Grover's Algorithm

Kharkiv Winter Quantum School 2025

The material was prepared by
Pavlo Pyrohov
PhD Computer Science student
Department of Theoretical and Applied Informatics
Faculty of Mathematics and Informatics
V. N. Karazin Kharkiv National University
pavlo.pyrohov@student.karazin.ua
mrvogorip@gmail.com

January 10, 2025

Grover's Algorithm

Lov Grover, working for the research (R&D) company "Bell Labs", in 1996 invented the first quantum algorithm for searching in an unstructured data set, the algorithm finds the correct solution with a high probability.

The meaning of Grover's algorithm is to "amplitude amplification" of the target state by reducing the amplitude of all other states.

Lov Grover

Introduction to Grover's Algorithm and the Search Problem

Grover's algorithm speeds up searching in unstructured data in fewer steps than any classical algorithm.

The search problem can be expressed in terms of an abstract function f(x) that takes search elements x.

If element x is a solution to the search problem, then f(x)=1.

If element x is not a solution, then f(x)=0.

The search problem is to find an element x0 such that f(x0)=1.

The problem solved by Grover's algorithm

It can be expressed as follows, this classical function:

$$f(x) : \{0,1\}^n \to \{0,1\},$$

where n is the bit size of the space, you need to know x0 at the input, for which f(x0)=1.

$$N = 2^n$$

In classical computing, the search is generally O(N) attempts.

Grover's quantum algorithm can solve this problem much faster, providing a quadratic speedup of O(√N).

Example of calculating complexity

4 bytes is 32 bits (int data type)

$$N = 2^{32} = 4\,294\,967\,295$$

The classical algorithm requires an average of two billion attempts

$$\frac{4\,294\,967\,295}{2} \approx 2^{31} = 2\,147\,483\,648$$

Grover's algorithm requires much fewer attempts

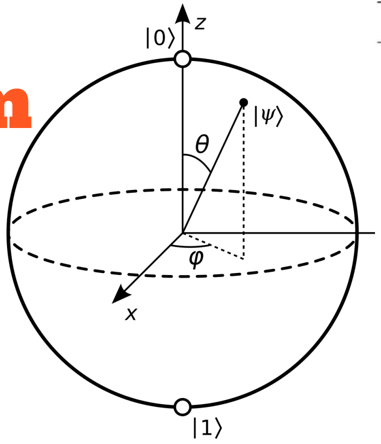
$$\frac{\pi}{4}\sqrt{4\,294\,967\,295} = \frac{\pi}{4} \times 2^{16} \approx 51\,472$$

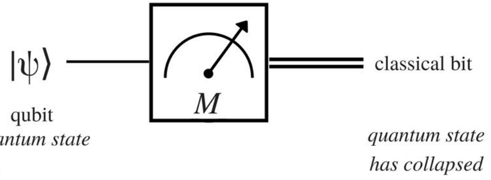
A small addition

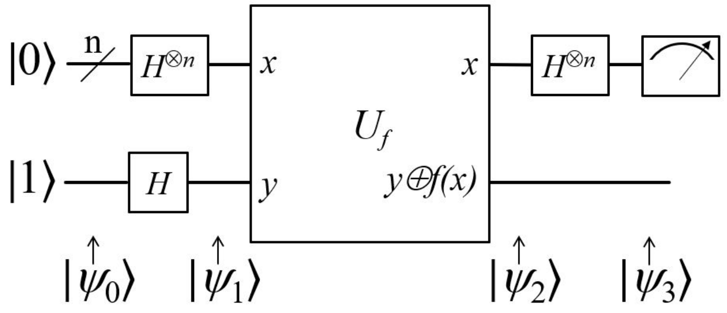
- Grover's algorithm allows searching a list in time less than the size of the list.
- Grover's algorithm itself does not work with integers, but a sequence of bits can represent an integer (binary encoding)
- Many sources often refer to Grover's algorithm as a database search, but these analogies work well from an illustrative point of view, but are not very practical from a real-world application perspective, given the limitations of current quantum hardware.
- Grover's algorithm could also be very useful for working on problems where a dataset will be created on a quantum computer while another algorithm is running.
- Advanced Encryption Standard (AES)

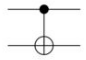
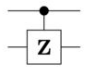
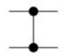
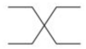
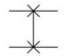
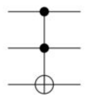
Quantum algorithm (Quantum circuit)

- Qubits
- Gates
- Measurement
- Oracle







Operator	Gate(s)		Matrix
Pauli-X (X)	$\boxed{\mathbf{X}}$	$\oplus$	$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$
Pauli-Y (Y)	$\boxed{\mathbf{Y}}$		$\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$
Pauli-Z (Z)	$\boxed{\mathbf{Z}}$		$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$
Hadamard (H)	$\boxed{\mathbf{H}}$		$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$
Phase (S, P)	$\boxed{\mathbf{S}}$		$\begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}$
$\pi/8$ (T)	$\boxed{\mathbf{T}}$		$\begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}$
Controlled Not (CNOT, CX)			$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$
Controlled Z (CZ)			$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -1\end{bmatrix}$
SWAP			$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)			$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$

Oracle for Grover's algorithm

Oracle (black box) is an operation that accepts input data and returns output data, used as an input parameter for quantum algorithms.

$\mathbf{U}_\omega$

For Grover's algorithm, we need an Oracle that can determine whether a given sequence of qubits is the result of the search, this is a phase oracle that changes the phase of the qubits using a Controlled Z-Gate, where ω corresponds to the solution we are looking for.

An important thing to note here is that the oracle does not need to know the answer, it only needs to be able to recognize it.

$$\mathrm{U}_\omega |x\rangle = \begin{cases} |x\rangle & \text{if } f(x) = 0 \\ -|x\rangle & \text{if } f(x) = 1 \end{cases} = \begin{cases} |x\rangle & \text{if } x \neq \omega = 0 \\ -|x\rangle & \text{if } x = \omega = 1 \end{cases}$$

$$\mathbf{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

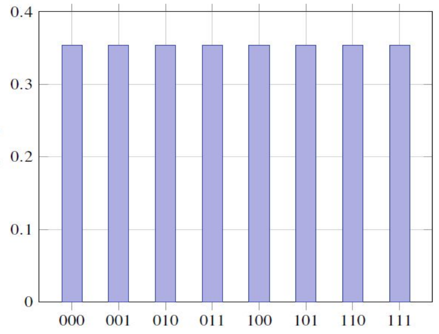
$$\mathrm{CZ}\,|00\rangle = |00\rangle, \quad \mathrm{CZ}\,|01\rangle = |01\rangle, \quad \mathrm{CZ}\,|10\rangle = |10\rangle, \quad \mathrm{CZ}\,|11\rangle = -\,|11\rangle$$

First step of Grover's algorithm

To achieve any computational advantage over classical computation, the input to the oracle must already be in a superposition of all possible states.

Since at the beginning of the algorithm we have no idea where exactly the answer ω is located, a uniform superposition with equal weighting of each possibility is a logical way to start the algorithm.

Therefore, the initial step in Grover's algorithm would be a Hadamard transformation of all qubits



$$|\psi_1\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle$$

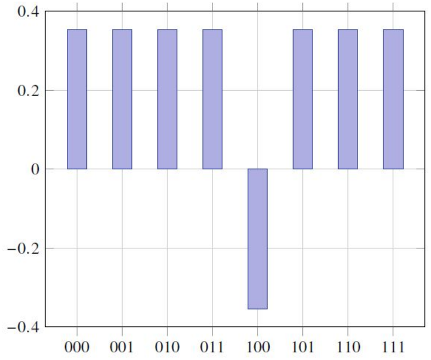
$$\mathbf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

Second step

The next step is to query the oracle Uω, passing the result of the first step as the input state. At this point, the overall state of the system can be described:

$$|\psi_2\rangle = \mathbf{U}_\omega |\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} (-1)^{f(x)} |x\rangle$$

We can also rewrite the state to better illustrate the oracle effect.

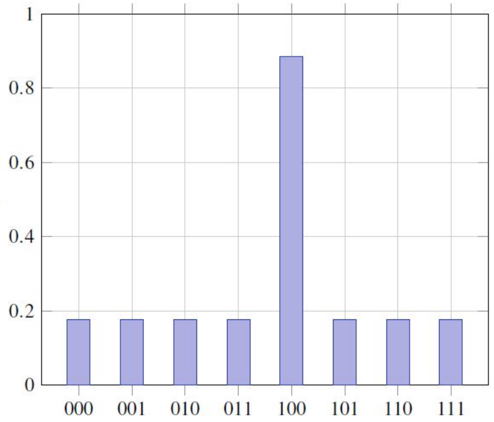


$$|\psi_2\rangle = -\frac{1}{\sqrt{N}} |\omega\rangle + \frac{1}{\sqrt{N}} \sum_{\substack{x \in \{0,1\}^n \\ x \neq \omega}} |x\rangle$$

Third Step Grover's Oracle

It is also the Grover diffusion operator or amplitude amplification or inversion to mean transformation.

$$\mathbf{U}_\psi = (2|\psi\rangle\langle\psi| - \mathbf{I})$$



If we look at this as a vector space, then:

I is the identity matrix, $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$2|\psi\rangle\langle\psi|$ is the projection matrix, with an amplitude twice the original.

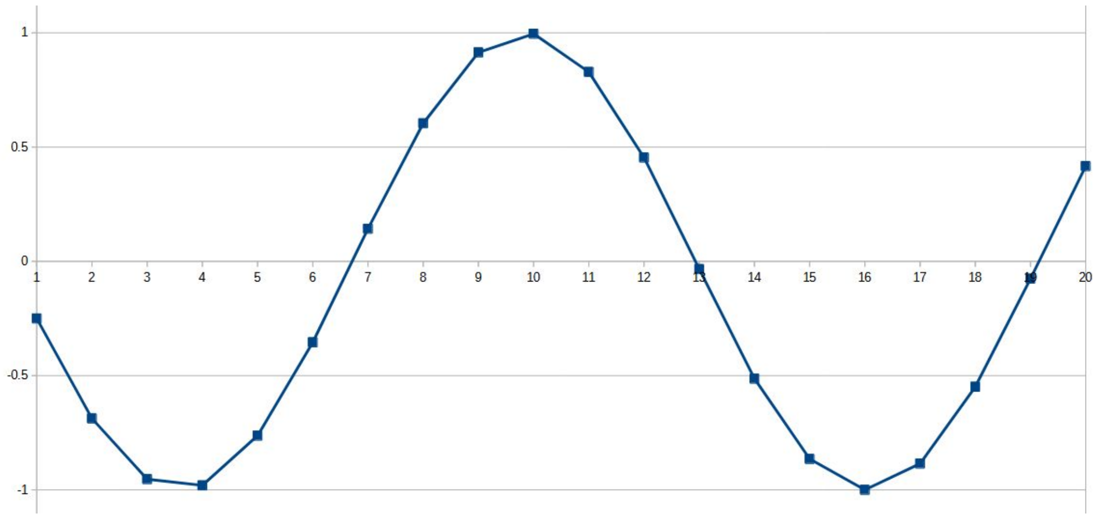
$$|\psi\rangle\langle\psi| = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Amplitude Amplification

https://github.com/MrVogorip/GroverSearchAlgorithm/blob/main/AmplitudeAmplificationExample.cs

https://dotnetfiddle.net/

If we interpret these numbers as the amplitudes of the probabilities of obtaining the correct solution, we can see that for a certain number of iterations we can obtain a probability close to 1.



Overview of Grover's algorithm steps

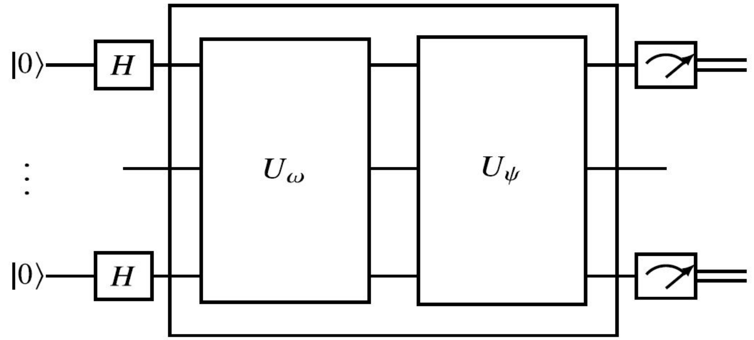
1. Prepare the input state, apply the transformation H to create a uniform superposition of all possible states.

2. Apply the Oracle Uω (marking)

$$\mathbf{G} = \mathbf{U}_{\psi}\mathbf{U}_{\omega}$$

3. Apply the Oracle Uψ (inversion to mean transformation)

4. Measure the register |x⟩ to obtain the result |ω⟩ with very high probability

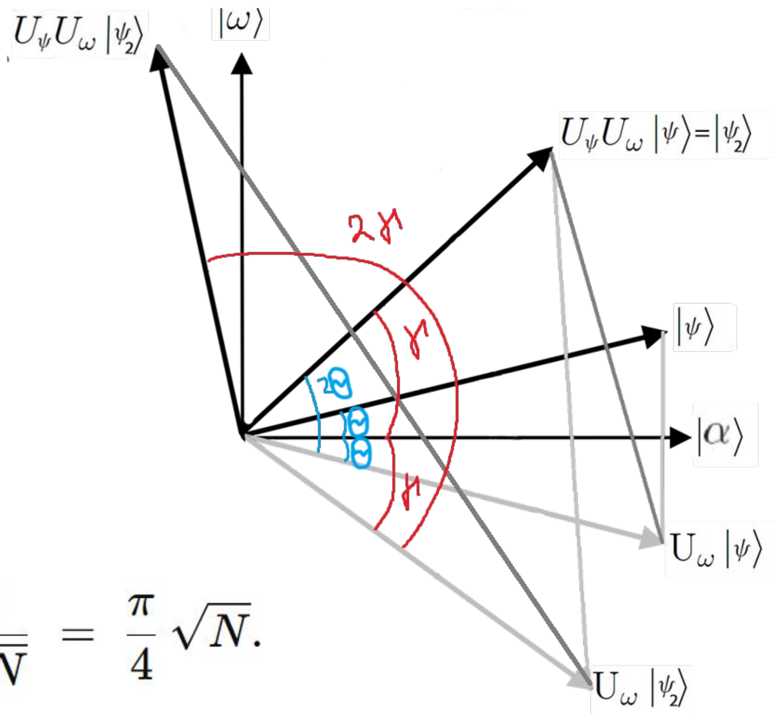


Geometric interpretation of Grover's algorithm

When the qubits are in equal superposition, they are represented by the vector |ψ⟩.

Initially, the vector is quite close to the |α⟩ axis, which is the "non-solution" axis.

The |ω⟩ axis is the "solution" axis — the closer the vector is to this axis, the higher the probability that the qubits will be the correct solution when measured.



$$\sin\theta \approx \frac{1}{\sqrt{N}} \qquad G = 2\theta \qquad \frac{\pi/2}{2\theta} = \frac{\pi/2}{2/\sqrt{N}} = \frac{\pi}{4}\sqrt{N}.$$

$U\omega$ and $U\psi$

$$U_\omega = \mathrm{X}^{\otimes n}\,\mathrm{CZ}\,\mathrm{X}^{\otimes n}$$

Apply X-transformation to all corresponding bits that correspond to 0 in the binary solution mask
Use CZ gate to invert state
Apply X-transformation again to return the register to its original state after inversion

$$U_\psi = H^{\otimes n} X^{\otimes n} \mathrm{CZ} X^{\otimes n} H^{\otimes n}$$

Apply the Hadamard transform H to each qubit.
Apply the X-transform to each qubit.
Use a CZ gate to invert the state
Apply the X-transform again.
Apply the Hadamard transform H to each qubit again.

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Example with 2 qubits where $|10\rangle$ was marked by oracle Uω as a solution

Initial state after Uω

$$|\psi_0\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle - |10\rangle + |11\rangle\right).$$

Apply (H⊗H)

$$|\psi_1\rangle = (H \otimes H)|\psi_0\rangle$$

$$|\psi_1\rangle = \frac{1}{2}\left[(H \otimes H)|00\rangle + (H \otimes H)|01\rangle - (H \otimes H)|10\rangle + (H \otimes H)|11\rangle\right]$$

$$(H \otimes H)|00\rangle = \tfrac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$
$$(H \otimes H)|01\rangle = \tfrac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$
$$(H \otimes H)|10\rangle = \tfrac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)$$
$$(H \otimes H)|11\rangle = \tfrac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)$$

$$|\psi_1\rangle = \frac{1}{2}\left(|00\rangle - |01\rangle + |10\rangle + |11\rangle\right)$$

$X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle \qquad \mathrm{CZ}|00\rangle = |00\rangle, \quad \mathrm{CZ}|01\rangle = |01\rangle, \quad \mathrm{CZ}|10\rangle = |10\rangle, \quad \mathrm{CZ}|11\rangle = -|11\rangle$

$$|\psi_1\rangle = \tfrac{1}{2}\big(|00\rangle - |01\rangle + |10\rangle + |11\rangle\big)$$

Apply (X⊗X)

$$\begin{aligned}|\psi_2\rangle &= (X\otimes X)\,|\psi_1\rangle \\ &= \tfrac{1}{2}\Big[(X\otimes X)|00\rangle - (X\otimes X)|01\rangle + (X\otimes X)|10\rangle + (X\otimes X)|11\rangle\Big]\end{aligned}$$

$$(X\otimes X)|00\rangle = |11\rangle, \quad (X\otimes X)|01\rangle = |10\rangle, \quad (X\otimes X)|10\rangle = |01\rangle, \quad (X\otimes X)|11\rangle = |00\rangle$$

$$|\psi_2\rangle = \tfrac{1}{2}\big(|00\rangle + |01\rangle - |10\rangle + |11\rangle\big)$$

Apply CZ

$$\begin{aligned}|\psi_3\rangle &= \mathrm{CZ}\,|\psi_2\rangle \\ &= \tfrac{1}{2}\Big[\mathrm{CZ}|00\rangle + \mathrm{CZ}|01\rangle - \mathrm{CZ}|10\rangle + \mathrm{CZ}|11\rangle\Big] \\ &= \tfrac{1}{2}\Big[|00\rangle + |01\rangle - |10\rangle - |11\rangle\Big].\end{aligned}$$

$$|\psi_3\rangle = \frac{1}{2}\Big[|00\rangle + |01\rangle - |10\rangle - |11\rangle\Big]$$

Apply (X⊗X)

$$|\psi_4\rangle = (X \otimes X)\,|\psi_3\rangle$$
$$= \frac{1}{2}\Big[(X \otimes X)|00\rangle + (X \otimes X)|01\rangle - (X \otimes X)|10\rangle - (X \otimes X)|11\rangle\Big]$$

$$|\psi_4\rangle = \frac{1}{2}\left(|11\rangle + |10\rangle - |01\rangle - |00\rangle\right)$$

Apply (H⊗H)

$$|\psi_5\rangle = (H \otimes H)\,|\psi_4\rangle$$
$$= \frac{1}{2}\Big[(H \otimes H)|11\rangle + (H \otimes H)|10\rangle - (H \otimes H)|01\rangle - (H \otimes H)|00\rangle\Big]$$

$$|\psi_5\rangle = \frac{1}{2}\Big[\frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle) + \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)$$
$$- \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) - \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)\Big]$$

$$|\psi_5\rangle = \frac{1}{4} \times \left(-4\,|10\rangle\right) = -\,|10\rangle$$

Software implementation

https://github.com/MrVogorip/GroverSearchAlgorithm/blob/main/GroverSearchAlgorithmExample.qs

https://quantum.microsoft.com/en-us/tools/quantum-coding

Q# is an open-source, high-level programming language for developing and running quantum algorithms.

It is included in the Azure Quantum Development Kit (QDK) along with a quantum simulator, resource estimator, and access to real quantum hardware via cloud services.

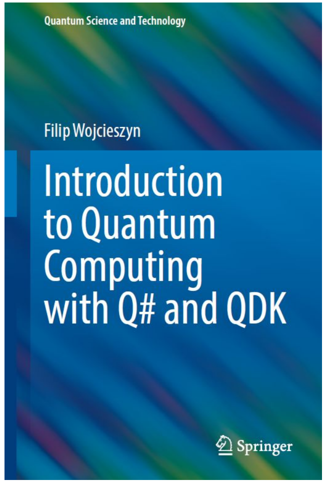
References

https://learn.microsoft.com/en-us/azure/quantum/tutorial-qdk-grovers-search

https://learn.microsoft.com/en-us/azure/quantum/concepts-grovers

Introduction to Quantum Computing with Q# and QDK Filip Wojcieszyn

Learn Quantum Computing with Python and Q#: A hands-on approach by Sarah C. Kaiser, Christopher Granade


Quantum Science and Technology
Filip Wojcieszyn
Introduction to Quantum Computing with Q# and QDK
Springer