CS 506 : Intro to Quantum Computing

Scribe: Nidhi Ravishankar Jois

Date: 11/12/2025

1 Grover's Search Algorithm

Suppose we need to find an element in an unstructured database of N elements. We are given a black-box query function

$$f : \{0, 1, \ldots, N-1\} \rightarrow \{0, 1\},$$

The goal is to find the unique x^* such that

$$f(x^*) = 1.$$

Classically (non-quantum), deterministic algorithms need N queries and randomized algorithms require $\Omega(N)$ queries to find x^* with a success probability $\geq 2/3$.
Grover's Search Algorithm is a quantum algorithm that solves this unstructured search problem using only $O(\sqrt{N})$ queries and $O(\sqrt{N} logN)$ 1-qubit and 2-qubit gates with a success probability $\geq 2/3$.

1.1 Example: Database Search

Consider an array $A[0], \ldots, A[N-1]$, and a query q. The task is to find x^* such that $A[x^*] = q$.

We define the function as:

$$f(x) = \begin{cases} 1 & \text{if A}[x^*] = \text{q}, \\ 0 & \text{otherwise.} \end{cases}$$

Function f(x): if A[x^*] = q then return 1 else return 0

Classically, searching requires N comparisons, whereas Grover's algorithm finds x^* in approximately $O(\sqrt{N})$ queries.

1.2 Example: Satisfiability

We are given n variables $x_1, \ldots, x_n$ and m clauses, where each clause is an OR of literals.
Goal: Find an assignment $x^* = (x_1^*, \ldots, x_n^*)$ such that each clause is True(1).

Example:

- $n = 3,\ m = 4$
- Variables: x_1, x_2, x_3
- Clauses:

$$\begin{aligned} C_1 &= \bar{x_1} \lor x_2 \\ C_2 &= x_1 \lor x_2 \lor \bar{x_3} \\ C_3 &= \bar{x_2} \lor x_3 \\ C_4 &= \bar{x_1} \lor x_3 \end{aligned}$$

Where $f(x_1 x_2 x_3) = 1$ if all clauses are satisfied, else 0.
For example: for $f(101)$ it returns 0.
The search space is defined by $f : \{0, 1, \ldots, 2^n - 1\} \to \{0, 1\}$ and $N = 2^n$.
Grover's algorithm solves this problem by using $O(\sqrt{N}) = O(\sqrt{2^n}) = O(2^{n/2})$ queries to f.

Classical algorithms take exponential time $O(2^n)$ to solve the satisfiability problem.
Grover's algorithm solves it faster in $O(2^{n/2})$ but still in exponential time although there is a quadratic speedup.
Grover's search algorithm does not solve NP-complete problems in polynomial time!

2 Key Components of Grover's Search Algorithm

2.1 Oracle Gate

The oracle gate in Grover's algorithm is the U_f gate.
It marks the correct solution by inverting its amplitude.

$U_f : |x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$

$U_f : |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = (-1)^{f(x)}|x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$

When $x = x^*$, $(-1)^{f(x^*)} = (-1)^1 = -1$

When $x \neq x^*$, $(-1)^{f(x)} = (-1)^0 = +1$

2.2 Diffusion Operator

After the oracle gate (U_f gate), we use the diffusion operator to increase the amplitude of x^* while reducing the others.

3 Amplitude Amplification

Given a set of vectors $|X_1\rangle, |X_2\rangle, \ldots, |X_N\rangle$ in $\mathbb{R}^d$ and a target vector $|Y\rangle$ in $\mathbb{R}^d$,ε we want to find $|X_i\rangle$ such that $\langle X_i|Y\rangle \geq \varepsilon$

3.1 Assumptions

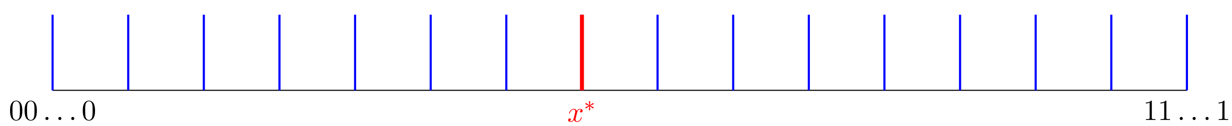
- N is a power of 2 and $n = log_2 N$.
- There is a unique x^* such that $f(x^*) = 1$

3.2 Initial State

We begin with a uniform superposition where all states have equal amplitudes.

$$\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle$$

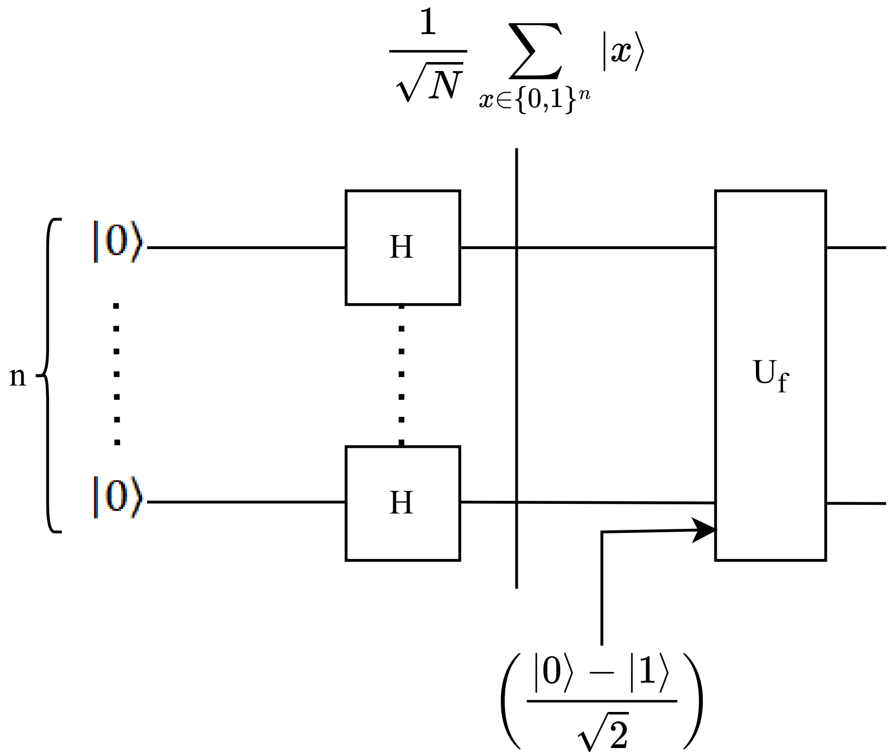
The probability of finding the solution x^* is $\frac{1}{N}$.



3.3 Applying the Oracle Gate (U_f)

The first step is to use the oracle gate or U_f gate on the initial state $\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle$.

$$U_f \left(\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle \right) = \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle$$



Mathematically, this operation translates to:

$$
\begin{aligned}
U_f\left(\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle\right) &= \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)}|x\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} (-1)^{f(x)}|x\rangle \\
&= \frac{1}{\sqrt{N}} \left[(-1)^{f(x^*)}|x^*\rangle + \sum_{x \neq x^*} (-1)^{f(x)}|x\rangle\right] \\
&= \frac{1}{\sqrt{N}} \left[(-1)^{1}|x^*\rangle + \sum_{x \neq x^*} (-1)^{0}|x\rangle\right] \\
&= \frac{1}{\sqrt{N}} \left[-|x^*\rangle + \sum_{x \neq x^*} |x\rangle\right] \\
&= -\frac{1}{\sqrt{N}}|x^*\rangle + \frac{1}{\sqrt{N}} \sum_{x \neq x^*} |x\rangle
\end{aligned}
$$

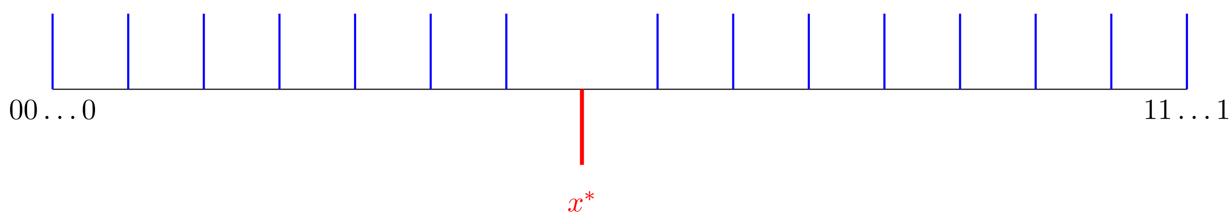
- For the solution x^*: $f(x^*) = 1 \Rightarrow (-1)^{f(x^*)} = (-1)^1 = -1$
- For all other $x \neq x^*$: $f(x) = 0 \Rightarrow (-1)^{f(x)} = (-1)^0 = +1$

This phase flip marks the solution x^* with a negative amplitude while preserving the positive amplitudes of all other states.

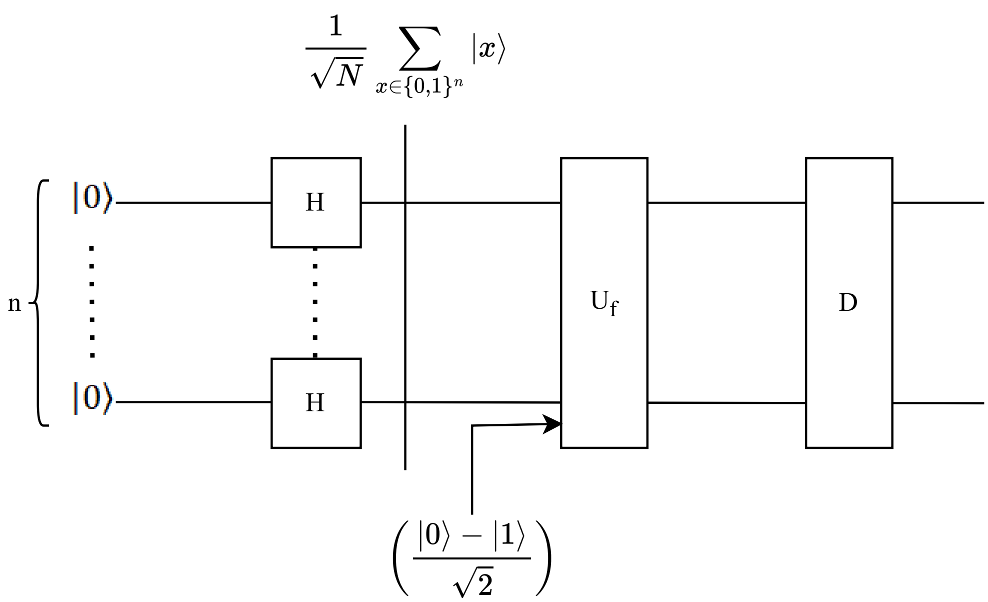
After applying the oracle (U_f) gate:

$$\alpha_{x^*} = -\frac{1}{\sqrt{N}} \quad \text{(amplitude of the solution } x^*\text{)}$$

$$\alpha_x = +\frac{1}{\sqrt{N}} \quad \text{for all } x \neq x^* \text{ (amplitude of the other states)}$$



3.4 Amplitude Amplification using the Diffusion Operator



Let μ be the average of all α_x*s.

Then,

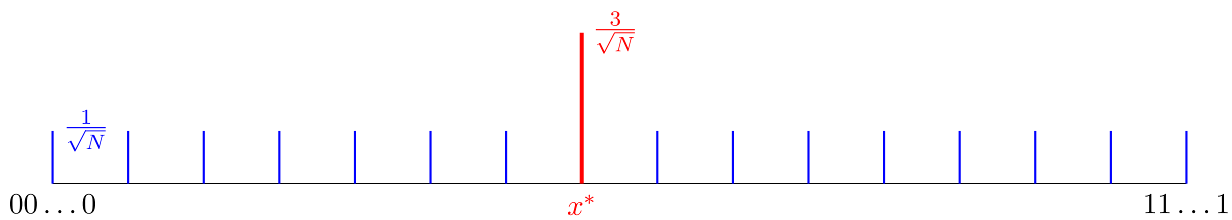
$$\mu = \frac{\frac{-1}{\sqrt{N}} + \frac{N-1}{\sqrt{N}}}{N} = \frac{N-2}{N\sqrt{N}} \approx \frac{1}{\sqrt{N}}$$

Suppose we design a quantum circuit such that α_x becomes $2\mu - \alpha_x$ for all x.

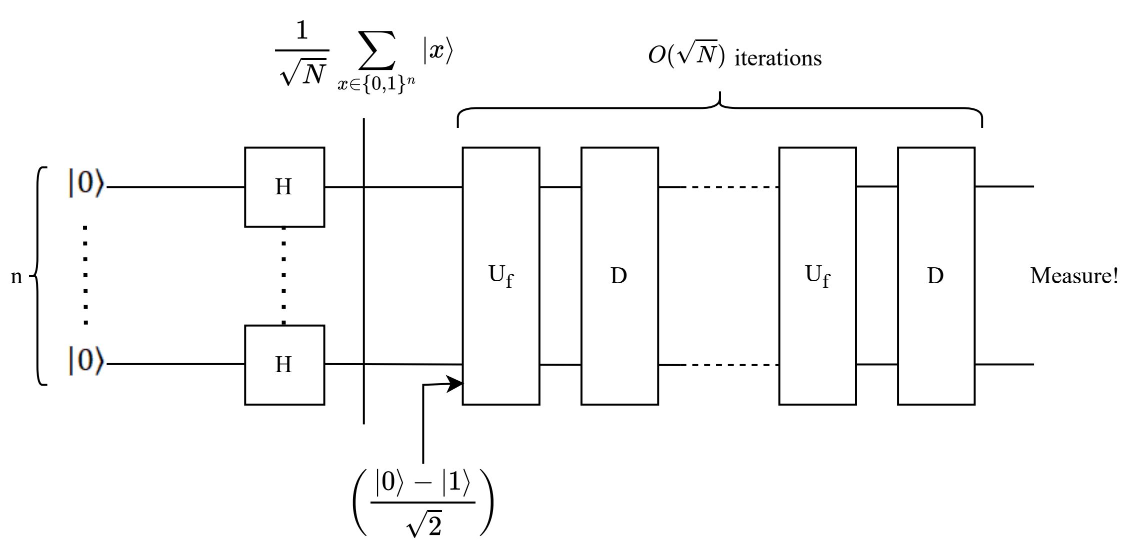
$$D\left(\sum_{x\in\{0,1\}^n} \alpha_x|x\rangle\right) = \sum_{x\in\{0,1\}^n} (2\mu - \alpha_x)|x\rangle$$

When $x = x*$, α_x* becomes $2\mu - \alpha_x* \approx \frac{2}{\sqrt{N}} - \left(-\frac{1}{\sqrt{N}}\right) = \frac{3}{\sqrt{N}}$

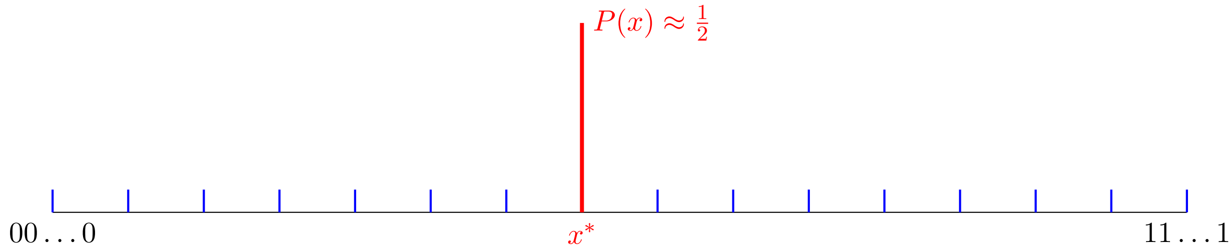
When $x \neq x*$, α_x becomes $2\mu - \alpha_x \approx \frac{2}{\sqrt{N}} - \frac{1}{\sqrt{N}} = \frac{1}{\sqrt{N}}$



3.5 Multiple Iterations of Amplitude Amplification



We repeat the oracle (U_f) and diffusion operator combination multiple times until the probability of finding x^* becomes $\frac{1}{2}$ (after roughly $\frac{\sqrt{N}}{4}$ iterations). Then we stop!



Why do we stop when the amplitude of x^* becomes half?

- If we continue iterating beyond this point, the amplitude α_{x^*} becomes very large.
- The average amplitude becomes negative.

$$\mu = \frac{-\alpha_{x^*} + \sum_{x \neq x^*} \alpha_x}{N} < 0$$

- This happens when $\alpha_{x^*} > \sum_{x \neq x^*} \alpha_x$
- When the average μ becomes negative, it works against our efforts. 2μ is negative and α_x* is very large. So the result, $2\mu - \alpha_x*$ decreases, reducing the amplitude of $x*$.
- Therefore the probability of finding x^* decreases instead of increasing!