USING GROVER'S ALGORITHM TO SOLVE SUDOKU

In this presentation we will solve a 2x2 sudoku puzzle

TABLE OF CONTENTS

01

BACKGROUND

SUDOKU

- Sudoku is a logic-based, combinatorial number-placement puzzle, wherein no number should be repeated in a row and column.
- We are going to solve a 2x2 sudoku using binary numbers.
- This has only two solutions:

0	1
1	0

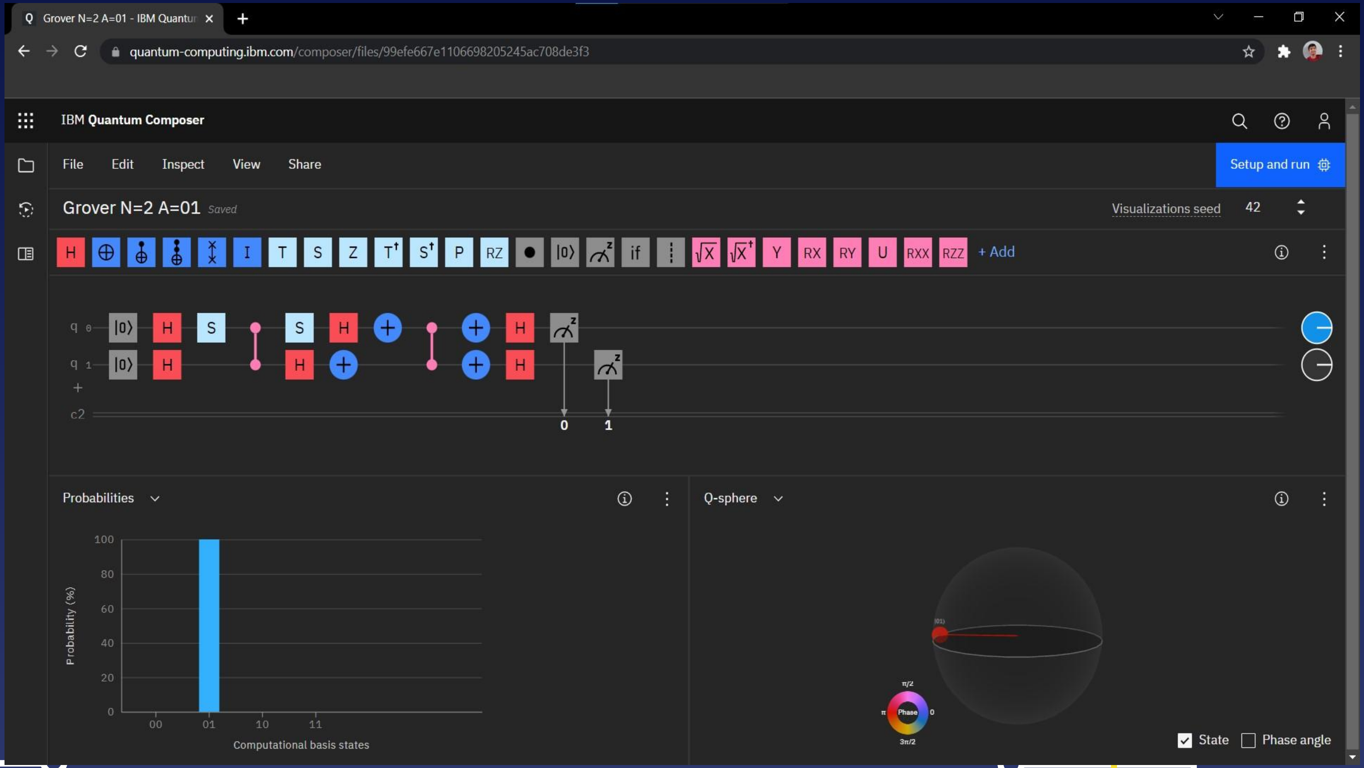
1	0
0	1

Note that, while this approach of using Grover's algorithm to solve this problem is not practical (we can probably find the solution in your head!), the purpose of this example is to demonstrate the conversion of classical decision problems into oracles for Grover's algorithm. This can further be extended into an algorithm to solve even bigger grids.

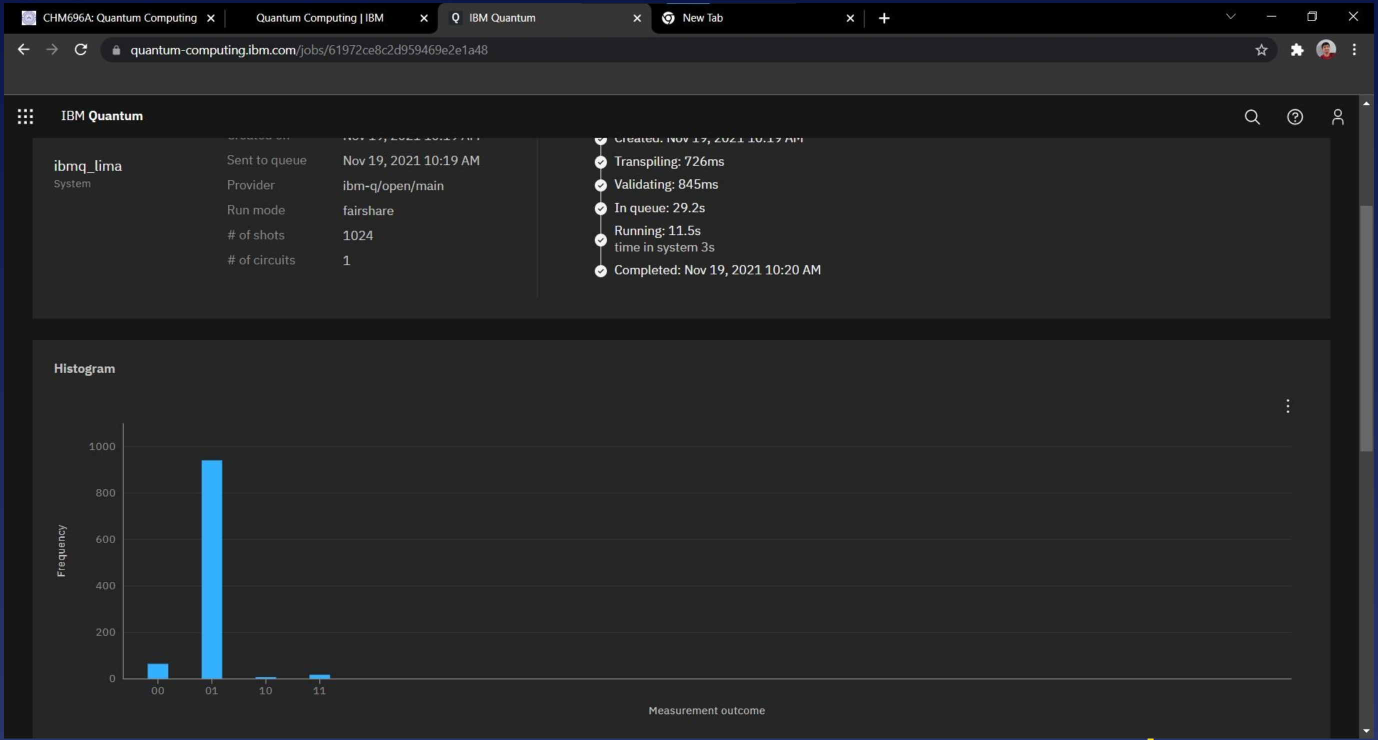
ORACLE

- It is a black box used extensively in quantum algorithms for the estimation of functions using qubits.
- It is represented as $O(|x\rangle \otimes |y\rangle) = |x\rangle \otimes |y \oplus f(x)\rangle)$
- They can recognize the solutions to the search problem.

Grovers algo for 2 qubits(|01>)



Results


CHM696A: Quantum Computing
Quantum Computing | IBM
IBM Quantum
New Tab
quantum-computing.ibm.com/jobs/61972ce8c2d959469e2e1a48
IBM Quantum
ibmq_lima
System
Sent to queue
Nov 19, 2021 10:19 AM
Provider
ibm-q/open/main
Run mode
fairshare
# of shots
1024
# of circuits
1
Transpiling: 726ms
Validating: 845ms
In queue: 29.2s
Running: 11.5s
time in system 3s
Completed: Nov 19, 2021 10:20 AM
Histogram
1000
800
600
400
200
0
Frequency
00
01
10
11
Measurement outcome

As you can see from the results, we got 01 with a very high probability than any other state, but along 01, we got some other state. However, in the case of 2 qubits, it was expected to get the results with 100% probability. This is because there is some decoherence and noise in the real world circuit.

PROCESS

1 CIRCUIT

Create a classical circuit that identifies the correct solution.

2 ORACLE

To turn the circuit into an oracle.

3 ALGORITHM

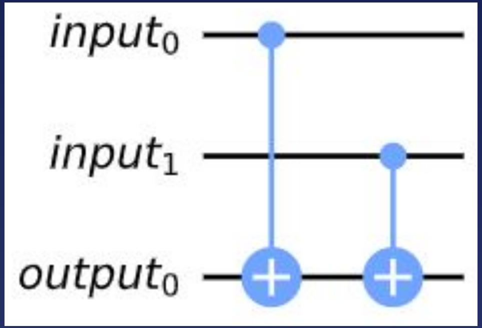
Using the Grover's algorithm to solve the oracle.

02

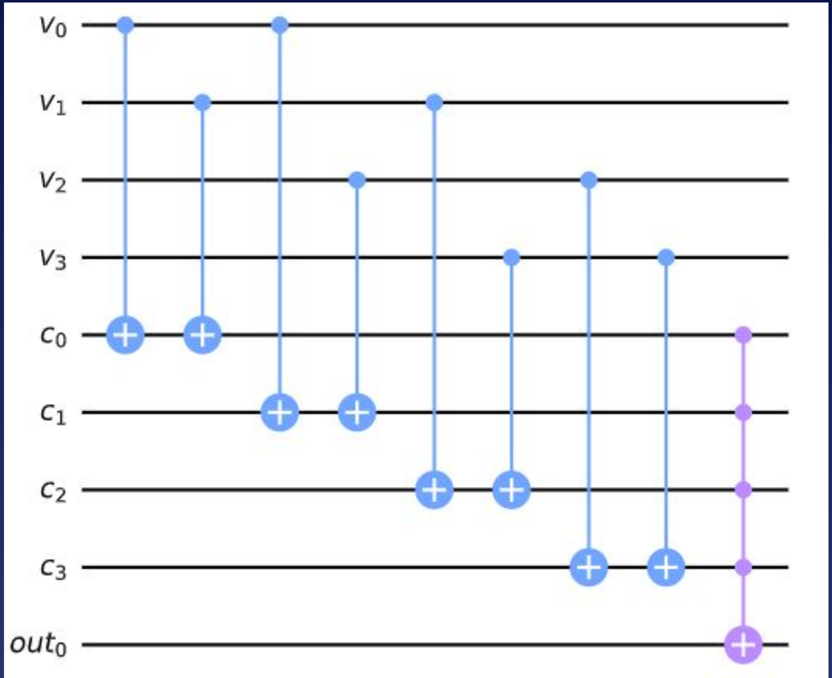
CIRCUIT

- Each cell is assigned a variable.
- Clauses:
    - V0 not equal to V1
    - V0 not equal to V2
    - V1 not equal to V3
    - V2 not equal to V3
- We create a circuit that checks the clauses and stores it in an output bit.
- This is achieved using the CNOT or XOR operation.
- If the clauses is satisfied the value of output bit is 1 else 0.

V0	V1
V2	V3



- To check every clause we repeat this circuit for every pairing.
- The circuit above takes as input an initial assignment of the bits v0, v1, v2 and v3, and all other bits should be initialized to 0. After running the circuit, the state of the out0 bit tells us if this assignment is a solution or not; out0 = 0 means the assignment is not a solution, and out0 = 1 means the assignment is a solution.



Toffoli gate

03

ORACLE & ALGORITHM

ORACLE (U_w)

Now in order to implement the oracle we will use three registers:

1. Sudoku variables (x = v3, v2, v1, v0; x = w be the solution)
2. Clauses (initial state: $|0000\rangle$ which we will abbreviate as $|0\rangle$)
3. Output qubit $|out0\rangle$ (initial state: $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$

So the transformation is:

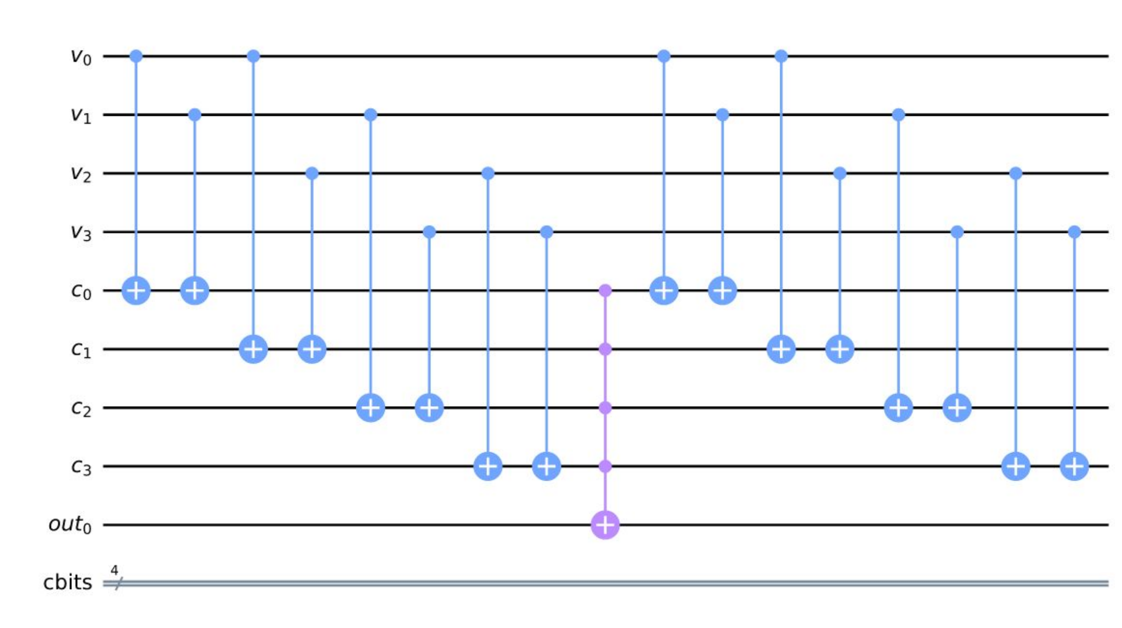
$$U_w |x\rangle |0\rangle |out0\rangle = |x\rangle |0\rangle |out0 \oplus f(x)\rangle$$

SO OUR ORACLE COMES OUT TO BE:

$$U_\omega|x\rangle|0\rangle|-\rangle = \begin{cases} |x\rangle|0\rangle|-\rangle & \text{for } x \neq \omega \\ -|x\rangle|0\rangle|-\rangle & \text{for } x = \omega \end{cases}$$

To adapt our checking circuit into a Grover oracle, we need to guarantee the bits in the second register (c) are always returned to the state $|0000\rangle$ after the computation. To do this, we simply repeat the part of the circuit that computes the clauses which guarantees $c0 = c1 = c2 = c3 = 0$ after our circuit has run. This is because quantum logics are reversible.

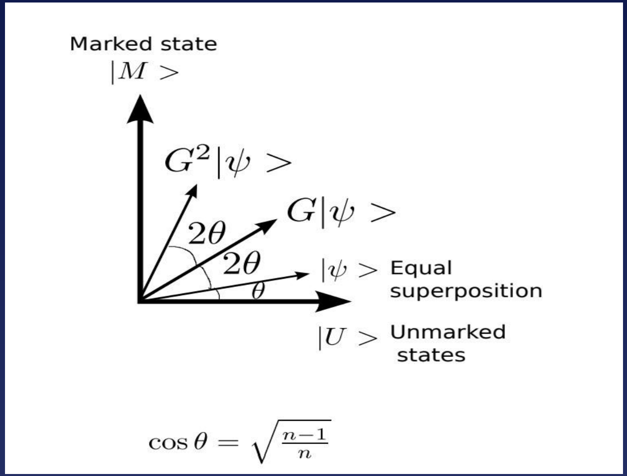
Finally the Grover Oracle is:



FINAL ALGORITHM

Wait!

Now, what do you think how many times do have to Amplify the amplitude or iterate the algo ?
Is it sqrt(N)?

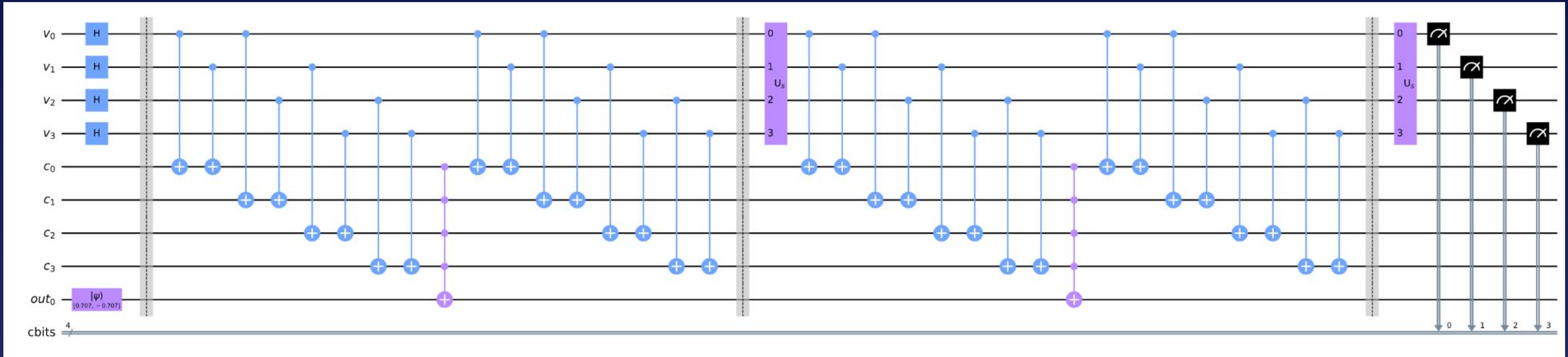


REMEMBER:-
Rotating more (or less) than the required amount will take us away from the marked state. So, it seems that we need to know the number of marked items to execute Grover's algorithm.

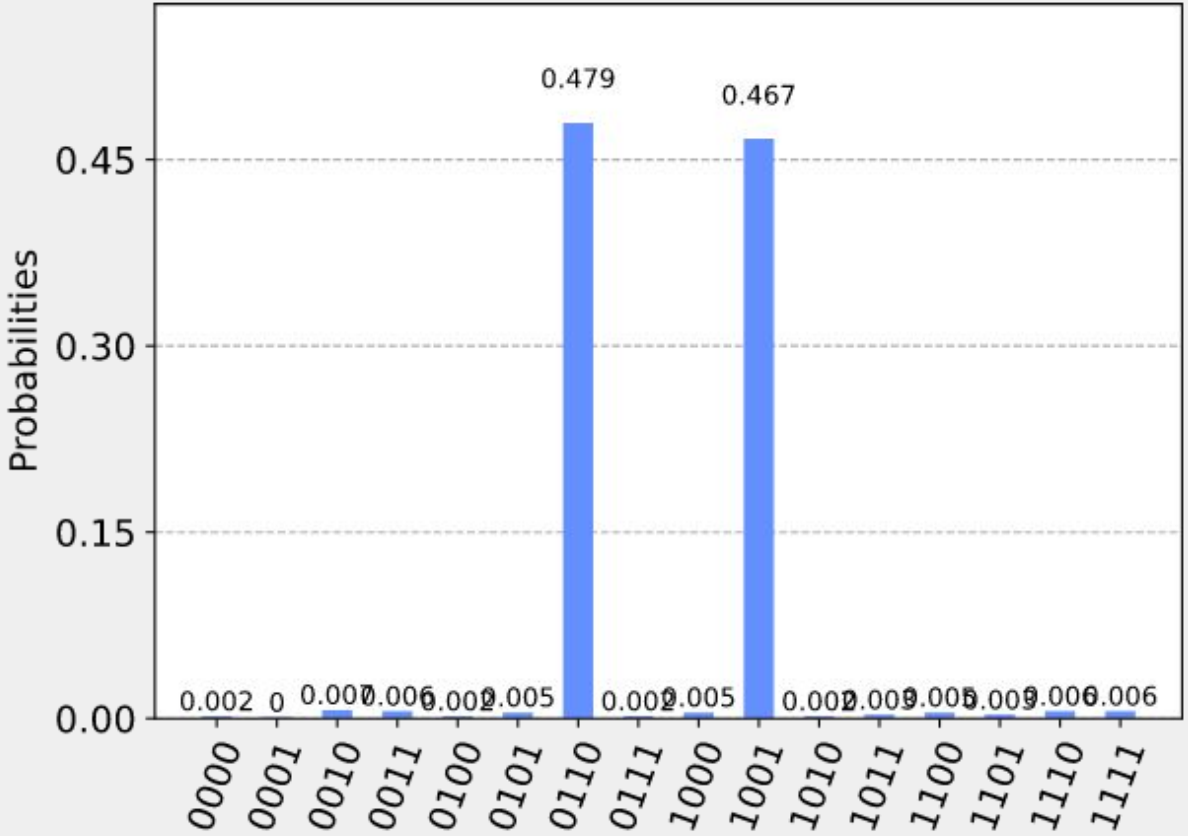
In this case, for two required states, the rotation necessary will be different from if the answer was one because we know that the no of rotation s required is:

$$l = \lfloor \left(\frac{\pi}{2\cos^{-1}\sqrt{\frac{n-m}{n}}} - 1 \right) /2 \rceil .$$

In this case its 2.



This gives us two solutions (v0, v1, v2, v3): (0,1,1,0) and (1,0,0,1).


Probabilities
0.45
0.30
0.15
0.00
0.479
0.467
0.002
0
0.007
0.006
0.002
0.005
0.002
0.005
0.002
0.003
0.005
0.003
0.006
0.006
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

THANKS!

KARAN

References:
Wikipedia-
https://en.wikipedia.org/wiki/Grover%27s_algorithm
Qiskit-
https://qiskit.org/textbook/ch-algorithms/grover.html