. Appendix C includes python code for Grover's algorithm in QISKIT. Appendix D includes python code for Classical Random Walk in 1-D in QISKIT. Appendix E includes python code for Coined Quantum Random Walk in 1-D in QISKIT. Appendix F includes python code for Staggered Quantum Walk on 1D lattice in QISKIT. Appendix G includes python code for Staggered Quantum Walk on 2-D square lattice in QISKIT.

2 Basics of Quantum Computation

2.1 Qubits

A classical computer operates on strings of bits. A quantum computer, on the other hand, works with qubits. We treat qubits as abstract mathematical objects. As a classical bit has a state – either 0 or 1 – a qubit also has two posible state $|0\rangle$ or $|\uparrow\rangle$ and $|1\rangle$ or $|\downarrow\rangle$

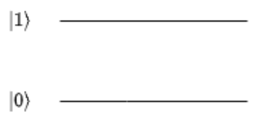


Figure 1: Quantum Two level system: Qubit

So a qubit is a two level system spanned by two states $|0\rangle, |1\rangle$ or $|\uparrow\rangle, |\downarrow\rangle$. Or we can say that the states is spanned by n-bit strings $|x_1x_2x_3...x_{ni}\rangle$ with $x_i\epsilon$ 0, 1. It is also possible to form linear combinations of states, often called superpositions:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where α and β are complex numbers. Hence we can say that, the state of a qubit is a vector in a two-dimensional complex vector space and the states $|0\rangle$ and $|1\rangle$ are known as computational basis states, and form an orthonormal basis for this vector space. A quantum computer state can be in any superposition of these basis states.

2.2 Multiple particle Quantum States

We use tensor product to describe Multi-particle states:

$$|a\rangle \otimes |b\rangle = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \otimes \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1b_1 \\ a_2b_2 \\ a_2b_1 \\ a_2b_2 \end{pmatrix}$$

Example: system A is in state $|1\rangle_A$ and system B is in state $|0\rangle_B$

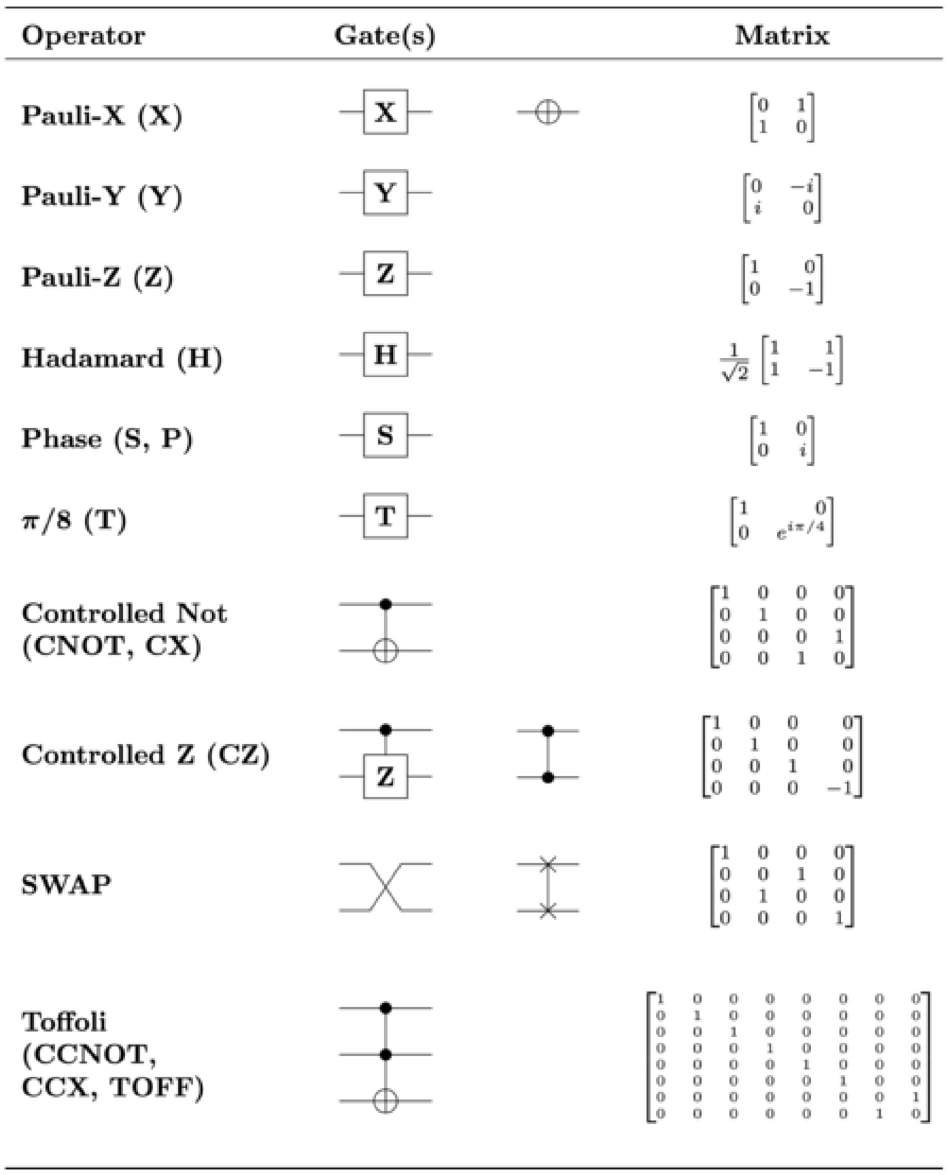
So the total biparticle state is $—10\rangle_{AB} = |1\rangle_A \otimes |0\rangle_B = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \otimes \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1b_1 \\ a_2b_2 \\ a_2b_1 \\ a_2b_2 \end{pmatrix}$

Remark:
States of this form are called uncorrelated, but there are also bi-particle states that cannot be written as $|\psi\rangle_A \otimes |\psi\rangle_B$ These states are correlated and sometimes even Entangled, Example:

$$\begin{aligned} |\psi\rangle_{AB} &= \frac{1}{\sqrt{2}} \left(|00\rangle_{AB} + |11\rangle_{AB}\right) \\ &\neq |\psi_A\rangle \otimes |\psi\rangle_B \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \end{aligned}$$

2.3 Quantum Gates

A quantum computer is formed from a *quantum circuit* comprising wires and elementary quantum gates to carry around and manipulate quantum information, similar to how a conventional computer is built from an electrical circuit containing wires and logic gates. As Quantum theory is unitary, quantum gates are represented by Unitary matrices.[5]



Operator	Gate(s)	Matrix
Pauli-X (X)	$\boxed{\mathbf{X}}$	$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$
Pauli-Y (Y)	$\boxed{\mathbf{Y}}$	$\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$
Pauli-Z (Z)	$\boxed{\mathbf{Z}}$	$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$
Hadamard (H)	$\boxed{\mathbf{H}}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$
Phase (S, P)	$\boxed{\mathbf{S}}$	$\begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}$
$\pi/8$ (T)	$\boxed{\mathbf{T}}$	$\begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$
Controlled Z (CZ)	$\boxed{\mathbf{Z}}$	$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -1\end{bmatrix}$
SWAP		$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$

Figure 2: Quantum Logic Gates

Real dirac notation,

$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix}$$

$U = U_{00}|0\rangle\langle 0| + U_{01}|0\rangle\langle 1| + U_{10}|1\rangle\langle 0| + U_{11}|1\rangle\langle 1|$

2.3.1 Single Qubit gates

1. Pauli-X Gate

   Also known as Classical NOT Gate

$$\begin{aligned}
\sigma_x &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0| \\
\sigma_x|0\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \qquad (1) \\
\sigma_x|0\rangle &= |1\rangle \\
\sigma_x|1\rangle &= |0\rangle
\end{aligned}$$

   Pauli-X gate flips the bit and thus is called bit flip gate. It represents rotation around x-axis by angle π

   since $|+\rangle$ and $|-\rangle$ states lie on x-axis hence there will be no effect of bit flip on them.

2. Pauli-Z gate

$$\begin{aligned}
\sigma_x &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1| \\
\sigma_x|+\rangle &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix} = |1\rangle \qquad (2) \\
\sigma_z|+\rangle &= |-\rangle \\
\sigma_z|-\rangle &= |+\rangle
\end{aligned}$$

   Pauli-Z gate flips the state $|+\rangle to —-\rangle and vice-versa and thus is called phase flip gate. It represents ro$ $axis by angle \pi$

3. Pauli-Y gate

$$\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = i\sigma_x\sigma_z \qquad (3)$$

   Pauli-Y gate is the combination of bit flip and phase flip. It represents rotation around y-axis by angle π

$$\sigma_i^2 = I = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{4}$$

σ_x,σ_y,σ_z are Pauli Matrices
Together with identity they form basis of 2×2 matrices

4. Hadamard Gate

$$\begin{aligned} H &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= |0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1| \\ H|0\rangle &= |+\rangle \\ H|1\rangle &= |-\rangle \end{aligned} \tag{5}$$

This gate is used to creates superposition states.It is also used to change between x and z basis

5. S Gate

$$\begin{aligned} S &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \\ &= |0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1| \\ S|+\rangle &= |+i\rangle \\ S|-\rangle &= |-i\rangle \end{aligned} \tag{6}$$

S gate adds 90 to the phase ϕ. S.H is applied to change from z to y basis.

2.3.2 Multi qubit gates

1. CNOT Gate(CX)
Acts on a pair of qubits, with one acting as 'control' and the other as 'target'. Performs a NOT on the target whenever the control is $|1\rangle$. If the control qubits is in superposition state, this gate creates entanglement.

2. Toffoli Gate(CCX)
   $\rightarrow$ Double Controlled-Not
   It has two control qubits and one target.
   Applies NOT to target only when both controks are in state $|1\rangle$
   Toffoli gate with Hadamard Gate is a universal gate set for quantum.

3. SWAP Gate
   Swaps the state of two qubits.

4. Identity Gate
   Is actually the absence of gate. It ensures nothing is applied to a qubit for one unit of gate time.

Refer Figure 2 for more Quantum gates and their corresponding matrix notation.

2.4 Entanglement

If a pure state $|\psi\rangle_{AB}$ on systems A,B, connot be written as $|\psi\rangle_A \otimes |\phi_B\rangle$ then it is entangled.

2.4.1 Bell States

There are four Bell states that are maximally entangled and build an orthogonal basis.

$$\begin{aligned}
|\psi^{00}\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \\
|\psi^{01}\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right) \\
|\psi^{10}\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) \\
|\psi^{11}\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right)
\end{aligned} \tag{7}$$

In general, we can write

$$|\psi^{ij}\rangle = \left(I \otimes \sigma_x^j \sigma_z^i\right)|\psi^{00}\rangle$$

Creation of Bell States

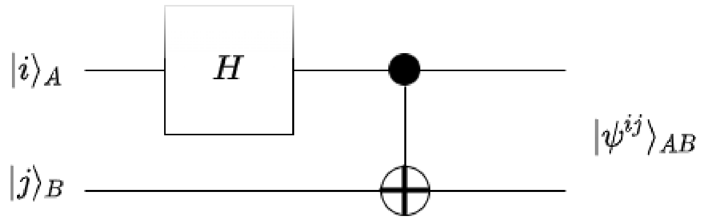


Figure 3: Creation of bell states

Initial state	H_A	$CNOT_{AB}$
$|\psi^{00}\rangle$	$\left(\frac{1}{\sqrt{2}}|00\rangle + |10\rangle\right)$	$\left(\frac{1}{\sqrt{2}}|00\rangle + |11\rangle\right)$
$|\psi^{00}\rangle$	$\left(\frac{1}{\sqrt{2}}|01\rangle + |11\rangle\right)$	$\left(\frac{1}{\sqrt{2}}|01\rangle + |10\rangle\right)$
$|\psi^{00}\rangle$	$\left(\frac{1}{\sqrt{2}}|00\rangle - |10\rangle\right)$	$\left(\frac{1}{\sqrt{2}}|00\rangle - |11\rangle\right)$
$|\psi^{00}\rangle$	$\left(\frac{1}{\sqrt{2}}|01\rangle - |11\rangle\right)$	$\left(\frac{1}{\sqrt{2}}|01\rangle - |10\rangle\right)$

BELL STATES

```
import numpy as np
from qiskit import *
from qiskit.quantum_info import Statevector
from qiskit.visualization import plot_state_qsphere
qreg = QuantumRegister(2, 'q')
qc=QuantumCircuit(qreg)
qc.reset(qreg[0])
qc.reset(qreg[1])
qc.h(0)
qc.cx(0,1)
qc.x(0)
qc.z(0)
sv=Statevector.from_instruction(qc).data
print(sv)
plot_state_qsphere(sv)
```

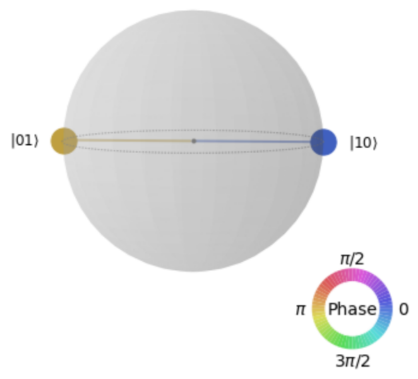


Figure 4: Formation of Bell states on Q-sphere

2.5 Teleportation

GOAL:

Alice and Bob met long ago and they generated an EPR pair, each taking one qubit of the EPR pair when they separated. Alice wants to send her unknown state to Bob. She can only send him classical information."

Solution: They both share the maximally entangled state.

$$|\psi^{00}\rangle_{AB} = \frac{1}{\sqrt{2}}\left[|00\rangle_{AB} + |11\rangle_{AB}\right]$$

Initial state of the total system

$$|\phi\rangle_s \otimes |\psi^{00}\rangle_{AB} = \frac{1}{\sqrt{2}}\left(\alpha|011\rangle_{SAB} + \beta|100\rangle_{SAB} + \beta|111\rangle_{SAB}\right)$$

$$\begin{aligned}
= \frac{1}{2\sqrt{2}}[&(|00\rangle_{SA} + |11\rangle_{SA}) \otimes (\alpha|0\rangle_B + \beta|1\rangle_B) \\
&+(|01\rangle_{SA} + |110\rangle_{SA}) \otimes (\alpha|1\rangle_B + \beta|0\rangle_B) \\
&+(|00\rangle_{SA} - |11\rangle_{SA}) \otimes (\alpha|0\rangle_B - \beta|1\rangle_B) \\
&+(|01\rangle_{SA} - |10\rangle_{SA}) \otimes (\alpha|1\rangle_B - \beta|0\rangle_B)] \\
= \frac{1}{\sqrt{2}}[&(|\psi^{00}\rangle_{SA} \otimes |\phi\rangle_B \\
&+(|\psi^{01}\rangle_{SA} \otimes \sigma_x|\phi\rangle_B \\
&+(|\psi^{10}\rangle_{SA} \otimes \sigma_z|\phi\rangle_B \\
&+(|\psi^{11}\rangle_{SA} \otimes \sigma_x\sigma_z|\phi\rangle_B]
\end{aligned}$$

PROTOCOL

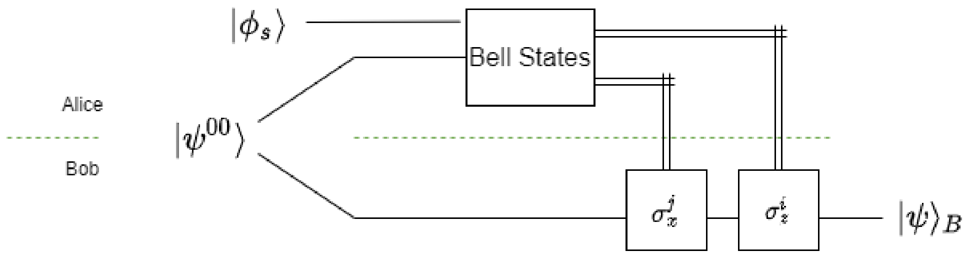


Figure 5: Teleportation protocol

1. Alice measures on S and A in bell basis.

Alice measures state	$\rightarrow$	corresponding Bob's state
$\|\psi^{00}\rangle$		$\|\phi\rangle_B$
$\|\psi^{01}\rangle$		$\sigma_x\|\phi\rangle_B$
$\|\psi^{10}\rangle$		$\sigma_z\|\phi\rangle_B$
$\|\psi^{11}\rangle$		$\sigma_x\sigma_z\|\phi\rangle_B$

2. She sends her classical output i, j to Bob.
3. Bob applies $\sigma_z^i\sigma_x^j$ to his qubit and gets $|phi\rangle$.
4. Bob applies $\implies$ Bob's final state.

		i	j	$Bob's applies$	$Bob's final state$
$\|\psi^{00}\rangle$	$\|\phi\rangle_B$	0	0	I	$\|\phi\rangle_B$
$\|\psi^{01}\rangle$	$\sigma_x\|\phi\rangle_B$	0	1	σ_x	$\|\phi\rangle_B$
$\|\psi^{10}\rangle$	$\sigma_z\|\phi\rangle_B$	1	0	σ_z	$\|\phi\rangle_B$
$\|\psi^{11}\rangle$	$\sigma_x\sigma_z\|\phi\rangle_B$	1	1	$\sigma_z\sigma_x$	$\|\phi\rangle_B$

NOTE:
That Alice's state collapses during the measurement, so she does not have the initial state $|\phi\rangle$ anymore.
This is expected due to no-cloning theorem, as she cannot copy her state, but just send her state to Bob when destroying her own.
States gets teleported .

Refer Appendix A to see how to Implement Teleportation circuit in QISKIT.

3 Deutsch Josza Algorithm

GOAL:
To determine some property of the oracle (constant or balanced) using the minimal number of queries.
On a classical computer, such an oracle is given by a function: $f : (0,1)^n \rightarrow (0,1)^n$
On a Quantum computer, the oracle must be reversible.

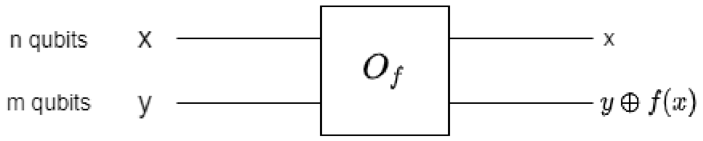


Figure 6: Reversible oracle

1. Let x and y are in state $|0\rangle$.

$$|x\rangle = |0\rangle, |y\rangle = |0\rangle$$
$$i.e.|\psi_{in}\rangle = |x\rangle|y\rangle = |0\rangle|0\rangle$$
$$|\psi_{out} = |x\rangle|y + f(x)\rangle = |0\rangle|0 \oplus f(0)\rangle = |0\rangle f(0)$$

2. Let state of y is changed from $|0\rangle$ to $|1\rangle$.

$$|x\rangle = |0\rangle, |y\rangle = |1\rangle$$
$$i.e.|\psi_{in}\rangle = |x\rangle|y\rangle = |0\rangle|1\rangle$$
$$|\psi_{out} = |x\rangle|y + f(x)\rangle = |0\rangle|1 \oplus f(0)\rangle = \begin{cases} |0\rangle|1\rangle, & \text{when} f(0) = 0 \\ |0\rangle|0\rangle, & \text{when} f(0) = 1 \end{cases}$$

3. Now let y is in superposition state.

$$|\psi_{in}\rangle = |x\rangle|y\rangle = |0\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$
$$|\psi_{out}\rangle = |0\rangle \left(\frac{|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle}{\sqrt{2}}\right) = \begin{cases} |0\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & \text{when} f(0) = 0 \\ -|0\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & \text{when} f(0) = 2 \end{cases}$$

$$|\psi_{out}\rangle = (-1)^{f(0)}|0\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \tag{8}$$

Equation 8 known as Deutsch Algorithm
U_f also called Bit flip oracle, can be seen as unitary which performs the map,

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

for$f : (0,1)^n \rightarrow (0,1),$ we can construct U_f.

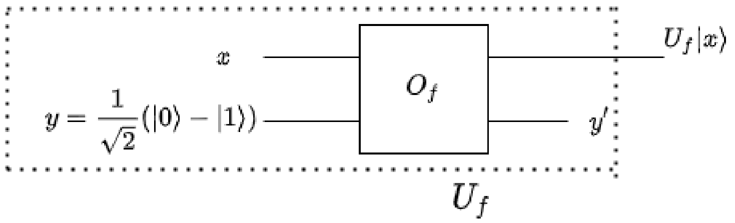


Figure 7: Deutsch Josza oracle

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

$$U_f|x\rangle = (-1)^{f(x)}|x\rangle \text{ Independent of } |y\rangle \tag{9}$$

4. When both Qubits are in superposition state:

$$\begin{aligned} |\psi_{in}\rangle &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \\ &= \frac{1}{\sqrt{2}}\left(|0\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) + |1\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\right) \end{aligned}$$

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

$$
\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)=\begin{cases}\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right); & \text{when} f(0)=f(1)\\ \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right); & \text{when} f(0)\neq f(1)\end{cases}
$$

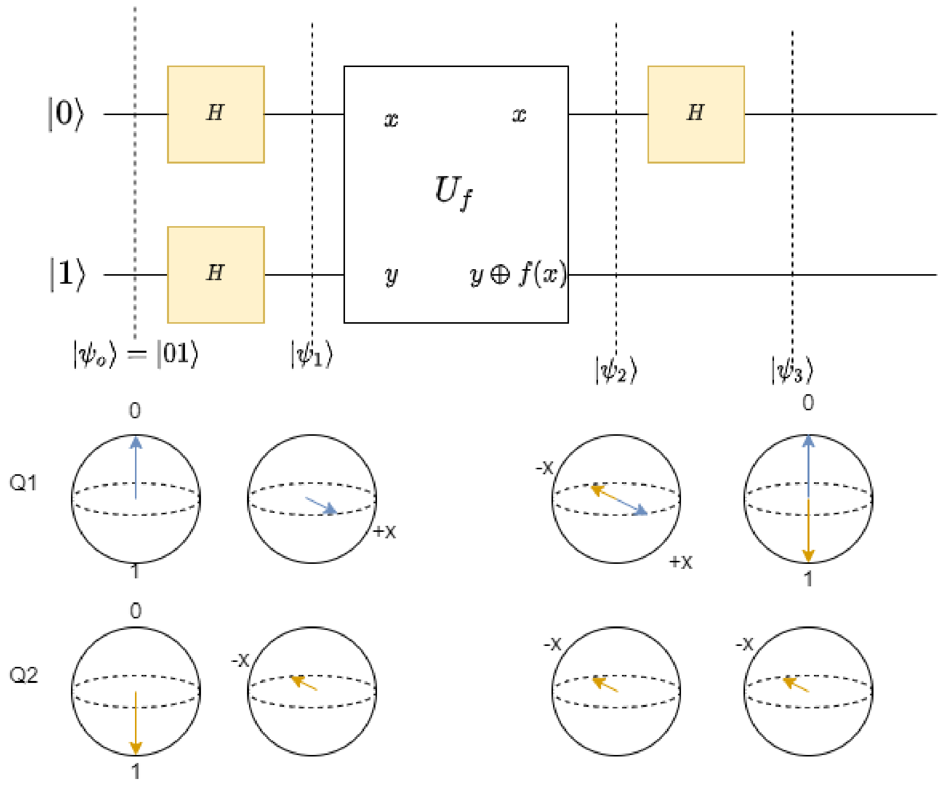


Figure 8: Deutsch Josza circuit for 2 qubits

From Figure 8 you can tell if f(0)=f(1) by checking the first bit at one go. Classical algorithm would require two checks.

N Qubit GENERALIZATION: DEUTSCH JOZSA ALGORITHM

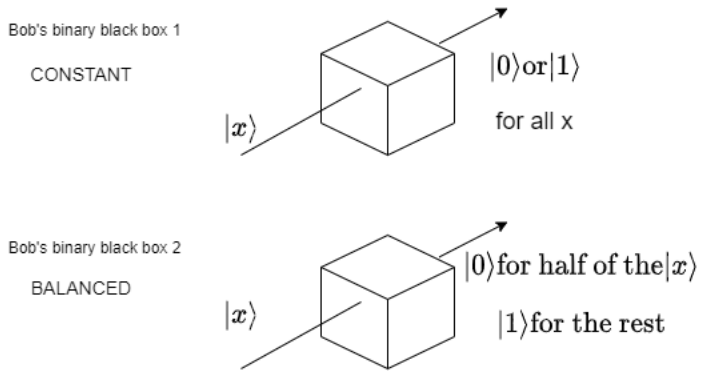


Figure 9: Deutsch Josza oracle

Alice can send bob a bunch of numbers from 0 to n and a bit to write the answer. Bob promises to use any one of the black boxes for all the numbers: Doesn't tell which one. Bob sends the bit back to Alice.
Alice needs to determine which one did he use? (With minimum no. of trials)

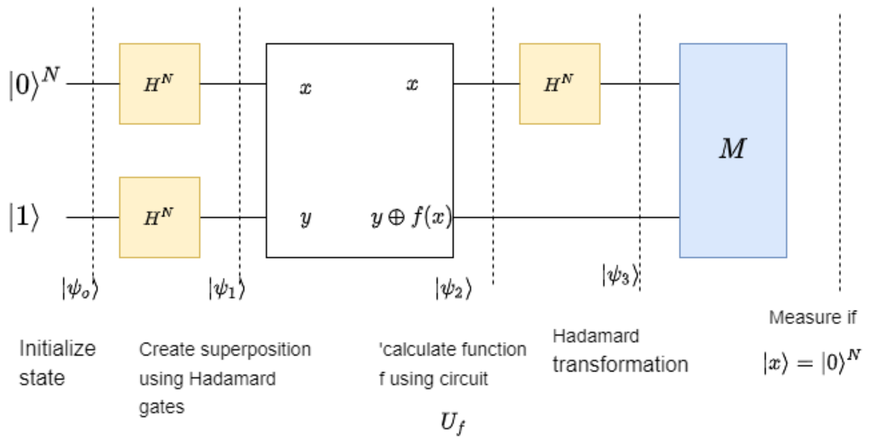


Figure 10: Deutsch Josza circuit

After measurement if $|x\rangle$ are 0 $\implies$ oracle is Constant;
If $|x\rangle$ are 1 $\implies$ oracle is Balanced
Quantum Algorithm evaluates in one go

Refer Appendix B to see how to implement Deutsch Josza Algorithm in QISKIT.

4 Grover's algorithm

Grover's algorithm is used for searching marked states from an unsorted database.

PROBLEM :Given a search space of size N, we want to find an element let say 'w' of that search space, given an oracle U_f with $f : (0,1)^n \to (0,1)$.

$$f(x) = \begin{cases} 1, & \text{if } x = w; \\ 0, & \text{else} \end{cases}$$

$$f_o(x) = \begin{cases} 0, & \text{if} x = 000...0 \\ 1, & \text{else} \end{cases}$$

$$Uf(x) = (-1)^{f(x)}|x\rangle \tag{10}$$

$$Uf|x\rangle \to \begin{cases} -|x\rangle; & \text{for} x = w \\ |x\rangle; & \forall x \neq w \end{cases}$$

$$U_f = I - 2|w\rangle\langle w| \tag{11}$$

$$U_{f_o}|o\rangle^{*n} \to \begin{cases} -|0\rangle^{*n}, & \text{for} x = w \\ |x\rangle, & \forall x \neq w \end{cases}$$

QUANTUM CIRCUIT:

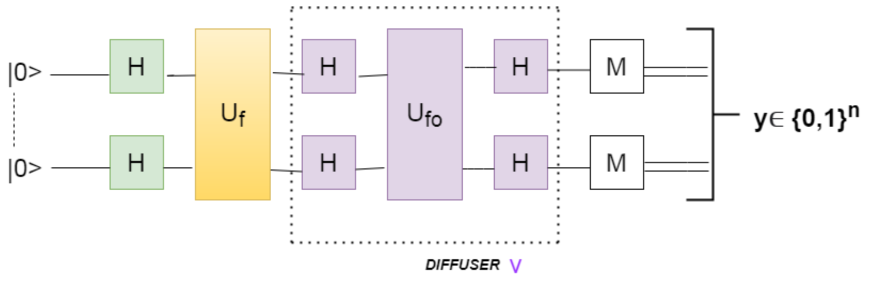


Figure 11: Grovers Algorithm Circuit

CLAIM: y=w with high probability

PROOF: let us define the uniform superposition state $|s\rangle$

$$|s\rangle = H^{*n}|0\rangle \tag{12}$$

and Diffuser V such that

$$\begin{aligned} V &= H^{*n}U_{fo}H^{*n} \\ &= H^{*n}2\langle 0|0\rangle^{*n}H^{*n} - H^{*n}IH^{*n} \\ &= 2\langle s|s\rangle - I \end{aligned} \tag{13}$$

V creates reflection at $|s\rangle$

Let $\sum$ be the plane spaced by $|s\rangle$ and $|w\rangle$ Let $|w^{`}\rangle$ be the state orthogonal to $|w\rangle$ in $\sum$: $|w\rangle = \frac{1}{\sqrt{2^n-1}}\sum_{x\neq w}|x\rangle$

$$\implies |s\rangle = \sqrt{\frac{2^n-1}{2^n}}|w^{`}\rangle + \frac{1}{\sqrt{2^n}}|w\rangle = \cos\frac{\theta}{2}|w^{'}\rangle + \sin\frac{\theta}{2}|w\rangle$$

where, $\theta = 2\sin^{-1}\left(\frac{1}{\sqrt{2^n}}\right)$

PROTOCOL

1. Prepare superposition state, $|s\rangle$
2. Apply $=I - 2|w\rangle\langle w| \rightarrow$ Reflection at $|w\rangle$
3. Apply $V = 2|s\rangle\langle s| - I \rightarrow$ Reflection at $|s>$

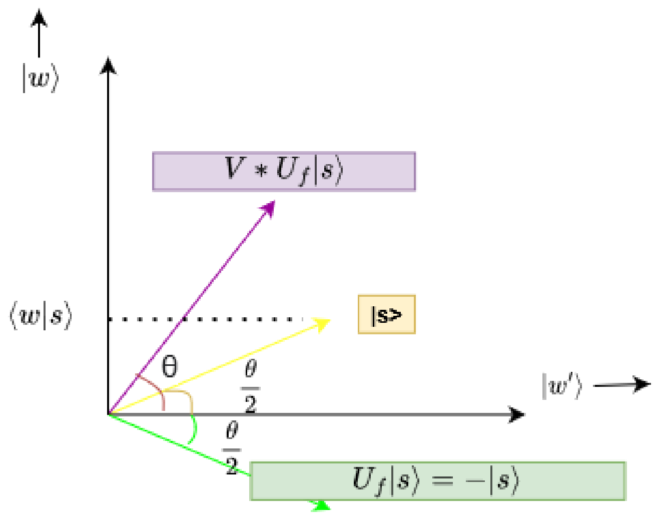


Figure 12: Grovers Algorithm Protocol

$V * U_f$ corresponds to a rotation by an angle θ.
After r application of step 2 and 3, the state is rotated by an angle $r \cdot \theta$.
Hence after r calls to the oracle, the final measure will result in state $|w\rangle$ with minimum probability, $p(w) \geq 1 - \sin^2 \frac{\theta}{2} = 1 - \frac{1}{2^n}$

Amplitude amplification

The general idea behind Grover's algorithm is amplitude amplification. Let us have a look at the amplitudes at each step in Grover's algorithm
1. Preparing superposition state $|s\rangle$
$|s\rangle = H^{*n}|0\rangle^{*n}$

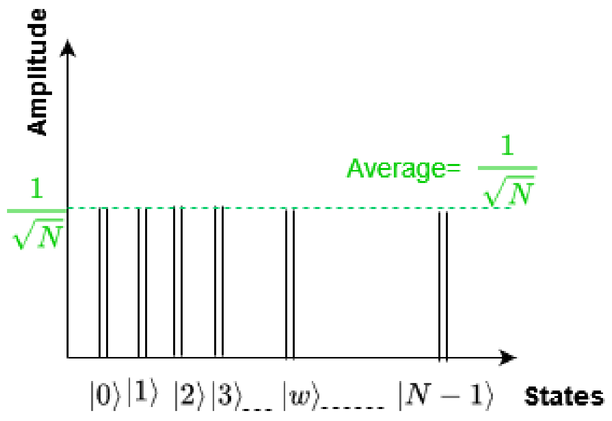


Figure 13: superposition states prepared

2. Applying Oracle to state $|s\rangle$
$|s\rangle = (I - 2|w\rangle\langle w|)|s\rangle$
This reflects the state $|w\rangle$ which is the required state.

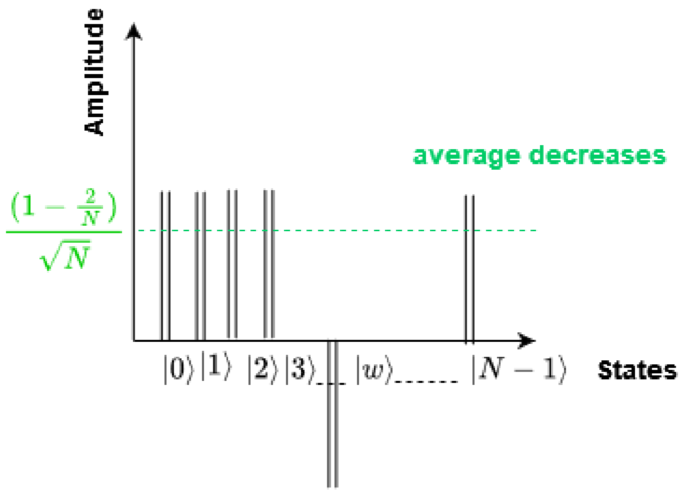


Figure 14: Reflection of $|w\rangle$

3. Applying Diffuser to the above state $|s\rangle$
$V|s\rangle = (2|s\rangle\langle s| - I)$

This reflects amplitude about the average amplitude

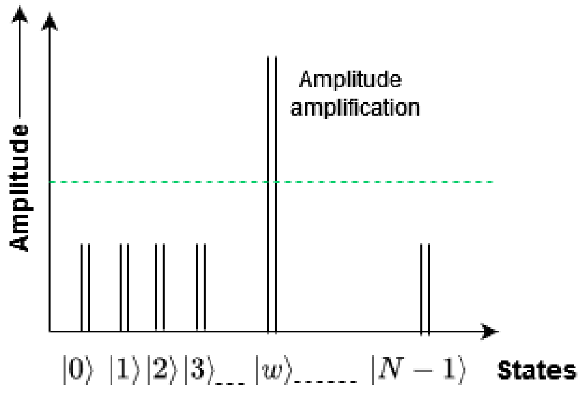


Figure 15: amplification about average amplitude

4. Repeating step 2 and 3 ,the amplitude of $|w\rangle$ will increase further
Hence in this way Amplitude amplification is achieved.

When we have more than one marked state.
Let say, we have M marked elements , we define the winning state as,

$$|w\rangle = \frac{1}{\sqrt{M}} \sum_{i=1}^{M} |w_i\rangle$$

and state orthogonal to $|w\rangle$ as,

$$|w'\rangle = \frac{1}{\sqrt{N-M}} \sum_{x \neq w_1, w_2 \dots w_m} |x\rangle$$

$$\implies |s\rangle = \frac{\sqrt{N-M}}{\sqrt{N}} |w'\rangle + \frac{\sqrt{M}}{\sqrt{N}} |w\rangle$$
$$= \cos\frac{\theta}{2} |w'\rangle + \sin\frac{\theta}{2} |w\rangle$$

$$\sin\frac{\theta}{2} = \frac{\sqrt{M}}{\sqrt{N}} \implies \text{Angle } \theta \text{ becomes larger}$$

$$r = \frac{\pi}{4\sin^{-1}}\left(\frac{\sqrt{N}}{\sqrt{M}}\right)$$

We can see this speed up also when looking at the amplitudes.

Refer Appendix C to see how to Implement Grover's Algorithm in QISKIT.

5 Quantum Fourier Transform

Quantum Fourier Transform is effectively a change of basis from the computational basis to the Fourier basis.
eg: 1 qubit computational basis states are $|0\rangle, |1\rangle$
Fourier basis for one qubit $|+\rangle, |-\rangle$
One qubit QFT that does this is Hadamard gate.

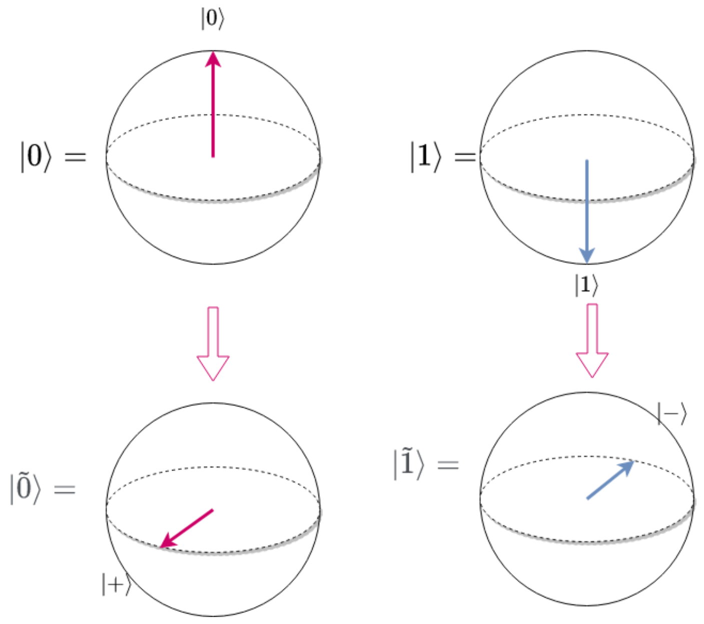


Figure 16: One qubit QFT

Building the quantum circuit that applies QFT

For n qubits we have 2^n basis states (let $N = 2^n$)

$$|\tilde{x}\rangle = QFT|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \exp\left(\frac{2\pi i x y}{2}\right)|y\rangle \tag{14}$$

For n=1 qubit case [N=2]

$$QFT|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{2-1} \exp\left(\frac{2\pi i x y}{2}\right)|y\rangle$$

$$QFT|x\rangle = \frac{1}{\sqrt{2}} \left(\exp\left(\frac{2\pi i x 0}{2}\right)|0\rangle + \exp\left(\frac{2\pi i x 1}{2}\right)|1\rangle\right)$$

$$QFT|x\rangle = |\tilde{x}\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \exp{(i\pi x)}|1\rangle\right)$$

For generalized case,

$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{\frac{2\pi ixy}{N}}|y\rangle \tag{15}$$

$$\text{where, } y = [y_1y_2y_3....y_n] = 2^{n-1}y_1 + 2^{n-2}y_2 + ...2^n y_n = \sum_{k=1}^{n} y_k 2^{n-k}$$

$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} \exp\left(2\pi i \sum_{k=1}^{n}\frac{y_k}{2^k}\right)|y_1y_2...y_n\rangle$$

$$= \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1}\prod_{k=1}^{n} \exp\left(\frac{2\pi ixy_k}{2^k}\right)|y_1y_2....y_n\rangle$$

$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}}(|0\rangle + e^{\frac{2\pi ix}{2^1}}|1\rangle)\otimes(|0\rangle + e^{\frac{2\pi ix}{2^2}}|1\rangle)\otimes(|0\rangle + e^{\frac{2\pi ix}{2^3}}|1\rangle)........\otimes(|0\rangle + e^{\frac{2\pi ix}{2^n}}|1\rangle) \tag{16}$$

e.g. n=3 qubits,$N = 2^3 = 8$

$$|x\rangle = 5 = |101\rangle$$

$$|\tilde{x}\rangle = |\tilde{5}\rangle = \frac{1}{\sqrt{8}}(|0\rangle + e^{\frac{2\pi i5}{2^1}}) \otimes (|0\rangle + e^{\frac{2\pi i5}{2^2}}) \otimes (|0\rangle + e^{\frac{2\pi i5}{2^3}})$$

Now we need to find a Quantum circuit that implements QFT.

$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}}(|0\rangle + e^{\frac{2\pi ix}{2^1}}|1\rangle)\otimes(|0\rangle + e^{\frac{2\pi ix}{2^2}}|1\rangle)\otimes(|0\rangle + e^{\frac{2\pi ix}{2^3}}|1\rangle)........\otimes(|0\rangle + e^{\frac{2\pi ix}{2^n}}|1\rangle)$$

Observations

Each qubit went from $|x_k\rangle \to |0\rangle + exp(\frac{2x}{2^k})|1\rangle$

$|\tilde{x}\rangle$ contains terms like $|000...0\rangle$ exp(0)

$$|000...01\rangle \to \exp(\tfrac{2\pi ix}{2^n})$$

$$|000...10\rangle \to \exp\left(\tfrac{2\pi ix}{2^{n-1}}\right)$$

This shows that Phase is qubit dependent

$$|111...11\rangle \to exp(2[\tfrac{x}{2^1} + \tfrac{x}{2^2} + ...\tfrac{x}{2^n}]$$

Need to add up more components with more 1' s

Ingredients to build a circuit

1. Hadamard gate

$$H|x_k\rangle = \begin{cases} \frac{(|0\rangle+|1\rangle)}{\sqrt{2}} \text{ for } x_k = 0 \\ \frac{(|0\rangle-|1\rangle)}{\sqrt{2}} \text{for } x_k = 1 \end{cases}$$
$$= \frac{(|0\rangle + exp(\frac{2x_k}{2})|1\rangle)}{\sqrt{2}}$$

2. Unitary rotation

$$UROT_k|x_j\rangle = \exp\left(\frac{2\pi i}{2^k}x_j\right)|x_j\rangle$$
$$= \begin{bmatrix} 1 & 0 \\ 0 & exp(\frac{2\pi i}{2^k}) \end{bmatrix}$$

$UROT_k|x_j\rangle$ applies phase $\exp\left(\frac{2\pi i}{2^k}\right)$ for state $|1\rangle$

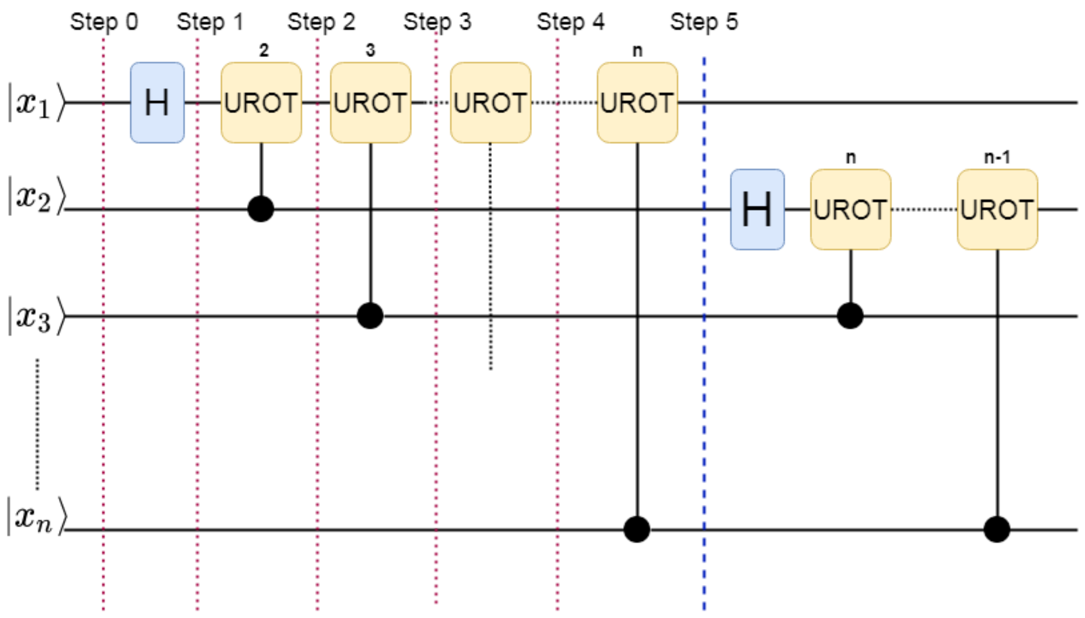


Figure 17: Quantum Fourier Transform

- Step 0: Initial state

$$|x_1\ x_2\ x_3\ ......x_n\rangle$$

- Step 1: After applying Hadamard to first qubit.

$$[|0\rangle + \exp\frac{2\pi i}{2}x_1|1\rangle] \otimes |x_2\ x_3\ x_4.......x_n\rangle$$

- Step 2: Controlled U rotation on first qubit with second qubit as target, we get

$$\left[|0\rangle + \exp\left(2\pi i\left(\frac{x_2}{2^2} + \frac{x_1}{2}\right)\right)|1\rangle\right] \otimes |x_2\ x_3\ x_4.......x_n\rangle$$

- Step 3: Controlled U rotation on first qubit with third qubit as control, we get

$$\left[|0\rangle + \exp\left(2\pi i\left(\frac{x_3}{2^3} + \frac{x_2}{2^2} + \frac{x_1}{2}\right)\right)|1\rangle\right] \otimes |x_2\ x_3\ x_4.......x_n\rangle$$

- Step 5: Similarly, controlled U rotation on first qubit after all qubits as control respectively we obtain,

$$\left[|0\rangle + \exp\left(2\pi i\left(\frac{x_n}{2^n} + \frac{x_{n-1}}{2^{n-1}}.... + \frac{x_1}{2}\right)\right)|1\rangle\right] \otimes |x_2\ x_3\ x_4.......x_n\rangle$$

Hence after repeating the same process on all qubits and applying controlled U rotation with controls as all greater qubit one by one we will obtain quantum Fourier transform.

$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}}(|0\rangle + e^{\frac{2\pi ix}{2^1}}|1\rangle) \otimes (|0\rangle + e^{\frac{2\pi ix}{2^2}}|1\rangle) \otimes (|0\rangle + e^{\frac{2\pi ix}{2^3}}|1\rangle)........ \otimes (|0\rangle + e^{\frac{2\pi ix}{2^n}}|1\rangle)$$

This circuit in Figure 17 implements QFT (except in reverse order of qubits at output)

6 Quantum Phase Estimation

PROBLEM: Remember that a unitary matrix has eigenvalues of the form $e^i\theta$ and that it has eigenvectors that form an orthogonal basis.

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

Can we extract θ_ψ given the ability to prepare $|\psi\rangle$ and the ability to apply U?
SOLUTION: Yes, we use QPE. Phase estimation allows us to convert phase estimation into amplitudes that we can measure.
More the number of qubits, more will be the precision.

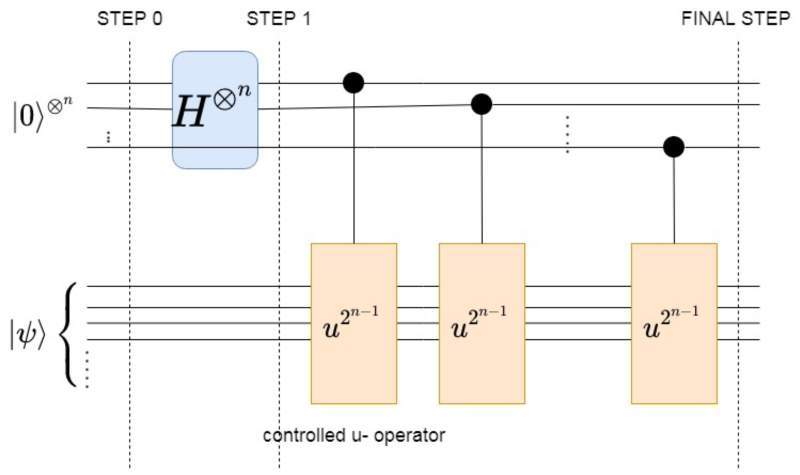


Figure 18: QPE trick using more qubits

- STEP 0: Preparing initial state.

$$|0\rangle^{\otimes n}|\psi\rangle$$

- STEP 1: After applying Hadamard to all qubits.

$$(\frac{1}{\sqrt{2}})^n(|0\rangle + |1\rangle)^{\otimes^n}|\psi\rangle$$

Using,

$$u^{2^x} = u^{2^{x-1}}u|\psi\rangle = u^{2^{x-1}}\exp(i\theta_\psi)|\psi\rangle = u^{2^{x-2}}\exp(2i\theta_\psi)|\psi\rangle = \exp(2xi\theta_\psi)|\psi\rangle$$

we get,

- FINAL STEP : After successive controlled U operations.

$$(\frac{1}{\sqrt{2}})^n[(|0\rangle+\exp(i\theta_\psi 2^{n-1})|1\rangle)\otimes(|0\rangle+\exp(i\theta_\psi 2^{n-2})|1\rangle).......\otimes(|0\rangle+\exp(i\theta_\psi 2^0)|1\rangle)] \tag{17}$$

Comparing Equation 17 with equation of QFT Equation 16

$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}}[(|0\rangle + \exp(\frac{2x}{2^1})|1\rangle) \otimes (|0\rangle + \exp(\frac{2x}{2^2})|1\rangle)....... \otimes (|0\rangle + \exp(\frac{2x}{2^n})|1\rangle)] \tag{18}$$

We can notice that QPE is same as QFT except $\theta_\psi \rightarrow (\frac{\theta_\psi}{2})2\pi$
If we apply QFT^{-1} at the end we can get our desired output.

QPE Protocol:

Given $u|\psi\rangle = e^{2\theta}$

QPE gives us $2^n\theta_\psi$, where n= number of qubits used to estimate θ_ψ

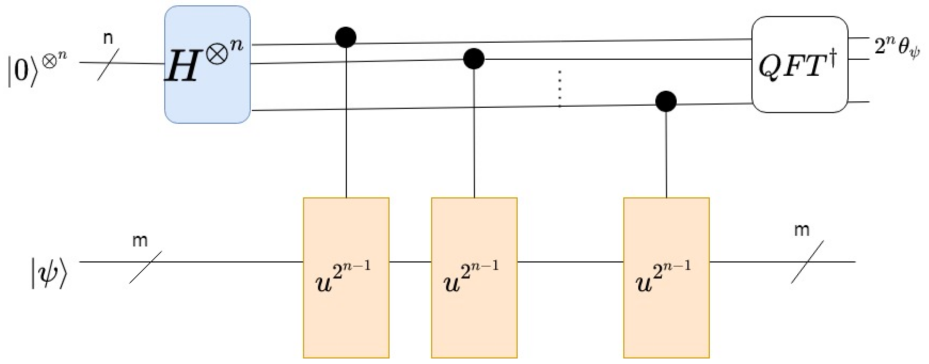


Figure 19: Quantum Phase Estimation

7 Classical Random Walk in 1-D

Consider a person walking on a 1 dimensional path with integral steps.

Let p be the probability that he takes a step to the right, i.e in the positive direction, and let $q = (1 - p)$ be the probability that he takes a step to the left, i.e in the negative direction.

Let n_l be the number of steps taken to the left and let n_r be the number of steps taken to the right. Then the position after N steps is given by the displacement, denoted by m, given by

$$m = n_r - n_l \tag{19}$$

We also have

$$N = n_r + n_l \tag{20}$$

The total number of configurations we have after N steps is 2^N.

The configuration of the state of the system is given by N and m, which we can use to calculate n_l and n_r. The probability of the man being at position m after N steps is the probability of obtaining the state (m,N), and it is given by

$$P(n_r, N) = \binom{N}{n_r} p^{n_r} q^{N-n_r} \tag{21}$$

where $n_r = \dfrac{N+m}{2}$

The expectation value of n_l and n_r is given by

$$\langle n_r \rangle = pN \tag{22}$$

$$\langle n_l \rangle = qN \tag{23}$$

The expectation value of m is given by,

$$\langle m\rangle = \langle n_r - n_l\rangle \tag{24}$$
$$= \langle n_r\rangle - \langle n_l\rangle \tag{25}$$
$$= N(p-q) \tag{26}$$

The uncertainty in n_r is Δn_r, which is given by,

$$\Delta n_r = \sqrt{\langle n_r^2\rangle - \langle n_r\rangle^2} \tag{27}$$

Now we need to compute $\langle n_r^2\rangle$. It is given by,

$$\langle n_r^2\rangle = \sum_{n_r=0}^{N} P(n_r, N) n_r^2 \tag{28}$$
$$= \sum_{n_r=0}^{N} \binom{N}{n_r} p^{n_r} q^{N-n_r} n_r^2 \tag{29}$$

Now, we write

$$n_r^2 p^{n_r} = \left(p\frac{\partial}{\partial p}\right)^2 (p^{n_r}) \tag{30}$$

$$\langle n_r^2\rangle = \sum_{n_r=0}^{N} \binom{N}{n_r} p^{n_r} q^{N-n_r} n_r^2 \tag{31}$$
$$= \sum_{n_r=0}^{N} \binom{N}{n_r} q^{N-n_r} \left(p\frac{\partial}{\partial p}\right)^2 (p^{n_r}) \tag{32}$$
$$= \left(p\frac{\partial}{\partial p}\right)^2 \sum_{n_r=0}^{N} \binom{N}{n_r} q^{N-n_r} (p^{n_r}) \tag{33}$$
$$\tag{34}$$

Recall the binomial expansion formula given by

$$(p+q)^N = \sum_{n=0}^{N} \binom{N}{n} p^n q^{N-n} \tag{35}$$

Thus we have,

$$\begin{aligned}
\langle n_r^2 \rangle &= \left(p\frac{\partial}{\partial p}\right)^2 (p+q)^N && (36)\\
&= \left(p\frac{\partial}{\partial p}\right) [pN(p+q)^{N-1}] && (37)\\
&= p[N(p+q)^{N-1} + pN(N-1)(p+q)^{N-2}] && (38)
\end{aligned}$$

Now, we put in $(p+q) = 1$, to get,

$$\begin{aligned}
\langle n_r^2 \rangle &= p[N + pN(N-1)] && (39)\\
&= pN[1 + pN - p] && (40)\\
&= pN[q + pN] && (41)\\
&= (pN)^2 + Npq && (42)
\end{aligned}$$

Therefore, the uncertainty Δn_r is given by,

$$\begin{aligned}
\Delta n_r &= \sqrt{\langle n_r^2 \rangle - \langle n_r \rangle^2} && (43)\\
&= \sqrt{(pN)^2 + Npq - (pN)^2} && (44)\\
&= \sqrt{Npq} && (45)\\
& && (46)
\end{aligned}$$

Similarly, $\Delta n_l = \sqrt{Npq}$

.

Now, we calculate the uncertainty in the displacement, m. We have,

$$\begin{aligned}
m &= n_r - n_l && (47)\\
&= 2n_r - N && (48)
\end{aligned}$$

where the second step is obtained using $n_l = \dfrac{N-m}{2}$.

We also have,

$$\langle m \rangle = 2\langle n_r \rangle - \langle N \rangle = 2\langle n_r \rangle - N \qquad (49)$$

$$\langle m^2 \rangle = 4\langle n_r^2 \rangle + N^2 - 4N\langle n_r \rangle \tag{50}$$

Then, we get,

$$\begin{align}
\Delta m &= \sqrt{\langle m^2 \rangle - \langle m \rangle^2} \tag{51}\\
&= \sqrt{4\langle n_r^2 \rangle + N^2 - 4N\langle n_r \rangle - 4\langle n_r \rangle^2 - N^2 + 4N\langle n_r \rangle} \tag{52}\\
&= 2\sqrt{\langle n_r^2 \rangle - \langle n_r \rangle^2} \tag{53}\\
&= 2\Delta n_r = 2\sqrt{Npq} \tag{54}
\end{align}$$

Now, in the special case, that $p = q = \frac{1}{2}$, we have,

$$P(n_r, N) = \binom{N}{n_r}\frac{1}{2^N} \tag{55}$$

$$\langle m \rangle = 0 \tag{56}$$

$$\Delta m = \sqrt{N} \tag{57}$$

Therefore, the the uncertainty in the displacement m goes as $\sqrt{N}$ in the special case, $p = q = \frac{1}{2}$.

7.1 Implementation of CRW in 1D

Refer Appendix D to see how Classical random walk code is implemented in python.

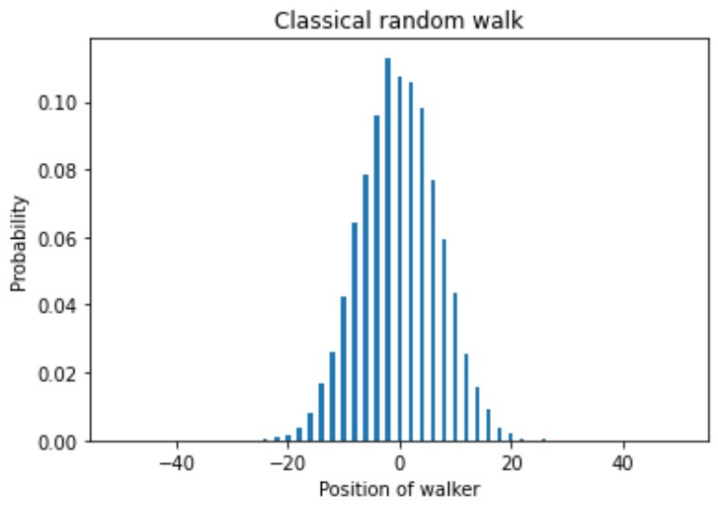


Figure 20: Classical random walk of a walker on a line with initial position at 0

Probability distribution plot in Figure 20 is obtained using inbuilt random function in python.

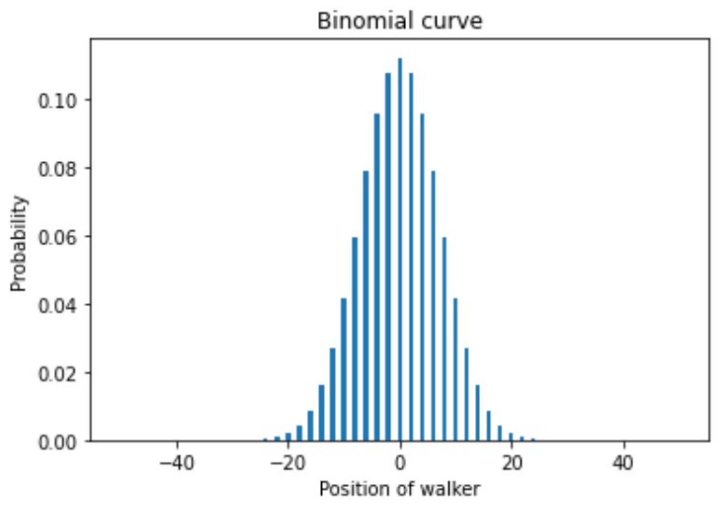


Figure 21: Binomial plot

Figure 21 is a binomial plot which is almost similar to Classical random walk probability distribution plot. Hence we verified experimentally which is exactly what the math predicts.

8 Quantum Random walk

8.1 Introduction

We'll start with an example to illustrate quantum random walks and give you a sense of what's to come. This example is based on the work of three physicists, Y. Aharonov, L. Davidovich, and N. Zagury, who published their findings in 1993. For the first time, their study coined the term "quantum random walk."[4]

Let us consider a particle on a line whose position is represented by a wave-packet $|\psi_{x0}\rangle$ localized around a position x_0, i.e. the function $\langle x|\psi_{x_0}\rangle$ corresponds to a wave-packet centralized around x_0. Let P be the momentum operator.

Translation operator,

$$U_l = \exp\left(\frac{-iPl}{\hbar}\right) \tag{58}$$

so that,

$$U_l|\psi_{x0}\rangle = |\psi_{x_0-l}\rangle \tag{59}$$

A spin $-\frac{1}{2}$ particle is usually described by a 2-vector, since $2 * s + 1 = \frac{1}{2}$ $|\psi\rangle = \left(|\tilde{\psi^{\uparrow}}\rangle, |\tilde{\psi^{\downarrow}}\rangle\right)^T$, where the first part corresponds the component of the wave-function of the particle in the spin- $|\uparrow\rangle$ space and the second one is the component in the $|\downarrow\rangle$-space.

Normalization condition gives, $\left(\||\tilde{\psi^{\uparrow}}\rangle\|^2 + \||\tilde{\psi^{\downarrow}}\rangle\|^2\right) = 1$

Total Wavefunction of the particle (spin and spatial part)

$$|\psi\rangle = \alpha^{\uparrow}|\uparrow\rangle \otimes |\psi^{\uparrow}\rangle + \alpha^{\downarrow}|\downarrow\rangle \otimes |\psi^{\downarrow}\rangle$$

(60)

The time development corresponding to a translation by l on the larger state space of the spin- $\frac{1}{2}$ particle can now be described by the unitary operator

$$U = \exp\left(-2iS_z \otimes \hat{P}l\right)$$

When initial state of the particle is in superposition

$$|\psi_{in}\rangle = \big(\alpha^{\uparrow}|\uparrow\rangle + (\alpha^{\downarrow}|\downarrow\rangle\big) \tag{61}$$

The translation operator's application U will cause a position superposition.

$$U|\psi_{in}\rangle = \alpha^{\uparrow}|\uparrow\rangle \otimes |\psi_{x_0-l}\rangle + \alpha^{\downarrow}|\downarrow\rangle \otimes |\psi_{x_0+l}\rangle \tag{62}$$

$\implies U|\psi_{in}\rangle$ corresponds to Random walk of particle on the line
If at this point if we measure the spin of the particle in the S_z basis, the particle will be either in the state $|\uparrow\rangle \otimes |\psi_{x_0 l}\rangle$, localized around $x_0 + l$ with probability $p^{\uparrow} = |\alpha^{\uparrow}|^2$ or in the state $|\downarrow\rangle \otimes |\psi_{x_0 l}\rangle$ localized around $x_0 l$ with probability $p^{\downarrow} = |\alpha^{\downarrow}|^2$
This procedure corresponds to a (biased) random walk of a particle on the line

We will now measure the spin in some rotated basis, supplied by two orthogonal vectors $|s+\rangle, |s\rangle$, rather than in the eigenbasis of S_z.Alternatively, we can rotate the spin by some angle θ first before measuring it in S_z eigenbasis .
Rotation of spin can be represented as

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{63}$$

Let us set up some more of the language used in quantum information theory.

$$|\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |\downarrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{64}$$

$$\begin{aligned} S_z &= \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ &= \frac{1}{2}(|\uparrow\rangle\langle\uparrow| - |\downarrow\rangle\langle\downarrow|) \end{aligned} \tag{65}$$

Let M_z denotes the measurement in the S_z basis.
Now we have to slightly rewrite the operator U to see the effect of operations $M_z R(\theta) U$ on the initial state $|\psi_{in}\rangle$.

$$\begin{aligned} U &= \exp\left(-2iS_z \otimes Pl\right) \\ &= \exp\left(-i(|\uparrow\rangle\langle\uparrow| - |\downarrow\rangle\langle\downarrow|) \otimes Pl\right. \\ &= (|\uparrow\rangle\langle\uparrow| \otimes \exp(iPl))(|\downarrow\rangle\langle\downarrow| \otimes \exp(iPl)). \end{aligned} \tag{66}$$

$$U|\psi_{in}\rangle = (\alpha\uparrow|\uparrow\rangle \otimes \exp -iPl + \alpha\downarrow|\downarrow\rangle \otimes \exp iPl)|\psi_{x_0}$$

Applying Rotation

$$\begin{aligned} R(\theta)U|\psi_{in}\rangle &= [(\alpha^{\uparrow}\cos\theta\exp -iPl - \alpha^{\downarrow}\sin\theta\exp iPl)|\uparrow\rangle \\ &\quad + (\alpha^{\uparrow}\sin\theta\exp -iPl + \alpha^{\downarrow}\cos\theta\exp iPl)|\downarrow\rangle] \otimes |\psi_{x_0\rangle} \end{aligned} \tag{67}$$

If width of $|\psi_{in}\rangle$ is s.t., $\Delta x >> l$
$\exp(\pm iPl)|\psi_{x_0} \approx (I \pm iPl)|\psi_{x_0}$

$$\begin{aligned} R(\theta)U|\psi_{in}\rangle &= [(\alpha^{\uparrow}\cos\theta(I - iPl) - \alpha^{\downarrow}\sin\theta(I + iPl))|\uparrow\rangle \\ &\quad + (\alpha^{\uparrow}\sin\theta(I - iPl) + \alpha^{\downarrow}\cos\theta(I + iPl))|\downarrow\rangle] \otimes |\psi_{x_0\rangle} \\ &= [[(\alpha^{\uparrow}\cos\theta - \alpha^{\downarrow}\sin\theta)I - (\alpha^{\uparrow}\cos\theta + \alpha^{\downarrow}\sin\theta)iPl]|\uparrow\rangle \\ &\quad + [(\alpha^{\uparrow}\sin\theta + \alpha^{\downarrow}\cos\theta)I - (\alpha^{\uparrow}\sin\theta - \alpha^{\downarrow}\cos\theta)iPl]|\downarrow\rangle] \otimes |\psi_{x_0\rangle} \end{aligned}$$

$$\implies M_z R(\theta)U|\psi_{in}\rangle = \begin{cases} |\uparrow\rangle \otimes (I - iPl\delta^{\uparrow})|\psi_{x_0} \\ |\downarrow\rangle \otimes (I - iPl\delta^{\downarrow})|\psi_{x_0} \end{cases} \tag{68}$$

where, displacements

$$l\delta^{\uparrow} = l\left(\frac{\alpha^{\uparrow}\cos\theta - \alpha^{\downarrow}\sin\theta}{\alpha^{\uparrow}\cos\theta - \alpha^{\downarrow}\sin\theta}\right)$$

$$l\delta^{\downarrow} = l\left(\frac{\alpha^{\uparrow}\sin\theta - \alpha^{\downarrow}\cos\theta}{\alpha^{\uparrow}\sin\theta - \alpha^{\downarrow}\cos\theta}\right)$$

And Probabilities

$$P^{\uparrow} = \|(\alpha^{\uparrow}\cos\theta - \alpha^{\downarrow}\sin\theta)\|^2$$

$$P^{\downarrow} = \|(\alpha^{\uparrow}\sin\theta - \alpha^{\downarrow}\cos\theta)\|^2$$

Displacement of the particle in one of the two cases, can be made much larger than l. . Let us take case of $l\delta^{\uparrow}$ We may choose, $\tan(\theta) = \|\frac{\alpha^{\uparrow}}{\alpha^{\downarrow}}\|(1+\epsilon)$

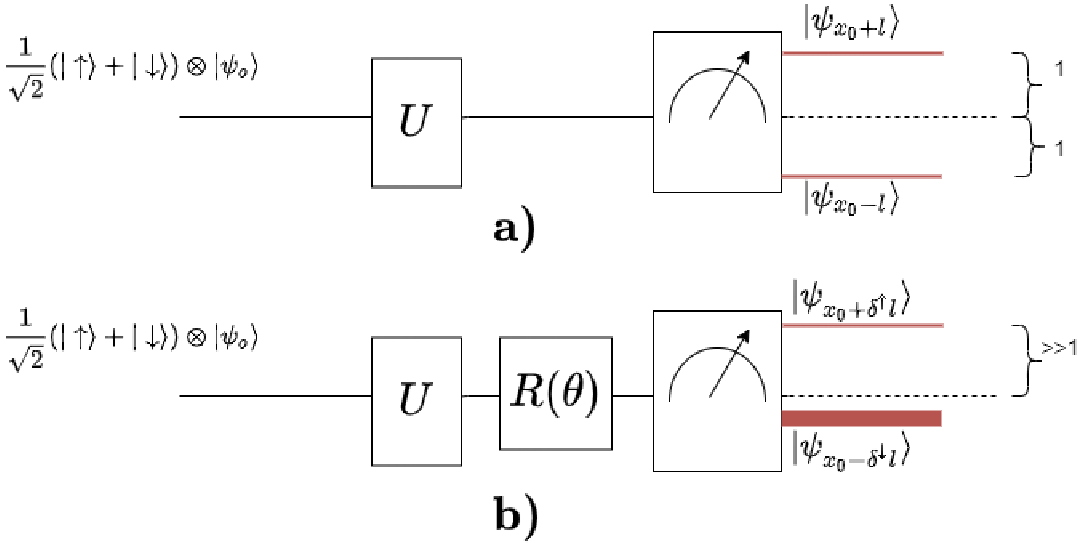
$\therefore l\delta^{\uparrow} = l\left(\frac{1+\frac{\alpha^{\uparrow}}{\alpha^{\downarrow}}\tan(\theta)}{1-\frac{\alpha^{\uparrow}}{\alpha^{\downarrow}}\tan(\theta)}\right)$

$\implies l\delta^{\uparrow} \approx \frac{-2l}{\epsilon}$

If $\Delta x >> l\delta^{\uparrow}$ and $l\delta^{\downarrow}$, again using approximation

$$\begin{aligned}(I - iPl\delta^{\uparrow,\downarrow})|\psi_{x_0}\rangle &\approx \exp(-iPl\delta^{\uparrow,\downarrow})|\psi_{x_0}\rangle \\ &= |\psi_{x_0 - l\delta^{\uparrow,\downarrow}}\rangle\end{aligned}$$

$\implies$ This concludes that even if translation operator displaces by l , in very rare cases the particle will jump much further than l



$\frac{1}{\sqrt{2}}(|\uparrow\rangle + |\downarrow\rangle) \otimes |\psi_o\rangle$
U
$|\psi_{x_0+l}\rangle$
1
1
$|\psi_{x_0-l}\rangle$
a)
$\frac{1}{\sqrt{2}}(|\uparrow\rangle + |\downarrow\rangle) \otimes |\psi_o\rangle$
U
$R(\theta)$
$|\psi_{x_0+\delta^{\uparrow}l}\rangle$
>>1
$|\psi_{x_0-\delta^{\downarrow}l}\rangle$
b)

Figure 22: **a)**The particle enters the conditional displacement box U after being localised in point x_0. It is displaced up or down by l based on its spin degree of freedom. When it exits its spin is measured in the z basis . (M_Z). With probability $\frac{1}{2}$ it is found displaced by l resp. l.
b) The identical setup as before, only the spin is rotated once the particle exits the U-box. The particle might now be displaced up by significantly more than l after the spin measurement M_z. (with small probability). It's shifted by less than l in the opposite direction (with large probability).

8.2 Coined Quantum Random walk

Let H_P be the position Hilbert space and H_P is augmented by a Coin space H_c spanned by two basis states $|\uparrow\rangle, |\downarrow\rangle$, which, in the preceding section, played the role of the spin- $\frac{1}{2}$ space.

States of the total system are in the space $H = H_C \otimes H_P$ as before.

The following unitary operation can be used to represent the system's conditional translation (the same function which $\exp(2iS_z \otimes Pl)$ was playing in previous section).

$$S = |\uparrow\rangle\langle\downarrow| \otimes \sum_i |i+1\rangle\langle i| + |\downarrow\rangle\langle\downarrow| \otimes \sum_i |i-1\rangle\langle i| \tag{69}$$

where the index i runs over $\mathbb{Z}$ in the case of a line.

$$\begin{aligned} S|\uparrow\rangle \otimes |i\rangle &= |\uparrow\rangle\langle\downarrow| \otimes \sum_i |i+1\rangle\langle i| + |\downarrow\rangle\langle\downarrow| \otimes \sum_i |i-1\rangle\langle i||\uparrow\rangle \otimes |i\rangle \\ &= |\uparrow\rangle \otimes |i+1\rangle \\ S|\downarrow\rangle \otimes |i\rangle &= |\uparrow\rangle\langle\downarrow| \otimes \sum_i |i+1\rangle\langle i| + |\downarrow\rangle\langle\downarrow| \otimes \sum_i |i-1\rangle\langle i||\downarrow\rangle \otimes |i\rangle \\ &= |\downarrow\rangle \otimes |i1\rangle \end{aligned}$$

$\implies$ transforms the basis state $|\uparrow\rangle \otimes |i\rangle$ to $|\uparrow\rangle \otimes |i+1\rangle$ and $|\downarrow\rangle \otimes |i\rangle$ to $|\downarrow\rangle \otimes |i1\rangle$.

The random walk's first step is a rotation in coin-space, which we'll call 'coin-flip' C. The unitary transformation C is extremely arbitrary, and altering it allows us to design a large family of walks with a wide range of behaviours.

IF we want unbiased C operation, i.e. to get equal probability of moving rightward or leftward after the coin-flip C followed by translation operation S, we can use most common Hadamard coin

The Hadamard coin H is a common balanced unitary coin.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{70}$$

$$\begin{aligned} |\uparrow\rangle \otimes |0\rangle &\xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \\ &\xrightarrow{S} \frac{1}{\sqrt{2}}(|\uparrow\rangle \otimes |1\rangle + |\downarrow\rangle \otimes |-1\rangle) \end{aligned} \tag{71}$$

The quantum random walk of T steps is defined as the transformation U^T , where U, acting on $H = H_C \otimes H_P$ is given by

$$U = S\dot{(}C \otimes I). \tag{72}$$

Let us evolve the quantum walk (without intermediate measurements) for some steps starting in the initial state $|\phi_{in}\rangle = |\downarrow\rangle \otimes |0\rangle$ and examine the induced probability distribution on the positions to show how the quantum walk differs from its classical counterpart.

$$\begin{aligned} |\phi_{in}\rangle &\xrightarrow{U} \frac{1}{\sqrt{2}}(|\uparrow\rangle \otimes |1\rangle - |\downarrow\rangle \otimes |-1\rangle) \\ &\xrightarrow{U} \frac{1}{\sqrt{2}}(|\uparrow\rangle \otimes |2\rangle - (|\uparrow\rangle - |\downarrow\rangle) \otimes |0\rangle + |\downarrow\rangle \otimes |-2\rangle \\ &\xrightarrow{U} \frac{1}{2\sqrt{2}}(|\uparrow\rangle \otimes |3\rangle + |\downarrow\rangle \otimes |1\rangle + |\uparrow\rangle \otimes |-1\rangle - 2|\downarrow\rangle \otimes |-1\rangle - |\downarrow\rangle \otimes |-3\rangle \\ &\xrightarrow{U} \frac{1}{2\sqrt{2}}(|\uparrow\rangle \otimes |4\rangle + (|\uparrow\rangle + \downarrow\rangle) \otimes |2\rangle - (3|\uparrow\rangle + |\downarrow\rangle) \otimes |0\rangle + (|\downarrow\rangle - |\uparrow\rangle) \otimes |2\rangle \\ &\qquad + |\downarrow\rangle \otimes |-4\rangle) \end{aligned} \tag{73}$$

1. Measuring the coin state in the standard basis gives each of $|\uparrow\rangle \otimes |1\rangle, |\downarrow\rangle \otimes |1\rangle$ with probability $\frac{1}{2}$. After this measurement there is no correlation between the positions left and the state would collapse to one of its eigenstate . We get the plain classical random walk on the line if we continue the quantum walk with such a measurement at each iteration.

2. We won't measure the coin register during intermediate iterations in the quantum random walk, but we will maintain the quantum correlations between distinct positions and let them interfere in following stages.

T \ i	-5	-4	-3	-2	-1	0	1	2	3	4	5
0						1					
1					$\frac{1}{2}$		$\frac{1}{2}$				
2				$\frac{1}{4}$		$\frac{1}{2}$		$\frac{1}{4}$			
3			$\frac{1}{8}$		$\frac{3}{8}$		$\frac{3}{8}$		$\frac{1}{8}$		
4		$\frac{1}{16}$		$\frac{1}{4}$		$\frac{3}{8}$		$\frac{1}{4}$		$\frac{1}{16}$	
5	$\frac{1}{32}$		$\frac{5}{32}$		$\frac{5}{16}$		$\frac{5}{16}$		$\frac{5}{32}$		$\frac{1}{32}$

Figure 23: After T steps of classical random walk, the probability of particle being at position i

T \ i	-5	-4	-3	-2	-1	0	1	2	3	4	5
0						1					
1					$\frac{1}{2}$		$\frac{1}{2}$				
2				$\frac{1}{4}$		$\frac{1}{2}$	0	$\frac{1}{4}$			
3			$\frac{1}{8}$		$\frac{5}{8}$		$\frac{1}{8}$		$\frac{1}{8}$		
4		$\frac{1}{16}$		$\frac{5}{8}$		$\frac{1}{8}$		$\frac{1}{8}$		$\frac{1}{16}$	
5	$\frac{1}{32}$		$\frac{17}{32}$		$\frac{1}{8}$		$\frac{1}{8}$		$\frac{5}{32}$		$\frac{1}{32}$

Figure 24: : The probability of being found at position i after T steps of the quantum random walk on the line, with the initial state $|\phi_{in}\rangle = |\downarrow\rangle \otimes |0\rangle$. As you can see that this distribution starts to differ from the classical distribution from T = 3 onwards . Furthermore the quantum random walk is asymmetric with a drift to the left which was also shown by Equation 73

.

This example shows that the probability distribution induced by the quantum walk differs from the classical one.

The quantum random walk create an asymmetric probability distribution, which is 'drifting' to the left. The Hadamard coin treats the two orientations $|\uparrow\rangle$ and $|\downarrow\rangle$ differently, resulting in the asymmetry ; it multiplies the phase by 1 only in the case of $|\downarrow\rangle$. Implicitly, this results in more cancellations for right-to-left routes (destructive interference), whereas particles moving left to right interfere constructively.

8.2.1 Implementation on IBM computer

Plotting the Graph of the result of our simulation with final position vector value on the x-axis and the number of occurrences of that position vector value on the y-axis. This provides us the probability distribution of the position of the walker.
Initially let our walker be at position $|1\rangle$.

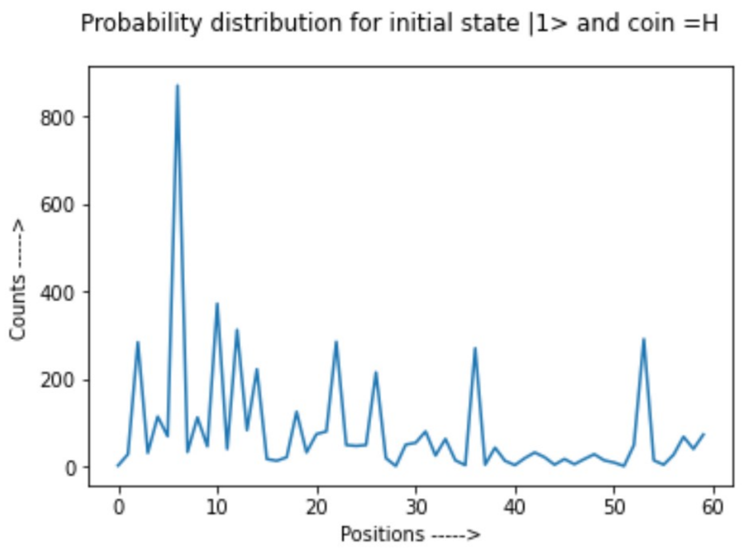


Figure 25: Probability distribution of walker with initial state of coin vector $|1\rangle$ and using H as coin operator.

Now, Changing the initial state of coin vector to $|0\rangle$ i.e. $|\uparrow\rangle$.

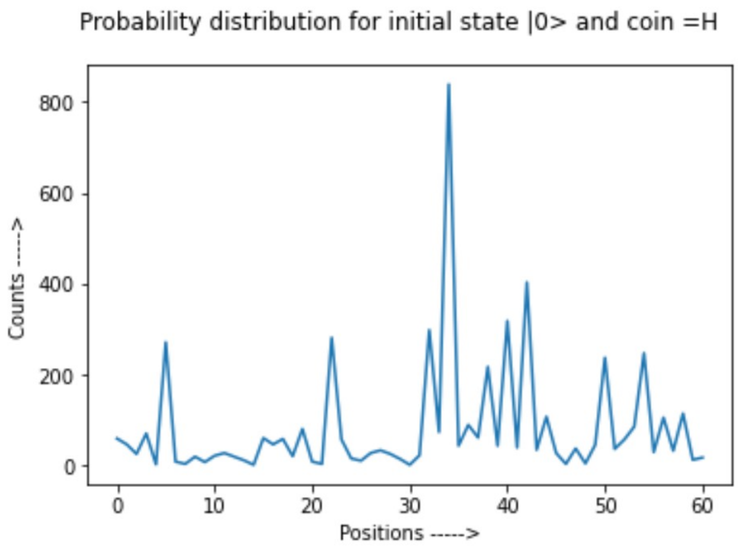


Figure 26: Probability distribution of walker with initial state of coin vector 1 and using H as coin operator.

Initialize the coin qubit in a balanced state $|i\rangle$, where $|i\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$.

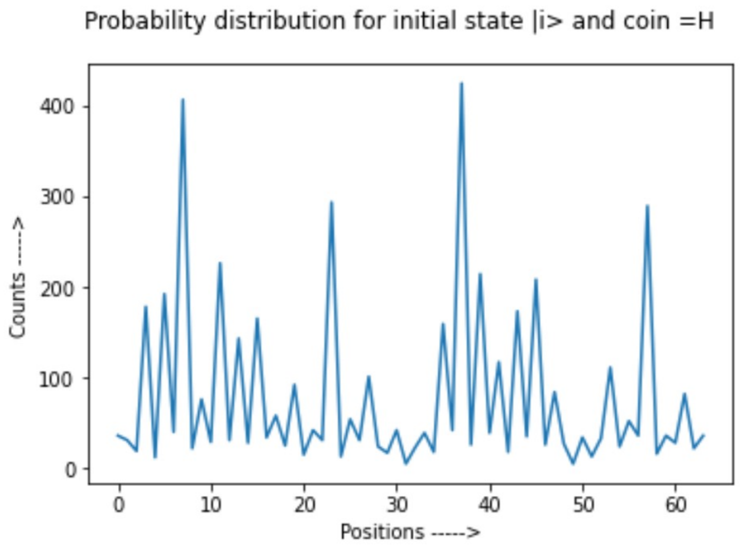


Figure 27: Probability distribution of walker with initial state of coin vector as $|i\rangle$ i.e. balanced state

Refer Appendix Appendix E for Python code of Coined Quantum Walk in QISKIT.

Conclusion

- This strange probability distribution is due to the repeated application of Hadamard gate on the coin qubit as H gate treats the the two states differently and applies a phase to $|\downarrow\rangle$ state. for figure Figure 25 interference causes our coin qubit to drift towards left but it also creates a large bump on the extreme right side which can be verified by the mathematics which we did in above section and from table Figure 23

- Mirrored probability distribution must be obtained when the coin vector is in opposite state i.e. $|\uparrow\rangle$ which is verified by figure Figure 26

- In figure Figure 27 we obtained a symmetric probability distribution as our coin state was balanced and thus interference over time does not lead to a drift in particular side

8.3 Staggered Quantum Walk

We construct this walk based on a graph tessellation, that is a set of disjoint cliques over all vertices; and a graph covering, that is a family of tessellations, where every edge belongs to, at least, one tessellation.[6]

Preliminaries

Clique

a clique of a graph G is an induced sub-graph of G that is complete .In a partition of graph into cliques, each element of the partition is a clique, and no two elements of the partition can share a vertex as shown in figure Figure 28. And every vertex of a clique should be connected.

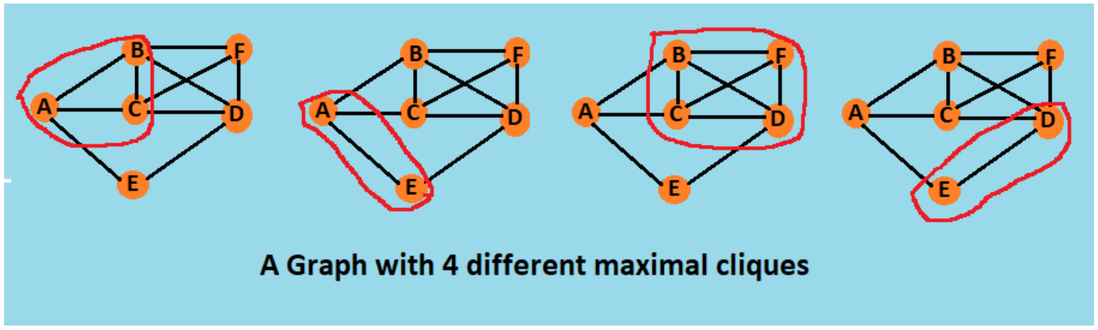


Figure 28: All possible cliques of given graph G

Tessellation

A tessellation or tiling is the covering of a surface, often a plane, using one or more geometric shapes, called tiles, with no overlaps and no gaps. In graph theory, a tessellation T is a partition of the graph into cliques, where each clique is called a polygon (or a cell), such that the union of the polygons covers the vertex set of graph. An edge belongs to the tessellation if and only if both endpoints of the edge belong to the same polygon. The set of edges belonging to T is denoted by E(T).

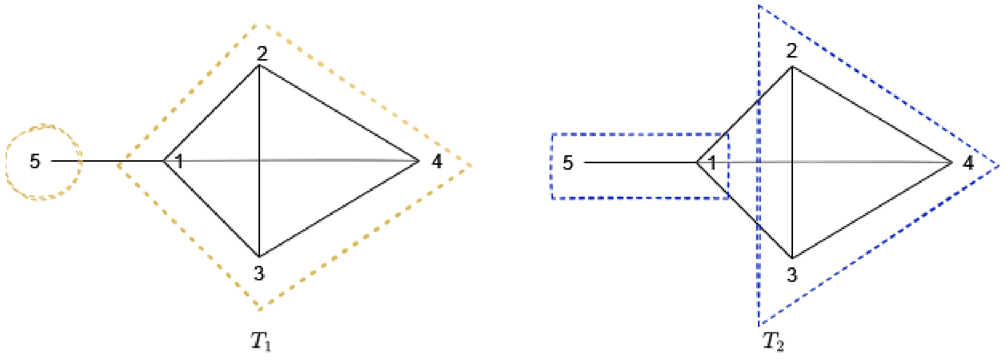


Figure 29: Possible tessellations of given graph G

8.3.1 SQW on 1D lattice

Considering an infinite line, we can tessellate this graph as,

$$\begin{aligned} T_\alpha &= \{\{2x, 2x+1\} : \ x \in \mathbb{Z}\} \\ T_\beta &= \{\{2x+1, 2x+2\} : \ x \in \mathbb{Z}\} \end{aligned} \tag{74}$$

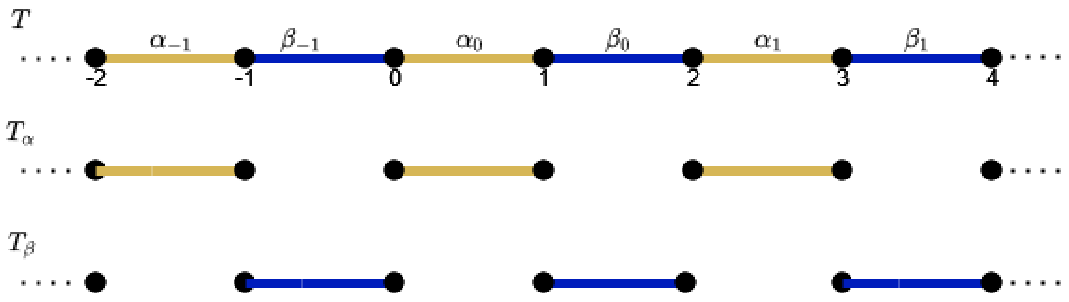


Figure 30: Caption

We may define state α_x associated to tessellation T_α and state β_x associated to tessellation T_β such as

$$\begin{aligned} |\alpha_x\rangle &= \frac{|2x\rangle + |2x+1\rangle}{\sqrt{2}} \\ |\beta_x\rangle &= \frac{|2x+1\rangle + |2x+2\rangle}{\sqrt{2}} \end{aligned} \tag{75}$$

and operators associated to each tesselation as

$$\begin{aligned} |H_\alpha\rangle &= 2\sum_{x=-\infty}^{+\infty} |\alpha_x\rangle\langle\alpha_x| - \mathbb{I} \\ |H_\beta\rangle &= 2\sum_{x=-\infty}^{+\infty} |\beta_x\rangle\langle\beta_x| - \mathbb{I} \end{aligned} \tag{76}$$

[1]Let us take a case of N-cycle whose vertices v, are labelled by 0,1,...N-1 and assume N to be even. Each vertex is associated with a basis vector $|v\rangle$ with a computational basis$\{|x\rangle : x = 0, 1, 2...N-1\}$ in a Hilbert space $\mathcal{H}^N$. we can tessellate this graph as,

$$\begin{aligned} T_\alpha &= \{\{2x, 2x+1\} : \ 0 \leq x \leq N/2 - 1\} \\ T_\beta &= \{\{2x+1, 2x+2\} : \ 0 \leq x \leq N/2 - 1\} \end{aligned} \tag{77}$$

Working with n=2 qubit case, i.e. $N = 2^n = 4$.

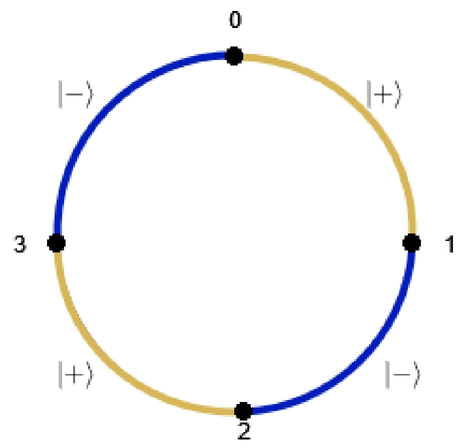


Figure 31: cycle with N=4 vertices

$$\begin{aligned}
|\alpha_x\rangle &= \frac{|2x\rangle + |2x+1\rangle}{\sqrt{2}} \\
|\alpha_0\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\\0\\0\end{pmatrix} \\
|\alpha_1\rangle &= \frac{|2\rangle + |3\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\0\\1\\1\end{pmatrix}
\end{aligned} \tag{78}$$

$$\begin{aligned}
|\beta_x\rangle &= \frac{|2x+1\rangle + |2x+2\rangle}{\sqrt{2}} \\
|\beta_0\rangle &= \frac{|1\rangle + |2\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\1\\0\end{pmatrix} \\
|\beta_1\rangle &= \frac{|3\rangle + |0\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}
\end{aligned} \tag{79}$$

Finding Hamiltonian operator for tessellation T_α

$$\begin{aligned}
|H_\alpha\rangle &= 2\sum_{x=0}^{\frac{N}{2}-1} |\alpha_x\rangle\langle\alpha_x| - \mathbb{I} \\
|H_\alpha\rangle &= 2\sum_{x=0}^{1} |\alpha_x\rangle\langle\alpha_x| - \mathbb{I} \\
&= 2(|\alpha_0\rangle\langle\alpha_0||\alpha_1\rangle\langle\alpha_1|) - \mathbb{I} \\
&= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
\implies |H_\alpha\rangle &= \mathbb{I} \otimes \mathbb{X}
\end{aligned} \tag{80}$$

Finding Hamiltonian operator for tessellation T_β

$$\begin{aligned}
|H_\beta\rangle &= 2\sum_{x=0}^{\frac{N}{2}-1} |\beta_x\rangle\langle\beta_x| - \mathbb{I} \\
|H_\beta\rangle &= 2\sum_{x=0}^{1} |\beta_x\rangle\langle\beta_x| - \mathbb{I} \\
&= 2(|\beta_0\rangle\langle\beta_0||\beta_1\rangle\langle\beta_1|) - \mathbb{I} \\
&= \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \\
\implies |H_\beta\rangle &= \begin{pmatrix} 0 & & & 1 \\ & 0 & 1 & \\ & 1 & 0 & \\ 1 & & & 0 \end{pmatrix}
\end{aligned} \tag{81}$$

Operators H_α and H_β in matrix form are given by,

$$|H_\alpha\rangle = \mathbb{I} \otimes \mathbb{X}$$

$$|H_\beta\rangle = \begin{pmatrix} 0 & & 1 \\ & \mathbb{I} \otimes \mathbb{X} & \\ 1 & & 0 \end{pmatrix}$$

Here, x=$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and empty entries are 0.
Evolution operator for SQW is given by

$$U = \exp\left(i\theta H_\alpha\right)\exp\left(i\theta H_\beta\right)$$

$$U_\alpha = \exp\left(\mathbb{I} \otimes X\right)$$

$$U_\alpha = \mathbb{I} \otimes R_x(2\theta) \tag{82}$$

and

$$U_\beta = \begin{pmatrix} \cos(\theta) & & -i\sin(\theta) \\ & \mathbb{I} \otimes \mathbb{X} & \\ -i\sin(\theta) & & \cos(\theta) \end{pmatrix} \tag{83}$$

Now we can decompose U_β into U_α via a similarity transpose $U_\beta = P^{-1}U_\alpha P$ where P is a permutation matrix which shifts the walker to the right and P^{-1} shifts to the left.

$$P = \sum_x |x+1\rangle\langle x| = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

So the Staggered Quantum walk evolution operator can thus be written as,

$$U = U_\beta U_\alpha = P^{-1}U_\alpha P U_\alpha \tag{84}$$

As we can see P is clearly an increment operator and P^{-1} is an decrement operator.

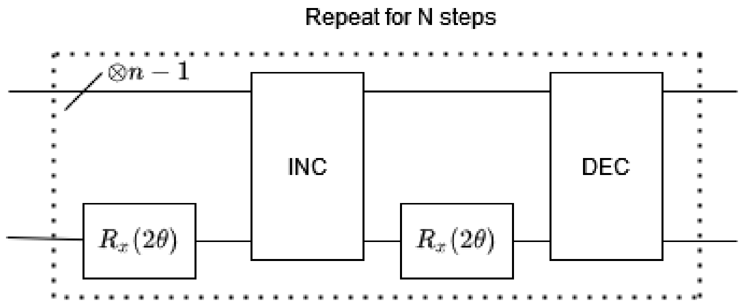


Figure 32: General circuit for Staggered Quantum Walk

8.3.2 Implementation on IBM Computers

Figure 33 refers to the 1D SQW circuit for one walk step on a lattice with 16 vertices using 4 qubits which is to be implemented using QISKIT.

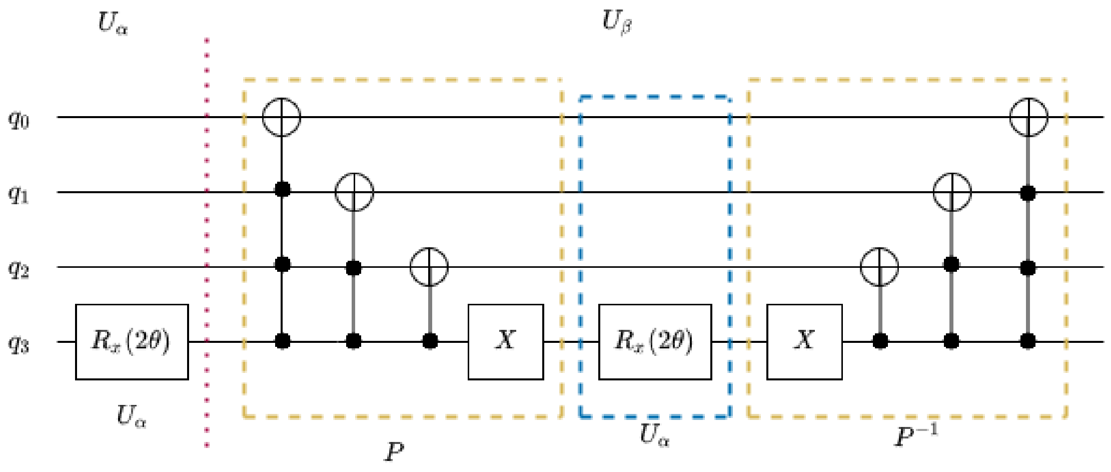


Figure 33: One walk step of SQW in 1D lattice using 4 qubits

Refer Appendix F for Python code of 1D SQW in QISKIT. Figure 34 and Figure 35 refers to Probability distribution plots of 1D SQW employing 7 qubits obtained using QISKIT. In fig:Figure 34 the action of evolution operator spreads the position among vertices 0,32,64 and 127.

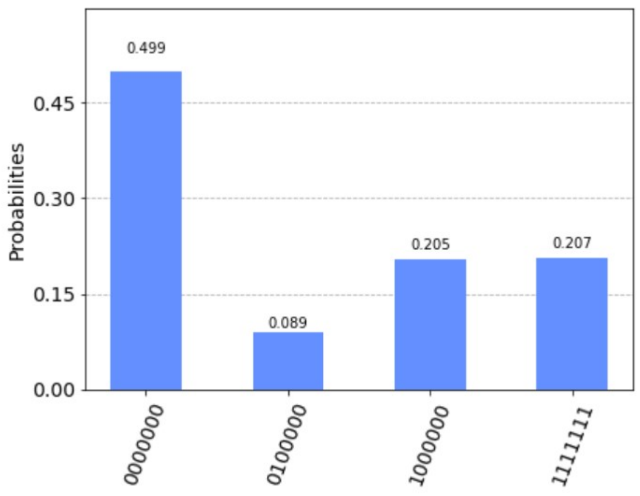


Figure 34: Probability distribution after **one** step of SQW using 7 qubits

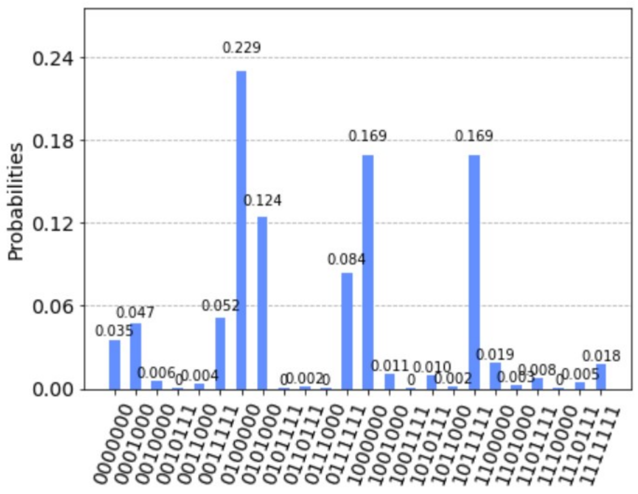


Figure 35: Probability distribution after **ten** step of SQW using 7 qubits

8.3.3 Quantum Walk on 2-D square lattice

Let us consider a 2D sqaure lattice with cyclic boundary conditions having $\sqrt{N} \times \sqrt{N}$ vertices which are labelled as 0,1,...N-1 where N is a square integer.

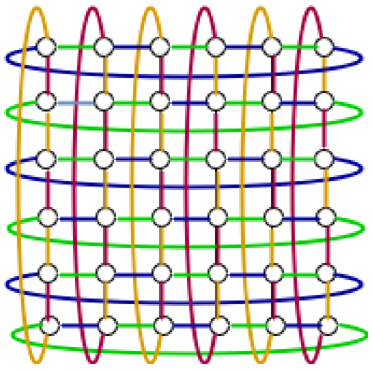


Figure 36: One walk step of SQW

A total of four tessellations are required to cover all edges of the graph and to define evolution operator of Staggered Quantum Walk on 2D lattice. Most simplest way to do so is given in Figure 37.

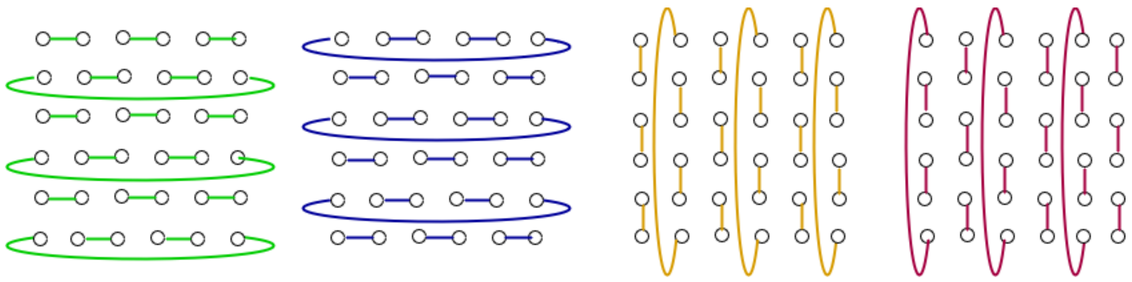


Figure 37: tessellation cover of the two-dimensional lattice with cyclic boundary

Using equation Equation 82 and Equation 89 the evolution operator for above tessellation is given as,

$$U_{00} = \left(\mathbb{I} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \right) \otimes U_\alpha + \left(\mathbb{I} \otimes \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes U_\beta$$

$$U_{10} = \left(\mathbb{I} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \right) \otimes U_\beta + \left(\mathbb{I} \otimes \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right) \otimes U_\alpha$$

where U_{00} and U_{10} is evolution operator for green and blue (first and second) tessellation respectively.

$$U_{01} = U_\alpha \otimes \left(\mathbb{I} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \right) + U_\beta \otimes \left(\mathbb{I} \otimes \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right)$$
$$U_{11} = U_\beta \otimes \left(\mathbb{I} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \right) + U_\alpha \otimes \left(\mathbb{I} \otimes \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

and U_{01} and U_{11} is evolution operator for yellow and brown (third and fourth) tessellation respectively.
So, the evolution operator for 2D SQW is given by,

$$U^{2D} = U_{11}U_{10}U_{01}U_{00}$$

8.3.4 Decomposition of evolution operator

$$\begin{aligned} U_{00} &= \mathbb{I} \otimes |0\rangle\langle 0| \otimes U_\alpha + \mathbb{I} \otimes |1\rangle\langle 1| \otimes U_\beta \\ &= Q_x^{-1}(\mathbb{I} \otimes U_\alpha)Q_x \end{aligned}$$

$$\begin{aligned} U_{10} &= \mathbb{I} \otimes |0\rangle\langle 0| \otimes U_\alpha + \mathbb{I} \otimes |1\rangle\langle 1| \otimes U_\beta \\ &= (\mathbb{I} \otimes X \otimes \mathbb{I})U_{00}(\mathbb{I} \otimes X \otimes \mathbb{I}) \end{aligned}$$

where Q_x is a controlled P gate with control qubit $|q_{(\frac{n}{2}-1)}\rangle$ and targets $|q_{\frac{n}{2}}....q_{n-1}\rangle$, and is given by,

$$Q_x = \mathbb{I} \otimes |0\rangle\langle 0| \otimes \mathbb{I} + \mathbb{I} \otimes |1\rangle\langle 1| \otimes P$$

similarly,

$$\begin{aligned} U_{01} &= U_\alpha \otimes \mathbb{I} \otimes |0\rangle\langle 0| + U_\beta \otimes \mathbb{I} \otimes |1\rangle\langle 1| \\ &= Q_y^{-1}(U_\alpha \otimes \mathbb{I})Q_y \end{aligned}$$

$$\begin{aligned}U_{11} &= U_\alpha \otimes \mathbb{I} \otimes |1\rangle\langle 1| + U_\beta \otimes \mathbb{I} \otimes |0\rangle\langle 0| \\ &= (\mathbb{I} \otimes X \otimes \mathbb{I}) U_{01} (\mathbb{I} \otimes X \otimes \mathbb{I})\end{aligned}$$

where Q_y is a controlled P gate with control qubit $|q_{(n-1)}\rangle$ and targets $|q_0....q_{(\frac{n}{2}-1)}\rangle$, and is given by,

$$Q_y = \mathbb{I} \otimes \mathbb{I} \otimes |0\rangle\langle 0| + P \otimes \mathbb{I} \otimes |1\rangle\langle 1|$$

8.3.5 Implementation on IBM computers

The evolution operator for 2D SQW is given by,

$$U^{2D} = U_{11} U_{10} U_{01} U_{00}$$

Figure 38 refers to the 2D SQW circuit for one walk step on a lattice with 16 vertices using 4 qubits which is to be implemented using QISKIT.

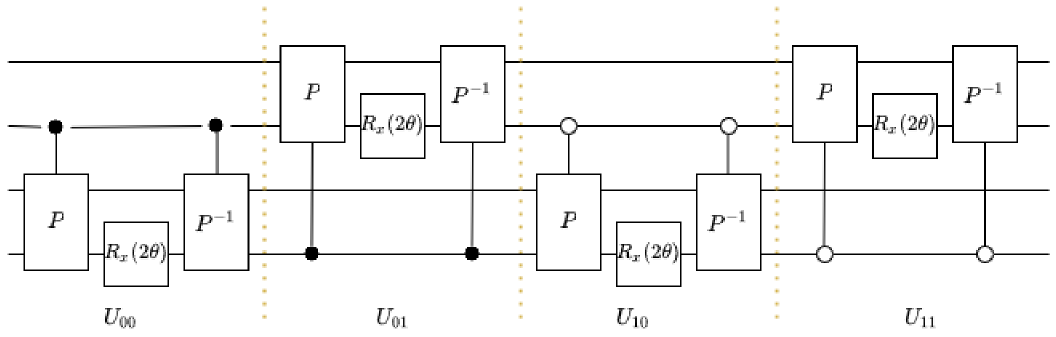


Figure 38: 2D SQW evolution operator circuit for one walk step on a lattice with 16 vertices using 4 qubits

Refer Appendix G for Python code of 2D SQW in QISKIT. Figure 39 and Figure 40 refers to Probability distribution plots of 2D SQW obtained using QISKIT.

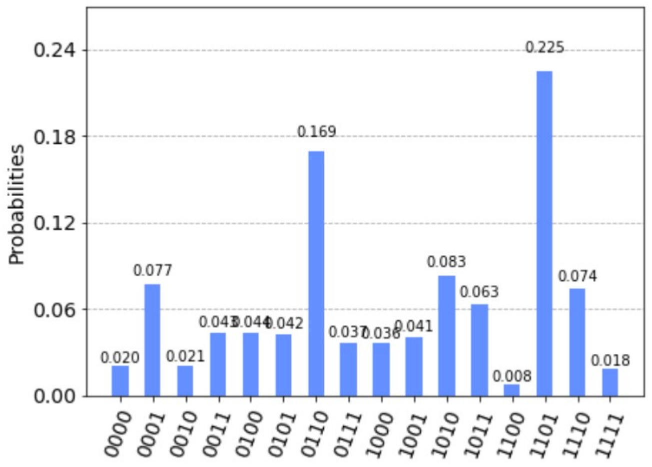


Figure 39: Probability distribution after four steps of 2D SQW with $|0000\rangle$ as walkers initial position.

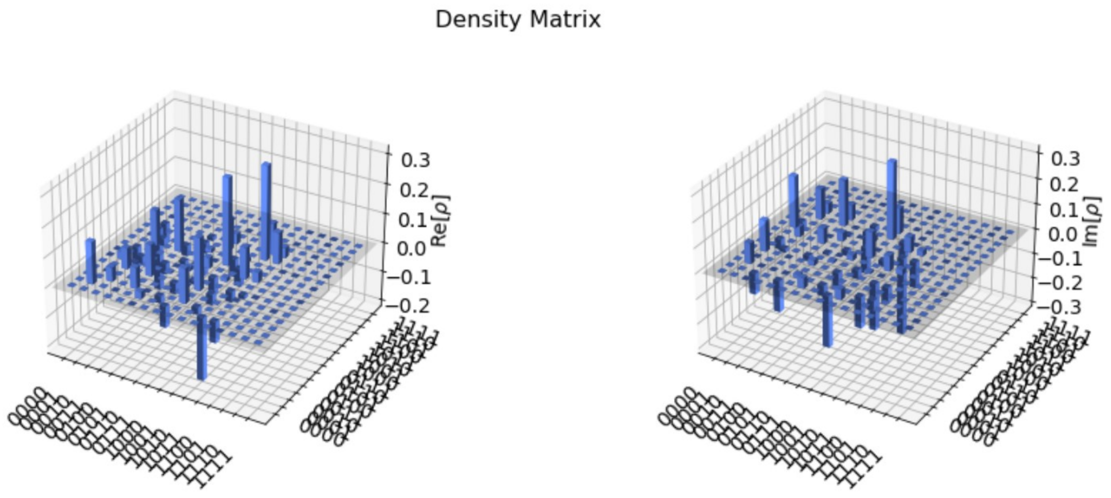


Figure 40: Probability distribution after one step of 2D Staggered Quantum walk, with $|0000\rangle$ as initial state using qasm simulator on 4 qubits

8.4 Szegedy's Quantum Walks

[8] Szegedy's Quantum Walk on a bipartite graph $\Gamma(X, Y, E)$ with biadjacent matrix is defined on a Hilbert space $H^{mn} = H^m \otimes H^n$, where $m = |X|$ and $n = |Y|$. The QW is driven by the unitary operator

$$U = R_0 R_1 \tag{85}$$

$$R_0 = 2 \sum_{x \epsilon X} |\phi_x\rangle\langle\phi_x| - \mathbb{I} \tag{86}$$

$$R_1 = 2 \sum_{y \epsilon Y} |\psi_y\rangle\langle\psi_y| - \mathbb{I} \tag{87}$$

for example, Let us take a line graph as in $L(\Gamma)$Figure 41.

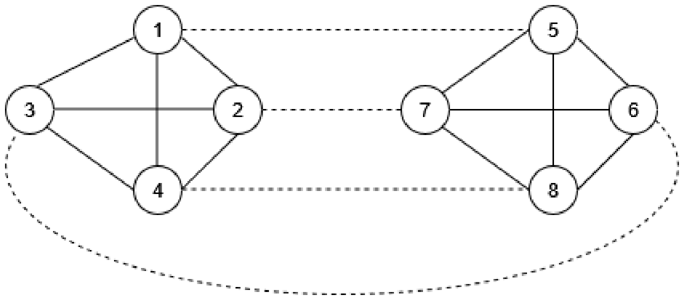


Figure 41: line graph $L(\Gamma)$

Here two tessellations are required to cover all the edges and vertices of this line graph. These tessellations are represented by orange and blue coloured tiles Figure 42.

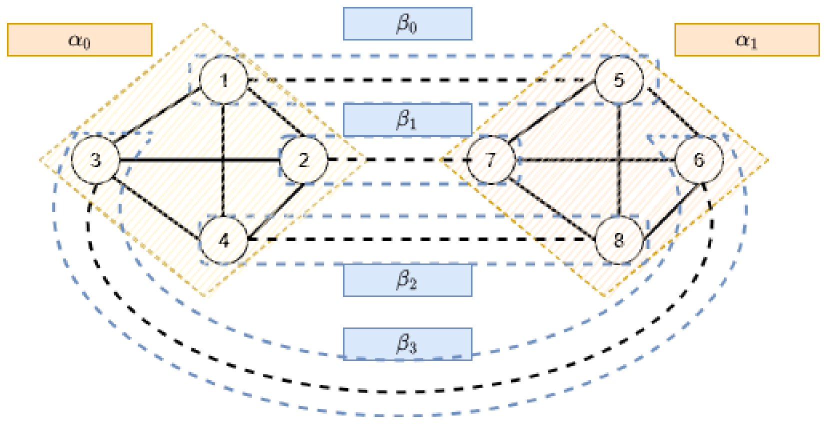


Figure 42: Line graph $L(\Gamma)$ with a two-colorable clique partition labeled by the vertices of Γ.

Staggered Quantum Walk vectors of tessellation α (orange) are

$$\begin{aligned}|\alpha_0\rangle &= \frac{1}{2}(|1\rangle + |2\rangle + |3\rangle + |4\rangle)\\ |\alpha_1\rangle &= \frac{1}{2}(|5\rangle + |6\rangle + |7\rangle + |8\rangle)\end{aligned} \tag{88}$$

and Staggered Quantum Walk vectors of tessellation β (blue) are

$$\begin{aligned}|\beta_0\rangle &= \frac{1}{\sqrt{2}}(|1\rangle + |5\rangle)\\ |\beta_1\rangle &= \frac{1}{\sqrt{2}}(|2\rangle + |7\rangle)\\ |\beta_2\rangle &= \frac{1}{\sqrt{2}}(|4\rangle + |6\rangle)\\ |\beta_3\rangle &= \frac{1}{\sqrt{2}}(|4\rangle + |8\rangle)\end{aligned} \tag{89}$$

Figure 43 represents a bipartite graph Γ of line graph $L(\Gamma)$. Notice that there is a edge-vertex correspondence $\alpha_0\beta_0 \leftrightarrow 1, \alpha_0\beta_1 \leftrightarrow 2, \alpha_0\beta_2 \leftrightarrow 4, \alpha_0\beta_3 \leftrightarrow 3, \alpha_1\beta_0 \leftrightarrow 5, \alpha_1\beta_1 \leftrightarrow 6, \alpha_1\beta_2 \leftrightarrow 8, \alpha_1\beta_3 \leftrightarrow 7$.

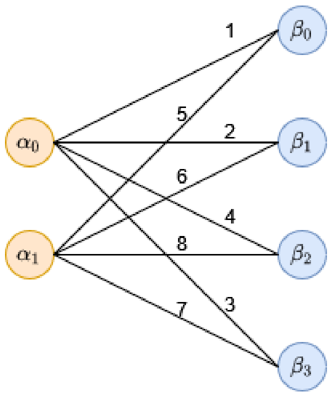


Figure 43: bipartite graph Γ of line graph $L(\Gamma)$

For a given two-colorable tessellations of a line graph with one vertex in each polygon intersection, it is possible to find the bipartite root graph.
As here for line graph $L(\Gamma)$, take polygon α_0 in $L(\Gamma)$. Selecting the polygons of tessellation β that have overlap with α_0, we obtain vector $|\phi_0\rangle$. Similarly we can obtain the other vectors used by szedegy's model.
Szegedy's model in Γ uses the following vectors,

$$\begin{aligned} |\phi_0\rangle &= \frac{1}{2}|\alpha_0\rangle(|\beta_0\rangle + |\beta_1\rangle + |\beta_2\rangle + |\beta_3\rangle) \\ |\phi_1\rangle &= \frac{1}{2}|\alpha_1\rangle(|\beta_0\rangle + |\beta_1\rangle + |\beta_2\rangle + |\beta_3\rangle) \end{aligned} \tag{90}$$

and

$$\begin{aligned} |\psi_0\rangle &= \frac{1}{\sqrt{2}}(|\alpha_0\rangle + |\alpha_1\rangle)|\beta_0\rangle \\ |\psi_1\rangle &= \frac{1}{\sqrt{2}}(|\alpha_0\rangle + |\alpha_1\rangle)|\beta_1\rangle \\ |\psi_2\rangle &= \frac{1}{\sqrt{2}}(|\alpha_0\rangle + |\alpha_1\rangle)|\beta_2\rangle \\ |\psi_3\rangle &= \frac{1}{\sqrt{2}}(|\alpha_0\rangle + |\alpha_1\rangle)|\beta_3\rangle \end{aligned} \tag{91}$$

From Equation 90 and Equation 91 we can find R_0 and R_1 using Equation 86 and Equation 87 and thus find evolution operator for Szegedy's Quantum walk U as given in Equation 85.

9 Future Scope

Random walks have applications in so many fields of scientific enquiry, ranging from physics, to computer science, to biology. For example we can have ion trap based implementation of random walk on a line.[9]
Implementation of random walk has been proposed by D¨ur et al.[3] Also large number of graphs have yet to be investigated in the context of quantum walks.
just like with classical random walks, we can use quantum random walks in the design of algorithms. This can give us complexity separations in the quantum query model by looking at glued trees (two binary trees attached leaf-to-leaf via a random permutation); there is no classical randomized algorithm (walk or otherwise) to get from one root to the other in a polynomial number of queries, but the quantum walk does it in polynomial number of queries.[4]
Also we can use Quantum Walk search to look for topological defects of the entire configuration space. Until now, the Quantum Walk search has only been used to find 'marked nodes,' as we'll study in the unsorted search Grover algorithm or good configurations within the configuration space that an oracle has specified. In this case, though, the QW search can be employed to hunt for topological faults, which are configuration space features. This suggests that instead of focusing on simple hole flaws in 2 D crystals, more general topological classification problems should be pursued. Grover search could be a natural occurrence, such as when fermions propagate in crystalline materials under specific conditions[7].
There are numerous reasons to concentrate on the Grover search among all quantum algorithms. For starters, because it transforms any brute force O(N) problem into an $O(\sqrt{N})$ problem. Even just having this quantum algorithm would be incredibly beneficial. Second, because of its extraordinary robustness: the algorithm is available in a variety of variations and has been reworked in a variety of ways, including in terms of resonance effect.
Hence we hope to use quantum random walk features to find more efficient algorithms on a quantum computer and apply to spin networks to study quantum gravity.

10 Conclusion

In this paper, we have formally studied about the basics of Quantum Computation and worked through some of the basic Quantum Algorithms and implemented it using qiskit. Then these simulations are executed on the Quantum Computing simulator and probability distributions are obtained which verified our results. From this we got the basic understanding of Quantum Algorithms and how to implement them on Quantum computers. Then we studied the most- straightforward one dimensional Random walks and came across some of the strange features of quantum random walk.

1. Here we came across the effect that even though our translation operator displaces by l, in some cases the particle will jump much further than l (the probability of this to happen is very less). Whereas classically it can not go beyond l

2. We can generate the Quantum Random Walk as per our requirement just by choosing the appropriate coin operator. Here we took example of Hadamard as coin operator, after several interference over time we noticed a drift to the one side in position of the walker. It started to differ from classical Random Walk after the T=3 time steps. We can obtain classical random walk if we take measurement after every step and do not allow the states to interfere.

3. In the purely quantum case the variance of the walk varies as $\sigma^2 \approx T^2$, whereas in classical case $\sigma^2 \approx T$ which shows the quadratic speed of random walk implementation in quantum case.

And then we studied about Staggered Quantum Walk and employed Staggered Quantum walk on 1D and 2D lattice.

References

[1] Frank Acasiete, Federica Agostini, Jalil Khatibi Moqadam, and Renato Portugal. Implementation of quantum walks on ibm quantum computers. *Quantum Inf. Process.*, 19:426, 2020.

[2] Y. Aharonov, L. Davidovich, and N. Zagury. Quantum random walks. *Phys. Rev. A*, 48:1687–1690, Aug 1993.

[3] W. Dür, R. Raussendorf, V. M. Kendon, and H.-J. Briegel. Quantum walks in optical lattices. *Phys. Rev. A*, 66:052319, Nov 2002.

[4] J Kempe. Quantum random walks: An introductory overview. *Contemporary Physics*, 44(4):307–327, Jul 2003.

[5] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[6] R. Portugal, R. A. M. Santos, T. D. Fernandes, and D. N. Gonçalves. The staggered quantum walk model. *Quantum Information Processing*, 15(1):85–101, oct 2015.

[7] Mathieu Roget, Stéphane Guillet, Pablo Arrighi, and Giuseppe Di Molfetta. Grover search as a naturally occurring phenomenon. *Physical Review Letters*, 124(18), May 2020.

[8] Mario Szegedy. Quantum speed-up of markov chain based algorithms. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 32–41, 2004. Proceedings - 45th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2004 ; Conference date: 17-10-2004 Through 19-10-2004.

[9] B. C. Travaglione and G. J. Milburn. Implementing the quantum random walk. *Phys. Rev. A*, 65:032310, Feb 2002.

A Teleportation circuit in QISKIT

Initializing the state to be teleported

First, create a quantum circuit that creates the state $\sqrt{0.70}|o\rangle + \sqrt{0.30}|1\rangle$

```
import numpy as np
from qiskit import IBMQ, Aer
from qiskit.quantum_info import Operator
from qiskit import QuantumCircuit, execute
from qiskit.visualization import plot_histogram
from qiskit.tools.jupyter import *

def initialize_qubit(given_circuit, qubit_index):
    import numpy as np
    given_circuit.initialize([np.sqrt(0.70), np.sqrt(0.30)],  qubit_index)
    return given_circuit

#create entanglement between Alice's and Bob's qubits.

def entangle_qubits(given_circuit, qubit_Alice, qubit_Bob):
    given_circuit.h(qubit_Alice)
    given_circuit.cx(qubit_Alice, qubit_Bob)
    return given_circuit

#do a Bell measurement of Alice's qubits.

def bell_meas_Alice_qubits(given_circuit, qubit1_Alice,  qubit2_Alice, clbit1_Alice,
    given_circuit.cx(qubit1_Alice, qubit2_Alice)
    given_circuit.h(qubit1_Alice)
    given_circuit.barrier()
    given_circuit.measure(qubit1_Alice, clbit1_Alice)
    given_circuit.measure(qubit2_Alice, clbit2_Alice)
    return given_circuit
```

Finally, we apply controlled operations on Bob's qubit.

1. an **X gate** is applied on Bob's qubit if the measurement outcome of Alice's second qubit, clbit2_Alice, is 1.

2. a **Z gate** is applied on Bob's qubit if the measurement outcome of Alice's first qubit, clbit1_Alice, is 1.

```
def controlled_ops_Bob_qubit(given_circuit, qubit_Bob, clbit1_Alice, clbit2_Alice):
    given_circuit.x(qubit_Bob).c_if(clbit2_Alice, 1)
    given_circuit.z(qubit_Bob).c_if(clbit1_Alice, 1)
    return given_circuit

#imports
from qiskit import QuantumRegister, ClassicalRegister
#set up the qubits and classical bits
all_qubits_Alice = QuantumRegister(2)
all_qubits_Bob = QuantumRegister(1)
creg1_Alice = ClassicalRegister(1)
creg2_Alice = ClassicalRegister(1)


#QUANTUM TELEPORTATION circuit here
# Initialize
mycircuit = QuantumCircuit(all_qubits_Alice, all_qubits_Bob,  creg1_Alice, creg2_Alic
initialize_qubit(mycircuit, 0)
mycircuit.barrier()
#Entangle
entangle_qubits(mycircuit, 1, 2)
mycircuit.barrier()
#Do a Bell measurement
bell_meas_Alice_qubits(mycircuit, all_qubits_Alice[0],  all_qubits_Alice[1], creg1_Al
mycircuit.barrier()
# Apply classically controlled quantum gates
controlled_ops_Bob_qubit(mycircuit, all_qubits_Bob[0], creg1_Alice, creg2_Alice)

#Look at the complete circuit
```

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

Out[14]:

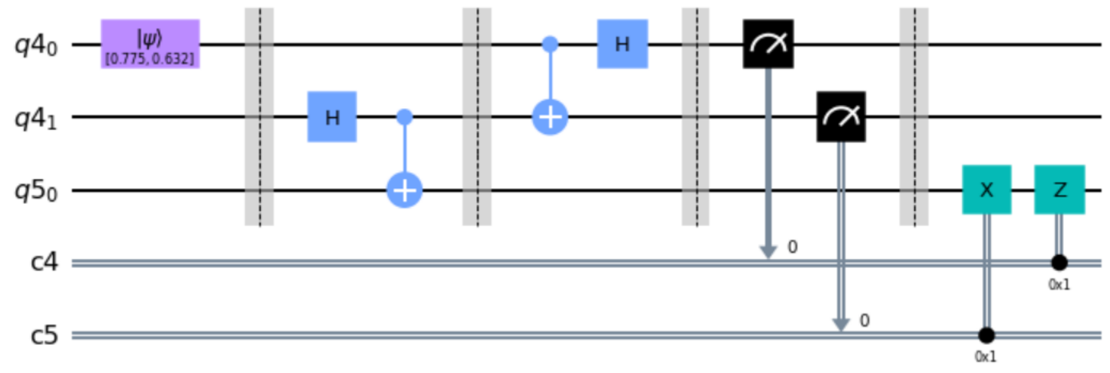


Figure 44: Teleportation circuit

B Deutsh Josza Algorithm in QISKIT

```
import numpy as np
from qiskit import IBMQ, BasicAer
from qiskit import QuantumCircuit, execute
from qiskit.visualization import plot_histogram
from qiskit.tools.jupyter import *

def dj_oracle(case,n):
    oracle_qc=QuantumCircuit(n+1)
    if case=="bal":
        for qubit in range(n):
            oracle_qc.cx(qubit,n)#if there are even no. of 1 then last qubit=0
    if case=="cons":
        out=np.random.randint(2)
        if out==1:
            oracle_qc.x(n)



    oracle_gate=oracle_qc.to_gate()
    oracle_gate.name="oracle"    #shows when we'll display the circuit
    return oracle_gate

#DEUTSCH JOZSA ALGORITHM
def dj_algorithm(n,case='random'):
    dj_ckt=QuantumCircuit(n+1,n) # n+1=q_reg, n=c_reg
    for qubit in range(n):
        dj_ckt.h(qubit)

    dj_ckt.x(n)
    dj_ckt.h(n) #it creates £|-\rangle£ state
#Append oracle circuit
    if case=='random':
        random=np.random.randint(2)
        if random==0:
            case="cons"
        else:
            case="bal"

    oracle=dj_oracle(case,n)
    dj_ckt.append(oracle,range(n+1))
#measuring n qubits
    for i in range(n):
        dj_ckt.h(i)
        dj_ckt.measure(i,i)
    return dj_ckt
n=4
dj_ckt=dj_algorithm(n)
dj_ckt.draw()
```

Out[39]:

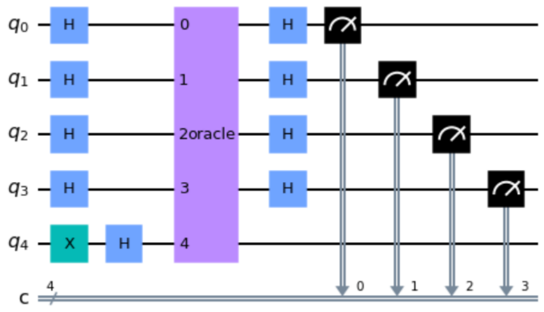


Figure 45: DJ circuit for n=4 qbits

TO CHECK RESULT

```
backend=BasicAer.get_backend('qasm_simulator')
shots=2084
sj_ckt=dj_algorithm(n,'bal')
results=execute(dj_ckt,backend=backend,shots=shots).result()
answer=results.get_counts()
plot_histogram(answer)
```

```
Out[37]:
```

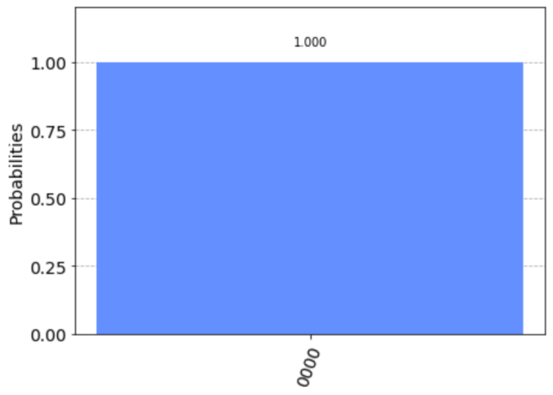


Figure 46: result

C Steps to write Grover's Algorithm

```
import numpy as np
from qiskit import IBMQ, Aer
from qiskit.quantum_info import Operator
from qiskit import QuantumCircuit, execute
from qiskit.visualization import plot_histogram
from qiskit.tools.jupyter import *

def phase_oracle(n,indm,name='oracle'):
    qc=QuantumCircuit(n,name=name)
    o_matrix=np.identity(2**n)
    for indm in indm:
        o_matrix[indm,indm]=-1
    qc.unitary(Operator(o_matrix),range(n))\ \ #turning unitary matrix into an opera
    return qc

def diffuser(n):
    qc=QuantumCircuit(n,name='Diff="V"')
    qc.h(range(n))
    qc.append(phase_oracle(n,[0]),range(n))
    qc.h(range(n))
    return qc

def grover(n,marked):
    qc=QuantumCircuit(n,n)
    r=int(np.round(np.pi/(4*np.arcsin(np.sqrt(len(marked)/2**n)))-1/2))
    print(f'{n  qubits,basis state {marked  marked},{r  rounds') }}
    qc.h(range(n)) #to get state £|s\rangle£
    for i in range(r):
        qc.append(phase_oracle(n,marked),range(n))
        qc.append(diffuser(n),range(n))
    qc.measure(range(n),range(n))
    return qc
```

```
n=4
x=np.random.randint(2**n)
marked=[x]
qc=grover(n,marked)
qc.draw(output='mpl')
```

```
4 qubits,basis state [13] marked,3 rounds
```

Out[4]:

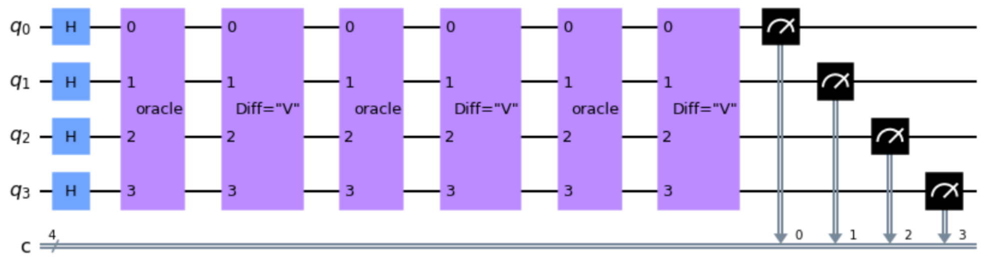


Figure 47: GA output for n=4 qbits

```
backend=Aer.get backend('qasm simulator')
result=execute(qc,backend,shots=10000).result()
counts=result.get counts(qc)
print(counts)
print(np.pi/(4*np.arcsin(np.sqrt(len(marked)/2**n)))-1/2)
plot histogram(counts)
```

```
{'001': 82, '101': 84, '010': 82, '110': 9433, '100': 82, '000': 93, '111': 67, '011': 77}
1.6734079041462837
```

Out[24]:

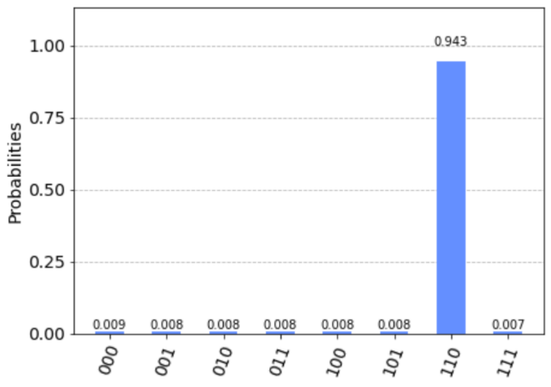


Figure 48: Marked state $|110\rangle$ has highest probability

D Classical random walk in QISKIT

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from numpy import random

#Defines all of the necessary parameters
N = 50        # Defines the total number of steps our walker will take
pr = 0.5      #Defines the probability of our walker stepping to the right
i = 0     # Defines the initial position of our walker
def random_walk(pr, N, i):
    position = i
    for j in range( 0 , N):
        coin_flip = list(np.random.choice(2, 1,p=[1-pr,pr]))   # Flips our weighted c
        position += 2*coin_flip[0]-1   #  Moves our walker according to the coin fli
    return position
print("The walker is located at: x = {var}".format(var = random_walk(pr, N, i)))
```

The walker is located at: x = -6

```
def dist(runs, N):
    positions = range(-1*N, N+1)
    instances = [0 for i in range(-1*N, N+1)]
    for k in range(0, runs):
        result = random\_walk(pr, N, i)
        instances[positions.index(result)] += 1
    plt.bar(positions, [n/runs for n in instances])
    plt.show()
dist(10000, N)
```

```
import scipy
def binomial(n,k):
    from math import factorial
    b=factorial(n)/((factorial(n-k))*(factorial(k)))
    return(int(b))
```

```
def height_calculate(x, N, pr):
    a = (N + x)/2
    b = (N - x)/2
    if (x%2 == 0):
        var = binomial(N, a)*(pr**a)*((1-pr)**b)
    else:
        var = 0
    return var
positions = range(-1*N, N+1)
heights = [height_calculate(x, N, pr) for x in positions]
plt.bar(positions, heights)
plt.show()
```

E Coined Quantum Random walk in 1D in QISKIT

```
#IMPORTS
import numpy as np
%matplotlib inline
from matplotlib import pyplot as plt
from qiskit import *
from qiskit import IBMQ, Aer
from qiskit.quantum_info import Operator
from qiskit import QuantumCircuit, execute
from qiskit.visualization import plot_state_city
from qiskit.tools.visualization import plot_histogram, plot_bloch_vector
import qiskit.quantum_info as qi
from qiskit.tools.jupyter import *
from qiskit.aqua.components.uncertainty_models import NormalDistribution,UniformDistr

#INCREAMENT OPERATOR
n=7
def inc_op(n):
    circuit=QuantumCircuit(n,name='inc')
    qr=circuit.qubits
    for i in range(n-1,0,-1):
        circuit.x(i)
        circuit.mct([*range(i,n)],i-1)
    circuit.x(n-1)
    return circuit
inc_gate = inc_op(n).to_gate()
inc_gate.definition.draw()

#DECREAMENT OPERATOR
def dec_op(n):
    circuit=QuantumCircuit(n,name='dec')
    qr=circuit.qubits
    for i in range(0,n-1,1):
        circuit.mct([*range(i+1,n)],i)
        if i<n-2:
            circuit.x(i+1)
    return circuit
dec_gate = dec_op(n).to_gate()
dec_gate.definition.draw()

#GENERATING WALK
Initial_state = str(input("Initial state? (Type 0/1/i): ")).lower()
def generate_walk(n,times,Initial_state):
    qc=QuantumCircuit(n,n)
    if Initial_state=='1':# down state
        qc.x(1)

    elif Initial_state=='0': # up state
        qc.x(1)
        qc.x(n-1)

    else:  #balanced state
        qc.x(1)
        qc.h(n-1)
        qc.s(n-1)
    for i in range(times):
        qc.append(qrw(n),range(n))

    qc.measure(range(n),range(n))
    return qc
n=7
times=60
qc=generate_walk(n,times,Initial_state)
qc.draw('mpl')

#PLOT OF PROBABILITY V/S POSITION
backend=Aer.get_backend('qasm_simulator')
result=execute(qc,backend,shots=5000).result()
counts=result.get_counts(qc)

print(counts)
sortedcounts = []
sortedkeys = sorted(counts)
for i in sortedkeys:
    for j in counts:
        if(i == j):
            sortedcounts.append(counts.get(j))
plt.suptitle('Uniform Distribution')
plt.xlabel("x")
plt.ylabel("Probability")
plt.plot(sortedcounts)
plt.show()

#PLOTTING HISTOGRAM
simulator=Aer.get_backend('qasm_simulator')
result=execute(qc,backend=simulator).result()
counts=result.get_counts(qc)
```

```
plot_histogram(result.get_counts(qc))

#CITYPLOT
def generate_walk(n,times):
    qc=QuantumCircuit(n,n)
    for i in range(times):
        qc.append(qrw(n),range(n))
    return qc
qc_AB= generate_walk(4,10)
psi_AB = qi.Statevector.from_instruction(qc_AB)
psi_AB.draw('latex', prefix='|\\psi_{AB}\\rangle = ')
rho_AB = qi.DensityMatrix.from_instruction(qc_AB)
rho_AB.draw('latex', prefix='\\rho_{AB} = ')

plot_state_city(rho_AB.data, title='Density Matrix')
```

F 1D Staggered Quantum walk in QISKIT

```
#IMPORTS

import numpy as np
%matplotlib inline
from matplotlib import pyplot as plt
from qiskit import *
from qiskit import IBMQ, Aer
from qiskit.quantum_info import Operator
from qiskit import QuantumCircuit, execute
from qiskit.tools.visualization import plot_histogram, plot_bloch_vector
from qiskit.visualization import plot_state_city
import qiskit.quantum_info as qi
from qiskit.tools.jupyter import *

#CIRCUIT
```

```
#Increament operator
def inc_op(n):
    circuit=QuantumCircuit(n,name='inc')
    qr=circuit.qubits
    for i in range(n-2):

        circuit.mcrx(np.pi,[*range(i+1,n)],i)
    circuit.cx(n-1,n-2)
    circuit.x(n-1)
    return circuit
inc_gate = inc_op(n).to_gate()

#Decreament Operator
def dec_op(n):
    circuit=QuantumCircuit(n,name='dec')
    qr=circuit.qubits
    circuit.x(n-1)
    circuit.cx(n-1,n-2)
    for i in range(n-3,-1,-1):
        circuit.mcrx(np.pi,[*range(i+1,n)],i)

    return circuit
dec_gate = dec_op(n).to_gate()

#Generating walk
theta=3*np.pi/2
def generate_walk(n,times):
    circuit=QuantumCircuit(n,n)
    qr=circuit.qubits

    for i in range(times):

        circuit.rx(theta/4,n-1)
        circuit.append(inc_op(n),range(n))
        circuit.rx(theta/4,n-1)
        circuit.append(dec_op(n),range(n))
    circuit.measure(range(n),range(n))
    return circuit
n=7
times=1
circuit= generate_walk(n,times)
circuit.draw('mpl')

#PLOTTING

backend=Aer.get_backend('qasm_simulator')
result=execute(circuit,backend,shots=5000).result()
counts=result.get_counts(circuit)

print(counts)
sortedcounts = []
sortedkeys = sorted(counts)
for i in sortedkeys:
    for j in counts:
        if(i == j):
            sortedcounts.append(counts.get(j))
##Output-{'0100000': 469, '0000000': 2376, '1111111': 1048, '1000000': 1107}

plt.suptitle('Uniform Distribution')
plt.xlabel("probability")
plt.plot(sortedcounts)
plt.show()

#PLOTTING HISTOGRAM

simulator=Aer.get_backend('qasm_simulator')
result=execute(circuit,backend=simulator).result()
counts=result.get_counts(circuit)
```

```
plot=plot_histogram(result.get_counts(circuit))
plot

#CITYPLOT

def generate_walk(n,times):
    qc_AB=QuantumCircuit(n,n)
    for i in range(times):

        qc_AB.rx(theta,n-1)
        qc_AB.append(inc_op(n),range(n))
        qc_AB.rx(theta,n-1)
        qc_AB.append(dec_op(n),range(n))
    return qc_AB
qc_AB= generate_walk(4,1)
psi_AB = qi.Statevector.from_instruction(qc_AB)
psi_AB.draw('latex', prefix='|\\psi_{AB}\\rangle = ')
rho_AB = qi.DensityMatrix.from_instruction(qc_AB)
rho_AB.draw('latex', prefix='\\rho_{AB} = ')
plot_state_city(rho_AB.data, title='Density Matrix')
```

G 2D Staggered Quantum walk in QISKIT

```
#IMPORTS
import numpy as np
%matplotlib inline
from qiskit import *
from qiskit import IBMQ, Aer
from qiskit.quantum_info import Operator
from qiskit import QuantumCircuit, execute
from qiskit.tools.visualization import plot_histogram, plot_bloch_vector
import qiskit.quantum_info as qi
from qiskit.quantum_info import Statevector
from qiskit.tools.jupyter import *
from qiskit import IBMQ

#INCREAMENT OPERATOR
def inc_op(n):
    circuit=QuantumCircuit(n,name='inc')
    qr=circuit.qubits
    for i in range(n-2):

        circuit.mcrx(np.pi,[*range(i+1,n)],i)
    circuit.cx(n-1,n-2)
    circuit.x(n-1)
    return circuit
qc = inc_op(2)
inc_gate = inc_op(2).to_gate()
cntrl_inc = inc_gate.control(1)
cntrl_inc.definition.draw()

#DECREAMENT OPERATOR
def dec_op(n):
    circuit=QuantumCircuit(n,name='dec')
    qr=circuit.qubits
    circuit.x(n-1)
    circuit.cx(n-1,n-2)
    for i in range(n-3,-1,-1):
        circuit.mcrx(np.pi,[*range(i+1,n)],i)

    return circuit
dec_gate = dec_op(2).to_gate()
cntrl_dec = dec_gate.control(1)
cntrl_dec.definition.draw()

#GENERATE WALK
n=4
theta = 2*np.pi/3
```

```
def qrw(n):
    circuit=QuantumCircuit(n,name='qrw')
    circuit.append(cntrl_inc,[1,2,3])
    circuit.rx(theta,3)
    circuit.append(cntrl_dec,[1,2,3])

    circuit.append(cntrl_inc,[3,0,1])
    circuit.rx(theta,1)
    circuit.append(cntrl_dec,[3,0,1])

    circuit.x(1)
    circuit.append(cntrl_inc,[1,2,3])
    circuit.rx(theta,3)
    circuit.append(cntrl_dec,[1,2,3])
    circuit.x(1)

    circuit.x(3)
    circuit.append(cntrl_inc,[3,0,1])
    circuit.rx(theta,1)
    circuit.append(cntrl_dec,[3,0,1])
    circuit.x(3)

    return circuit
def generate_walk(circuit,n,times):
    initState = Statevector.from_label('0000')
    circuit.initialize(initState)
    for i in range(times):
        circuit.append(qrw,range(n))
    circuit.measure(range(n),range(n))
    return circuit

n=4
times=4
circuit=QuantumCircuit(n,n)
```

```
qr=circuit.qubits
generate_walk(circuit,n,times)
circuit.draw(output='mpl')

#PLOTTING HISTOGRAM
simulator=Aer.get_backend('qasm_simulator')
result=execute(circuit,backend=simulator).result()
plot_histogram(result.get_counts(circuit))

#ON IBM COMPUTERS
IBMQ.load_account()

provider=IBMQ.get_provider('ibm-q')

qcomp= provider.get_backend('ibmq_qasm_simulator')

job=execute(circuit,backend=qcomp)

from qiskit.tools.monitor import job_monitor

job_monitor(job)

result=job.result()

plot_histogram(result.get_counts(circuit))

#CITYPLOT
def generatewalk(n):
    circuit=QuantumCircuit(n,n)
    initState = Statevector.from_label('0000')
    circuit.initialize(initState)
    circuit=QuantumCircuit(n,name='qrw')
    circuit.append(cntrl_inc,[1,2,3])
    circuit.rx(theta,3)
    circuit.append(cntrl_dec,[1,2,3])

    circuit.append(cntrl_inc,[3,0,1])
    circuit.rx(theta,1)
    circuit.append(cntrl_dec,[3,0,1])

    circuit.x(1)
    circuit.append(cntrl_inc,[1,2,3])
    circuit.rx(theta,3)
    circuit.append(cntrl_dec,[1,2,3])
    circuit.x(1)

    circuit.x(3)
    circuit.append(cntrl_inc,[3,0,1])
    circuit.rx(theta,1)
    circuit.append(cntrl_dec,[3,0,1])
    circuit.x(3)


    return circuit
qc_AB= generatewalk(4)
psi_AB = qi.Statevector.from_instruction(qc_AB)
psi_AB.draw('latex', prefix='|\\psi_{AB}\\rangle = ')
rho_AB = qi.DensityMatrix.from_instruction(qc_AB)
rho_AB.draw('latex', prefix='\\rho_{AB} = ')
from qiskit.visualization import plot_state_city
plot_state_city(rho_AB.data, title='Density Matrix')
```

H Link to Source code:

https://github.com/ShwetaDadhwal/quantumrandomwalk.git