

Grover's Algorithm

Ishanya¹ Rohan Mehra¹

¹Indian Institute of Science Education and Research Bhopal, Bhopal 462 066
Madhya Pradesh, India

Contact: ishanya21@iiserb.ac.in, rohan21@iiserb.ac.in

Instructor: Dr. Ankur Raina

November 18, 2024

Introduction to Grover's Algorithm

- Quantum algorithm for database search with quadratic speedup over classical methods.
- Utilizes amplitude amplification to locate a marked element in an unsorted database.
- Complexity: $O(\sqrt{N})$ compared to $O(N)$ for classical search.
- Applications in optimization, pattern matching and satisfiability problems like 3-SAT.

Circuit of Grover's Algorithm

- **Initialization:** Apply Hadamard gates to all qubits to create a superposition.
- **Oracle:** Encodes the solution to the search problem.
- **Diffusion Operator:** Reflects the state over the average amplitude.

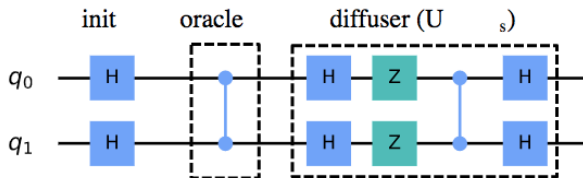


Figure: Circuit Diagram of Grover's Algorithm

The state $|\omega\rangle$ represents the marked state that Grover's algorithm aims to find. The state $|w^\perp\rangle$ is all n -bit strings except for a specific marked state $|w\rangle$. The state $|\omega\rangle$ and $|w^\perp\rangle$ are orthogonal to each other.

$$|w^\perp\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq w} |x\rangle,$$

. The initialization state $|s\rangle$ is the equal superposition state over all possible n -bit strings and $N = 2^n$.

$$|s\rangle = H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle = |s\rangle = \cos\left(\frac{\theta}{2}\right) |w^\perp\rangle + \sin\left(\frac{\theta}{2}\right) |w\rangle$$

$$|s\rangle = \sqrt{\frac{N-1}{N}} |w^\perp\rangle + \sqrt{\frac{1}{N}} |w\rangle$$

Geometric Representation of Grover's Algorithm

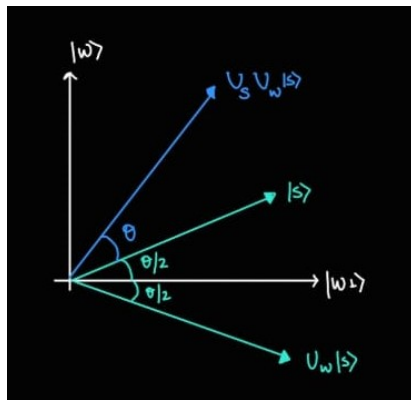


Figure: Geometric Representation of Grover's Algorithm (Drawn by Ishanya)

Oracle Operator U_w

The Oracle U_w acts as a quantum version of a function that checks whether an input is the desired solution (the "marked" state). The oracle U_w marks the state $|w\rangle$ by applying a phase shift:

$$U_w|x\rangle = (-1)^{f(x)}|x\rangle,$$

where $f(x)$ is a function such that:

$$U_w|x\rangle = \begin{cases} -|x\rangle & \text{if } f(x) = 1, \text{ i.e., } x = w, \\ |x\rangle & \text{if } f(x) = 0, \text{ i.e., } x \neq w. \end{cases}$$

In matrix form:

$$U_w = I - 2|w\rangle\langle w|.$$

Grover's Diffusion Operator U_s

The Grover diffusion operator U_s amplifies the amplitude of the marked state while decreasing the amplitudes of other states. This process helps to make the correct solution more likely to be measured after several iterations, (We will see optimal iterations in the later slides). It is defined as:

$$U_s = 2|s\rangle\langle s| - I,$$

where I is the identity matrix.

$$U_s = 2 \left(\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle\langle x| \right) - I = H^{\otimes n} (2|0\rangle^{\otimes n}\langle 0|^{\otimes n} - I) H^{\otimes n}.$$

Probability of Measuring the Marked State

The probability $P(\omega, r)$ of measuring the marked state $|\omega\rangle$ after r iterations of Grover's algorithm is given by:

$$P(\omega, r) = |\langle\omega|(U_s U_\omega)^r |s\rangle|^2.$$

This probability can be approximated by:

$$P(\omega, r) \approx \sin^2 \left(\theta \left(\frac{1}{2} + r \right) \right).$$

At the start of Grover's algorithm, i.e, $r = 0$:

$$P(\omega, 0) = |\langle\omega|s\rangle|^2 = \frac{1}{N}.$$

Optimal Number of Iterations

To maximize $P(\omega, r)$, the argument of the sine function $\theta \left(\frac{1}{2} + r \right)$ must be $\frac{\pi}{2}$. Solving for r :

$$\theta \left(\frac{1}{2} + r \right) = \frac{\pi}{2},$$

$$r = \frac{\pi}{2\theta} - \frac{1}{2}.$$

For small θ , we use the approximation:

$$\sin \left(\frac{\theta}{2} \right) \approx \frac{\theta}{2} = \sqrt{\frac{1}{N}}$$

Thus, the optimal number of iterations is approximately:

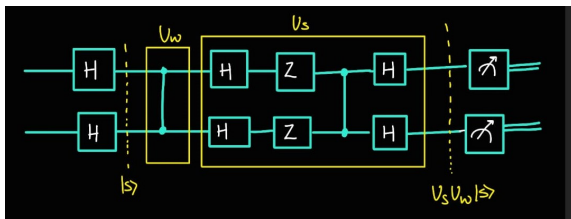
$$r_{\text{opt}} \approx \frac{\pi}{4} \sqrt{N}.$$

2-Qubit Implementation of Grover's Algorithm

Key Steps in the Algorithm:

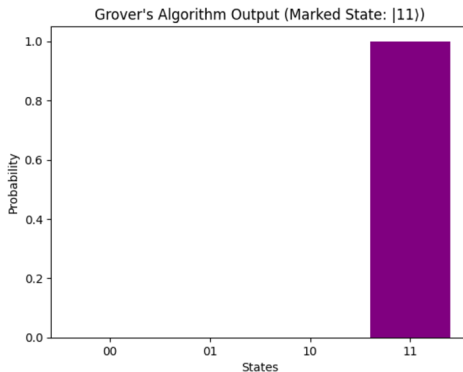
- **Initialization:** Apply Hadamard gates on $|00\rangle$ to create an equal superposition of all possible states.
- **Oracle:** Marks the target state.
- **Grover Diffusion Operator:** Reflects the states about the average amplitude to amplify the marked state.

Circuit for marked state = $|11\rangle$



Output for marked state = $|11\rangle$

Enter the marked state (0 to 3): 3



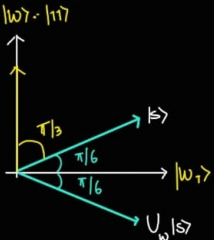
Quantum Circuit:

0: $\text{---H---}\bullet\text{---H---Z---}\bullet\text{---H---}$ (Probs
 1: $\text{---H---Z---H---Z---Z---H---}$ (Probs

Figure: Grover's Algorithm with two qubits

Verification of the Circuit for marked state = $|11\rangle$

$$\begin{aligned}
 |s\rangle &= H^{\otimes 2} |00\rangle = \frac{1}{2} \left[|00\rangle + |01\rangle + |10\rangle + |11\rangle \right] = \frac{1}{\sqrt{4}} \sum_{x \in \{0,1\}^2} |x\rangle = \frac{1}{\sqrt{2^2}} \sum_{x=0}^3 |x\rangle \\
 |w\rangle &= \frac{1}{\sqrt{3}} \left(|00\rangle + |01\rangle + |10\rangle \right) & |\langle w | s \rangle|^2 &= \frac{1}{4} \\
 & & Z^{\otimes 2} &= \mathcal{D}(1, -1, -1, 1) \\
 U_w &= CZ: \mathcal{D}(1, 1, 1, -1) = \mathcal{D}((-1)^0, (-1)^0, (-1)^1, (-1)^1) \\
 U_s &= H^{\otimes 2} Z^{\otimes 2} CZ H^{\otimes 2} = H^{\otimes 2} \left(2|00\rangle\langle 00| - I_4 \right) H^{\otimes 2} \\
 &= 2|s\rangle\langle s| - I_4
 \end{aligned}$$

Verification of the Circuit for marked state = $|11\rangle$


GEOMETRIC REPRESENTATION
FOR $|w\rangle = |11\rangle$

$$U_s U_w |s\rangle = \begin{bmatrix} \cos \pi/3 & -\sin \pi/3 \\ \sin \pi/3 & \cos \pi/3 \end{bmatrix} \begin{bmatrix} \cos \pi/6 \\ \sin \pi/6 \end{bmatrix}$$

$$= \begin{bmatrix} 1/2 & \sqrt{3}/2 \\ \sqrt{3}/2 & 1/2 \end{bmatrix} \begin{bmatrix} \sqrt{3}/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$R\left(\frac{\pi}{3}\right) = 0|w_\perp\rangle + 1|w\rangle$$

$$= \underline{|w\rangle = |11\rangle}$$

Hence, verified!

Implementation with 10 qubits

- Demonstrating the grover's algorithm for 10 qubit case while marking two out of them (decimal 1 and 2)

```
Enter the marked states as binary strings of length 10.  
Enter one state per line and type 'done' when finished:  
Marked state: 0000000001  
Marked state: 0000000010  
Marked state: done
```

```
Measured probabilities for each state:  
State 0000000000: Probability 0.000004  
State 0000000001: Probability 0.497896  
State 0000000010: Probability 0.497896  
State 0000000011: Probability 0.000004  
State 0000000100: Probability 0.000004  
State 0000000101: Probability 0.000004  
State 0000000110: Probability 0.000004  
State 0000000111: Probability 0.000004  
State 0000001000: Probability 0.000004  
State 0000001001: Probability 0.000004  
State 0000001010: Probability 0.000004  
State 0000001011: Probability 0.000004  
State 0000001100: Probability 0.000004  
State 0000001101: Probability 0.000004  
State 0000001110: Probability 0.000004  
State 0000001111: Probability 0.000004  
State 0000010000: Probability 0.000004  
State 0000010001: Probability 0.000004
```

Figure: Grover's Algorithm for ten qubit case

What is NP-Complete?

What is NP-Complete?

- **NP (Nondeterministic Polynomial Time):** Class of problems for which a given solution can be verified in polynomial time but a polynomial time algorithm does not exist to solve the problem.
- **NP-Complete:** A problem is NP-complete if:
 - It belongs to the class NP.
 - Every other problem in NP can be reduced to it in polynomial time.
- SAT was the first problem proven to be NP-complete (Cook-Levin theorem).

Introduction to SAT and 3-SAT

SAT Problem Overview:

- **SAT (Satisfiability Problem):** Given a Boolean formula, determine if there exists an assignment of variables such that the formula evaluates to **true**.
- SAT is an NP-complete problem, making it fundamental in the study of computational complexity.

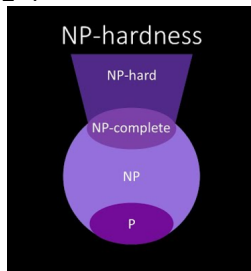
3-SAT Problem:

- **3-SAT:** A specific case of SAT where each clause in the formula contains exactly **three literals** (variables or their negations).
- The formula is expressed in **Conjunctive Normal Form (CNF)**, where:
 - Each clause is a disjunction (**OR**) of three literals.
 - The entire formula is a conjunction (**AND**) of these clauses.
- Example clause: $(x_1 \vee \neg x_2 \vee x_3)$

Number of Clauses and Assignments in 3-SAT

Number of Clauses and Assignments:

- **Clauses:** A 3-SAT problem has m clauses, each with 3 literals.
- **Variables:** The problem involves n variables (e.g., $x_1, x_2, x_3, \dots, x_n$).
- **Total Assignments:** The number of possible assignments for n variables is 2^n .



3-SAT Problem with Grover's Algorithm

- Grover's algorithm was used to search for an assignment of variables that satisfies the given Boolean formula.
- Encoded 3-SAT clauses into Grover's Oracle.

$$\begin{aligned} G(x) = & (x_1 \vee x_2 \vee x_3) \wedge \\ & (\neg x_1 \vee x_2 \vee x_3) \wedge \\ & (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge \\ & (\neg x_1 \vee \neg x_2 \vee x_3) \wedge \\ & (x_1 \vee x_2 \vee \neg x_3) \wedge \\ & (\neg x_1 \vee x_2 \vee \neg x_3) \end{aligned}$$

Steps in Grover's Algorithm for 3-SAT

Steps Involved:

■ Step 1: Initialization in Superposition:

- Initialize all qubits in a superposition state using Hadamard gates.

■ Step 2: Oracle (Problem-Specific):

- The oracle is a quantum operation that marks the solutions satisfying the 3-SAT formula by flipping the sign of their amplitude.

■ Step 3: Grover Diffusion Operator:

- Amplifies the probability amplitude of the marked solution by reflecting all states about the average amplitude.

■ Step 4: Iteration and Measurement:

- Repeat steps 2 and 3 multiple times to increase the probability of measuring the correct solution.

3-SAT Problem with Grover's Algorithm: Results

Results:

■ Grover's Algorithm Performance:

After applying Grover's search algorithm, we observe a high probability of finding the correct satisfying assignment for a given 3-SAT formula.

```
Starting 3-SAT solver using Grover's algorithm...  
Number of variables: 3
```

```
Oracle creation time: 0.0002 seconds
```

```
Results:
```

```
Found solution: 011
```

```
Probability: 0.5110
```

```
Solution is valid: 1
```

```
Variable assignments:
```

```
x1 = 0
```

```
x2 = 1
```

```
x3 = 1
```

```
All valid solutions found: ['010', '011']
```

```
Number of valid solutions: 2
```

```
Timing information:
```

```
Oracle creation time: 0.0002 seconds
```

```
Solution time: 0.0073 seconds
```

```
Total execution time: 0.0075 seconds
```

```
Verification: Solution found is correct and matches c
```

LINKS

Resources:

- **Report Link:** You can access the full report for this implementation at: 3-SAT Report
- **Code for Grover's Algorithm:** You can access the code for Grover's Algorithm at: Grover's Algorithm Code

Contributions

■ Ishanya:

- Mathematics of Grover's Algorithm (Slides 4-9)
- Worked on the analysis and implementation of 2 Qubit case Grover's Algorithm. (Slides 10-13)
- Implemented Grover's algorithm for 3-SAT problem and verified using classical brute-force algorithm. (Slides 15-20)
- Worked on the majority report and slides (4-13, 15-21)

■ Rohan:

- Mathematics and Implementation of Grover's Algorithm using PennyLane for $N = 4$.
- Extended the implementation to handle $N = 10$ case (base case).
- Conducted simulations and verified results with theoretical predictions.
- Contributed to overall documentation and presentation.

References

- M.A. Nielsen and I.L. Chuang, "Quantum Computation and Quantum Information," Cambridge University Press.
- Grover, L. K. (1996). A fast quantum mechanical algorithm for database search.
- PennyLane Documentation:
https://pennylane.ai/qml/demos/tutorial_grovers_algorithm
- Stephen Jordan, "Quantum Algorithm Zoo,"
<https://quantumalgorithmzoo.org>