Quantum Computing

An Introduction to Quantum Algorithm

Nishanka Das Debanjan Kola

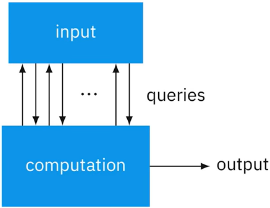
December 2023

Quantum Algorithm - Query Model

Query Model:

1. **Standard Computation:**
   - Input → $\boxed{\text{Computation}}$ → Output

   The entire input is provided to the computation, most typically as a string of bits with nothing hidden from the computation.
2. **Query model of computation:**



In the query model of computation , the input is made available in the form of a function , which the computation access by making queries . The input to query problems is represented by a function : $f : \Sigma^n \to \Sigma^m$ where $n, m \in \mathbb{Z}^+$ and $\Sigma = \{0, 1\}$

- **Queries :**

To say that a computation makes a query means that it evalues the function f once : $x \in \Sigma^m$ is made available to the computation . The efficiency of query algo is measured by counting the number of queries to the input they required .

- **Example of query problems :**

1. **OR :**
   - **Input :** $f : \Sigma^n \rightarrow \Sigma$
   - **Output :**

$$\begin{cases} 1 & \text{if there exists a string } x \in \Sigma^n \text{ such that } f(x) = 1 \\ 0 & \text{if there is no such string} \end{cases}$$

2. **Parity :**
   - **Input :** $f : \Sigma^n \rightarrow \Sigma$
   - **Output :**

$$\begin{cases} 0 & \text{if } f(x) = 1 \text{ for even number of strings } x \in \Sigma^n \\ 1 & \text{if } f(x) = 1 \text{ for odd number of strings } x \in \Sigma^n \end{cases}$$

3. **Minimum:**
   - **Input :** $f : \Sigma^n \to \Sigma^m$
   - **Output :**

     The string $y \in \{f(x) = x \in \Sigma^n\}$

     that comes first in the lexicographic ordering of Σ^n

Sometimes we also consider query problems where we have a promise on the input . Inputs that doesn't satisfy the promise are called don't care input .

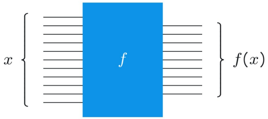
4. **Unique search:**
   - **Input :** $f : \Sigma^n \to \Sigma$
   - **Promise :**

     There is exactly one string $z \in \Sigma^n$

     for which f(z) = 1 , with f(x) = 0 for all strings $x \neq z$.

   - **Output :** The string z .

- **Query Gate :**

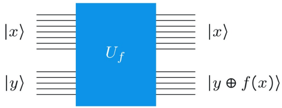


For the circuit model of computation , queries are made by **query gates** .For the quantum circuit model , we choose a different defination for query gates that makes them unitary - allowing them to be applied to quantum states .

- **Definition :**

The query gate U_f for any function $f : \Sigma^n \to \Sigma^m$ is defined as $U_f(|y\rangle\,|x\rangle) = |y\rangle \oplus |f(x)\rangle\,|x\rangle \quad \forall x \in \Sigma^n$ and $y \in \Sigma^m$.

- **Notation :**



The string $y \oplus f(x)$ is the bitwise XOR of y and f(x) .
$001 \oplus 101 = 100$
$|0^m\rangle \rightarrow$ All zero string .

- **Deutsch's Algo :** Deutsch's problem is parity problem of functions of the form $f : \Sigma \rightarrow \Sigma$.

a	$f(a)$	a	$f_2(a)$	a	$f_3(a)$	a	$f_4(a)$
0	0	0	0	0	1	0	1
1	0	1	1	1	0	1	1

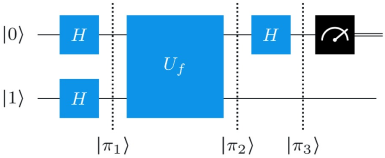
f_1 and f_4 are **constant** and f_2 and f_3 are **balanced** .

- **Problem :**
- **Input :** $f : \Sigma \to \Sigma$
- **Output :**

$$\begin{cases} 0 & \text{if f is constant} \\ 1 & \text{if f is balanced} \end{cases}$$

Every classical query algo must make two queries to f to solve this problem learning just one of two bits provids no information about their parity .

- **Notation :**



$$
\begin{aligned}
|\pi_1\rangle &= |-\rangle\,|+\rangle \\
&= \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)\frac{1}{\sqrt{2}}|0\rangle + \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)\frac{1}{\sqrt{2}}|1\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\,|0\rangle + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\,|1\rangle
\end{aligned}
$$

- $|\pi_2\rangle$ **:**
  U_f gate is performed . According to the definition of the U_f gate , the value of the function f for the classical state of the top / right most qbit is XORed onto the bottom/left most qbit , which transforms $|\pi_1\rangle$ to $|\pi_2\rangle$

$$
\begin{aligned}
|\pi_2\rangle &= \frac{1}{2}\left(|10 \oplus f(0)\rangle - |1 \oplus f(0)\rangle\right)|0\rangle \\
&\quad + \frac{1}{2}\left(|10 \oplus f(1)\rangle - |1 \oplus f(1)\rangle\right)|1\rangle \\
&\quad [|10 \oplus a\rangle - |1 \oplus a\rangle = (-1)^a\left(|0\rangle - |1\rangle\right)] \\
&= \frac{1}{2}(-1)^{f(0)}\left(|0\rangle - |1\rangle\right)|0\rangle + \frac{1}{2}(-1)^{f(1)}\left(|0\rangle - |1\rangle\right)|1\rangle \\
&= |-\rangle\left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right)
\end{aligned}
$$

$$= (-1)^{f(0)} \left|-\right\rangle \left(\frac{\left|0\right\rangle + (-1)^{(f(0) \oplus f(1))} \left|1\right\rangle}{\sqrt{2}} \right)$$

$$= \begin{cases} (-1)^{f(0)} & \left|-\right\rangle \left|+\right\rangle & f(0) \oplus f(1) = 0 \\ (-1)^{f(0)} & \left|-\right\rangle \left|-\right\rangle & f(0) \oplus f(1) = 1 \end{cases}$$

- **Phase Kickback:**

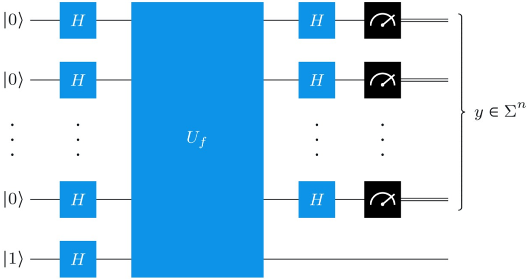
$$\left|b \oplus c\right\rangle = x^c \left|b\right\rangle U_f \left(\left|b\right\rangle \left|a\right\rangle\right) = \left|b \oplus f_a\right\rangle \left|a\right\rangle = \left(x^{f(a)} \left|b\right\rangle\right) \left|a\right\rangle$$

$$x \left|-\right\rangle = - \left|-\right\rangle$$

$$U_f \left(\left|-\right\rangle \left|a\right\rangle\right) = (-\mathbf{1})^{\mathbf{f(a)}} \left|-\right\rangle \left|\mathbf{a}\right\rangle$$

- **Deuctsch-Jozsa Algorithm :** The Deuctsch Jozsa Algo extends deuctsch's algorithm . To input functions of the form , $f : \Sigma^n \rightarrow \Sigma$ for any n$\geq$1

- **Quantum Circuit :**



- **Problem :**
- **Input :** $f : \Sigma^n \to \Sigma$
- **Promise :** f is either constant or balanced .
- **Output :**

$$\begin{cases} 0 & \text{if f is constant .} \\ 1 & \text{if f is balanced .} \end{cases}$$

- **Analysis:** We are taking a Hadamard operation:

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

$$H|0\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

We can express the two equations via one .

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

$$= \frac{1}{\sqrt{2}} \sum_{b \in \{0,1\}} (-1)^{ab} |b\rangle$$

Hadamard operation on each of n qubit :

$$H^{\oplus n}|X_{n-1} \cdots X_0\rangle$$

$$= (H|X_{n-1}\rangle) \oplus \cdots \oplus (H|X_0\rangle)$$

$$= \left(\frac{1}{\sqrt{2}} \sum_{y_{n-1} \in \Sigma} (-1)^{x_{n-1} y_{n-1}} |y_{n-1}\rangle \right) \bigoplus \cdots \bigoplus \left(\frac{1}{\sqrt{2}} \sum_{y_0 \in \Sigma} (-1)^{x_0 y_0} |y_0\rangle \right)$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y_{n-1} \cdots y_0 \in \Sigma} (-1)^{x_{n-1} y_{n-1} + \cdots + x_0 y_0} |y_{n-1} \cdots y_0\rangle$$

- **Binary Dot Product:** For binary strings $x = x_{n-1} \cdots x_0$ and $y = y_{n-1} \cdots y_0$,

$$x \cdot y = x_{n-1} y_{n-1} \bigoplus \cdots \bigoplus x_0 y_0$$

$$= \begin{cases} 1 & \text{if } x_{n-1} y_{n-1} + \cdots + x_0 y_0 \text{ is odd .} \\ 0 & \text{if } x_{n-1} y_{n-1} + \cdots + x_0 y_0 \text{ is even .} \end{cases}$$

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2}^n} \sum_{y \in \Sigma^n} (-1)^{x \cdot y} |y\rangle$$

$$\left|\pi_1\right\rangle = \left|-\right\rangle \otimes \frac{1}{\sqrt{2}^n} \sum_{x \in \Sigma^n} \left|x\right\rangle$$

$$\left|\pi_2\right\rangle = \left|-\right\rangle \otimes \frac{1}{\sqrt{2}^n} \sum_{x \in \Sigma^n} (-1)^{f(x)} \left|x\right\rangle$$

$$\left|\pi_3\right\rangle = \left|-\right\rangle \otimes \frac{1}{\sqrt{2}^n} \sum_{y \in \Sigma^n} \sum_{x \in \Sigma^n} (-1)^{f(x)+x\cdot y} \left|y\right\rangle$$

The probability of measurements given $y = 0^n$ is

$$p(0^n) = \left| \frac{1}{\sqrt{2}^n} \sum_{x \in \Sigma^n} (-1)^{f(x)} \right|^2$$

$$\begin{cases} 1 & \text{if f is constant .} \\ 0 & \text{if f is balanced .} \end{cases}$$

- **Bernstein-Vazirani Problem:**
- **Input:** $f : \Sigma^n \rightarrow \Sigma$
- **Promise:** There exists a binary string $s = s_{n-1} \cdots s_0$ for which $f(x) = s \cdot x$ for all $x \in \Sigma^n$
- **Output:** The string s

$$|\pi_3\rangle = |-\rangle \otimes \frac{1}{\sqrt{2}^n} \sum_{y \in \Sigma^n} \sum_{x \in \Sigma^n} (-1)^{f(x) + x \cdot y} |y\rangle$$

$$= |-\rangle \otimes \frac{1}{\sqrt{2}^n} \sum_{y \in \Sigma^n} \sum_{x \in \Sigma^n} (-1)^{s \cdot x + y \cdot x} |y\rangle$$

$$= |-\rangle \otimes \frac{1}{\sqrt{2}^n} \sum_{y \in \Sigma^n} \sum_{x \in \Sigma^n} (-1)^{(s \oplus y) \cdot x} |y\rangle$$

$$= |-\rangle \otimes |s\rangle$$

- **Simon's Algorithm:**
- **Input:** $f : \Sigma^n \to \Sigma^m$
- **Promise:** $\exists$ a string $s \in \Sigma^n$ such that

$$[f(x) = f(y)] \Leftrightarrow [(x = y) \vee (x \oplus s) = y] \quad \forall x, y \in \Sigma^n$$

- **Output:** The string s
- case 1 : $s = 0^n$ the condition in the promise simplifies to

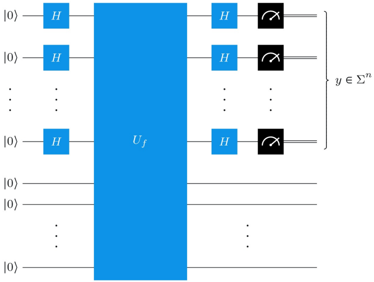
$$[f(x) = f(y)] \Leftrightarrow [x = y]$$

  this is equivalent to f being one-to-one .
- case 2 : $s \neq 0^n$ the function f must be two-to-one to satisfy the promise

$$f(x) = f(x \oplus s)$$

  with distinct output strings for each pair .

- **Quantum circuit :**



$$|\pi_1\rangle = \frac{1}{\sqrt{2}^n} \sum_{x \in \Sigma^n} |0^m\rangle |x\rangle$$

$$|\pi_2\rangle = \frac{1}{\sqrt{2}^n} \sum_{x \in \Sigma^n} |f(x)\rangle |x\rangle$$

$$|\pi_3\rangle = \frac{1}{\sqrt{2}^n} \sum_{x \in \Sigma^n} |f(x)\rangle \oplus \left(\frac{1}{\sqrt{2}^n} \Sigma (-1)^{x \cdot y} |y\rangle \right)$$

$$= \frac{1}{2^n} \sum_{y \in \Sigma^n} \sum_{x \in \Sigma^n} (-1)^{x \cdot y} |f(x)\rangle |y\rangle$$

$$p(y) = \left\| \frac{1}{2^n} \sum_{x \in \Sigma^n} (-1)^{x \cdot y} |f(x)\rangle \right\|^2$$

$$\begin{cases} \text{range}(f) = \{f(x) : x \in \Sigma^n\} \\ f^{-1}(\{z\}) = \{x \in \Sigma^n : f(x) = z\} \end{cases}$$

$$= \left\| \frac{1}{2^n} \sum_{z \in \text{range}(f)} \left(\sum_{x \in f^{-1}(\{z\})} (-1)^{x \cdot y} |z\rangle \right) \right\|^2$$

$$= \frac{1}{2^{2n}} \sum_{z \in \text{range}(f)} \left| \sum_{x \in f^{-1}(\{z\})} (-1)^{x \cdot y} \right|^2$$

- Case 1 : $s = 0^n$ Because f is a one-to-one , there a single element $x \in f^{-1}(\{z\})$ for every $z \in \text{range}(f)$:

$$\left| \sum_{x \in f^{-1}(\{z\})} (-1)^{x \cdot y} \right|^2 = 1$$

There are 2^n elements in range f , so

$$p(y) = \frac{1}{2^{2n}} \cdot 2^n = \frac{1}{2^n} \qquad \forall y \in \Sigma^n$$

- Case 2 : $s \neq 0^n$ There are two strings in the set $f^{-1}(\{z\})$ for each $z \in$ range (f) if $w \in f^{-1}(\{z\})$ either one of them , then $w \oplus s$ is the other .

$$\left| \sum_{x \in f^{-1}(\{z\})} (-1)^{x \cdot y} \right|^2 = \left| (-1)^{w \cdot y} + (-1)^{(w \oplus s) \cdot y} \right|^2$$

$$= \left|1 + (-1)^{s \cdot y}\right|^2$$

$$= \begin{cases} 4 & s \cdot y = 0 \\ 0 & s \cdot y = 1 \end{cases}$$

There are 2^{n-1} elements in range (f) so:

$$p(y) = \frac{1}{2^{2n}} \sum_{z \in \text{range}} \left| \sum_{x \in f^{-1}(\{z\})} (-1)^{x \cdot y} \right|^2$$

$$= \begin{cases} \frac{1}{2^{n-1}} & s \cdot y = 0 \\ 0 & s \cdot y = 1 \end{cases}$$

- **Integer Factorization:**
- **Input:** An integer $N \geq 2$
- **Output:** The prime factorization of N.
  The Prime Factorization of N is the list of prime factors of N and the powers to which they must be raised to obtain N by multiplication.
- **Greatest Common Divisor:**
- **Input:** Non-negative integers N and M not both 0.
- **Output:** The Greatest Common Divisor of N and M.
- **Measuring Computational Cost:** Abstract overview of computation:
  1. Turing Mechine
  2. Boolean Circuits
  3. Quantum Circuits
  4. Python Programs

- **Encoding and input length :**
- Inputs and outputs are binary strings .
- Through binary strings we can encode .
  1. Numbers
  2. Vectors
  3. Matrices
  4. Graphs
  5. Description of molecules
- Example :

Number	Encode	Length
0	0	1
1	1	1
2	10	2

Length of binary encoding of N :

$$\log N = \begin{cases} 1 & N = 0 \\ 1 + \log_2 N & N \geq 1 \end{cases}$$

In general , Input length is the length of the binary String encoding of the input , with respect to whatever encoding scheme has been selected .

- **Elementory Operation :** For circuit based models of computation it is typical that we view each gate as being an elementory operation .
- A standard quantum gate set :
  1. Single-qubit unary gates from $X, Y, Z, H, S, S^T, T, T^T$
  2. Controlled-NOT gates
  3. Single-qubit standard basis measurements .

  The Unitary gates in this set are universal-any Unitary operation can be closely approximate by a circuit of the gates .
- **A Standard boolean gate set :**
  1. AND
  2. OR
  3. NOT
  4. FANOUT

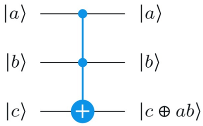
- **Circuit Size :** A **size** of a circuit is the total number of gates it includes, we may write size(c) to refer the size of a circuit .
  Circuit size corresponds to sequential running time .
  The **depth** of a circuit is the maximum number of gates encountered on any path from an input to an output wire .
- **Cost as a $f(x)$ of input length** Each circuit has a fixed size so we need a family $c_1, c_2 \cdots$ of circuit to describe an algorithm,typically one circuit for each input length.
- Example : A classical algorithm for integer factorization could be described by a family of boolean circuits.
  The cost of such an algorithm is described by a function .
  $t(n) = size(C_n)$
- **Asymptotic Notation :**
- **Big-O-Notation :** For two functions $g(n)$ and $h(n)$ we write that $g(n) = O(h(n))$ if there exist a positive real number $c > 0$ and positive integer no such that :

$$g(n) \leq c \cdot h(n) \quad \forall n \geq n_0$$

- Eample :
  1. Addition of n-bit integer can be computed at cost O(n).
  2. Multiplication of n-bit integer can be computed at cost $O(n^2)$.
  3. Division of n-bit integer can be computed at cost $O(n^2)$.
  4. GCD of n-bit integer can be computed at cost $O(n^2)$.
  5. Moduler exponential cost of n-bit integer can be computed at cost $O(n^3)$.
- **Polynomial vs Exponential cost :** An algorithm's cost is polynomial if it is $O(n^b)$ for some fixed constant $b > 0$
  An algorithm's cost scales sub exponentially if it is :

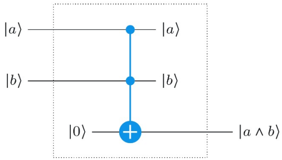
$$O\left(2^{n^{\epsilon}}\right) \quad \forall \epsilon > 0$$

  otherwise exponential (or super exponential).
- **Toffoli gates :** Toffoli gates are controlled-controlled-NOT gate .

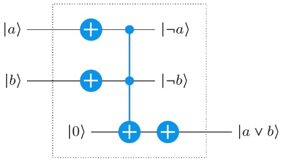


- **Simulating Boolean Gates :**
- **AND gate and OR gate :**

AND gate

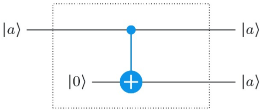


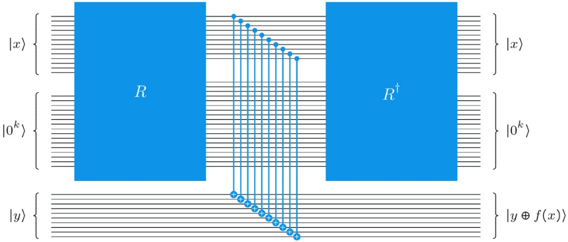
OR gate



- **FANOUT gate :**

FANOUT gate





$|x\rangle$
$|0^k\rangle$
$|y\rangle$
R
$R^\dagger$
$|x\rangle$
$|0^k\rangle$
$|y \oplus f(x)\rangle$

Spectral Theorem

Suppose U is an NxN unitary matrix
There exists an orthonormal basis $|\psi_1\rangle \dots |\psi_N\rangle$ of vector along with complex numbers
$\lambda_1 = e^{2\pi i\theta_1}, \dots, \lambda_N = e^{2\pi i\theta_N}$ such that

$$U = \sum_{k=1}^{N} \lambda_k |\psi_k\rangle \langle\psi_k|$$

Each of the vector $|\psi_k\rangle$ is an eigenvector of U having eigenvalue λ_k

$$U|\psi_k\rangle = \lambda_k |\psi_k\rangle = e^{2\pi i\theta_k} |\psi_k\rangle$$

Phase Estimation Problem

Input: A unitary quantum circuit for an n-qubit operation U and an n qubit quantum state $|\psi\rangle$

Promise: $|\psi\rangle$ is an eigenvector of U

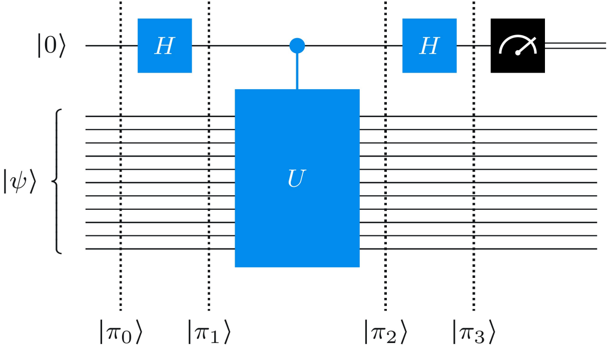
Output: An approximation to the number $\theta \in [0, 1)$ satisfying $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$

We can approximate θ by fraction

$$\theta \approx y/2^m \quad \forall \quad y \in \{0, 1, ..., 2^{m-1}\}$$

Phase Estimation Procedure

Approximating phase with low precision using the phase kick back



$$|\pi_0\rangle = |\psi\rangle\,|0\rangle$$
$$|\pi_1\rangle = 1/\sqrt{2}\,|\psi\rangle\,|0\rangle + 1/\sqrt{2}\,|\psi\rangle\,|1\rangle$$
$$|\pi_2\rangle = |\psi\rangle \otimes (1/\sqrt{2}\,|0\rangle + e^{2\pi i\theta}/\sqrt{2}\,|1\rangle)$$
$$|\pi_3\rangle = |\psi\rangle \otimes ((1+e^{2\pi i\theta})/2 + ((1-e^{2\pi i\theta})/2)\,|1\rangle)$$

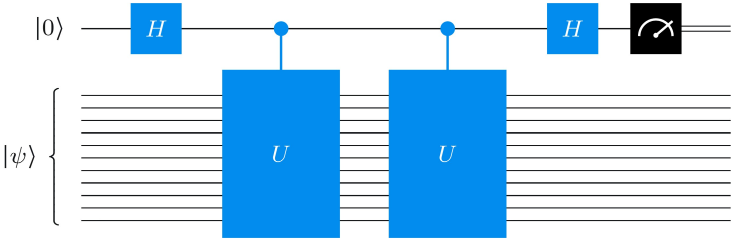
Phase Estimation Procedure

Measuring Probability

$$P_0 = \left|\frac{1+e^{2\pi\theta i}}{2}\right|^2 = cos^2(\pi\theta)$$

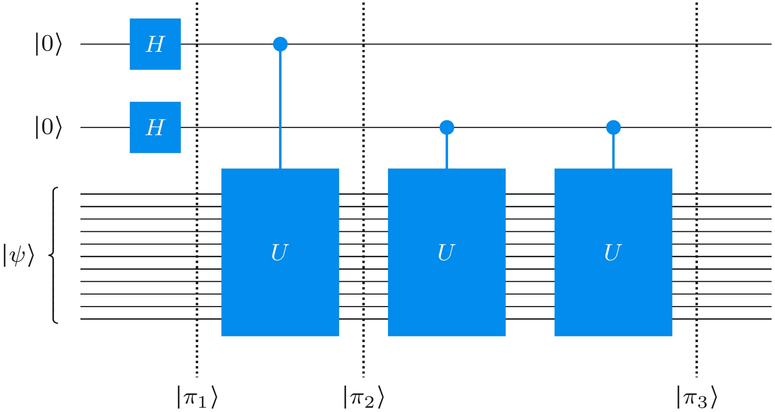
$$P_1 = \left|\frac{1-e^{2\pi\theta i}}{2}\right|^2 = sin^2(\pi\theta)$$

Doubling the phase



By this circuit the value of θ is doubled only

Two qubit phase estimation



$$|\pi_1\rangle = |\psi\rangle \otimes \frac{1}{2}\sum_{a_0=0}^{1}\sum_{a_1=0}^{1} |a_1 a_0\rangle$$
$$|\pi_2\rangle = |\psi\rangle \otimes \frac{1}{2}\sum_{a_0=0}^{1}\sum_{a_1=0}^{1} e^{2\pi i a_0 \theta} |a_1 a_0\rangle$$

$$|\pi_3\rangle = |\psi\rangle \otimes \frac{1}{2}\sum_{a_0=0}^{1}\sum_{a_1=0}^{1} e^{2\pi i(a_0+2a_1)\theta}\,|a_1 a_0\rangle$$

$$|\pi_3\rangle = |\psi\rangle \otimes \frac{1}{2}\sum_{x=0}^{3} e^{2\pi i x\theta}\,|x\rangle$$

We will consider a special case where $\theta = \frac{y}{4}$ for y $\in \{0, 1, 2, 3\}$
$\therefore \theta \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$

$$|\phi_y\rangle = \frac{1}{2}\sum_{x=0}^{3} e^{2\pi i x\frac{y}{4}}\,|x\rangle$$

where y$\in$ {0,1,2,3}

Results :

$$|\phi_0\rangle = \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle + \frac{1}{2}|2\rangle + \frac{1}{2}|3\rangle$$

$$|\phi_1\rangle = \frac{1}{2}|0\rangle + \frac{i}{2}|1\rangle - \frac{1}{2}|2\rangle - \frac{i}{2}|3\rangle$$

$$|\phi_2\rangle = \frac{1}{2}|0\rangle - \frac{1}{2}|1\rangle + \frac{1}{2}|2\rangle - \frac{1}{2}|3\rangle$$

$$|\phi_3\rangle = \frac{1}{2}|0\rangle - \frac{i}{2}|1\rangle - \frac{1}{2}|2\rangle + \frac{i}{2}|3\rangle$$

Here we are taking the columns of V to be the states $|\phi_0\rangle \ldots |\phi_3\rangle$

$$V = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

This V $4x4$ matrix is named as QFT_4 (Quantum Fourier Transformation)

Quantum Fourier Transformation

To define the quantum fourier transformation ω_N , complex number have to be defined first, for positive integer N

$$\omega_N = e^{\frac{2\pi i}{N}} = cos(\frac{2\pi}{N} + isin(\frac{2\pi}{N}))$$

The N dimensional quantum fourier transformation, which is described by NxN matrix whose rows and columns are associated with standard basis state $|0\rangle \ldots |N-1\rangle$

$$QFT_N = \frac{1}{\sqrt{N}}(\sum_{x=0}^{N-1}\sum_{x=0}^{N-1}\omega_N^{xy}|x\rangle\langle y|)$$

32-dimensional quantum Fourier transform

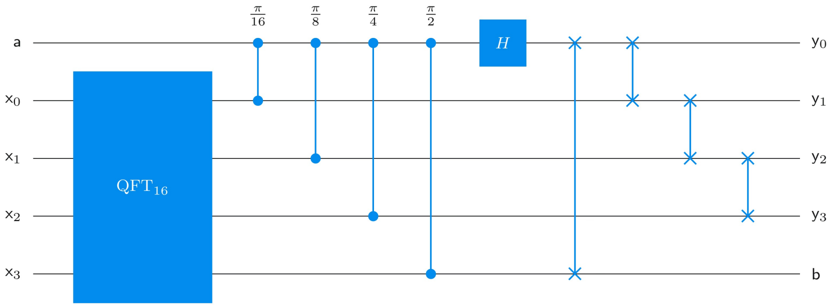


Figure: 32 dimentional QFT

Computational Cost:

Let S_m denotes the total number of gates in a m qubit QFT
For m $= 1$, a single Hadamard gate is required
For m ≥ 2 , these are the gates requires

S_m for the QFT_{m-1} gates
m-1 controlled phase gate
m-1 swap gate
1 Hadamard gate

$$S_m = \begin{cases} 1, & \text{if } m = 1 \\ S_{m-1} + 2m - 1, & \text{if } m \geq 2 \end{cases}$$

Phase estimation for any choice of m
$P_y = |\frac{1}{2^m}\sum_{x=0}^{2^m-1} e^{2\pi i x(\theta - \frac{y}{2^m})}|^2$

Best Case: Suppose $y/2^m$ is the best approximation to θ
$|\theta - \frac{y}{2^m}| \leq 2^{-(m+1)}$
$P_y \geq \frac{4}{\pi^2} \approx 0.405$

Worst Case: Suppose there is a better approximation to θ between $\frac{y}{2^m}$ to θ
$|\theta - \frac{y}{2^m} \geq 2^{-m}|$
$P_y \leq \frac{1}{4}$

Shor's algorithm

Order Finding Problem :

Input: Positive integers N and a satisfying gcd(N, a)=1

Output: The smallest positive integer r such that $a^r \equiv 1 \pmod{N}$

Factoring by Order-Finding

If N is odd and not a prime power, order-finding allows us to split N. Iterate the following steps:

- Randomly choose a$\in$ 2,...,N-1
- Compute d $= gcd(a, N)$ if d $\geq$ 2 then output id d and stop
- Compute the order r of a modulo N.
- If r is even then compute d $= gcd(a^{\frac{r}{2}} - 1, N)$ if d $\geq$ 2 then output id d and stop
- If this step reached, The method failed

There is a 50% chance of achiving success in this method.

Introduction to Grover's Algo

Grover's algorithm, which is a quantum algorithm for so-called unstructured search problems that offers a quadratic improvement over classical algorithms. What this means is that Grover's algorithm requires a number of operations on the order of the square-root of the number of operations required to solve unstructured search classically — which is equivalent to saying that classical algorithms for unstructured search must have a cost at least on the order of the square of the cost of Grover's algorithm.

Unstructured Search

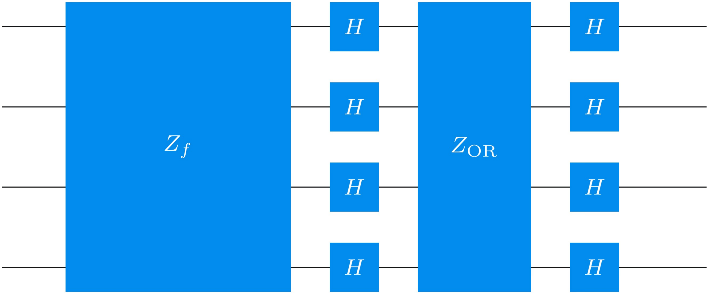
- Input: $f : \sum^n \rightarrow \sum$
- Output: A string x$\in \sum^n$ satisfying f(x) $=$ 1 or "no solution" if no such string exists.

This is unstructured search because f is and there is no promise and also we can't rely on it having a structure that makes finding solution easy.

Grover's algorithm

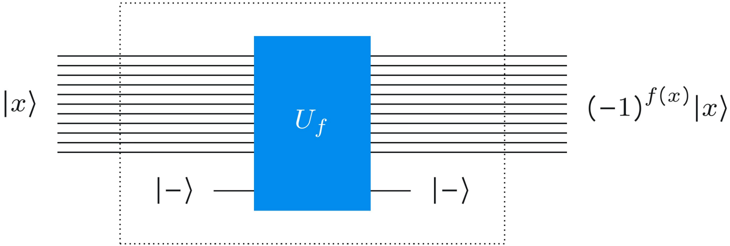
- Initialize : Set n qubit to state $H^{\otimes n} |0^n\rangle$
- Iterate : Apply the Grover operation t times
- Measure : A standard basis measurement yields a candidate solution.

The Grover Operation looks like $G = H^{\otimes n} Z_{OR} H^{\otimes n} Z_f$



- $Z_f\,|X\rangle$: $\quad Z_f\,|X\rangle = -1^{f(X)}\,|X\rangle$
- $Z_{OR|X\rangle}$:

$$Z_{OR|X\rangle} = \begin{cases} |X\rangle\,, & \text{if } X = 0^n \\ -\,|X\rangle\,, & \text{if } X \neq 0^n \end{cases}$$



Solution and Non-Solution

We will refer to the n qubit begin used for Grover's Algo as a register Q

Q is initialized to the state $H^{\otimes n}|0^n\rangle$ and Grover operation G is performed iteratively

$A_0 = \{x \in \sum^n : f(x) = 0\}$ Non Solutions

$A_1 = \{x \in \sum^n : f(x) = 1\}$ Solutions

$$|A_0\rangle = \frac{1}{\sqrt{|A_0|}} \sum_{x \in A_0} |x\rangle$$

$$|A_1\rangle = \frac{1}{\sqrt{|A_1|}} \sum_{x \in A_1} |x\rangle$$

$$|u\rangle = H^{\otimes n}|0^n\rangle = \frac{1}{\sqrt{N}} \sum_{x \in \sum^n} |x\rangle$$

$$|u\rangle = \sqrt{\frac{|A_0|}{N}}|A_0\rangle + \sqrt{\frac{|A_1|}{N}}|A_1\rangle$$

Action of Grover Operation

$$G|A_0\rangle = (2|u\rangle\langle u| - 1)Z_r|A_0\rangle$$
$$G|A_0\rangle = 2\sqrt{\frac{|A_0|}{N}}|u\rangle - |A_0\rangle$$
$$G|A_0\rangle = \frac{|A_0|-|A_1|}{N}|A_0\rangle + \frac{2\sqrt{|A_0|.|A_1|}}{N}|A_1\rangle$$

$$G|A_1\rangle = (2|u\rangle\langle u| - 1)Z_r|A_1\rangle$$
$$G|A_1\rangle = |A_1\rangle - 2\sqrt{\frac{|A_1|}{N}}|u\rangle$$
$$G|A_1\rangle = -\frac{2\sqrt{|A_0|.|A_1|}}{N}|A_0\rangle + \frac{|A_0|-|A_1|}{N}|A_1\rangle$$

Rotation by an angle

$$M = \begin{pmatrix} \frac{|A_0|-|A_1|}{N} & -\frac{2\sqrt{|A_0|.|A_1|}}{N} \\ \frac{2\sqrt{|A_0|.|A_1|}}{N} & \frac{|A_0|-|A_1|}{N} \end{pmatrix}$$

$$= \begin{pmatrix} \sqrt{\frac{|A_0|}{N}} & -\sqrt{\frac{|A_1|}{N}} \\ \sqrt{\frac{|A_1|}{N}} & \sqrt{\frac{|A_0|}{N}} \end{pmatrix}^2$$

$$= \begin{pmatrix} cos(2\theta) & -sin(2\theta) \\ sin(2\theta) & cos(2\theta) \end{pmatrix}^2$$

$$|u\rangle = cos(\theta)\,|A_0\rangle + sin(\theta)\,|A_1\rangle$$
$$G\,|u\rangle = cos(3\theta)\,|A_0\rangle + sin(3\theta)\,|A_1\rangle$$
$$G^t\,|u\rangle = cos((2t+1)\theta)\,|A_0\rangle + sin((2t+1)\theta)\,|A_1\rangle$$

Setting Targate

Consider a quantum state $\alpha |A_0\rangle + \beta |A+1\rangle$

$x \in A_1$ with probability $|\beta|^2$

Measuring after t iteration given an outcome $x \in A_1$ with probability

$\sin^2((2t+1)\theta)$

To make this probability close to 1 and minimize t

$(2t+1)\theta \approx \frac{\pi}{2} => t = \lfloor \frac{\pi}{4\theta} \rfloor$

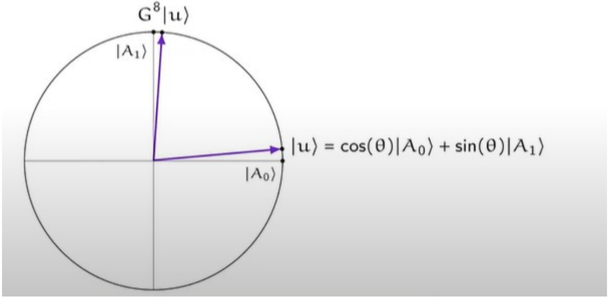
Unique Search

For Unique search we have $s = |A_1| = 1$ and therefore

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

For N = 128

$$t = 8$$

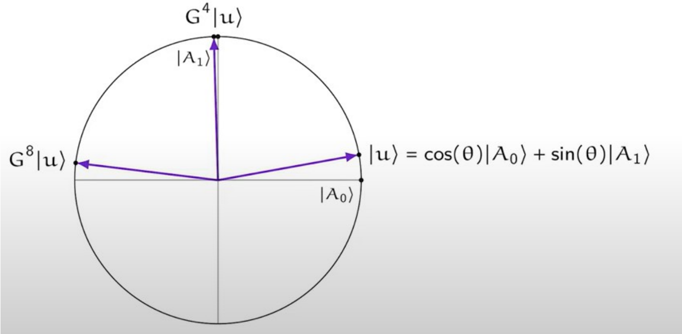


Multiple Search

Let's Consider N $= 128$ and s $= 4$

$$\theta = \sin^{-1}(\sqrt{\tfrac{s}{N}}) = 0.177...$$

$t = \lfloor \frac{\pi}{4\theta} \rfloor = 4$



Conclusion

- Grover's Algorithm is asymptotically optimal
- Grover's Algorithm is broadly applicable
- The technique used in Grover's Algorithm can be generalized