search_algorithm

October 21, 2024

1 SEARCH ALGORITHM (GROVER's ALGORITHM)

[104]:

```
from qiskit import QuantumCircuit, transpile, ClassicalRegister, QuantumRegister
from qiskit.visualization import plot_histogram, plot_bloch_multivector
from qiskit_aer import AerSimulator
import math
```

1.0.1 A few oracles to test the algorithm

[105]:

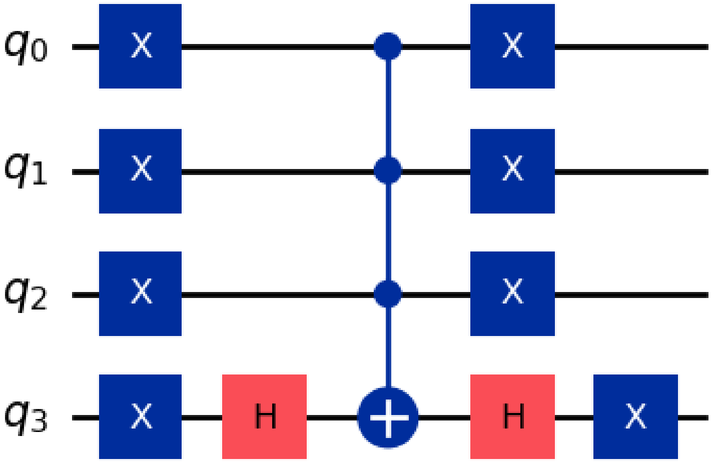
```
def oracle_0000():
    '''
    Creates Oracle to find state 0000.
    Parameters
    ----------
    register: Register of input qubits.
    solution: The true marked state.
    Return
    ------
    Quanttum Circuit based on the oracle.
    '''
    n = 4
    oracle = QuantumCircuit(n)
    oracle.x(range(n))
    oracle.h(n-1)
    oracle.mcx(list(range(n-1)), n-1)
    oracle.h(n-1)
    oracle.x(range(n))
    return oracle

def oracle_101():
    '''
    Creates Oracle to find state |101>.
    Parameters
    ----------
    register: Register of input qubits.
    solution: The true marked state.
    Return
    ------
    Quantum Circuit based on the oracle.
    '''
    n = 3
    oracle = QuantumCircuit(n)
    oracle.x(1)
    oracle.h(2)
    oracle.mcx([0,1], 2)
    oracle.h(2)
    oracle.x(1)
    return oracle
```

|0000> Oracle

[106]:

```
oracle = oracle_0000()
oracle.draw('mpl')
```

[106]:

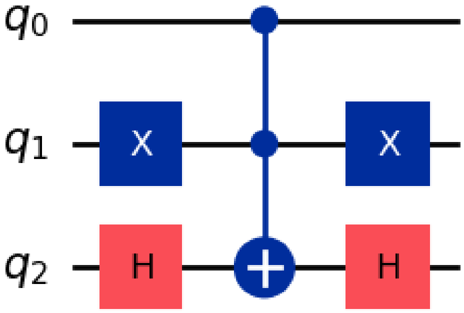


|101> Oracle

[107]:

```
oracle = oracle_101()
oracle.draw('mpl')
```

[107]:

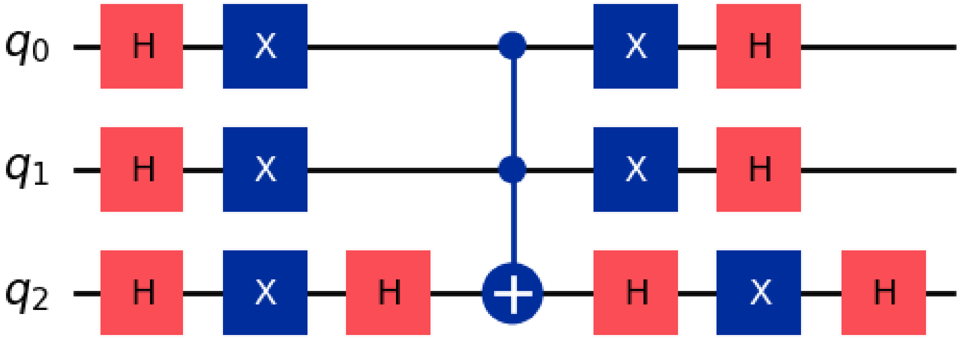


1.0.2 Diffuser

[108]:

```
def diffuser(n):
    '''

    '''
    diffuser = QuantumCircuit(n)
    diffuser.h(range(n))
    diffuser.x(range(n))
    diffuser.h(n-1)
    diffuser.mcx(list(range(n-1)), n-1)
    diffuser.h(n-1)
    diffuser.x(range(n))
    diffuser.h(range(n))
    return diffuser
dif = diffuser(3)
dif.draw('mpl')
```

[108]:



2 Search Algorithm

[109]:

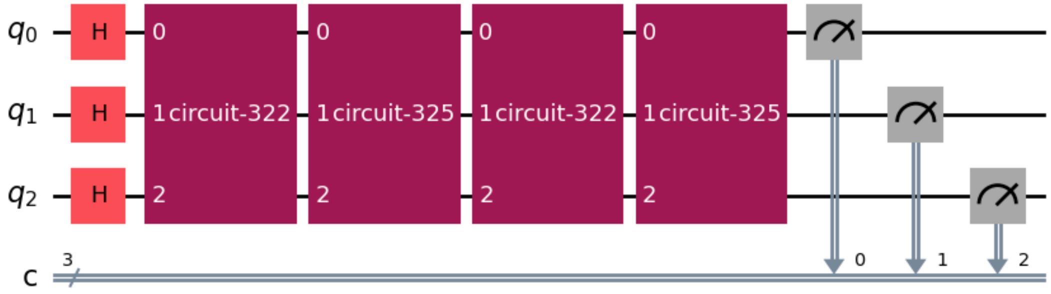
```
def search_algorithm(n, oracle, diffuser):
    '''

    '''
    qc = QuantumCircuit(n,n)
    qc.h(range(n))
    N = 2 ** n
    iters = (math.pi / 4) * math.sqrt(N)
    iters = math.floor(iters)
    for _ in range(iters):
        qc.append(oracle, range(n))
        qc.append(diffuser, range(n))
    qc.measure(range(n),range(n))
    return qc
```

2.0.1 Apply the Quantum Circuit to develop the Search Algorithm (Grover's)

[110]:

```
oracle = oracle_101().to_gate()
diff = diffuser(3).to_gate()
qc = search_algorithm(3, oracle, diff)
qc.draw('mpl')
```

[110]:

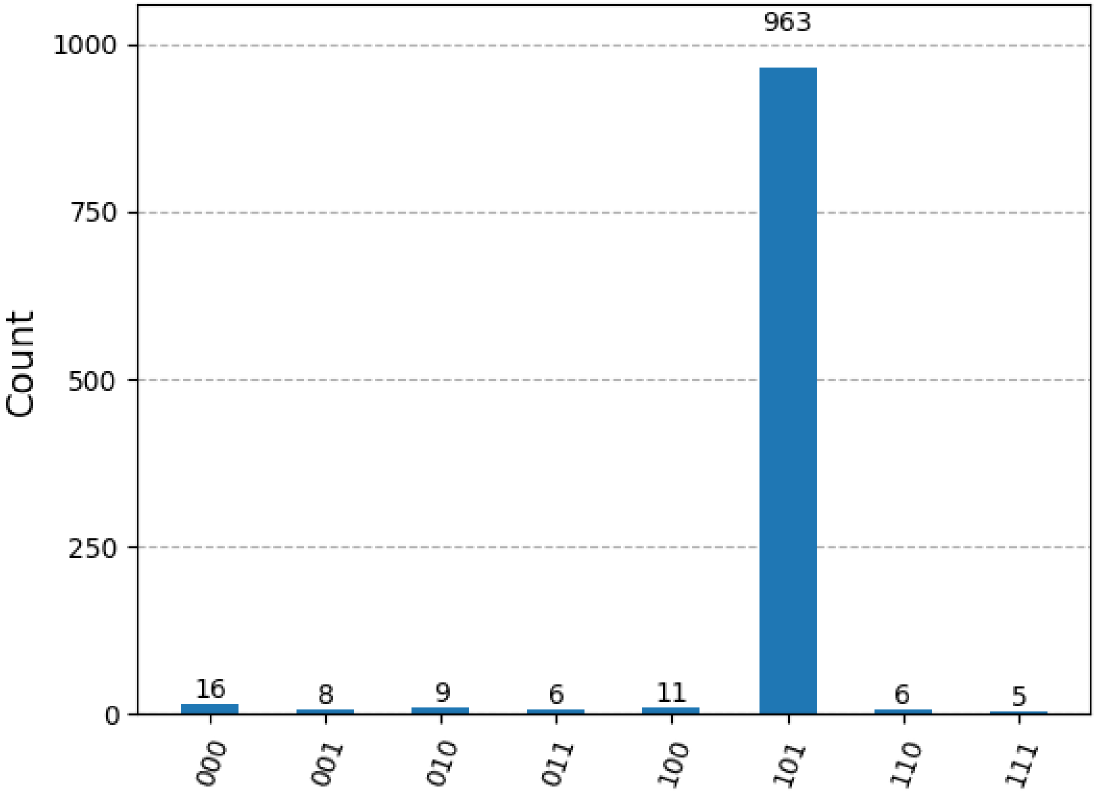


2.0.2 Simulate and Plot the counts of the solution state (marked state)

[114]:

```
aer = AerSimulator()
transpiled = transpile(qc,aer)
results = aer.run(transpiled).result()
counts = results.get_counts()
plot_histogram(counts)
```

[114]:


Count
1000
750
500
250
0
16
8
9
6
11
963
6
5
000
001
010
011
100
101
110
111