SudoQ

Ángel Gómez González, Alberto Martínez Gallardo, Javier López Roda

Indice

1. Grover's Algorithm
2. Solving a Sudoku
3. Algorithm Theory
4. Implementation in Qiskit

Grover's Algorithm

The Grover algorithm is a search algorithm.
This algorithm is able to find a specific object within an unordered set.

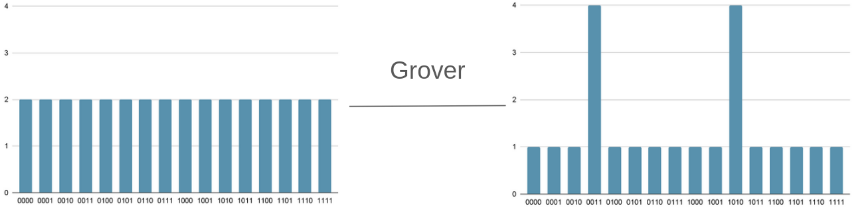


Figure: Operation of Grover's Algorithm

Grover Step by Step

1. First, a Hadamard gate is applied to all states, initializing them with equal probability.
2. Then, an Oracle is applied to invert the state that is being searched.
3. Finally, a Reflector circuit is used to amplify the amplitude of that state.
4. Steps 2 and 3 can be repeated multiple times ($\sqrt{N}$) to increase the probability of obtaining the correct state.

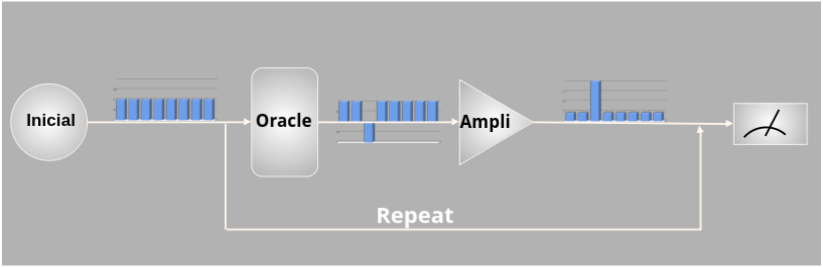
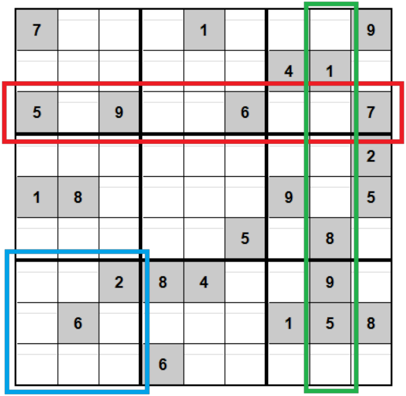


Figure: Diagram of the Algorithm's Operation

Solving a Sudoku

A Sudoku is a mathematical puzzle consisting of a grid with numbers and empty spaces. The way to solve it is by knowing the conditions that must be met. These are:

- The same number cannot be repeated in a row.
- The same number cannot be repeated in a column.
- The same number cannot be repeated in a block.



7				1				9
						4	1	
5		9			6			7
								2
1	8					9		5
					5		8	
		2	8	4			9	
	6					1	5	8
			6					

Figure: Sudoku Conditions

Algorithm Theory

- First Step:

The first step is to generate our Sudoku. After this, we need to subtract 1 from all Sudoku values to match Python's counting order.

	1	3	2
3			4
	4		3
2		4	

Figure: Our SudoQ.

⇒

	0	2	1
2			3
	3		2
1		3	

Figure: SudoQ with indices 0,1,2,3.

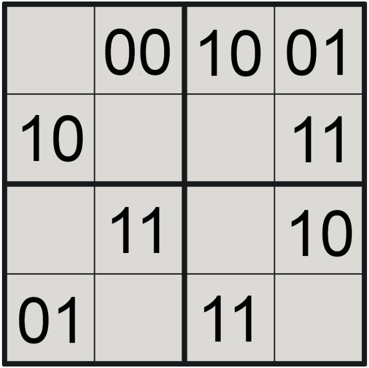
Algorithm Theory

- Second Step:

The second step is to encode Sudoku values in binary, where:

- 0 - 00
- 1 - 01
- 2 - 10
- 3 - 11

These binary values represent the numbers we need to obtain in our circuit and work with in the program.



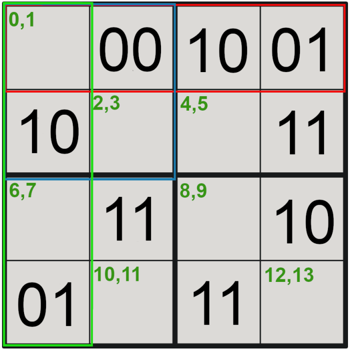
	00	10	01
10			11
	11		10
01		11	

Figure: SudoQ encoded in binary.

Algorithm Theory

- Third Step:

We also encode each bit of the empty spaces and define the conditions for each. Then, we apply the algorithm for each empty space and its conditions.



0,1	00	10	01
10	2,3	4,5	11
6,7	11	8,9	10
01	10,11	11	12,13

- Considering the first row, there is a blank space that cannot be 00, 10, or 01, so it must be 11.
- Considering the column, there are two blank spaces that cannot be 10 or 01. Additionally, these spaces cannot have the same number, so the possible states are 1100 and 0011.

Algorithm Theory

- Fourth Step:

To interpret Qiskit's result, we must consider that it returns a state with all values in it, represented as:

$$|0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\rangle \tag{1}$$

However, Qiskit provides it in reverse order, meaning we receive:

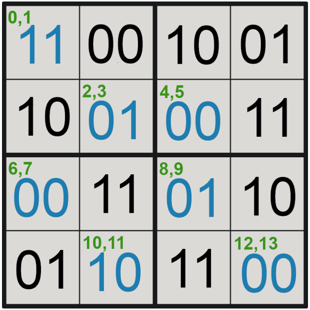
$$|13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0\rangle \tag{2}$$

After executing the circuit, we obtain the most probable state:

$$|00011000001011\rangle \xrightarrow{\textit{invert}} |11010000011000\rangle \tag{3}$$

Algorithm Theory

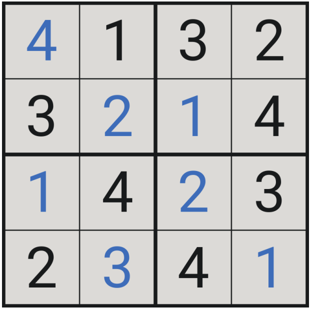
$$|11\ 01\ 00\ 00\ 01\ 10\ 00\rangle$$



0,1 11	00	10	01
10	2,3 01	4,5 00	11
6,7 00	11	8,9 01	10
01	10,11 10	11	12,13 00

Figure: SudoQ solved in binary.

$\Rightarrow$



4	1	3	2
3	2	1	4
1	4	2	3
2	3	4	1

Figure: Solved SudoQ.

Implementation in Qiskit

Let's see the implementation in Qiskit.