

Internship Report

Submitted in partial fulfillment of the requirements for the award of the degree

Of

-BACHELOR OF TECHNOLOGY-

In

-COMPUTER SCIENCE AND ENGINEERING-

By

Kritika Jha

10601012023

Guided by

Dr. Harish Kumar Sahu

Scientist 'E'

Scientific Analysis Group (SAG),

Defence Research and Development Organisation (DRDO)



**INDIRA GANDHI DELHI TECHNICAL UNIVERSITY
FOR WOMEN**

NEW DELHI – 110006

JUNE 2024 - JULY 2024 INDEX

INDEX

Certificate.....	3
Undertaking regarding anti plagiarism.....	5
Acknowledgement	6
Declaration	7
Abstract.....	8
1. Introduction	
1.1 Introduction to Quantum Computing.....	9
1.2.Details of Quantum Algorithms.....	9
1.3.Types of Quantum Algorithm.....	10
2. Internship Objectives.....	11
3. Weekly Activities	
3.1 Week 1.....	12
3.2 Week 2.....	14
3.3 Week 3.....	16
3.4 Week 4.....	18
3.5 Week 5.....	20
3.6 Week 6.....	22
3.7 Week 7.....	24
3.8 Week 8.....	27
4. Future Scope.....	28
5. Conclusion.....	30
6. References.....	32

CERTIFICATE

This is to certify that the project entitled “**Quantum Maximum Search**” is being submitted at IGDTUW, Delhi for the award of **Bachelor of Technology in Computer Science and Engineering** degree. It contains the record of bonafide work carried out by **Kritika Jha and 10601012023** under my supervision and guidance. It is further certified that the work presented here has reached the standard of B.Tech and to the best of my knowledge has not been submitted anywhere else for the award of any other degree or diploma.

Dr. Harish Kumar Sahu

Scientist ‘E’

Scientific Analysis Froup (SAG)

Defence Research and Development Organisation (DRDO)

Date: 05 November 2024

Place: Delhi

DEFENCE RESEARCH & DEVELOPMENT ORGANISATION
SCIENTIFIC ANALYSIS GROUP



CERTIFICATE

This is to certify that **Ms. KRITIKA JHA** a student of **INDIRA GANDHI DELHI TECHNICAL UNIVERSITY FOR WOMEN, DELHI** has carried out industrial training on the topic "**QUANTUM MAXIMUM SEARCH**" at Scientific Analysis Group, DRDO, Ministry of Defence, Metcalfe House Complex, Delhi-110054, as part of **B.TECH (CSE)** program requirement. The student carried out this work sincerely during the period from **04 JUNE 2024** to **31 JULY 2024** and completed the training successfully.


(**SUNIL SAGAR**)
Scientist 'F' / Head HR
Date: 05 November 2024



Harish Sahu
(**Dr. HARISH KUMAR SAHU**)
Guide / Scientist 'E'

UNDERTAKING REGARDING ANTI-PLAGIARISM

I, ***Kritika Jha*** hereby, declare that the content presented in the report are free from plagiarism and is properly cited and written in my own words. In case, plagiarism is detected at any stage, I shall be solely responsible for it.

Kritika Jha

(10601012023)

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to **Dr. Harish Kumar Sahu, Scientist 'E', SAG, DRDO**, for his exceptional guidance and mentorship during my internship. His deep knowledge, continuous encouragement, and constructive feedback were instrumental in shaping my understanding of quantum computing and its applications.

I would also like to thank the entire team at SAG, DRDO, for providing me with a supportive environment and valuable resources that significantly enhanced my learning experience.

Furthermore, I extend my sincere thanks to **Dr. S.R.N Reddy**, Head of the Department of Computer Science, for giving me the golden opportunity to pursue this internship and supporting me throughout the journey.

Finally, I am deeply grateful to everyone who contributed to this enriching experience and helped me refine my skills and knowledge.

Kritika Jha

10601012023

DECLARATION

I, **Kritika Jha**, solemnly declare that the project report, **IMPLEMENTATION OF GROVER'S ALGORITHM FOR OPTIMIZED SEARCH AND MAXIMUM ELEMENT IDENTIFICATION**, is based on my own work carried out during the course of my study under the supervision of **Dr. Harish Kumar Sahu**, Scientist 'E', SAG, DRDO. I assert that the statements made and conclusions drawn in this report are an outcome of my research work.

I further certify that:

1. The work contained in this report is original and has been completed by me under the guidance of my supervisor.
2. The work has not been submitted to any other institution for any degree, diploma, or certificate in this university or any other university in India or abroad.
3. I have followed the guidelines provided by the university in writing this report.
4. Wherever materials (text, data, theoretical analysis, equations, codes, figures, tables, pictures, etc.) from other sources have been used, due credit has been given to them in the report, along with their details in the references section.

Kritika Jha

10601012023

Abstract

This project explores the application of quantum computing to solve the maximum-finding problem in an unsorted dataset using Grover's algorithm. The algorithm achieves a significant improvement in efficiency, reducing the complexity from $O(N)$ in classical methods to $O(\sqrt{N})$ through quantum superposition and amplitude amplification. By iteratively refining guesses for the maximum value, the algorithm converges on the correct result with high probability.

The implementation, carried out using Qiskit, involved constructing quantum circuits, defining an oracle to identify elements larger than the current guess, and applying Grover's diffusion operator to amplify the target state. Simulations validated the algorithm's effectiveness, with results supported by probability histograms highlighting the amplified likelihood of the maximum element.

This project demonstrates the potential of quantum algorithms for solving optimization problems efficiently, providing a strong basis for further exploration as quantum technology advances.

CHAPTER 1: INTRODUCTION

1.1 Introduction to Quantum Computing

Quantum computing represents a revolutionary approach to computation, based on the principles of quantum mechanics, which governs the behaviour of matter and energy at the smallest scales—atoms and subatomic particles. Unlike classical computers, which process information using binary bits (0s and 1s), quantum computers use **quantum bits** or **qubits**. The core principles of quantum computing, such as **superposition** and **entanglement**, provide a significant advantage over classical computing in solving certain types of complex problems.

- **Superposition** enables a qubit to exist in multiple states simultaneously, unlike classical bits, which are limited to being either 0 or 1. This ability allows quantum computers to perform calculations in parallel, drastically speeding up computations.
- **Entanglement** is another powerful phenomenon where qubits become intertwined in such a way that the state of one qubit directly influences the state of another, even if they are separated by large distances. This property enhances the computational power of quantum systems.

Quantum computing takes advantage of these quantum properties to perform computations at speeds that would be impossible for classical computers, opening the door to advancements in fields like cryptography, simulation, and optimization.

1.2 Details of Quantum Algorithms

Quantum algorithms leverage the unique properties of quantum mechanics to solve problems more efficiently than classical algorithms. One of the most notable quantum algorithms is **Grover's Algorithm**, which is designed to search unsorted databases in **quadratic time**, making it much faster than classical search algorithms. This algorithm's potential has been extended beyond basic search tasks, including applications like **finding the maximum element in an unsorted dataset**.

In my internship at DRDO, I focused on understanding and applying Grover's Algorithm. I began by exploring the basics of quantum search algorithms and then delved deeper into how Grover's Algorithm could be adapted for specific problems, such as finding the maximum element in an unsorted dataset. I studied relevant research papers, including those by **L. K. Grover** and others, which laid the foundation for understanding the efficiency of quantum search methods compared to classical techniques.

1.3 Types of Quantum Algorithms

Quantum computing encompasses various types of algorithms, each designed to exploit different aspects of quantum mechanics. Some key types of quantum algorithms include:

- **Search Algorithms:** These algorithms, such as **Grover's Algorithm**, are designed to efficiently search through unsorted datasets or solve search problems in databases. Grover's Algorithm, in particular, provides a quadratic speedup over classical search methods.
- **Shor's Algorithm:** This famous quantum algorithm is used for factoring large numbers efficiently, offering exponential speedup compared to the best-known classical algorithms. It has significant implications for cryptography, particularly in breaking widely used encryption systems.
- **Quantum Optimization Algorithms:** These algorithms leverage quantum mechanics to solve complex optimization problems more efficiently than classical algorithms. Examples include the **Quantum Approximate Optimization Algorithm (QAOA)**, which can be applied to problems in logistics, finance, and machine learning.

In the subsequent weeks of my internship, I explored these algorithms in greater detail, starting with the implementation of Grover's Algorithm in a 2-qubit system and progressively scaling up to more complex systems, ultimately focusing on its application for finding the maximum element in an unsorted dataset.

CHAPTER 2: INTERNSHIP OBJECTIVES

The primary objectives of my internship at DRDO were to develop a strong understanding of quantum computing fundamentals, implement quantum algorithms, and explore their real-world applications. The specific goals for this internship were as follows:

1. Understand Quantum Computing Fundamentals

The first objective was to gain a solid understanding of quantum mechanics and quantum computing principles. This included learning about the basic building blocks of quantum systems, such as **qubits**, **quantum gates**, **superposition**, and **entanglement**. A foundational understanding of these concepts was crucial to effectively working with quantum computing tools and algorithms.

2. Implement Grover's Algorithm Using Qiskit

The second objective was to implement **Grover's Algorithm** on a quantum simulator using **Qiskit**, IBM's open-source quantum computing framework. Grover's Algorithm is an essential quantum search algorithm, and by implementing it, I aimed to deepen my understanding of quantum search algorithms and their applications. This involved coding and testing the algorithm for simple use cases, including a 2-qubit system to search for a marked state.

3. Research and Adapt Grover's Algorithm for Maximum-Finding in Unsorted Datasets

The third goal was to research how Grover's Algorithm can be adapted to solve the **maximum-finding problem** in unsorted datasets. This involved studying research papers and previous work, especially by **L. K. Grover**, on how to modify the basic search algorithm to find the maximum element in a set. I then applied this knowledge to create a quantum algorithm capable of solving this problem efficiently, testing it with small datasets to validate its effectiveness.

These objectives laid the foundation for the work I conducted during my internship and were integral to understanding the practical applications of quantum computing in defense technology and other fields.

CHAPTER 3: WEEKLY ACTIVITIES

3.1: Week 1: Introduction to Quantum Computing

Quantum computing is a revolutionary field within computer science that uses the principles of quantum mechanics to solve problems that classical computers struggle with, or can't solve in a reasonable amount of time. Unlike classical computers, which rely on bits to process information, quantum computers use qubits, which can hold more information at once, thanks to quantum phenomena like superposition and entanglement.

Quantum mechanics, the branch of physics that deals with the behavior of subatomic particles, underpins quantum computing. Some of the most important concepts in quantum mechanics include superposition, entanglement, decoherence, and interference. These principles allow quantum computers to perform computations in ways that classical computers cannot.

Key Principles of Quantum Mechanics

To understand quantum computing, it's essential to grasp the following four principles of quantum mechanics:

1. **Superposition**: This is the idea that a quantum particle, or qubit in the case of quantum computing, can exist in more than one state simultaneously. For example, a qubit can be in a state of both 0 and 1 at the same time, unlike a classical bit, which can only be in one state at a time. This ability allows quantum computers to process multiple possibilities at once.
2. **Entanglement**: Entanglement occurs when two or more quantum particles become linked in such a way that the state of one particle can instantly affect the state of another, regardless of the distance between them. This phenomenon is crucial for quantum computers as it enables faster and more complex computations.
3. **Decoherence**: Decoherence is the loss of quantum properties like superposition, which happens when a quantum system interacts with its environment. This is a major challenge in building stable quantum computers, as it can cause errors in computations.

4. **Interference:** Interference is the phenomenon where quantum states interact with one another, either amplifying or diminishing certain possibilities. This allows quantum computers to find the most likely solution to a problem by boosting correct answers and cancelling out wrong ones.

Qubits and Their Role in Quantum Computing

Qubits are the fundamental units of information in quantum computers. They are usually created by manipulating quantum particles like electrons, photons, or atoms. Unlike classical bits, which store data as either 0 or 1, qubits can represent a combination of these states simultaneously. This unique property makes quantum computers exponentially more powerful than classical ones for specific problems.

For example:

- 1 qubit = 2 states
- 2 qubits = 4 states
- 3 qubits = 8 states

As the number of qubits increases, the computational possibilities grow exponentially.

Quantum Computers vs. Classical Computers

Quantum computers are fundamentally different from classical computers:

Feature	Classical Computers	Quantum Computers
Storage Unit	Bits (0 or 1)	Qubits (0, 1, or a combination)
Computation Process	Sequential and logical	Parallel and probabilistic
Problem-Solving Ability	Deterministic (single solution)	Probabilistic (range of solutions)

3.2 : Week 2: Setting Up Qiskit and Exploring IBM Quantum Experience

As part of learning quantum computing, I was introduced to **Qiskit**, an open-source Python library that allows users to write, simulate, and execute quantum programs. The steps involved in setting up Qiskit were straightforward and user-friendly, making it accessible even for beginners.

1. Installing Qiskit

I installed Qiskit in a Python virtual environment using the command:

```
pip install qiskit
```

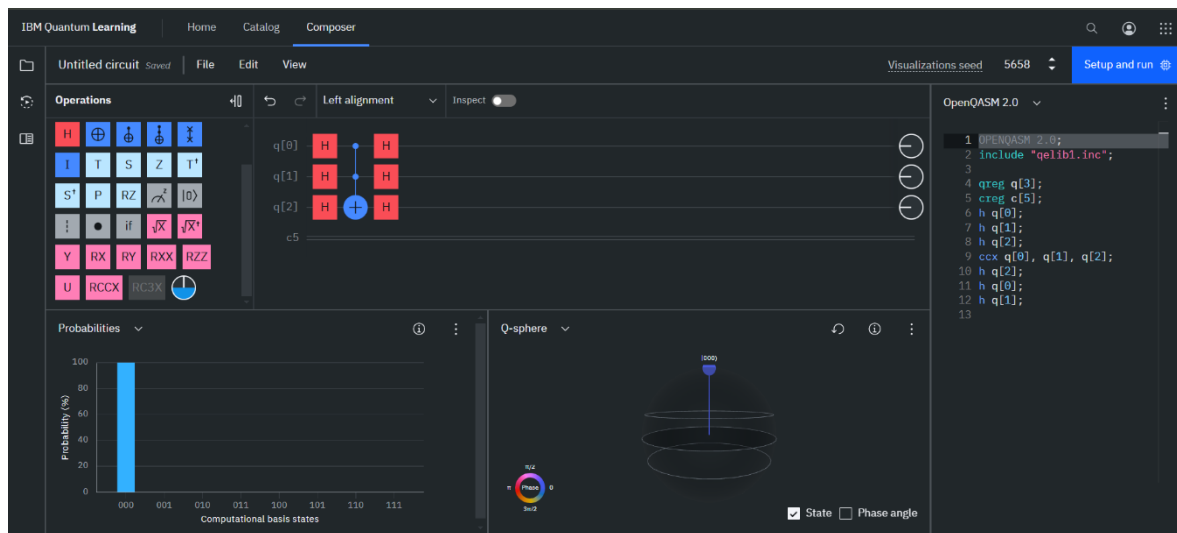
2. Learning Qiskit Basics

I explored its modules, including `qiskit.circuit` for creating quantum circuits and `qiskit.visualization` for plotting and analyzing quantum states. I also learned how to simulate quantum circuits using AerSimulator.

3. Using the IBM Quantum Experience

IBM Quantum Experience provides a platform to run quantum circuits on actual quantum computers. I created an account, accessed the dashboard, and obtained an API token to connect my Qiskit code to IBM's quantum systems. This hands-on approach helped me understand how quantum circuits are executed on real hardware.

- **Building Circuits Online:** Using the drag-and-drop interface, I designed simple quantum circuits to experiment with gates like Hadamard, X, and CNOT.
- **Running Circuits:** I executed basic quantum algorithms, first on simulators and then on real quantum hardware, observing the differences in results due to noise and decoherence.



Hello World Program:

```

+ Code + Markdown | ▶ Run All ↺ Restart ≡ Clear All Outputs | 📄 Variables 📖 Outline ...
+ Code + Markdown

from qiskit import QuantumCircuit
from qiskit.quantum_info import SparsePauliOp
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager
from qiskit_ibm_runtime import EstimatorV2 as Estimator

# Create a new circuit with two qubits
qc = QuantumCircuit(2)

# Hadamard gate to qubit 0
qc.h(0)
# controlled-X gate on qubit 1, controlled by qubit 0
qc.cx(0, 1)

qc.draw("mpl")

[4] ✓ 0.1s

```

The screenshot shows a Jupyter Notebook interface with a code editor and a quantum circuit diagram. The code defines a QuantumCircuit with two qubits, applies a Hadamard gate to qubit 0, and then a controlled-X gate from qubit 0 to qubit 1. The circuit is drawn using the 'mpl' backend. Below the code, there's a quantum circuit diagram showing two qubits, q0 and q1. q0 has a Hadamard gate (H) and is then connected to q1 via a controlled-X gate (CX).

3.3: Week 3: Exploring Grover's Search Algorithm

Introduction to Grover's Algorithm

Grover's algorithm is a quantum algorithm designed to search through an unordered set more efficiently than classical methods.

Unlike classical approaches, which may require examining every item, Grover's algorithm reduces the number of operations by achieving a quadratic speedup, solving the problem in roughly the square root of the number of operations.

It uses quantum superposition and amplitude amplification to find the target item faster. While current quantum technology is not yet fully able to exploit this advantage, Grover's algorithm holds great potential for solving complex search problems more efficiently as quantum technology advances.

Understanding Unstructured Search

Unstructured search refers to finding a specific input (target) where a function returns 1, while it returns 0 for all others.

In **classical computing**, identifying the target typically requires evaluating all possible inputs one by one. For a space of 2^n inputs, this involves up to 2^n evaluations in the worst case, which becomes inefficient for large datasets.

Grover's algorithm, however, uses quantum principles like superposition to evaluate multiple possibilities at once. Through amplitude amplification, the probability of the correct input is increased, making the search significantly faster. This approach reduces the number of evaluations to approximately $(2^n)^{\frac{1}{2}}$ providing a substantial speedup over classical methods.

Mechanics of Grover's Algorithm

- Purpose: Efficiently find a target in an unordered set, reducing evaluations from 2^n (classical) to $(2^n)^{\frac{1}{2}}$ (quantum), offering a quadratic speedup.
- Mechanism:
 1. Initialization: Creates a quantum superposition of all possible states.

2. Amplitude Amplification: Boosts the target state's probability using quantum operations.
 3. Repetition: Iterates the process to maximize the probability of measuring the target state.
- Outcome: The target state becomes the most likely outcome, drastically speeding up the search for large datasets.

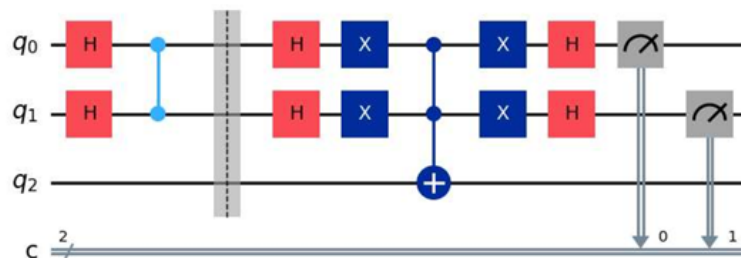
3.4: Week 4: Searching for '11' State (2-Qubit System)

- **Objective:** Implement Grover's algorithm to search for the '11' state in a 2-qubit system.
- **Steps Taken:**
 1. Set up the quantum circuit for a 3-qubit system.
 2. Implemented Grover's algorithm with the oracle designed to mark the '11' state.
 3. Applied amplitude amplification to increase the probability of the target state.
 4. Ran simulations using Qiskit to observe the behaviour of the quantum system.

```
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator
from qiskit import transpile
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt
```

```
def oracle(qc, q):
    # Apply controlled-Z (CZ) gate to mark the '11' state
    qc.cz(q[0], q[1])
    qc.barrier()
```

```
qc.draw('mpl')
```



```
# Define the inversion about the mean (IAM) operator
def iam(qc, q):
    qc.h(q[0])
    qc.h(q[1])
    qc.x(q[0])
    qc.x(q[1])
    qc.ccx(q[0], q[1], q[2]) # Use a different qubit (q[2]) as the target
    qc.x(q[0])
    qc.x(q[1])
    qc.h(q[0])
    qc.h(q[1])
```

```
# Create a quantum circuit with 2 qubits and 2 classical bits
qc = QuantumCircuit(3, 2) # 3 qubits, 2 classical bits
```

```
# Initialize the qubits in a superposition state
qc.h(0)
qc.h(1)
```

```
# Apply the oracle function
oracle(qc, [0, 1])
```

```
# Apply the IAM operator
iam(qc, [0, 1, 2])
# Measure the qubits
qc.measure([0, 1], [0, 1])
```

```

backend = AerSimulator()
qc_compiled = transpile(qc, backend)
job = backend.run(qc_compiled, shots = 1024)
result = job.result()
counts = result.get_counts(qc)
print(counts)

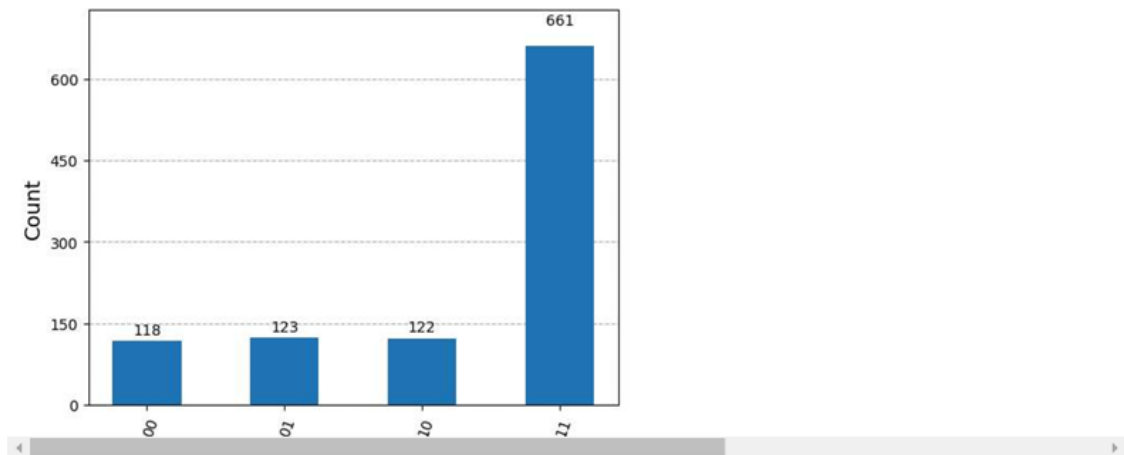
# Get the measurement counts
# counts = result.get_counts(qc)
# print("Counts:", counts) # Print the number of occurrences for each state

# Plot the counts
plot_histogram(counts)
plt.show() # Get the measurement counts
counts = result.get_counts(qc)
print("Counts:", counts) # Print the number of occurrences for each state

# Plot the counts
plot_histogram(counts)
# plt.show()

In [ ]: {'11': 661, '10': 122, '01': 123, '00': 118}
Counts: {'11': 661, '10': 122, '01': 123, '00': 118}

```



- **Results:**

- The graph plotted shows the state probabilities, with the highest probability at the '11' state.
- This confirms that the algorithm successfully amplifies the correct state.

3.5: Week 5: Finding the '101' State in a 3-Qubit System

I implemented Grover's algorithm to locate the '101' state in a 3-qubit system.

1. **Environment Setup:** Qiskit and NumPy were utilized to build quantum circuits and handle calculations.
2. **Oracle Definition:** The oracle was constructed to mark the '101' state as the target using controlled gates.
3. **Circuit Initialization:** A quantum circuit with 5 qubits was initialized. The first 3 qubits were put into superposition using Hadamard gates.
4. **Diffuser Creation:** The diffuser, responsible for amplitude amplification, was implemented to boost the probability of the target state.
5. **Grover Iterations:** The oracle and diffuser were applied iteratively to enhance the likelihood of measuring the '101' state.
6. **Measurement:** The first 3 qubits were measured to determine the system's final state.
7. **Simulation Execution:** The circuit was run on a quantum simulator, and results were analyzed through a histogram.

```
from qiskit.quantum_info import Operator
from qiskit import QuantumCircuit
import numpy as np

def phase_oracle(n, name = 'Uf'):
    qc = QuantumCircuit(n, name = name)
    qc.x(1)
    qc.ccx(0,1,3)
    qc.ccx(2,3,4)
    qc.ccx(0,1,3)
    qc.x(1)

    return qc

n = 5
qc = QuantumCircuit(n)
for i in range(n-2):
    qc.x(i)
    qc.ccx(0,1,3)
    qc.ccx(2,3,4)
    qc.ccx(0,1,3)
for i in range(n-2):
    qc.x(i)

def diffuser(n, name = 'V'):
    qc = QuantumCircuit(n, name = name)

    for qb in range(n-2):
        qc.h(qb)

    for i in range(n-2):
        qc.x(i)
        qc.ccx(0,1,3)
        qc.ccx(2,3,4)
        qc.ccx(0,1,3)
    for i in range(n-2):
        qc.x(i)
    for qb in range(n-2):
        qc.h(qb)
    return qc

n = 5
qc = QuantumCircuit(n, n-2)
nsol = 1

r = int(np.floor(np.pi/4*np.sqrt(2**(n-2)/nsol)))
```

8. **Result:** The '101' state appeared with the highest frequency, confirming successful implementation of Grover's algorithm.

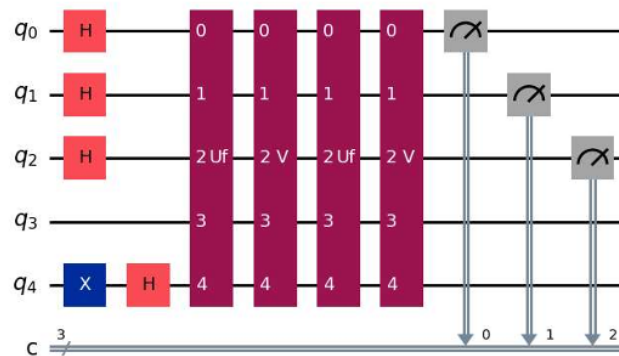
```
qc.h(range(n-2))

qc.x(n-1)
qc.h(n-1)

for i in range(r):
    qc.append(phase_oracle(n), range(n))
    qc.append(diffuser(n), range(n))

qc.measure(range(n-2), range(n-2))
qc.draw('mpl')
```

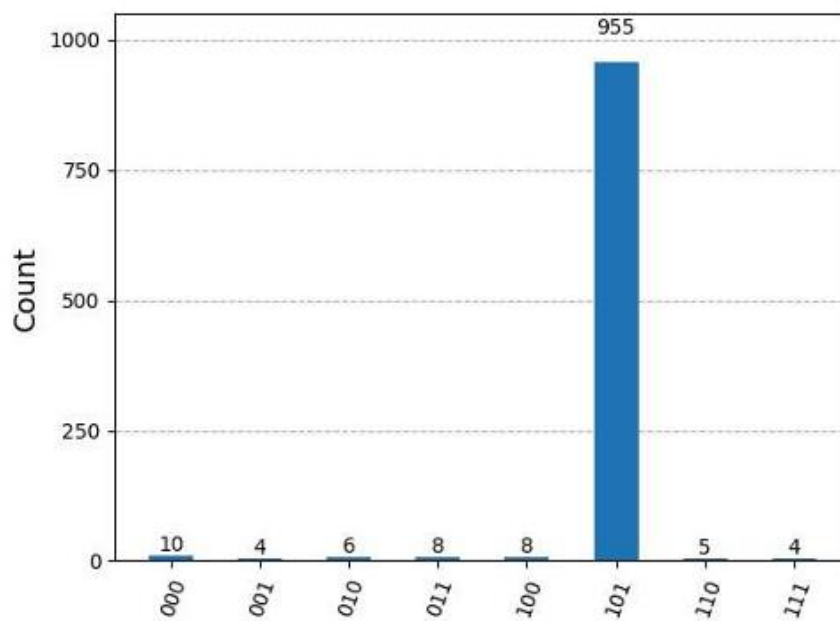
2



```
from qiskit_aer import AerSimulator
from qiskit import transpile
from qiskit.visualization import plot_histogram

backend = AerSimulator()
gr_compiled = transpile(qc, backend)
job_sim = backend.run(gr_compiled, shots = 1000)
# simulator = Aer.get_backend('qasm_simulator')
# job = execute(gr, backend = simulator, shots = 1000)
counts = job_sim.result().get_counts()
print(counts)
plot_histogram(counts)
```

{'101': 955, '000': 10, '001': 4, '010': 6, '011': 8, '100': 8, '110': 5, '111': 4, '010': 6}



3.6: Week 6: Maximum Search Algorithm: Theory and Understanding

This week was dedicated to studying the theoretical aspects of a quantum algorithm designed to find the maximum element in an unsorted dataset. By adapting Grover's algorithm, this method iteratively improves guesses about the maximum value, offering a significant efficiency boost over classical approaches.

Problem Overview

The task is to determine the index of the maximum value in an unsorted array $T[0 \dots N-1]$, containing N distinct elements.

- **Classical Approach:**

A conventional algorithm compares each element, requiring $O(N^2)$ steps in the worst case.

- **Quantum Advantage:**

The quantum algorithm reduces this complexity to $O((2^n)^{\frac{1}{2}})$ by exploiting quantum principles like superposition and amplitude amplification.

Core Principles of the Algorithm

The algorithm operates by iteratively refining a guess y , which represents the index of the current maximum candidate. It uses Grover's search algorithm to identify indices of elements greater than the current guess, eventually converging on the maximum value.

1. **Oracle Function:**

- The oracle marks elements that are larger than the current guess $T[y]$.
- This is achieved by setting $f_y(x) = 1$ if $T[x] > T[y]$ and $f_y(x) = 0$ otherwise.

2. **Superposition and Search:**

- The quantum state is initialized to represent all indices in superposition.
- Grover's search algorithm is applied to amplify the probabilities of indices satisfying the oracle condition ($T[x] > T[y]$).

3. Measurement and Update:

- After amplification, a measurement collapses the quantum state to a new index x_0 which becomes the updated guess y .
- This process is repeated, refining the guess at each step.

4. Termination:

- The algorithm continues for approximately *square root* N iterations, ensuring the final guess corresponds to the maximum element with high probability.

3.7: Week 7: Implementation of Maximum Search Algorithm

This week focused on translating the theoretical understanding of the maximum search algorithm into a practical implementation. The algorithm was developed step-by-step to efficiently find the index of the maximum element in an unsorted dataset using quantum principles.

Detailed Steps of the Algorithm

1. Initialization:

- A random index y was selected as the starting guess for the maximum value.
- The quantum state was initialized to represent all possible indices in a superposition, with y as the current guess.

2. Oracle Application:

- An oracle was constructed to identify indices x where the value $T[x]$ was greater than the value at the current guess $T[y]$. These indices were marked as candidates for the next iteration.

3. Amplitude Amplification:

- Using Grover's diffusion operator, the probabilities of the marked indices were amplified, increasing the chances of selecting one of these indices in the next step.

4. Measurement:

- The quantum state was measured to collapse into a single index x_0 representing a new and improved guess for the maximum element.

5. Iteration:

- The process was repeated, updating the guess y to the new value x_0 after each measurement. With each iteration, the algorithm focused more closely on the largest element in the dataset.

6. Output:

- After repeating the process for approximately the square root of the dataset size, the final index y was returned as the most probable index of the maximum element.

```

from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator
import numpy as np

# Function to create the Oracle for Grover's algorithm
def create_oracle(qc, dataset, guess_element, num_qubits):
    for i in range(len(dataset)):
        if dataset[i] > guess_element:
            binary_index = format(i, f'0{num_qubits}b')

            # Apply X gates to the qubits where the binary index has '0'
            for qubit in range(num_qubits):
                if binary_index[qubit] == '0':
                    qc.x(qubit)

            # Apply the multi-controlled X gate (oracle)
            qc.mcx(list(range(num_qubits - 1)), num_qubits - 1)

            # Reapply the X gates to the qubits to restore their state
            for qubit in range(num_qubits):
                if binary_index[qubit] == '0':
                    qc.x(qubit)

    return qc

# Function to apply the Grover Diffusion Operator
def diffusion_operator(qc, num_qubits):
    # Apply Hadamard gates to all qubits
    qc.h(range(num_qubits))

    # Apply X gates to all qubits
    qc.x(range(num_qubits))

    # Apply Hadamard gate to the last qubit
    qc.h(num_qubits - 1)

    # Apply multi-controlled X gate (diffusion operator)
    qc.mcx(list(range(num_qubits - 1)), num_qubits - 1)

    # Apply Hadamard gate to the last qubit again
    qc.h(num_qubits - 1)

    # Reapply X gates to all qubits
    qc.x(range(num_qubits))

    # Apply Hadamard gates to all qubits again
    qc.h(range(num_qubits))

    return qc

# Function to construct the Grover's algorithm circuit
def grover_algorithm(dataset, guess_element):
    num_qubits = int(np.ceil(np.log2(len(dataset))))

    qc = QuantumCircuit(num_qubits, num_qubits)

    # Apply Hadamard gates to create superposition
    qc.h(range(num_qubits))

    # Apply the Oracle
    qc = create_oracle(qc, dataset, guess_element, num_qubits)

    # Apply the Grover Diffusion Operator
    qc = diffusion_operator(qc, num_qubits)

    # Measure the qubits
    qc.measure(range(num_qubits), range(num_qubits))

    return qc

# Function to execute the quantum circuit
def execute_circuit(qc):
    backend = AerSimulator()
    compiled_qc = transpile(qc, backend)
    result = backend.run(compiled_qc, shots=1000).result()
    counts = result.get_counts(qc)

```

```

# Function to run Grover's algorithm and get the results
def execute_grover(qc):
    counts = execute_circuit(qc)
    return counts

# Example dataset
dataset = [3, 1, 2, 80, 6, 70, 4]
n = len(dataset)
remaining_indices = list(range(n))

while remaining_indices:
    # Randomly select an index from the remaining indices
    random_index = np.random.choice(remaining_indices)
    guess_element = dataset[random_index]

    # Apply Grover's algorithm to find indices with elements greater than the guess_element
    qc = grover_algorithm(dataset, guess_element)

    counts = execute_grover(qc)

    print(f"Iteration with guess element {guess_element}: {counts}")


    new_remaining_indices = [int(index, 2) for index in counts if counts[index] > 0]

    # Filter out indices where the dataset value is not greater than the guess_element
    remaining_indices = [index for index in new_remaining_indices if index < len(dataset) and dataset[index] > guess_element]

    # If only one index remains, it's the maximum element
    if len(remaining_indices) == 1:
        max_index = remaining_indices[0]
        break

print("\n")
# Print the final result
print(f"The index of the maximum element is: {max_index}")
print(f"The maximum element is: {dataset[max_index]}")


```

 Iteration with guess element 4: {'011': 116, '000': 134, '010': 124, '110': 128, '100': 146, '101': 117, '111': 115, '001': 120}
 Iteration with guess element 80: {'000': 136, '101': 115, '100': 118, '110': 135, '011': 119, '001': 127, '010': 129, '111': 121}

The index of the maximum element is: 3
 The maximum element is: 80

3.8: Week 8: Finalizing the Project and Submission

The last week was focused on completing the project, refining the implementation, and preparing the final report for submission.

1. Improving the Algorithm

- Ran additional tests to validate the algorithm's performance on different datasets.
- Optimized parameters like the number of iterations to ensure accurate results.

2. Analysing Results

- Confirmed that the simulation outputs matched the theoretical predictions, with the maximum element being identified correctly.
- Finalized histograms to clearly show the amplified probability result.

3. Project Review

- Checked the implementation for consistency and resolved any remaining issues.
- Explored potential scalability and challenges for real-world quantum hardware applications.

4. Compiling the Report

- Organized all sections: introduction, theory, implementation, results, and conclusions.
- Included clear graphs and well-commented code snippets to support the explanations.

5. Final Submission

- Ensured all files, including the report, code, and visuals, were complete and ready for submission.

This week concluded with a polished project showcasing the successful implementation and analysis of the maximum search algorithm using Grover's quantum approach.

Chapter 4: Future Scope

This project demonstrated how quantum computing can efficiently solve the maximum-finding problem. While the algorithm worked well on small datasets, there are several areas where it can be improved and extended in the future:

1. Handling Larger Datasets:

Currently, the algorithm was tested on small datasets using a simulator. In the future, it can be scaled to handle much larger datasets. This will help explore its performance in real-world scenarios and test its limits as quantum hardware becomes more advanced.

2. Better Oracle Design:

The oracle is a crucial part of the algorithm that identifies elements larger than the current guess. Future work can focus on designing more efficient oracles that can work with complex datasets, making the algorithm suitable for solving more diverse problems.

3. Testing on Real Quantum Hardware:

This project used simulations, but future work could test the algorithm on actual quantum computers, like those provided by IBM or Google. This would provide insights into how the algorithm behaves on physical devices and help refine its implementation.

4. Combining with Classical Computing:

A hybrid approach can combine the strengths of classical and quantum computing. Classical methods can preprocess the data to reduce its size, and the quantum algorithm can then focus on finding the maximum efficiently. This can make the algorithm more practical for current systems.

5. Applications in Real Problems:

This algorithm has potential applications in many fields. For example:

- **Finance:** Finding the best investment option from a large set of possibilities.

- **Logistics:** Optimizing delivery routes or schedules.
- **Machine Learning:** Improving algorithms by selecting the best features or hyperparameters.

6. **Refining the Algorithm:**

Improvements can be made to reduce the number of iterations required or make the algorithm more efficient with the resources it uses, such as the number of qubits or gates. This will make it more practical for use with current quantum hardware.

Chapter 5: Conclusion

During my summer internship at DRDO, I had the privilege of immersing myself in the dynamic and groundbreaking field of quantum computing. This journey has been both intellectually stimulating and practically enriching, offering me valuable insights into how quantum technology can revolutionize computing.

5.1 Quantum Computing in a Nutshell

Quantum computing harnesses the principles of quantum mechanics, such as superposition and entanglement, to perform computations in ways that classical computers cannot. With quantum bits (qubits) capable of existing in multiple states at once, quantum computers hold the promise of solving certain problems exponentially faster than traditional methods.

5.2 Grover's Algorithm

One of the most exciting aspects of my internship was working with Grover's algorithm. This quantum algorithm significantly accelerates the search through large, unsorted datasets. Unlike classical search methods that might require exhaustive checks, Grover's algorithm utilizes quantum superposition and amplitude amplification to quickly identify the desired item, showcasing the potential efficiency gains of quantum computing.

5.3 My Experience

The hands-on experience with Grover's algorithm was both challenging and rewarding. I implemented the algorithm to solve practical problems, such as finding specific quantum states and locating the maximum element in an unsorted dataset. This practical application allowed me to witness firsthand how quantum algorithms can outperform classical techniques, offering faster and more efficient solutions.

Throughout the internship, I also had the opportunity to collaborate with experts in the field, participate in in-depth discussions, and tackle real-world problems. The experience not only deepened my understanding of quantum computing concepts but also honed my skills in quantum programming using tools like Qiskit.

5.4 Key Takeaways

- Quantum computing has the potential to transform various industries by offering solutions to complex problems that are currently infeasible for classical computers.

- Grover's algorithm exemplifies the advantages of quantum computing in search and optimization tasks, highlighting its potential to outperform traditional methods.
- The internship experience has been invaluable, providing me with practical skills, real-world applications, and a deeper appreciation for the future of quantum technology. Overall, my time at DRDO has been an enlightening journey, equipping me with the knowledge and experience to contribute to the rapidly evolving field of quantum computing.

Chapter 6: References

1. A Fast Quantum Mechanical Algorithm for Database Search, Lov K. Grover.
2. A Quantum Algorithm for Finding the Maximum, Ashish Ahuja and Sanjiv Kapoor.
3. Quantum Computation and Quantum Information, Michael A. Nielsen and Isaac L. Chuang.
4. Quantum Computing for Computer Scientists, Noson S. Yanofsky and Mirco A. Mannucci.
5. Grover's Algorithm: Implementations and Implications, [Research paper].
6. Grover's Algorithm: Fundamentals and Implementations, IBM. Retrieved from <https://learning.quantum.ibm.com/course/fundamentals-of-quantum-algorithms/grovers-algorithm>.
7. Quantum Computing Overview, IBM. Retrieved from <https://www.ibm.com/topics/quantum-computing>

