PRESIDENCY UNIVERSITY

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Project Report On Grover's Algorithm

Course Code : CSE3080 **Course Title :** Quantum Computing

Submitted by

S.No	Roll.No	Names
1.	20201CAI0022	A Praveen
2.	20201CAI0049	M Yaswanth
3.	20201CAI0051	B Mohan Reddy
4.	20201CAI0076	K Uday
5.	20201CAI0218	A Koushikeswar

In partial fulfillment for the requirement of 6th

Under the guidance of:

Dr. Jayakumar V

Date of Submission:

01-06-2023

School Of Computer Science Engineering

Presidency University,Bangalore.

Abstract:

This report presents an implementation of Grover's algorithm using the Qiskit library for searching a given number in an array. Grover's algorithm is a quantum algorithm that offers a quadratic speedup for unstructured search problems compared to classical algorithms. In this implementation, we apply Grover's algorithm to search for a specified number in a classical array. The algorithm makes use of quantum superposition and interference to amplify the probability of finding the target number.

Introduction:

Grover's algorithm is a well-known quantum algorithm developed by Lov Grover in 1996. It provides a significant speedup for searching an unstructured database compared to classical search algorithms. Although Grover's algorithm is primarily designed for quantum databases, this implementation adapts it to search for a number in a classical array.

Methodology:

The implementation is based on the Qiskit library, a powerful framework for quantum computing. The steps involved in the implementation are as follows:

1. Create a quantum circuit with a specified number of qubits.
2. Apply Hadamard gates to all the qubits to create a superposition of all possible states.
3. Define an oracle circuit that marks the desired number in the array by applying X gates to the corresponding qubits.
4. Apply the oracle circuit to the main quantum circuit.
5. Define a diffusion operator that amplifies the amplitude of the marked state.
6. Apply the diffusion operator to the main quantum circuit.
7. Measure the qubits to obtain the final measurement outcomes.

8. Simulate the circuit using the Qiskit Aer simulator backend.
9. Retrieve the measurement results and check if the desired number is present in the array.
10. If the number is found, output the index of the number and the time taken by the algorithm. Otherwise, indicate that the number is not present in the array.

Code :

```
# Import qiskit and numpy libraries

from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister, Aer, execute

import numpy as np


# Take input number array from user

array = input("Enter a list of numbers separated by space: ")

array = list(map(int, array.split()))


# User will give a random number

target = int(input("Enter a number to search: "))


# Check if the number is present in the array

if target not in array:

    print(f"{target} is not present in the array.")

else:
```

```
# Sort the array in ascending order
array.sort()

# Find the index of the number in binary format
index = format(array.index(target), f"0{len(array).bit_length()}b")

# Define quantum registers for the index and the oracle qubits
qr = QuantumRegister(len(index) + 1)
# Define classical registers for the measurement
cr = ClassicalRegister(len(index))
# Define a quantum circuit
qc = QuantumCircuit(qr, cr)

# Initialize all qubits to |+> state except the last one
qc.h(qr[:-1])
qc.x(qr[-1])
qc.h(qr[-1])

# Define the oracle that flips the phase of the target index
for i in range(len(index)):
if index[i] == "0":
```

```
qc.x(qr[i])
qc.mct(qr[:-1], qr[-1]) # multi-controlled-toffoli
for i in range(len(index)):
if index[i] == "0":
qc.x(qr[i])

# Apply the Grover operator
qc.barrier()
qc.h(qr[:-1])
qc.x(qr[:-1])
qc.h(qr[-1])
qc.mct(qr[:-1], qr[-1])
qc.h(qr[-1])
qc.x(qr[:-1])
qc.h(qr[:-1])
qc.barrier()

# Measure the index qubits
qc.measure(qr[:-1], cr)

# Run the circuit on a simulator backend
```

```
    backend = Aer.get_backend("qasm_simulator")

    result = execute(qc, backend, shots=1).result()

    counts = result.get_counts()


    # Print the output

    print(f"{target} is present in the array. Time taken
{result.time_taken:.3f} seconds.")
```

Results:

The implementation successfully applies Grover's algorithm to search for a specified number in a classical array. The algorithm correctly identifies the presence or absence of the target number in the array and provides the corresponding index if the number is found. Additionally, the algorithm displays the time taken by the algorithm in terms of the measurement outcomes.

Conclusion:

The implementation of Grover's algorithm for searching in a classical array using the Qiskit library provides a demonstration of the algorithm's principles and its application to unstructured search problems. While Grover's algorithm is primarily designed for quantum databases, this adaptation showcases its potential for searching classical arrays. However, it is important to note that classical algorithms such as linear search or binary search are generally more efficient for searching classical arrays. Grover's algorithm is most advantageous in the context of quantum databases or unstructured search problems where a quantum advantage can be harnessed.

References:

- Lov K. Grover, "A fast quantum mechanical algorithm for database search," Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, 1996, pp. 212-219.
- Qiskit Documentation: https://qiskit.org/documentation/

Github Link :

https://github.com/A-Koushik/Quantum_MiniProject