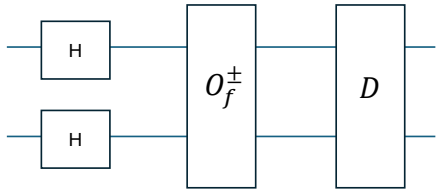
Quantum Computing – The Grover Algorithm for n Qubits

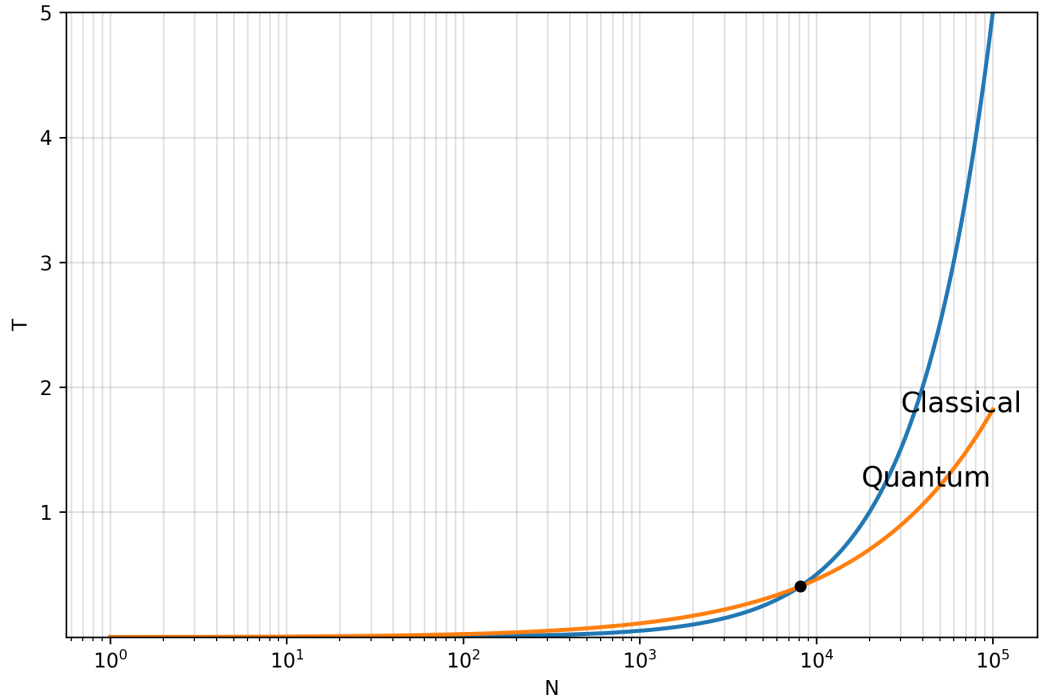
Bill Celmaster

July 14, 2026

Why doesn't a 2-qubit Grover algorithm prove anything?



- This circuit starts with $|00\rangle$ and ends with $|pwd\rangle$ in "one step".
- We say the classical circuit requires "four steps" (test each "pwd")
- But a quantum "step" takes longer than 4 classical "steps".
- We **want** to show for N possible passwords something like



Problem setup and algorithm outline

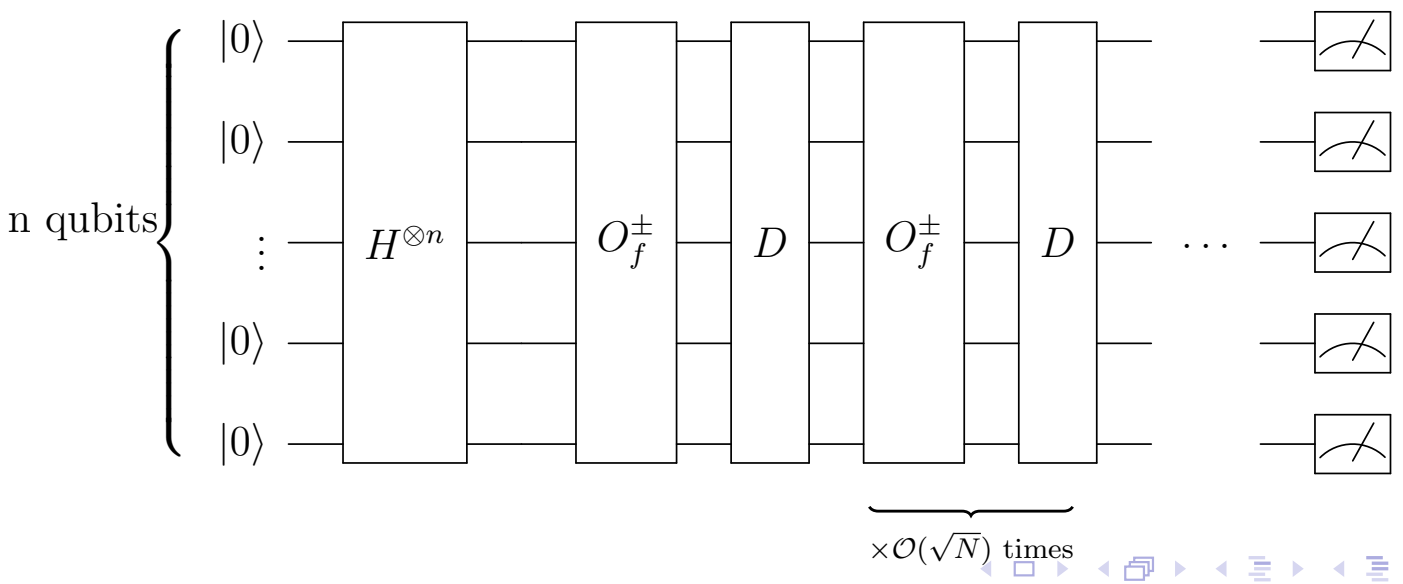
- Consider an n-qubit system with computational basis states:

$$|00...00\rangle,\ |000...01\rangle, ...\ |11..1101\rangle,\ |11...1111\rangle$$

- The total number of basis states is $N = 2^n$
- Choose a target ("password") – we'll use the following:

$$x^* = 100...001$$

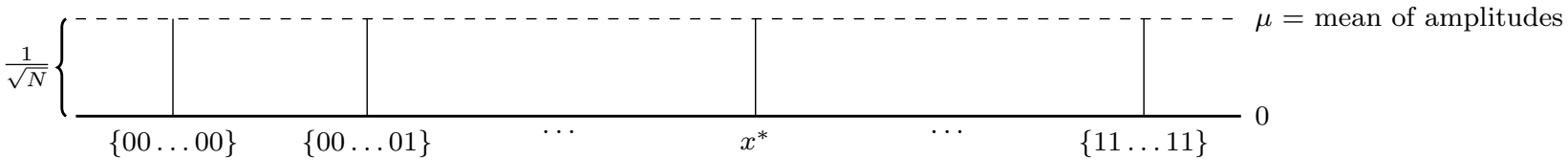
- Our goal is to identify the state $|100...001\rangle$.
- Our algorithm, to be explained, will be:



n-qubit Superposition

- Apply a Hadamard gate to all qbits
- Tensor product:

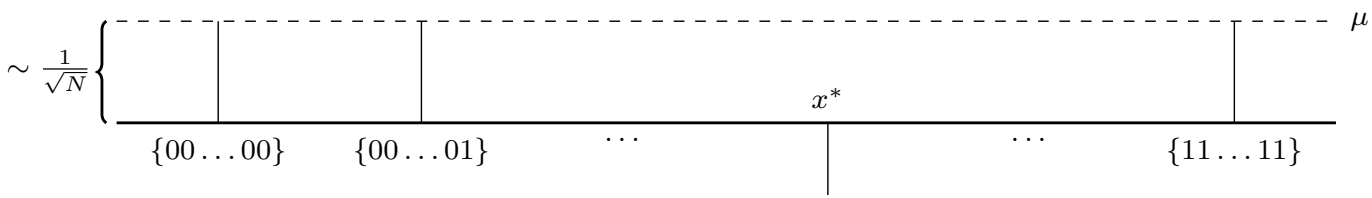
$$
\begin{aligned}
|00...00\rangle &= |0\rangle_0 \otimes ... \otimes |0\rangle_{n-1} \\
&\rightarrow \frac{1}{\sqrt{2}}(|0\rangle_0 + |1\rangle_0) \otimes ... \otimes \frac{1}{\sqrt{2}}(|0\rangle_{n-1} + |1\rangle_{n-1}) \\
&= \frac{1}{\sqrt{2^n}}(|00....00\rangle + |00...01\rangle + |11...11010\rangle + |11...1111\rangle) \\
&= \frac{1}{\sqrt{N}}(|00....00\rangle + |00...01\rangle + |11...11010\rangle + |11...1111\rangle)
\end{aligned}
$$



n-qubit Oracle

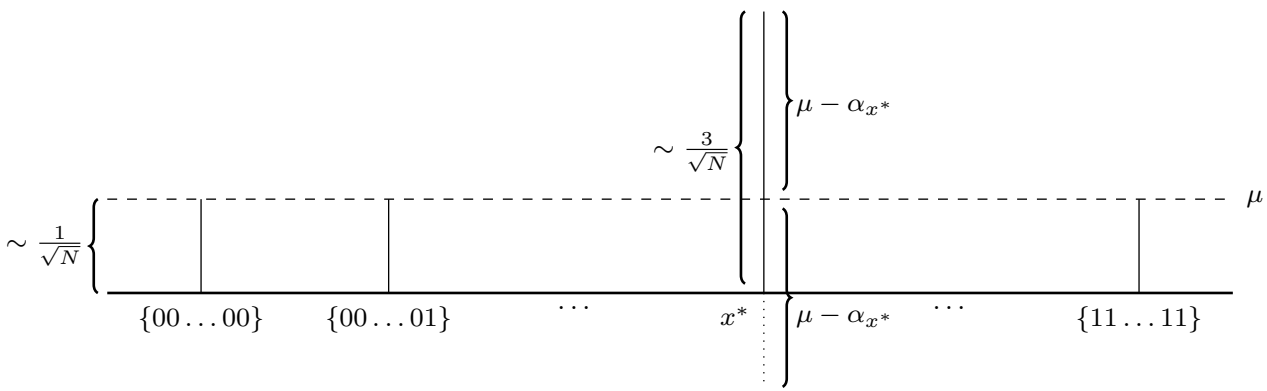
An example oracle gate $O_f^{\pm}$ is defined by:

- $|100...001\rangle \rightarrow -|100...001\rangle$
- $|i_0...i_{n-1}\rangle \rightarrow |i_0...i_{n-1}\rangle$ when $|i_0...i_{n-1}\rangle \neq |100...001\rangle$



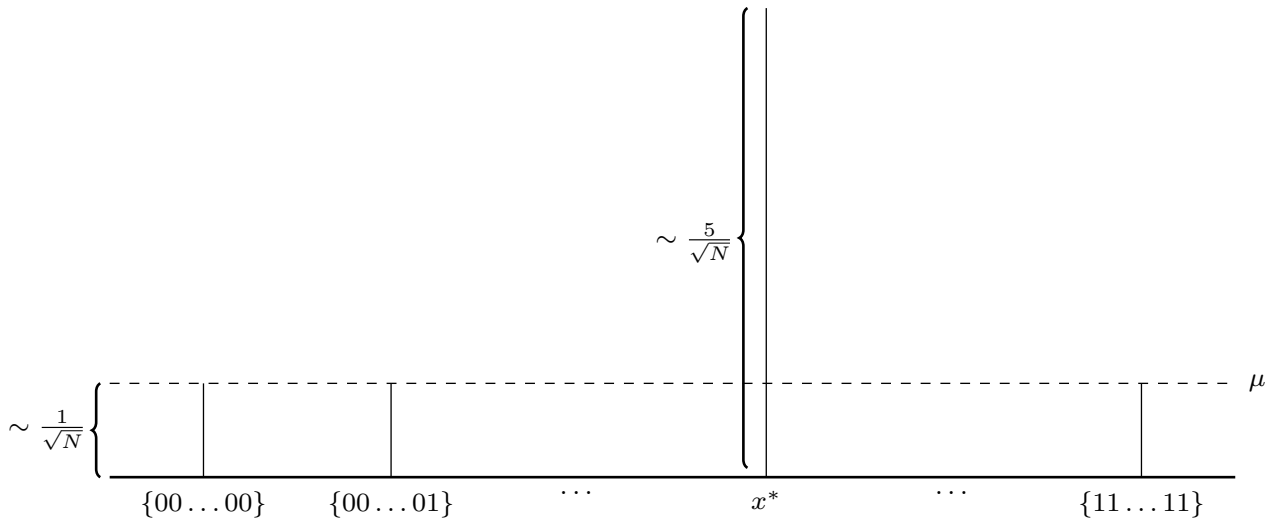
n-qubit Diffusion

- For ease of notation, relabel the basis vectors as $|x_i\rangle$
  - Example: $|1001\rangle$ is relabeled $|x_9\rangle$.
  - Then a vector $\psi\rangle$ has $|\psi\rangle = \sum_i^N \alpha_i |x_i\rangle$
- Define $\mu = \frac{1}{N} \sum_i^N \alpha_i$
- Define $\mathbf{D} : |\psi\rangle \rightarrow \sum_i^N (2\mu - \alpha_i) |x_i\rangle$
- Example : if $x_i \neq x^*$ then $\alpha_i = \frac{1}{\sqrt{N}}$ otherwise $\alpha_i = -\frac{1}{\sqrt{N}}$
  - $\mu = \left(1 - \frac{2}{N}\right) \frac{1}{\sqrt{N}}$
  - $\mathbf{D}|\psi\rangle = \sum_{i:\, x_i \neq x^*}^N \left(1 - \frac{4}{N}\right) \frac{1}{\sqrt{N}} |x_i\rangle + \left(3 - \frac{4}{N}\right) \frac{1}{\sqrt{N}} |x^*\rangle$



Repeat: oracle plus diffusion

- Oracle + Diffusion amplifies amplitude of $|x^*\rangle$ relative to other $|x_i\rangle$.
- Apply Oracle + Diffusion to $\mathbf{D}|\psi\rangle$

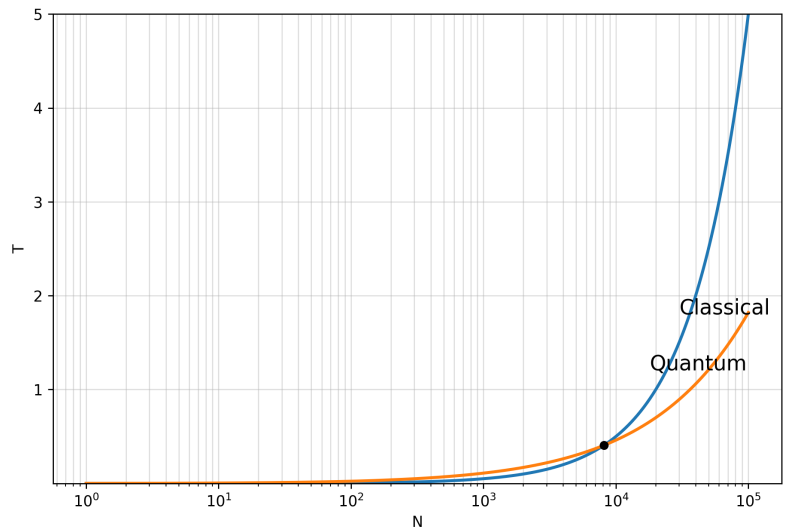


- Repeat again $\mathcal{O}(\sqrt{N})$ times so that amplitude of $|x^*\rangle$ is ≈ 1.
- Unlike 2-qubit case, answer is approximate.
  - But can repeat the whole thing to get arbitrary precision.

Does this prove that quantum is faster than classical?

NO!

- So far, we don't know how the time-per-gate changes with N
  - Suppose processing time for a diffusion gate is $k * 2^n = k * N$?
  - Then the time-to-solution grows with N, same as classical
    - And K will certainly be larger than for a classical gate
- **We will show** that the time-per-gate grows as $n = log(N)$.
- Also, remember there are $\approx \sqrt{N}$ oracle-diffusion repetitions.
- So the time-per-circuit grows as $\sqrt{N}log(N)$
- **THIS IS CALLED <u>COMPLEXITY ANALYSIS</u>**



Review of 2-qubit matrix representations of product gates

A matrix $\mathbf{M}^{(m)}$ is an m-dimensional matrix.

By default with no superscript, $\mathbf{M}$ is an 2-dimensional matrix.

- Let $|\mathbf{v}\,,\mathbf{w}\rangle = \mathbf{v}\otimes\mathbf{w}$ and define $\mathbf{U}^{(4)}$ by $\mathbf{U}^{(4)}|\mathbf{v}\,,\mathbf{w}\rangle = |\mathbf{A}\mathbf{v}\,,\mathbf{B}\mathbf{w}\rangle$.
  - $\mathbf{A}$ and $\mathbf{B}$ represent 1-qubit gates
  - We say $\mathbf{U}^{(4)} = \mathbf{A}\otimes\mathbf{B}$ and call it a *product gate*.
- With the 4 basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \otimes \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \text{ is represented as } \alpha = \begin{pmatrix} v_1 w_1 \\ v_1 w_2 \\ v_2 w_1 \\ v_2 w_2 \end{pmatrix}$$

- Thus $\mathbf{U}^{(4)} = \mathbf{A}\otimes\mathbf{B}$ acting on $\alpha\otimes\mathbf{w}$ has representation

$$\mathbf{U}^{(4)}\alpha = \mathbf{A}\otimes\mathbf{B}\left(\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \otimes \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}\right) = \left(\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \otimes \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}\right)$$

$$= \begin{pmatrix} A_{11}B_{11} & A_{11}B_{12} & A_{12}B_{11} & A_{12}B_{12} \\ A_{11}B_{21} & A_{11}B_{22} & A_{12}B_{21} & A_{12}B_{22} \\ A_{21}B_{11} & A_{21}B_{12} & A_{22}B_{11} & A_{22}B_{12} \\ A_{21}B_{21} & A_{21}B_{22} & A_{22}B_{21} & A_{22}B_{22} \end{pmatrix}\begin{pmatrix} v_1 w_1 \\ v_1 w_2 \\ v_2 w_1 \\ v_2 w_2 \end{pmatrix}$$

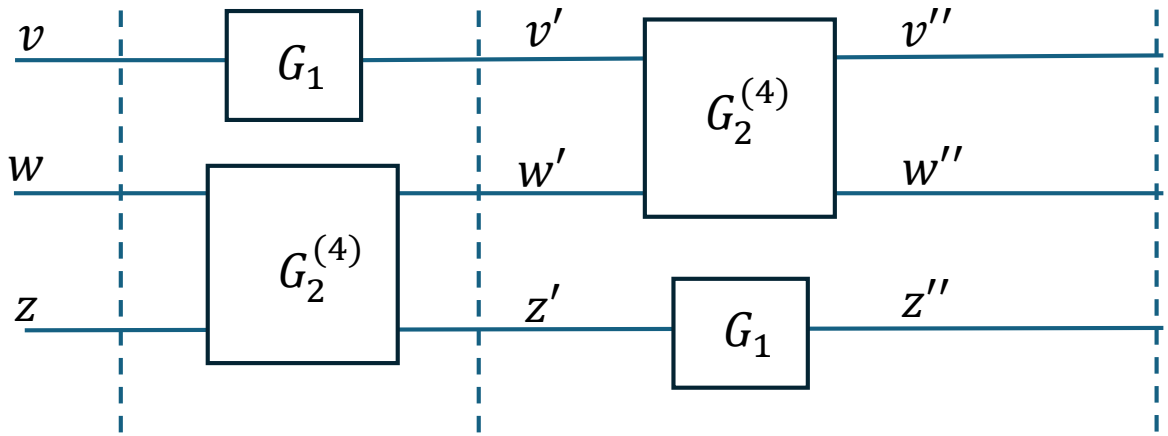
- Note $\dfrac{\mathbf{U}^{(4)}_{11}}{\mathbf{U}^{(4)}_{22}} = \dfrac{\mathbf{U}^{(4)}_{33}}{\mathbf{U}^{(4)}_{44}}$, so a general 2-qubit gate is not a product gate.

3-qubit product gates

- Example: Let $|\mathbf{v}, \mathbf{w}, \mathbf{z}\rangle = \mathbf{v} \otimes \mathbf{w} \otimes \mathbf{z}$ and define $\mathbf{U}^{(8)}$ by

$$\mathbf{U}^{(8)}|\mathbf{v}, \mathbf{w}, \mathbf{z}\rangle = \left(\mathbf{A}^{(4)}(\mathbf{v} \otimes \mathbf{w})\right) \otimes (\mathbf{Bz}).$$

  - We say $\mathbf{U}^{(8)} = \mathbf{A}^{(4)} \otimes \mathbf{B}$ and call it a *product gate*.
- A matrix representation of $\mathbf{U}^{(8)}$ is with respect to the 8 basis states $|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$, $|100\rangle$, $|101\rangle$, $|110\rangle$, $|111\rangle$
- 3-qubit product gates can be composed in a circuit.



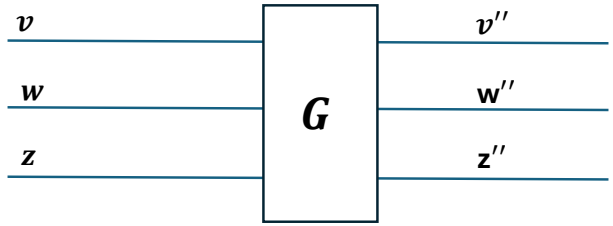
$\mathbf{V'} = v' \otimes w' \otimes z'$ $\qquad$ $\mathbf{V''} = v'' \otimes w'' \otimes z''$

$\mathbf{V'} = U_1^{(8)}\mathbf{V} = G_1 v \otimes G_2^{(4)}(w \otimes z))$ $\qquad$ $\mathbf{V''} = U_2^{(8)}\mathbf{V'} = (G_2^{(4)}(v' \otimes w')) \otimes G_1 z'$

$= U_2^{(8)}\, U_1^{(8)}\mathbf{V}$

3-qubit product gates cont'd

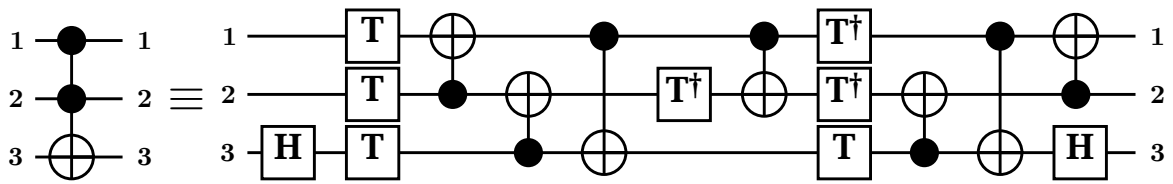
- The circuit, $\mathbf{G}_2^{(4)} \otimes \mathbf{G}_2^{(4)}$ followed by $\mathbf{G}_2^{(4)} \otimes \mathbf{G}_2^{(4)}$, is equivalent to



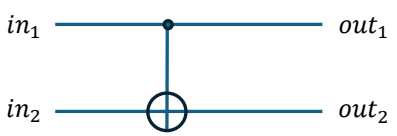
$$G = (G_1 \otimes G_2^{(4)})(G_2^{(4)} \otimes G_1) \stackrel{\text{def}}{=} U_2^{(8)}\, U_1^{(8)}$$

- Consider the 3-qubit gate CCX $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$

- CCX is a sequence of products of 1-qubit and 2-qubit gates.



where $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}$ and $CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} =$



Complexity analysis

- Review of 2-qubit gates.
  - Some gates are product gates
  - Some gates (e.g. CX (=CNOT)) are not product gates
- Assume that the time for all 1- and 2-qubit gates is less than T.
- Analysis of 3-qubit gates.
  - CCX-gate is a sequence of products of 1- and 2-qubit gates
- n-qubit gates: represented by $N \times N$ unitary matrices with $N = 2^n$.
- Theorem without proof:
  "Let U represent an n-qubit gate. Then $U = U_1 U_2 ... U_m$ where U_i represents a product-gate (i.e. a tensor product) whose tensor components are 1- and 2-qubit gates, and $m = \mathcal{O}(n)$."
- The Grover algorithm consists of an n-qubit superposition gate followed by $\mathcal{O}(\sqrt{N})$ repetitions of n-qubit oracle-superposition pairs.
- So the Grover *complexity* is $\mathcal{O}(n\sqrt{N}) = \mathcal{O}(log(N)\sqrt{N})$
- Since each 1- and 2-qubit gates takes less than time T
  The time for completion is $\mathcal{O}(log(N)\sqrt{N}T)$