CSBB 311 : QUANTUM COMPUTING

LAB ASSIGNMENT 6 : Grover’s Search with an unknown number of solutions

Submitted By:

Name: KARTIK MITTAL

Roll No: 221210056

Branch: CSE

Semester: 5th Sem

Group: 2

Submitted To: Dr. VS Pandey

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY DELHI

2024

Theory -

1. Introduction to Grover's Search Algorithm

- Grover's Search Algorithm is a quantum algorithm that efficiently searches an unsorted database or solves black-box search problems. It provides a quadratic speedup over classical algorithms, reducing the number of queries required to find a solution.

2. Grover’s Search with an Unknown Number of Solutions

- The central idea remains the same: Grover’s algorithm uses quantum parallelism to evaluate multiple possibilities at once, and then iteratively amplifies the amplitude of the correct solutions. The search process is repeated O(sqrt(N/M)) times, where M is the number of solutions in the database, providing an efficient way to locate all solutions.

3. Workflow of Grover's Search Algorithm

- **Quantum Circuit Initialization**:Initialize a superposition state over all possible database entries using Hadamard gates, creating an equal amplitude state.
- **Oracle Query**: Apply the oracle function, which marks the correct solutions by flipping their amplitudes. In this case, the oracle can mark multiple solutions but will flip the phase of each one.
- **Amplitude Amplification**: The diffusion operator acts by inverting the amplitude of each state about the average amplitude of all states.
- **Measurement**: After sufficient iterations, measure the state of the quantum register. The measurement will yield one of the solutions with high probability, and further measurements can be made to find additional solutions.

4. Importance of Grover’s Algorithm

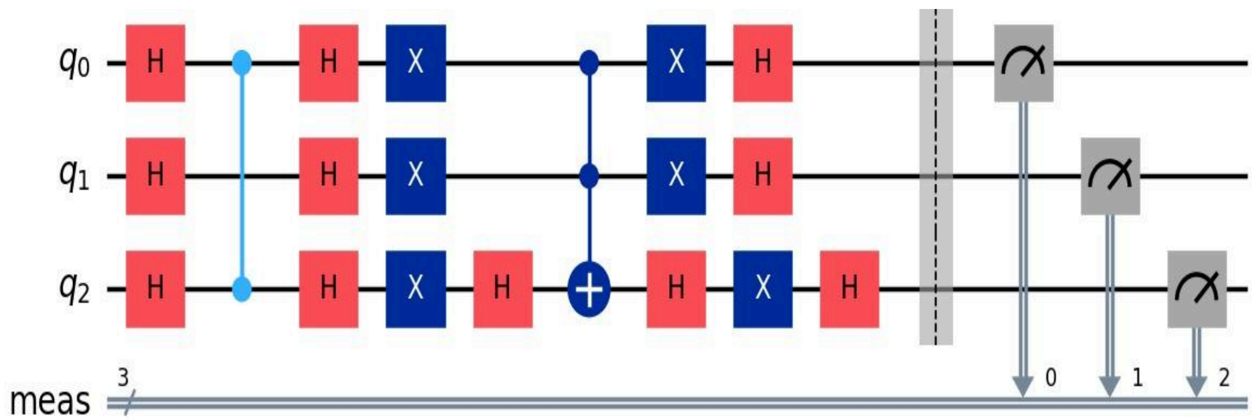
- Grover's algorithm with an unknown number of solutions is crucial for efficiently solving search problems in situations where classical methods would require a linear number of queries.

Code (Grover Search) -

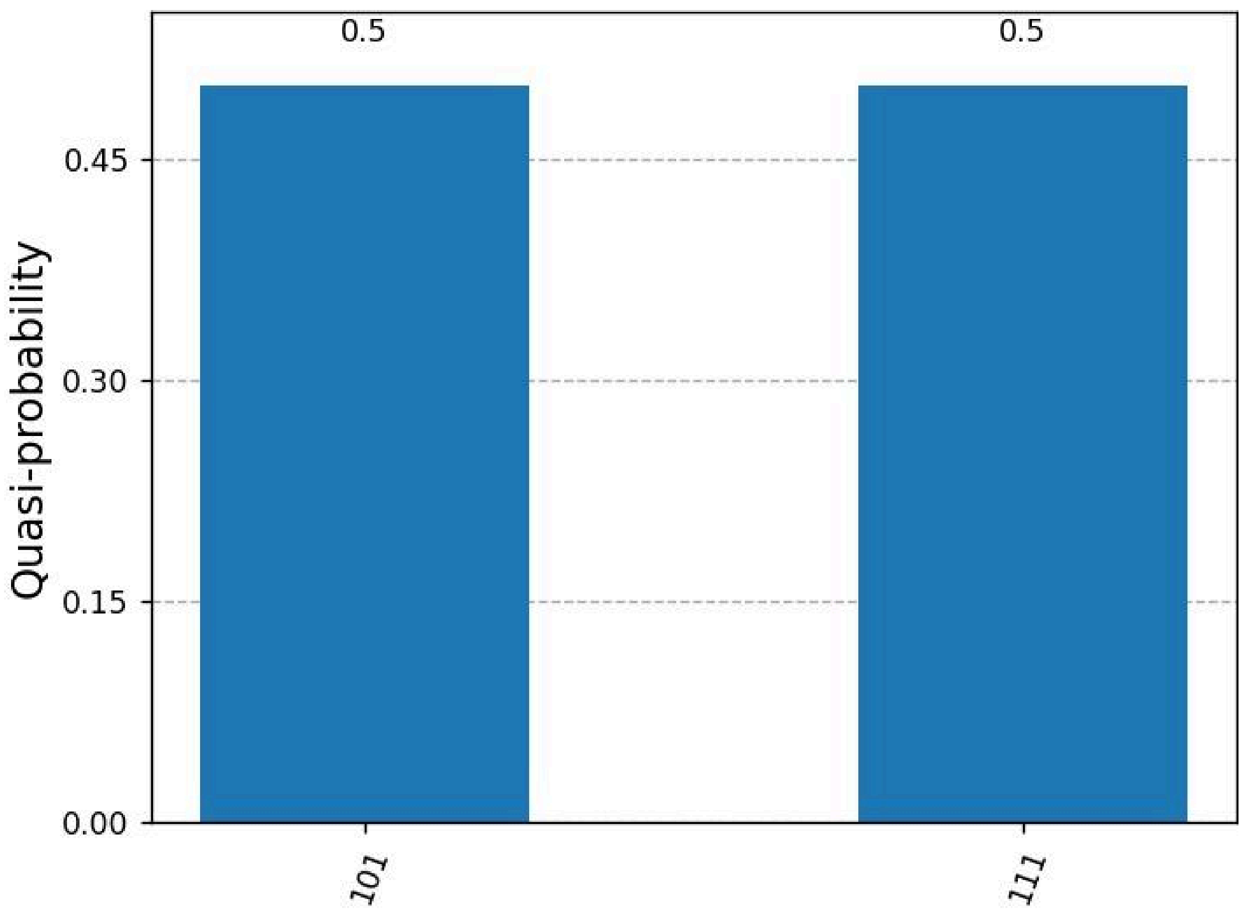
```
from qiskit import QuantumCircuit , execute
from qiskit_aer  import Aer
from qiskit.visualization import plot_histogram, circuit_drawer
from qiskit.circuit.library import MCXGate
import matplotlib.pyplot as plt

# Create a 3-qubit quantum circuit for Grover's search
n = 3  # Number of qubits
qc = QuantumCircuit(n)

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

# Example oracle for marking the state |101>
qc.cz(0, 2)

# Apply the diffusion operator (inversion about the mean)
qc.h(range(n))
qc.x(range(n))
qc.h(n - 1)

# Add a multi-controlled Toffoli gate using MCXGate
mct_gate = MCXGate(num_ctrl_qubits=n-1)  # Create an MCX gate with (n-1) control qubits
qc.append(mct_gate, range(n))  # Append the gate to the circuit

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

```
# Measure the qubits
qc.measure_all()

# Visualize the quantum circuit
circuit_diagram = circuit_drawer(qc, output='mpl')
plt.show()  # Display the circuit diagram

# Use Aer simulator to simulate and get results
simulator = Aer.get_backend('qasm_simulator')
job = execute(qc, backend=simulator, shots=1024)
result = job.result()
counts = result.get_counts()

# Plot and visualize the result
plot_histogram(counts)
plt.show()
```

Output -



q_0
q_1
q_2
meas
3
H
X
0
1
2

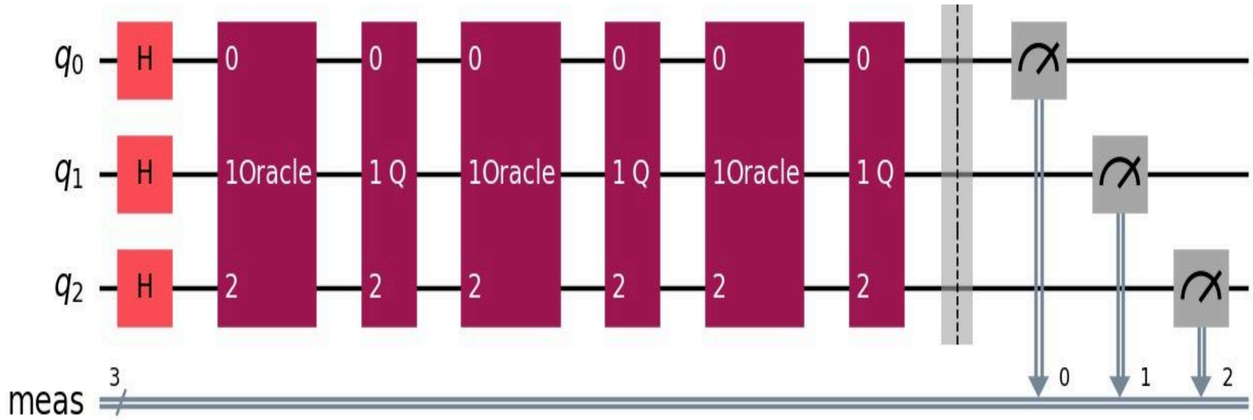


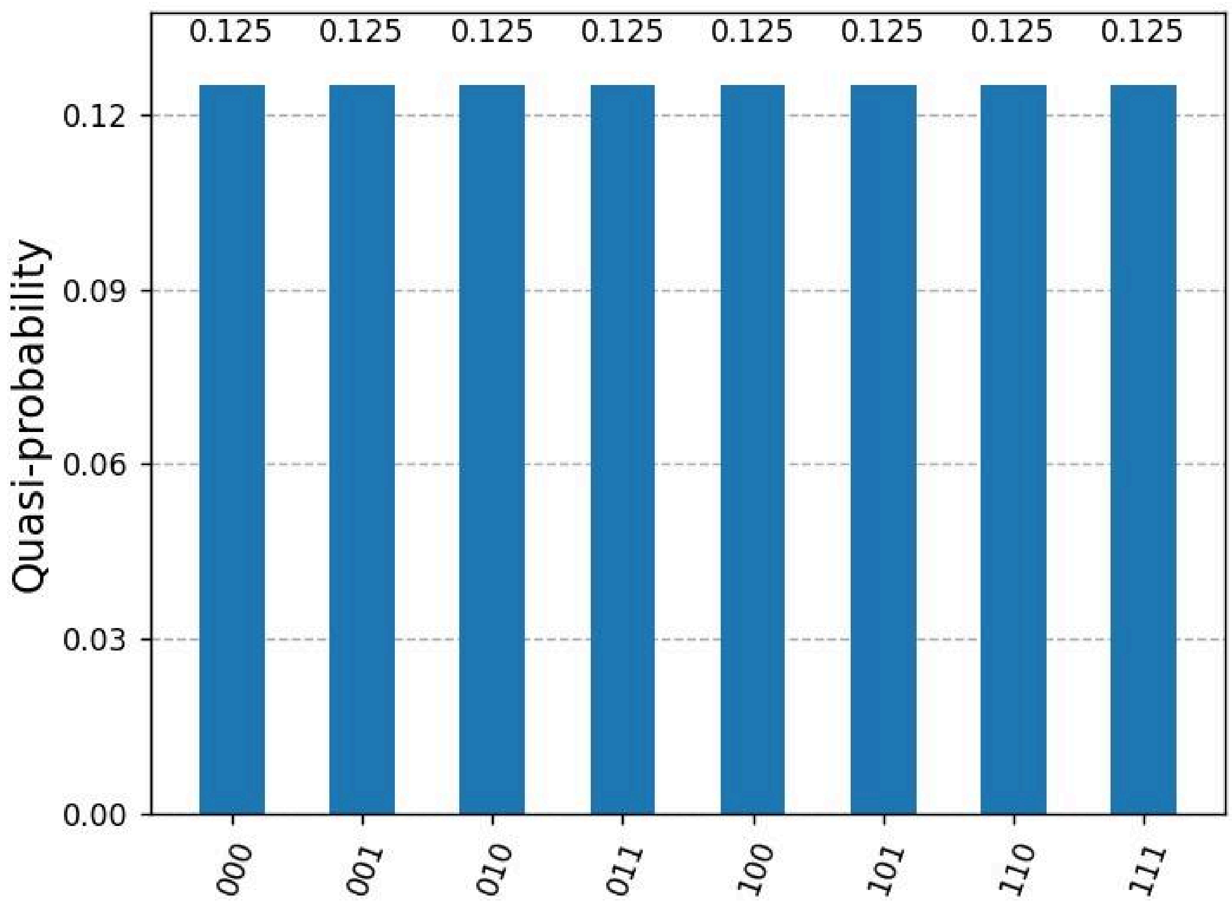
0.5
0.5
Quasi-probability
0.45
0.30
0.15
0.00
101
111

Code -

```
from qiskit import QuantumCircuit, transpile
from qiskit_aer import Aer
from qiskit.primitives import Sampler
from qiskit.circuit.library import GroverOperator
from qiskit.visualization import plot_histogram, circuit_drawer
import matplotlib.pyplot as plt

# Define a custom oracle for multiple solutions
def custom_oracle(n):
    oracle = QuantumCircuit(n)
    # Mark states |001> and |110> as solutions
    oracle.cz(0, 2)
    oracle.cz(1, 2)
    oracle.name = "Oracle"
    return oracle

# Set up the circuit for 3 qubits
n = 3
oracle = custom_oracle(n)

# Create the Grover diffusion operator
grover_operator = GroverOperator(oracle)

# Initialize Grover's search circuit
iterations = 3  # Number of iterations for Grover's algorithm
qc = QuantumCircuit(n)
qc.h(range(n))  # Initial Hadamard gates for superposition

# Apply Grover iterations
for _ in range(iterations):
    qc.append(oracle, range(n))
    qc.append(grover_operator, range(n))

# Measure all qubits
qc.measure_all()

# Visualize the quantum circuit
circuit_diagram = circuit_drawer(qc, output='mpl')
plt.show()

# Use Sampler to simulate and get results
sampler = Sampler()
backend = Aer.get_backend('aer_simulator')
transpiled_qc = transpile(qc, backend)
result = sampler.run(transpiled_qc).result()
counts = result.quasi_dists[0].binary_probabilities()

# Display the results
plot_histogram(counts)
plt.show()
```

Output -


q_0
q_1
q_2
meas
H
Oracle
Q
3


Quasi-probability
0.125
0.00
0.03
0.06
0.09
0.12
000
001
010
011
100
101
110
111

Conclusion -

- **Efficiency-Scalability Trade-off**: Grover's Search Algorithm offers a trade-off between efficiency and scalability when applied to problems with an unknown number of solutions. As the number of solutions increases, the number of required iterations grows, which impacts the quantum resources needed for execution.
- **Algorithmic Significance**:Grover’s Search Algorithm is a cornerstone in quantum computing, showcasing the power of quantum parallelism for solving search problems in unsorted databases.