# Grover algorithm for universal set of clasuses

Alibek Zhakubayev
Bikram Khanal

August 8, 2021

**Abstract**

The complexity of searching algorithms in classical computing is a perpetual researched field. It is proven that Quantum computers and quantum algorithms can compute these problems faster. Grover Algorithm is a quantum search algorithm that achieves quadratic speedup over optimal classical search implementation. In this paper, we propose an application of the Grover algorithm that searches for the sequences of input qubits in a circuit, such that it forces the output to be 1 with a high probability. Our algorithm considers that the circuit contains only AND, XOR, and OR gates.

## 1 Introduction

The searching problem to locate the distinct record from a sizeable unstructured dataset with $N = 2^n$ is often phrased intricacy [1]. The probability theory claims that given $x$ records, we can obtain the particular record $a$ in $O(N)$ complexity. However, Grover's quantum Algorithm can find the same probability in $O(\sqrt{N})$ complexity. In this paper, we aim to provide as straightforward as possible introductory on Grover's Algorithm.

Two registers used in the Grover's algorithm, $n$ qubits in the first register and one qubits in the second register is the key architectural structure to achieve this speedup complexity over the classical algorithm[3]. We start the circuit for Grover's Algorithm by creating a superposition of $2^n$ computational basis states in the top register. All the qubits in the first register are initialized to state $|0, ....., 0\rangle$. After applying the $n$ Hadamard gate, $H^{\otimes n}$ on the first we have the state,

$$|\psi\rangle = H^{\otimes n}|0, 0, 0, ....., 0\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle \tag{1}$$

Note that, $|\psi\rangle$ is the superposition here. If we start the second register with a single qubit in state $|0\rangle$ or $|1\rangle$, after the Hadamard we achieve the respective Hadamard bases[4]. Let $f : 0, 1^N \to 0, 1$ be a function defined as:

$$f(x) = \begin{cases} 1 & \text{if x is the searched element} \\ 0 & \text{Otherwise.} \end{cases}$$

We define $U_f$ as an oracle often referred to as a black box, defined as,

$$U_f(|i\rangle|j\rangle) = |i\rangle|j \otimes f(i)\rangle \tag{2}$$

When we apply $U_f$ on the state $\phi$, the state of the second register does not changes[4] but the state of the first register changes, and we call this state $\phi_1$. We assume that the Hadamard basis in the second qubit is $|-\rangle$.

$$|\psi_1\rangle|-\rangle = U_f|\psi\rangle|-\rangle = \frac{1}{N}\sum_{i=0}^{N-1}(-1)^{f(i)}|i\rangle|-\rangle \tag{3}$$

Due to *Quantum. parallelism* we can observe all the database elements simultaneously at the quantum level. If the position of the searched element is known, then it will be labeled as the negative value of $i$ in equation 3. It is impossible to get this result in the classical level. Before we perform the measurement and collapse our superposition in the classical bits, we apply another set of Hadamard gate, unitary operator and $n$ Hadamard gates for $O(\sqrt{N})$ times. Let us define this unitary operator, $U_{f_0}$ as,

$$U_{f_0} = 2|\psi\rangle\langle\psi| - I \tag{4}$$

When we apply this operator on state $\psi_1$ we have,

$$\psi_2 = 2|\psi\rangle\langle\psi| - I\psi_1 = \frac{2^{n-2}-1}{2^{n-2}}\psi + \frac{2}{\sqrt{2^n}}|i_0\rangle \tag{5}$$

Equation 5 is the state of the first register and the second register is still on state $|-\rangle$ by assumption.
Thus, we can measure this state on both the register to evaluate the function $f$ and get the probability finding the $x$ record. Below, we present the implementation of this algorithm. Assuming we have an output of 1 for the function $f$ with high probability, we calculate the probability of all possible input qubits on both the register. It is guaranteed to achieve the mentioned output.

# 2 Models, methods or materials

First, we need to get input information from the user. The user will input the number of input qbits, number of clauses, and the clauses themselves. We use these clauses in our oracle. We call our algorithm universal because the user can input any combination of the clauses. The algorithm will output final probabilities based on the user inputs (evaluating the user clauses). The user is asked to input the clauses in a specific format. One example is presented here. The user can input the following details:

- Number of qubits: 5.
- Number of Clauses: 3.

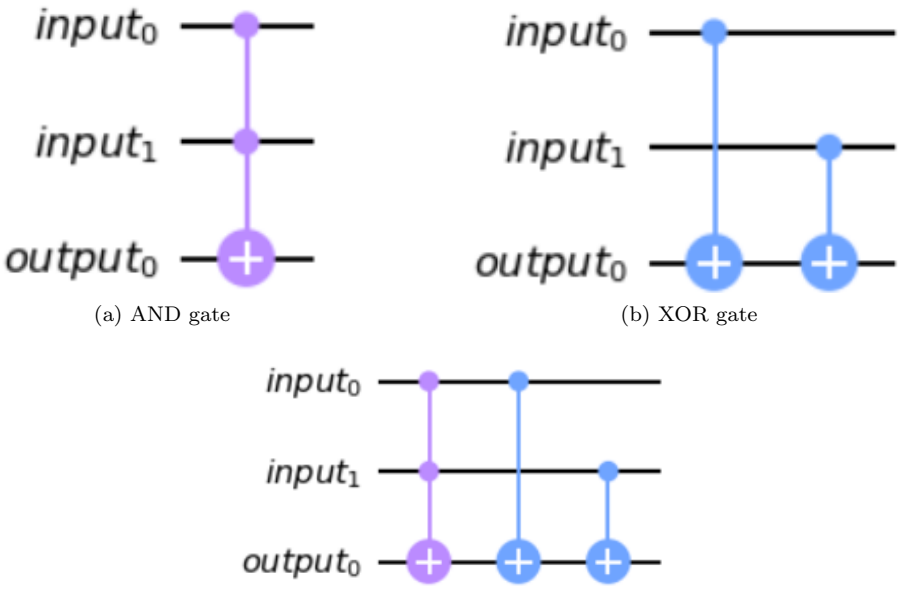
- Clause 1: 0 AND 1.
- Clause 2: 1 XOR 2.
- Clause 3: 2 OR 3.

Qbit indices are separated by the gate. We have implemented XOR, OR, and AND gate. Then, possible input clauses are 0 AND 1, 1 XOR 2, and 2 OR 3 in any order of the qubits.

We implemented the XOR gate using implemented using two CNOT gates. Our input qubits will be control variables, and they will output to one same qubit. Then, the output qubit will have value one if only one of the input qubits has value one.

AND gate is implemented using one Toffoli gate. If both input variables have value one, the output variable will also have value one. Otherwise, zero will be returned.

OR gate is implemented using one Toffoli and 2 CNOT gates. The output is one, if and only if one of the input gates have value 1. All of the codind implementation is done in IBM Quantum framework.




(a) AND gate

(b) XOR gate

(c) OR gate

Figure 1: Quantum gates

Figure 1 show the implementation of AND, XOR, and OR gates as described above.

For each clause gate, we use one clause qubit. Then, using Toffoli gate, we check if all of the clause qubits have value one, we return one. Otherwise, zero is returned. We construct our oracle under this definition and the mathematical abstraction described in the introduction.

Before oracle, each input qbits is put into superposition by applying the Hadamard gate as described in the introduction. Then, we have constructed an oracle. Finally, the diffuser is used.

The diffuser is an essential part of Grover's algorithm. The diffuser function is universal. That means any two oracles that have the same number of input qbits can use the same diffuser. The function consists of the H-gates, X-gates, and multi-controlled Z-gate.

# 3 Results

In this section we present two different examples.

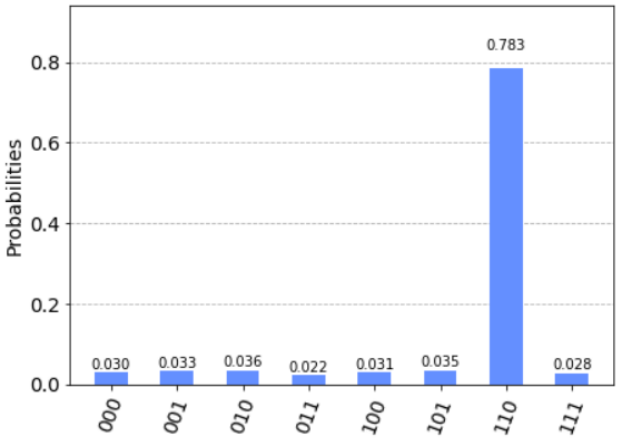



Figure 2: 0 AND 1, 1 XOR 2

Figure 2 illustrates a histogram of probabilities when the input clauses are *0 AND 1* and *1 XOR 2* on 3 qubits circuit. 110 is the only answer and it has a probability of 0.783.

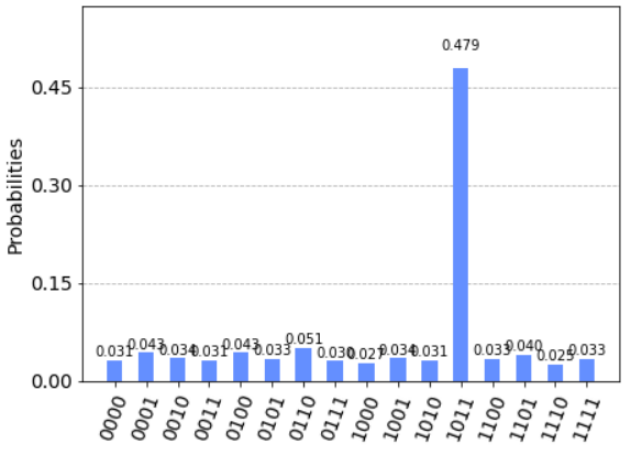



Figure 3: 0 AND 3, 1 XOR 2, 2 AND 3

Figure 3 shows a histogram of the second example. Input clauses are *0 AND 3*, *1 XOR 2*, *2 AND 3* with 4 qubits. 1011 is the only answer and it has a probability of 0.479.

# 4 Conclusion

In this article, we presented a way to calculate the probabilities of possible input sequences such that the output of the function always yields 1. As we saw on table 2 and table3 the probability of the input qubit decreases with the increase in input qubits. To construct an oracle, we presented Grover's quantum algorithm application using $XOR, AND$, and, $OR$ gates. We believe, provided with access to more working qubits and powerful classical computers to simulate the process, this research can expand further and exploit Grover's algorithm.

# 5 References

[1] Team, The Qiskit. "Grover's Algorithm." Qiskit, Data 100 at UC Berkeley, 5 Aug 2021, qiskit.org/textbook/ch-algorithms/grover.html.
[2] Jozsa, Richard. "Searching in Grover's Algorithm." 9 Jan 1999.
[3] Lavor C., Manssur L.R.U, and Portugal R. "Grover's Algorithm: Quantum Database Search." 16 Jan 2003.
[4] Nielsen, Michael A., and Isaac L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 2019.