Name: Gudur Krishna Chaitanya Roll no: 102117049 email used:
gchaitanya_be21@thapar.edu

Grover's Algorithm:

Problem: The classical problem of searching an unsorted database with N items typically requires O(N) queries. Grover's algorithm aims to perform this search with only O(N*0.5) queries, providing a quadratic speedup. Quantum Oracle: Grover's algorithm uses a quantum oracle, which is a black box function that marks the desired solution(s). The oracle reflects the input states corresponding to the marked solution(s) about the average state. Amplitude Amplification: The core of Grover's algorithm is amplitude amplification, a technique that increases the amplitude of the marked states and decreases the amplitude of the unmarked states. This amplification process is performed iteratively to enhance the probability of measuring a marked state.

```
from qiskit_ibm_provider import IBMProvider

provider = IBMProvider(token='f55702335547d565b44eb80fd6708f3b82b4d0147236062

active_account = provider.active_account()

print("Active Account Details:")

print(active_account)
```

```
<ipython-input-2-76aa4e895954>:1: DeprecationWarning: The package qiskit_ib
  from qiskit_ibm_provider import IBMProvider
Active Account Details:
{'channel': 'ibm_quantum', 'token': 'f55702335547d565b44eb80fd6708f3b82b4d0
```

```
import matplotlib.pyplot as plt
import numpy as np
import math
from qiskit import transpile
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
import qiskit_aer
from qiskit.visualization import plot_histogram
```

```
n = 2
grover_circuit = QuantumCircuit(n)
```

```
def initialize_s(qc, qubits):
    for q in qubits:
        qc.h(q)
    return qc
```

```
grover_circuit = initialize_s(grover_circuit, [0,1])
grover_circuit.draw(output='text',style='bw')
```

```
     ┌───┐
q_0: ┤ H ├
     ├───┤
q_1: ┤ H ├
     └───┘
```

```
grover_circuit.cz(0,1)
grover_circuit.draw(output='text',style='bw')
```

```
     ┌───┐   
q_0: ┤ H ├─■─
     ├───┤ │ 
q_1: ┤ H ├─■─
     └───┘   
```

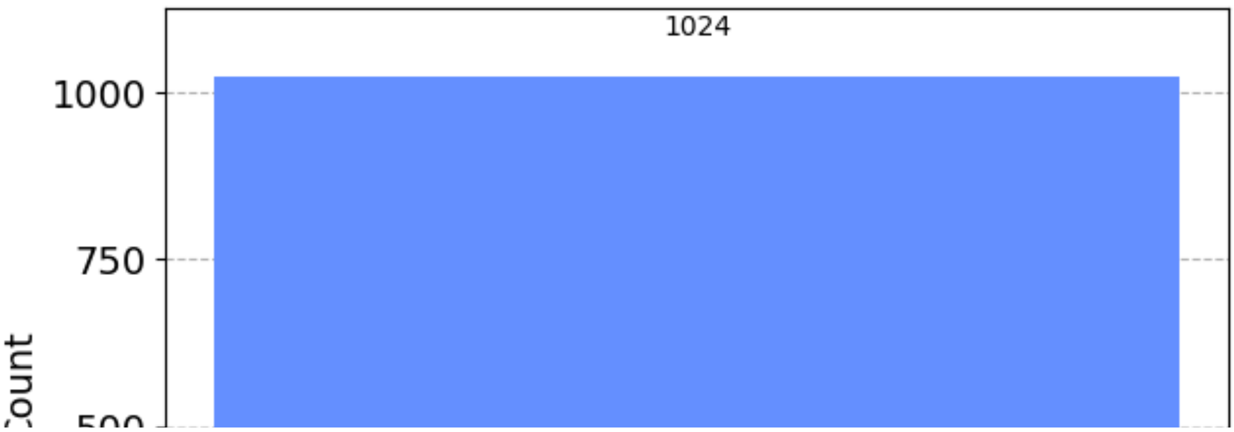
```
grover_circuit.h([0,1])
grover_circuit.z([0,1])
grover_circuit.cz(0,1)
grover_circuit.h([0,1])
grover_circuit.draw(output='text',style='bw')
```

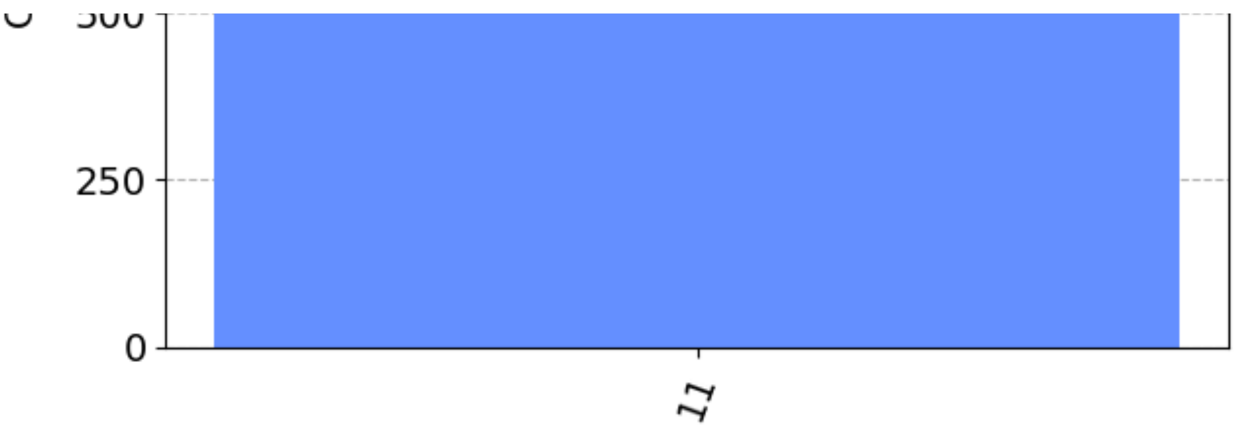
```
     ┌───┐   ┌───┐┌───┐   ┌───┐
q_0: ┤ H ├─■─┤ H ├┤ Z ├─■─┤ H ├
     ├───┤ │ ├───┤├───┤ │ ├───┤
q_1: ┤ H ├─■─┤ H ├┤ Z ├─■─┤ H ├
     └───┘   └───┘└───┘   └───┘
```

```
sv_sim = qiskit_aer.Aer.get_backend('statevector_simulator')
result = sv_sim.run(grover_circuit).result()
statevec = result.get_statevector()
from qiskit.visualization import array_to_latex
array_to_latex(statevec, prefix="|\\psi\\rangle =")
```

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

```
grover_circuit.measure_all()
qasm_sim = qiskit_aer.Aer.get_backend('qasm_simulator')
result = qasm_sim.run(grover_circuit).result()
counts = result.get_counts()
plot_histogram(counts)
```





Start coding or generate with AI.

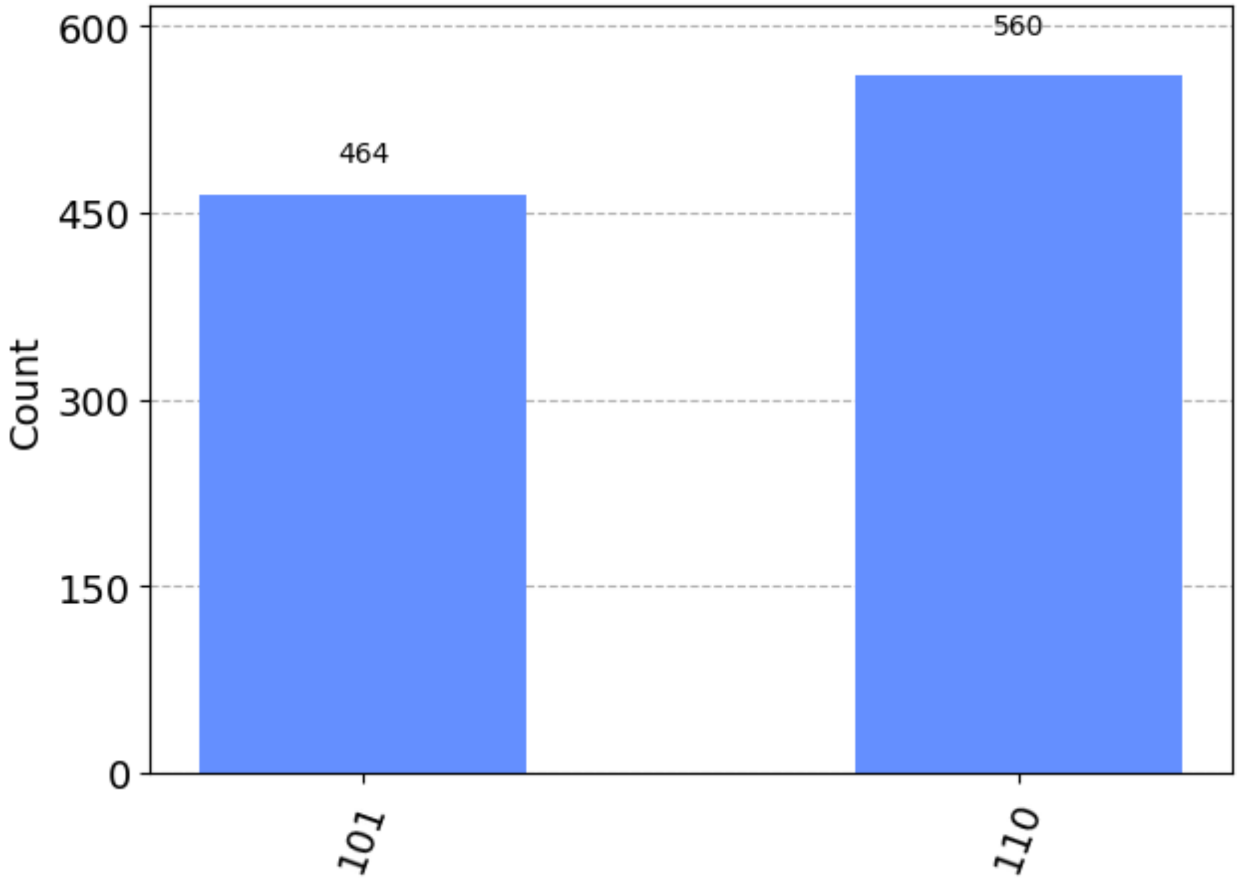
```
qc = QuantumCircuit(3)
qc.cz(0, 2)
qc.cz(1, 2)
oracle_ex3 = qc.to_gate()
oracle_ex3.name = "U$_\omega$"
```

```
def diffuser(nqubits):
    qc = QuantumCircuit(nqubits)
    for qubit in range(nqubits):
        qc.h(qubit)
    for qubit in range(nqubits):
        qc.x(qubit)
    qc.h(nqubits-1)
    qc.mcx(list(range(nqubits-1)), nqubits-1)
    qc.h(nqubits-1)
    for qubit in range(nqubits):
        qc.x(qubit)
    for qubit in range(nqubits):
        qc.h(qubit)
    U_s = qc.to_gate()
    U_s.name = "U$_s$"
    return U_s
```

```
n = 3
grover_circuit = QuantumCircuit(n)
grover_circuit = initialize_s(grover_circuit, [0,1,2])
grover_circuit.append(oracle_ex3, [0,1,2])
grover_circuit.append(diffuser(n), [0,1,2])
grover_circuit.measure_all()
grover_circuit.draw(output='text',style='bw')
```

```
        ┌───┐┌──────────────┐┌─────────┐ ░ ┌─┐      
   q_0: ┤ H ├┤0             ├┤0        ├─░─┤M├──────
        ├───┤│              ││         │ ░ └╥┘┌─┐   
   q_1: ┤ H ├┤1 U$_\omega$  ├┤1 U$_s$  ├─░──╫─┤M├───
        ├───┤│              ││         │ ░  ║ └╥┘┌─┐
   q_2: ┤ H ├┤2             ├┤2        ├─░──╫──╫─┤M├
        └───┘└──────────────┘└─────────┘ ░  ║  ║ └╥┘
meas: 3/════════════════════════════════════╩══╩══╩═
                                            0  1  2 
```

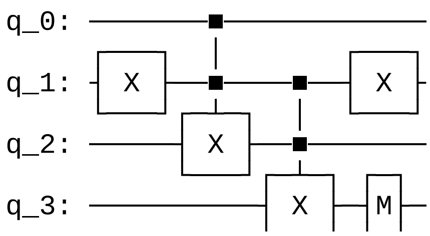
```
qasm_sim = qiskit_aer.Aer.get_backend('qasm_simulator')
transpiled_grover_circuit = transpile(grover_circuit, qasm_sim)
results = qasm_sim.run(transpiled_grover_circuit).result()
counts = results.get_counts()
plot_histogram(counts)
```




```
# oracle for '101':
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi

qreg_q = QuantumRegister(4, 'q')
creg_c = ClassicalRegister(4, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)

circuit.x(qreg_q[1])
circuit.ccx(qreg_q[0], qreg_q[1], qreg_q[2])
circuit.ccx(qreg_q[1], qreg_q[2], qreg_q[3])
circuit.x(qreg_q[1])
circuit.measure(qreg_q[3], creg_c[3])
circuit.draw(output='text',style='bw')
```



```
                └───┘ └╥┘
c: 4/═════════════════╩══
                      3
```