Running Grover Algorithm to search an item from given array

Sukhsagard@iisc.ac.in

Feb 2022

1 Introduction

There is always need to perform a search on some kind of database. Classically, it is well known that when searching through N items, a linear search takes O(N) queries to find the index of item. In the mid-nineties, Lov Grover came up with a new algorithm which performs search through this unstructured array using a Quantum Computer. Grover's algorithm takes only $O(\sqrt{N})$ queries, a quadratic speed up over a classical search. An array of 8 randomly placed items is considered and purpose is to find out index of given item using Grover algo.

Arr=[IIT-K, IIT-D, IIT-B,IIT-H,IIT-R,IISC, IIT-M, IIT-G]

This paper shows implementation of Grover algorithm using Qiskit, and run on a simulator and device.

2 Grover Algorithm Theoretical Aspect

At low level, Instruction to quantum computer is given in terms of pulses, Qubit is manipulated in terms of pulses. But at high level, we design circuit comprise of quantum gates to give Instruction to Quantum computer. Quantum circuits are the foundation of quantum computing. When you run a quantum circuit, the gates manipulate Qubit's in the quantum computer. Quantum circuit in Qiskit can be designed either by composer or by simple commands of python. Circuit for Grover Algorithm is shown below-

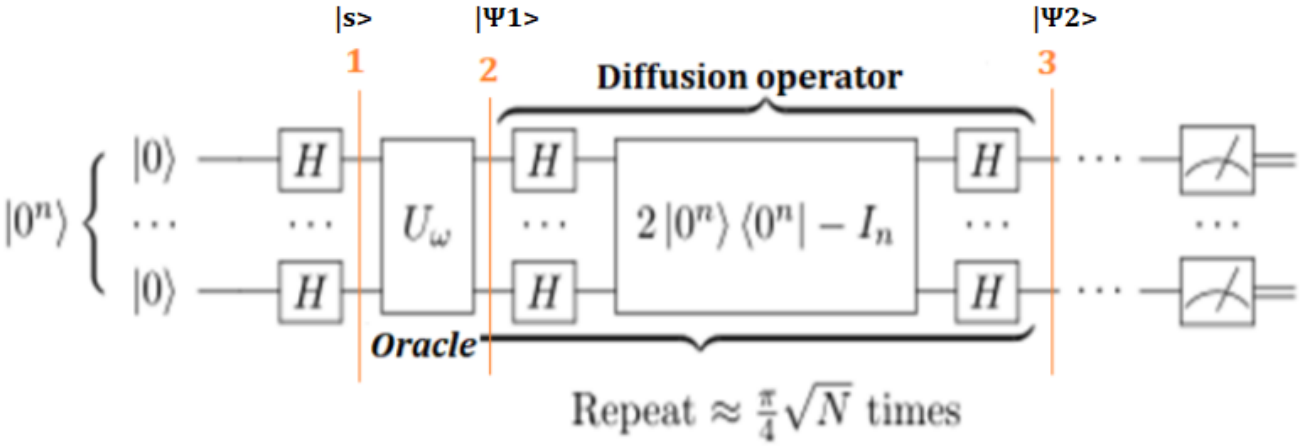

Figure 1: Quantum circuit representation of Grover's algorithm

2.1 Grover Algorithm step

1. Number of Qubit:
   Number of Qubit (n) which depends on array length $N = 2^n$.
   Initialize the system to the uniform superposition over all states using Hadamard Gate.

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

   for N=8

$$|s\rangle = \frac{1}{\sqrt{8}} \sum_{x=0}^{7} |x\rangle$$

   Here for database design,extra n+1 qubit (with 1 ancilla qubit) is considered.

2. Apply Oracle (Operator U_w) :
   Task of Oracle is to identify target and invert it's phase.
   $U_w = I - |w\rangle \langle w|$

$$|\Psi_1\rangle = U_w |s\rangle$$

   Refer section 3.1 for Oracle circuit design.

3. Apply Grover diffusion operator:
   Purpose of diffusion operator is to amplify the amplitude of target. General diffusion operator can be written-

$$U_s = 2 |s\rangle \langle s| - I$$
$$U_s = H^{(\otimes n)} X^{(\otimes n)} (MCZ) X^{(\otimes n)} H^{(\otimes n)}$$
$$|\Psi_2\rangle = U_s |\Psi_1\rangle$$

   Here MCZ is Multi-contrl Z gate. MCZ can be implemented with the help of Multi-control Not gate beacuse of Z=HXH identity.

   Diffusion circuit for N=8 is shown below-

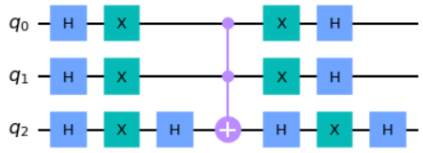

Figure 2: Diffuser for N=8

4. Apply Measurement
   After Grover operator $U_s U_w$ operation for $\frac{\pi}{4} * \sqrt{N}$ time probability of target state reaches close to 100%.In last measurement is applied on state $|\psi_2\rangle$ in the computational basis.Basis with maximum probability is output. It is index of target item.

2.2 Geometrical interpretation

Below diagram shows two reflection operation which lead to $|s\rangle$ to target state $|w\rangle$.

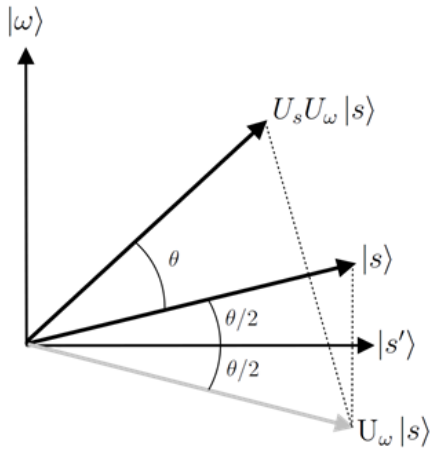

Figure 3: Geometric interpretation of the first iteration of Grover's algorithm. The state vector $|s\rangle$ is rotated towards the target vector $|\omega\rangle$ as shown.

3 Grover's Three Qubit Algorithm Implementation

3.1 Oracle Design for Three Qubit

3.1.1 Encoding

I will use Grover's search algorithm to search the database shown in below Table. This database, contains a list of records of college, with each record having two main fields. The index field takes the place of the (sorted) names in a list book, and the name-code field takes the place of the (unsorted) name code. In Grover's search algorithm, both fields must be represented by binary numbers. Objective is find out index with given name-code.

Index	Name	Name_code(b)	Name_code(D)
000	IIT-K	001	1
001	IIT-D	000	0
010	IIT-B	011	3
011	IIT-H	101	5
100	IIT-R	110	6
101	IISC	100	4
110	IIT-M	111	7
111	IIT-G	010	2

3.1.2 Oracle

Here last 4 Qubit are required to implement database.I will have concern on first 3 qubit. When oracle is applied on $|s\rangle$ equal superposition state of first 3 Qubit, target ket phase get inverted. let's take target is 'IISC' so target ket correspond to 'IISC' is $|101\rangle$ which get inverted when oracle is applied.Oracle comprise of Database and Marking gate.

$|w\rangle$=$|5\rangle$, $|\Psi_1\rangle$=Oracle $|s\rangle$=$\frac{1}{\sqrt{8}}$($|0\rangle$+$|1\rangle$+$|2\rangle$+$|3\rangle$+$|4\rangle$-$|5\rangle$+$|6\rangle$+$|7\rangle$)

Oracle circuit is shown below

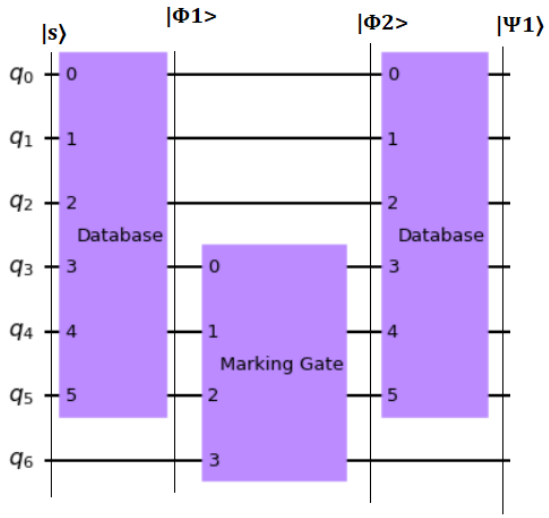

Figure 4: Oracle

Steps from $|s\rangle$ to $|\psi_1\rangle$-

$$|s\rangle = (\frac{1}{\sqrt{8}})\sum_{x=0}^{7}|x\rangle \bigotimes |000\rangle \bigotimes (\frac{1}{\sqrt{2}})(|0\rangle - |1\rangle)$$

$$|\Phi_1\rangle = DB\,|s\rangle = (\frac{1}{\sqrt{8}})*(\frac{1}{\sqrt{8}})(|0\rangle\,|1\rangle+|1\rangle\,|0\rangle+|2\rangle\,|3\rangle+|3\rangle\,|5\rangle+|4\rangle\,|6\rangle+|5\rangle\,|4\rangle+|6\rangle\,|7\rangle+|7\rangle\,|2\rangle) \bigotimes (\frac{1}{\sqrt{2}})(|0\rangle-|1\rangle)$$

$$|\Phi_2\rangle = MG\,|\Phi_1\rangle = (\frac{1}{\sqrt{8}})*(\frac{1}{\sqrt{8}})(|0\rangle\,|1\rangle+|1\rangle\,|0\rangle+|2\rangle\,|3\rangle+|3\rangle\,|5\rangle+|4\rangle\,|6\rangle-|5\rangle\,|4\rangle+|6\rangle\,|7\rangle+|7\rangle\,|2\rangle) \bigotimes (\frac{1}{\sqrt{2}})(|0\rangle-|1\rangle)$$

$$|\Psi_1\rangle = DB\,|\phi_2\rangle = (\frac{1}{\sqrt{8}})(|0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle - |5\rangle + |6\rangle + |7\rangle) \bigotimes |000\rangle \bigotimes (\frac{1}{\sqrt{2}})(|0\rangle - |1\rangle)$$

Consider first 3 Qubit-

$$|\Psi_1\rangle = (\frac{1}{\sqrt{8}})(|0\rangle + |1\rangle + |2\rangle + |3\rangle + |4\rangle - |5\rangle + |6\rangle + |7\rangle)$$

3.2 Database Design

Load the database of Table in terms of Qubit. The left most control gate operates when the controlling index qubits are in the state $|000\rangle$ and turns the database qubits into the state $|001\rangle$, since 001 is the Name-code for IIT-K. The next gate operates when the controlling index qubits are in the state $|001\rangle$ and turns the database qubits into the state $|000\rangle$ for IIT-D.The final gate correspond to IIT-G complete the loading of the database into the database of qubits.

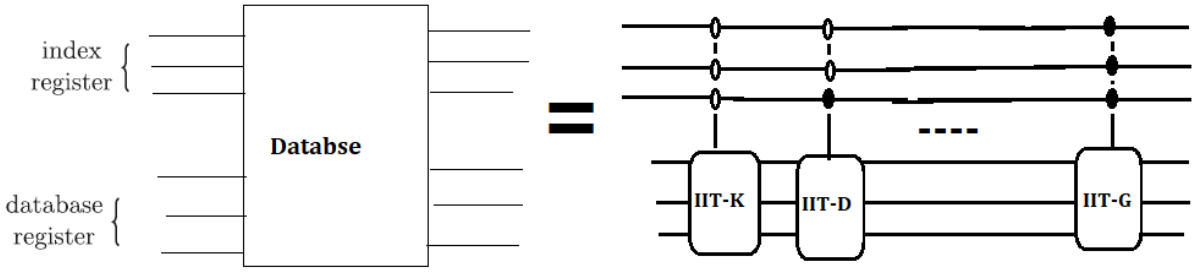

Figure 5: Database

3.3 Marking Gate Circuit Design

Marking Gates, 8 Circuits are shown below. Choice of Circuit depends on target. For example for targer 'IISC'$|100\rangle$ circuit with first black and rest two white dot will be selected. A black dot in below Fig. means the control qubit must be set to $|1\rangle$, and a white dot means the control qubit must be set to $|0\rangle$.

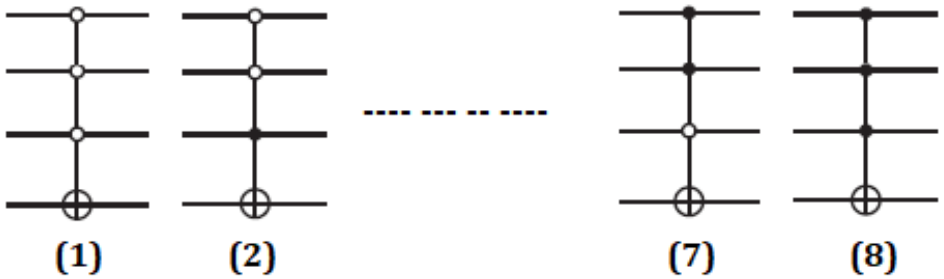

Figure 6: Marking Circuit

3.4 Qiskit Code

Code is applicable for general n Qubit.Circuit seems more complex due to database design. Grover Algo. take $O(\sqrt{N})$ query for given database. Database is designed once.

```
1.#Importing standard Qiskit libraries
import numpy as np
from qiskit import QuantumCircuit, transpile, Aer, IBMQ, QuantumRegister ,ClassicalRegis-
ter, execute
from qiskit.tools.jupyter import *
from qiskit.visualization import *
from ibm_quantum_widgets import *
from qiskit.providers.aer import QasmSimulator

#Loading your IBM Quantum account(s)
provider = IBMQ.load_account()

2.#Database creation
n=3
name_codes =[1, 0, 3 ,5, 6, 4, 7, 2]
database ={}
for i in range(2**n):
    database[i]= name_codes[i]
desired_code = 4 # Here put coded value corresponding to iisc
binary=bin(desired_code).replace("0b", "")
desired_code='0'*(n-len(binary))+binary
DB = QuantumCircuit(2*n)
def DB_function(QC, name_code):
    binary=bin(name_code).replace("0b", "")
```

```
    x='0'*(n-len(binary))+binary
    for j in range(n):
      if x[j]=='1': DB.mct(list(np.arange(n)), n+j)
for i in range(2**n):
    binary=bin(i).replace("0b", "")
    y='0'*(n-len(binary))+binary
    for j in range(n):
      if y[j]=='0': DB.x(j)
    DB_function(DB, database[i])
    for j in range(n):
      if y[j]=='0': DB.x(j)
Datab_ckt=DB.to_gate(label='Database')

3.#Marking Gate Circuit
MG = QuantumCircuit(n+1)
for i in range(n):
    if desired_code[i]=='0': MG.x(i)
MG.mct(list(np.arange(n)), n)
for i in range(n):
    if desired_code[i]=='0': MG.x(i)
Marking_gate=MG.to_gate(label='Marking Gate')
oracle = QuantumCircuit(2*n+1)
oracle.append(Datab_ckt,list(np.arange(2*n)))
oracle.append(Marking_gate,list(np.arange(n,2*n+1)))
oracle.append(Datab_ckt,list(np.arange(2*n)))

4.#Diffusion Circuit for Amplitude Amplification
phase = QuantumCircuit(n)
for i in range(n):
    phase.h(i)
    phase.x(i)
phase.h(n-1)
phase.mct(list(np.arange(n-1)), n-1)
phase.h(n-1)
for i in range(n):
    phase.x(i)
    phase.h(i)
Diffuser=phase.to_gate(label='Diffuser')

5.#Combine all to make Grover Operator Grover = QuantumCircuit(2*n+1)
Grover = Grover + oracle
Grover.append(Diffuser,list(np.arange(n)))

6.#Main Circuit circuit=QuantumCircuit(2*n+1,n)
#MAKE all qubit in equal superpostion state
for i in range(n):
circuit.h(i)
#To make ancilla bit in state 1 and further equal superposition
circuit.x(2*n)
circuit.h(2*n)
circuit=circuit+Grover
circuit.measure(list(np.arange(n)),list(np.arange(n))[::-1])
circuit.draw()

7.#Compiling circuit on simulator and retrieving result
```

```
backend=Aer.get_backend('qasm_simulator')
job=execute(circuit,backend, shots=1024)
result=job.result().get_counts()
plot_histogram(result)

8.#Compiling circuit on real device and retrieving result
provider = IBMQ.load_account()
provider = IBMQ.get_provider(hub="ibm-q-education")
device = provider.get_backend('ibmq_jakarta')
from qiskit.tools.monitor import job_monitor
transpiled_grover_circuit = transpile(circuit, device, optimization_level=3)
job = device.run(transpiled_grover_circuit)
job_monitor(job, interval=2)
#When job run status is successful
results = job.result()
answer = results.get_counts(circuit)
plot_histogram(answer)
```

3.5 Qiskit Circuit and Result

Figure 7. shows complete circuit for Grover Algorithm. Output is Target index which is shown as highest probability.Figure 8 shows Simulation result. Result in one query(iteration) is correct up to 77.2% and in two query 95.7%. For $N = 8$ approx. $\frac{\pi}{4} * \sqrt{8} = 2$ are required.Figure 9 shows result obtained from real IBM Quantum computer 'ibmq_jakarta'. Due to high error result is not correct. it has only 12.2% accuracy.

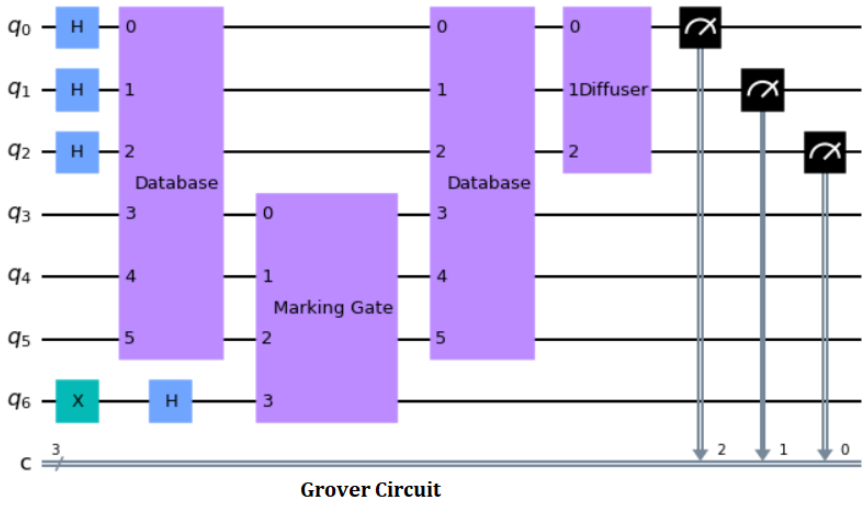

Figure 7: Grover circuit for first iteration

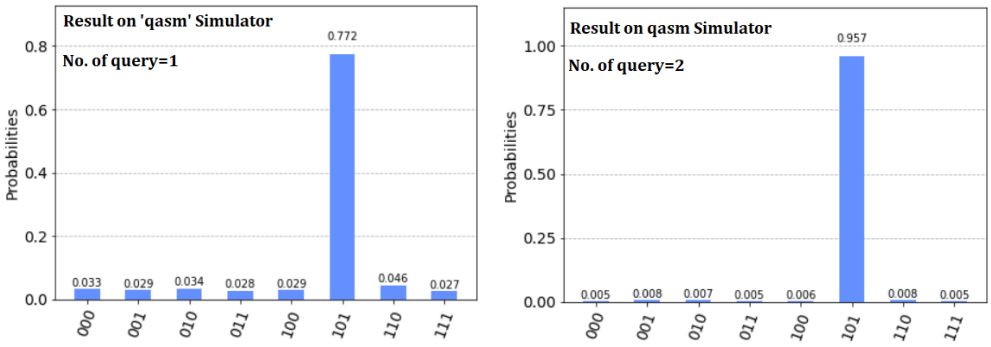

Figure 8: Simulation Result

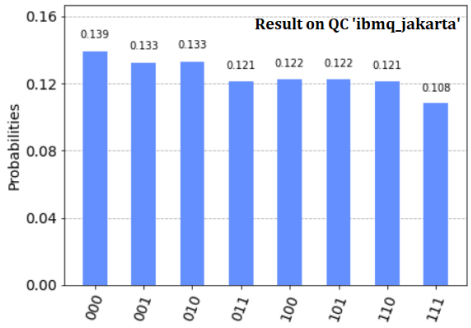

Figure 9: Real device Result

4 Conclusion

This paper shows various fundamental aspect of array searching problem by Quantum Computer. It gives immense feel to use Quantum computer to solve a practical problem. Encoding to database design and finally application of Grover operator and measurement all is shown. It can be easily scaled for any N (Array length). Due to $O(\sqrt{N})$ complexity of Grover algorithm it is possible exhaustive search which make feasible to find out key in many cryptography encryption schemes. If, instead of 1 matching entry, there are k matching entries, the same algorithm works, but the number of iterations must be $\frac{\pi}{4}\left(\frac{N}{k}\right)^{1/2}$ instead of $\frac{\pi}{4}N^{1/2}$.