Quantum computing

(1) Quantum state (superposition of classical states)

- A classical state can be found in one quantum system if we observe it.
- A quantum state $|\phi>$ is a superposition of classical states. It can be written as $|\phi> = \alpha_1|1> + \alpha_2|2> + \ldots + \alpha_N|N>$, where α_i is a complex number.
  It is in state $|1>$ with amplitude α_1, in state $|2>$ with amplitude α_2, and so on.

(2) Measurement in the computational basis

- We can say that we will see the state $|j>$ with probability $|\alpha_j|^2$, and the total probability is equal to 1 (i.e., $\sum_{j=1}^{N} |\alpha_j|^2 = 1$).
- If we measure $|\phi>$ and see the classical state $|j>$ as a result, then $|\phi>$ collapses the quantum superposition $|\phi>$ to the classical state $|j>$. That is to say, $|\phi>$ itself disappeared.

(3) Unitary operation

- If we want to do some state transformation, then we can apply unitary operation U to change a quantum state $|\phi>$ to another quantum state $|\psi>$. ($|\psi> = U|\phi>$)
- A matrix U is unitary if U^{-1} (inverse U) = U^* (conjugate transpose of U).
- A unitary transformation must be reversible because U always has U^{-1}. For example, a state after applying U twice sequentially is invariant.
  $UU|\phi> = UU^{-1}|\phi> = I|\phi> = |\phi>$
- A measurement is NOT reversible because we cannot reconstruct $|\phi>$ from the observed classical state $|j>$.

(4) A register of n qubits has 2^n basis states $|0\ldots0>, \ldots, |1\ldots1>$ or $|0>, \ldots, |2^n -1>$

- A quantum bit is a superposition of 0 and 1.
- A simple system with 2 basis states, $|0>$ and $|1>$. We identify these basis states with the vectors $\binom{1}{0}$ and $\binom{0}{1}$, respectively.
- A 2-qubit system has 4 ($=2^2$) basis states: $|0>\otimes|0>$, $|0>\otimes|1>$, $|1>\otimes|0>$, and $|1>\otimes|1>$, where $\otimes$ stands for tensor product.
- $|0>\otimes|0> = |0>|0> = |00>$
- We give an example to show how tensor product works as follows.

$$|0> \otimes |1> = \begin{pmatrix} 1\binom{0}{1} \\ 0\binom{0}{1} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01>$$

(5) Entanglement refers to quantum correlations between different qubits

- A state $|\phi>$ is called entangled if it cannot be written as a tensor product $|A> \otimes |B>$ where $|A>$ lives in the first space and $|B>$ lives in second one.

(6) Elementary quantum gates: 1-bit gates:

- Bit flip gate X: swap $|0>$ and $|1>$.
  If a state $|\phi> = \alpha_0|0> + \alpha_1|1>$, then the state after applying X will be $|\phi> = \alpha_1|0> + \alpha_0|1>$.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad X\begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_0 \end{pmatrix}$$

- Phase gates Z: $|1>$ becomes $-|1>$.
  If a state $|\phi> = \alpha_0|0> + \alpha_1|1>$, then the state after applying Z will be $|\phi> = \alpha_0|0> - \alpha_1|1>$.

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad Z\begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} \alpha_0 \\ -\alpha_1 \end{pmatrix}$$

- Gate T: is a phase gate R_ϕ with $\phi = \pi/4$

$$R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{pmatrix}, \qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

- Hadamard gate H:

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

- 2-bit gates CNOT: negates the second bit if the first bit is 1.
  CNOT $|0>|b> = |0>|b>$
  CNOT $|1>|b> = |1>|1-b>$
  If a state $|\phi> = \alpha_0|00> + \alpha_1|01> + \alpha_2|10> + \alpha_3|11>$, then the state after applying CNOT will be $|\phi> = \alpha_0|00> + \alpha_1|01> + \alpha_3|10> + \alpha_2|11>$

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \text{control } U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{11} & U_{12} \\ 0 & 0 & U_{21} & U_{22} \end{pmatrix}$$

- 3-bit gate: CCNOT (Toffoli gate): negates the third bit if the first and second bits are all 1. (i.e., CCNOT $|1>|1>|b> = |1>|1>|1-b>$)

The circuit model and Deutsch-Jozsa

(1) Quantum circuit

- A quantum circuit replaces the AND, OR, and NOT gates by elementary quantum gate. For example, 2-qubit gate CNOT, 3-qubit gate CCNOT.

(2) Elementary quantum gates can be composed into bigger unitary operations by taking tensor products (if gates are applied in parallel to different parts of the register), and ordinary matrix products (if gates are applied sequentially)

- Parallel: if we apply H to each quantum bit, then it can be denoted as $H^{\otimes n} | j >$

$$H^{\otimes n}|i\rangle = \frac{1}{\sqrt{2^n}} \sum_{j \in \{0,1\}^n} (-1)^{i \cdot j} |j\rangle$$

For example, j = 101, and follow the above equation, you will get

$$\boxed{\frac{1}{\sqrt{2^3}} (|000> - |001> + |010> - |011> - |100> + |101> - |110> + |111>)}$$

We can apply H to each individually, first we apply to the first bit

$|101> \Rightarrow \frac{1}{\sqrt{2}}(|001> - |101>)$ NOTE: $|1> = \frac{1}{\sqrt{2}} (|0> - |1>)$

And then apply H to the second bit

$\frac{1}{\sqrt{2}}(|001> - |101>) \Rightarrow \frac{1}{\sqrt{2^2}} [(|001> + |011>) - (|101> + |111>)]$

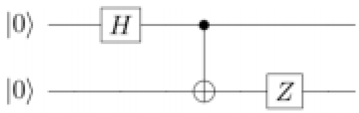
Finally, apply H to the third bit.

$\frac{1}{\sqrt{2^2}} [(|001> + |011>) - (|101> + |111>)] \Rightarrow$

$$\boxed{\frac{1}{\sqrt{2^3}} [(|000> - |001>) + (|010> - |011>) - ((|100> - |101>) + (|110> - |111>))]}$$

- Sequential: if we apply H and then U, then it can be denoted as $UH | j >$

(3) Simple circuit for turning $| 00 >$ into an entangled state $\frac{1}{\sqrt{2}}(| 00 > - | 11 >)$



Applying H to the first qubit: $| 00 > \Rightarrow \frac{1}{\sqrt{2}}(| 0 > + | 1 >) | 0 > = \frac{1}{\sqrt{2}}(| 00 > + | 10 >)$

Applying CNOT to both qubits: $\frac{1}{\sqrt{2}}(| 00 > + | 10 >)$ becomes $\frac{1}{\sqrt{2}}(| 00 > + | 11 >)$

Applying Z: $\frac{1}{\sqrt{2}}(|00> + |11>)$ becomes $\frac{1}{\sqrt{2}}(|00> - |11>)$

(6) Oracles

- $O_x : |i, b> \to |i, b \otimes x_i>$, where $i \in \{0,1\}^n$ (called the address bits) and $b \in \{0,1\}$ (called the target bit).
- $O_{x,\pm}$: $|i>|-> \to (-1)^{x_i}|i>|->$, where $|-> = \frac{1}{\sqrt{2}}(|0> - |1>) = H|1>$

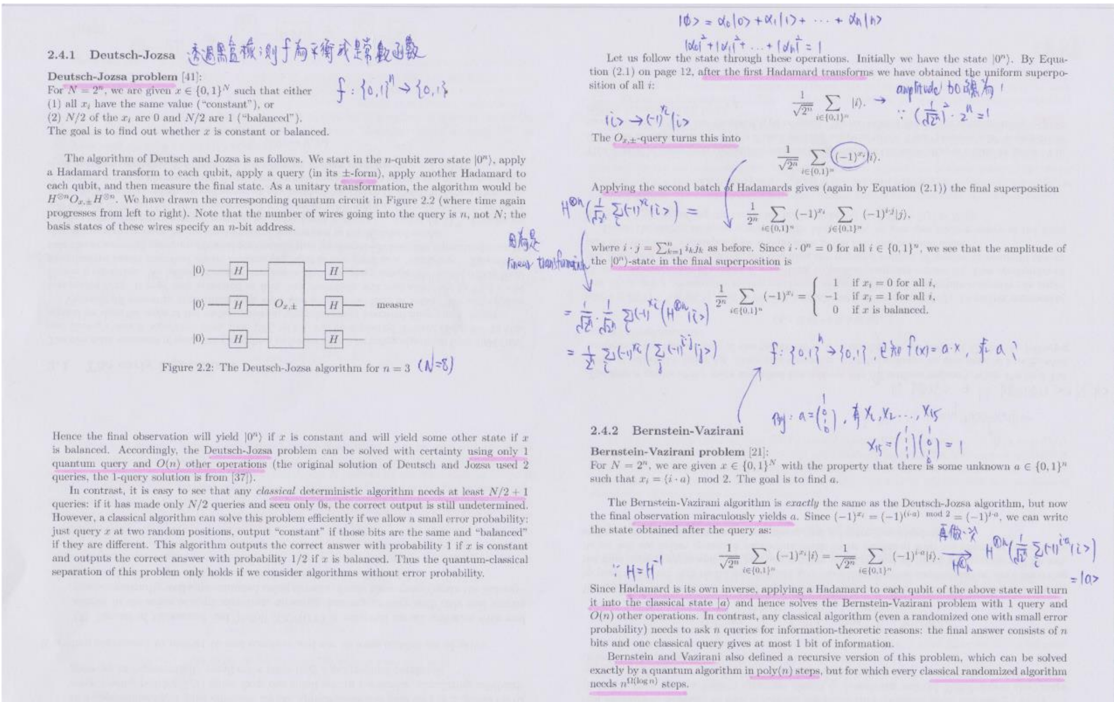
(7) Deutsch-Jozsa problem

- $N = 2^n$, given a function f: $\{0, 1\}^N \to \{0, 1\}$. To check if f is constant or balanced.
  f is constant if all x_i get the same values. (i.e., $f(x_i)$ all equal to 0 or 1).
  f is balanced if half of x_i get 1 while the remaining ones get 0.
- The algorithm is $H^{\otimes n}O_{x,\pm}H^{\otimes n}$ (involve 1 quantum query and O(n) operations). We only need to measure the amplitude of the state $|0^n>$ in the final superposition to determine f is constant or balanced.
- $|0^n>$ (initial state)

  $\Rightarrow \frac{1}{\sqrt{2^n}}\sum_{i\in\{0,1\}^n}|i>$ (after H gate, we obtain the uniform superposition of all i)

  $\Rightarrow \frac{1}{\sqrt{2^n}}\sum_{i\in\{0,1\}^n}(-1)^{x_i}|i>$ (after oracle, mapping relation: $|i> \to (-1)^{x_i}|i>$)

  $\Rightarrow \frac{1}{2^n}\sum_{i\in\{0,1\}^n}(-1)^{x_i}\sum_{j\in\{0,1\}^n}(-1)^{i\cdot j}|j>$ (after H gate)



2.4.1 Deutsch-Jozsa 透過黑盒檢測f為平衡或是常數函數

Deutsch-Jozsa problem [41]: $f: \{0,1\}^n \to \{0,1\}$
For $N = 2^n$, we are given $x \in \{0,1\}^N$ such that either
(1) all x_i have the same value ("constant"), or
(2) $N/2$ of the x_i are 0 and $N/2$ are 1 ("balanced").
The goal is to find out whether x is constant or balanced.

The algorithm of Deutsch and Jozsa is as follows. We start in the n-qubit zero state $|0^n\rangle$, apply a Hadamard transform to each qubit, apply a query (in its $\pm$-form), apply another Hadamard to each qubit, and then measure the final state. As a unitary transformation, the algorithm would be $H^{\otimes n}O_{x,\pm}H^{\otimes n}$. We have drawn the corresponding quantum circuit in Figure 2.2 (where time again progresses from left to right). Note that the number of wires going into the query is n, not N; the basis states of these wires specify an n-bit address.

$|0\rangle$ — H — $O_{x,\pm}$ — H — measure

Figure 2.2: The Deutsch-Jozsa algorithm for $n = 3$ ($N=8$)

$|\phi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + \cdots + \alpha_n|n\rangle$

$|\alpha_0|^2 + |\alpha_1|^2 + \ldots + |\alpha_n|^2 = 1$

Let us follow the state through these operations. Initially we have the state $|0^n\rangle$. By Equation (2.1) on page 12, after the first Hadamard transforms we have obtained the uniform superposition of all i:

$$\frac{1}{\sqrt{2^n}}\sum_{i\in\{0,1\}^n}|i\rangle. \to$$

amplitude 加總為1 $\because (\frac{1}{\sqrt{2^n}})^2 \cdot 2^n = 1$

$|i\rangle \to (-1)^{x_i}|i\rangle$

The $O_{x,\pm}$-query turns this into

$$\frac{1}{\sqrt{2^n}}\sum_{i\in\{0,1\}^n}(-1)^{x_i}|i\rangle.$$

Applying the second batch of Hadamards gives (again by Equation (2.1)) the final superposition

$$H^{\otimes n}\left(\frac{1}{\sqrt{2^n}}\sum_i(-1)^{x_i}|i\rangle\right) = \frac{1}{2^n}\sum_{i\in\{0,1\}^n}(-1)^{x_i}\sum_{j\in\{0,1\}^n}(-1)^{i\cdot j}|j\rangle,$$

因為是 linear transformation

where $i \cdot j = \sum_{k=1}^n i_k j_k$ as before. Since $i \cdot 0^n = 0$ for all $i \in \{0,1\}^n$, we see that the amplitude of the $|0^n\rangle$-state in the final superposition is

$$= \frac{1}{\sqrt{2^n}}\cdot\frac{1}{\sqrt{2^n}}\sum_i(-1)^{x_i}(H^{\otimes n}|i\rangle)$$

$$= \frac{1}{2^n}\sum_i(-1)^{x_i}\left(\sum_j(-1)^{i\cdot j}|j\rangle\right)$$

$$\frac{1}{2^n}\sum_{i\in\{0,1\}^n}(-1)^{x_i} = \begin{cases} 1 & \text{if } x_i = 0 \text{ for all } i, \\ -1 & \text{if } x_i = 1 \text{ for all } i, \\ 0 & \text{if } x \text{ is balanced.} \end{cases}$$

Hence the final observation will yield $|0^n\rangle$ if x is constant and will yield some other state if x is balanced. Accordingly, the Deutsch-Jozsa problem can be solved with certainty using only 1 quantum query and $O(n)$ other operations (the original solution of Deutsch and Jozsa used 2 queries, the 1-query solution is from [37]).

In contrast, it is easy to see that any *classical* deterministic algorithm needs at least $N/2 + 1$ queries: if it has made only $N/2$ queries and seen only 0s, the correct output is still undetermined. However, a classical algorithm can solve this problem efficiently if we allow a small error probability: just query x at two random positions, output "constant" if those bits are the same and "balanced" if they are different. This algorithm outputs the correct answer with probability 1 if x is constant and outputs the correct answer with probability 1/2 if x is balanced. Thus the quantum-classical separation of this problem only holds if we consider algorithms without error probability.

$f: \{0,1\}^n \to \{0,1\}$, 已知 $f(x) = a\cdot x$, 求 a

例: $a = \begin{pmatrix}1\\0\\ \vdots\\0\end{pmatrix}$, 有 $x_1, x_2, \ldots, x_{15}$

$x_{15} = \begin{pmatrix}1\\1\\ \vdots\\1\end{pmatrix}\begin{pmatrix}1\\0\\ \vdots\\0\end{pmatrix} = 1$

2.4.2 Bernstein-Vazirani

Bernstein-Vazirani problem [21]:
For $N = 2^n$, we are given $x \in \{0,1\}^N$ with the property that there is some unknown $a \in \{0,1\}^n$ such that $x_i = (i \cdot a) \mod 2$. The goal is to find a.

The Bernstein-Vazirani algorithm is *exactly* the same as the Deutsch-Jozsa algorithm, but now the final observation miraculously yields a. Since $(-1)^{x_i} = (-1)^{(i\cdot a) \bmod 2} = (-1)^{i\cdot a}$, we can write the state obtained after the query as:

$$\frac{1}{\sqrt{2^n}}\sum_{i\in\{0,1\}^n}(-1)^{x_i}|i\rangle = \frac{1}{\sqrt{2^n}}\sum_{i\in\{0,1\}^n}(-1)^{i\cdot a}|i\rangle. \xrightarrow[H^{\otimes n}]{\text{再做一次}} H^{\otimes n}\left(\frac{1}{\sqrt{2^n}}\sum_i(-1)^{i\cdot a}|i\rangle\right) = |a\rangle$$

$\because H = H^{-1}$

Since Hadamard is its own inverse, applying a Hadamard to each qubit of the above state will turn it into the classical state $|a\rangle$ and hence solves the Bernstein-Vazirani problem with 1 query and $O(n)$ other operations. In contrast, any classical algorithm (even a randomized one with small error probability) needs to ask n queries for information-theoretic reasons: the final answer consists of n bits and one classical query gives at most 1 bit of information.

Bernstein and Vazirani also defined a recursive version of this problem, which can be solved exactly by a quantum algorithm in poly(n) steps, but for which every classical randomized algorithm needs $n^{\Omega(\log n)}$ steps.

- The classical deterministic algorithm needs at least $N/2+1$ queries.

(8) Bernstein-Vazirani problem

- $N = 2^n$, given a function f: $\{0, 1\}^N \to \{0, 1\}$ satisfying $f(i) = (i \cdot a)$. To find a.
- The algorithm is the same as Deutsch-Jozsa algorithm.
- $|\, 0^n >$ (initial state)

$\Rightarrow \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i>$ (after H gate, we obtain the uniform superposition of all i)

$\Rightarrow \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} (-1)^{x_i} |i>$ (after oracle, have a mapping relation: $i \to (-1)^{x_i}$)

$\Rightarrow \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} (-1)^{i \cdot a} |i>$ (because $x_i = i \cdot a$)

$\Rightarrow |\, a >$ (after H gate) because $|\, a >$ after H gate is as presented in previous step.

The Bernstein-Vazirani algorithm is *exactly* the same as the Deutsch-Jozsa algorithm, but now the final observation miraculously yields a. Since $(-1)^{x_i} = (-1)^{(i \cdot a) \bmod 2} = (-1)^{i \cdot a}$, we can write the state obtained after the query as: 再做一次

$$\frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} (-1)^{x_i} |i\rangle = \frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} (-1)^{i \cdot a} |i\rangle. \xrightarrow[H^{\otimes n}]{} H^{\otimes n}\left(\frac{1}{\sqrt{2^n}} \sum_i (-1)^{i \cdot a} |i>\right) = |a>$$

$\because H = H^{-1}$

Since Hadamard is its own inverse, applying a Hadamard to each qubit of the above state will turn it into the classical state $|a\rangle$ and hence solves the Bernstein-Vazirani problem with 1 query and

Simon's algorithm

- $N = 2^n$, given a function f: $\{0, 1\}^N \to \{0, 1\}^N$ with an unknown $s \in \{0,1\}^n$, for all $x, y \in \{0,1\}^n$ such that $f(x) = f(y)$ if and only if $x \otimes y = s$. To find s.

Chapter 3

Simon's Algorithm

$\boxed{\frac{1}{\sqrt{2^n}} \sum |i\rangle |0^n\rangle}$

e.g. $n=3$, $s=011$

$f(000) = f(011) = 010$
$f(100) = f(111) = 110$
$f(001) = f(010) = 101$
$f(101) = f(110) = 001$

$\boxed{\text{i.e., } f(i) = f(i \oplus s)}$

$\Rightarrow \frac{1}{\sqrt{8}}(|000000> + |001000> + |010000> + |011000> + |100000> + |101000> + |110000> + |111000>)$

$\downarrow$ Oracle

$\boxed{\frac{1}{\sqrt{2^n}} \sum |i\rangle |x_i\rangle}$

$\frac{1}{\sqrt{8}}(|000010> + |011010> + |100110> + |111110> + |001101> + |010101> + |101001> + |110001>)$

$\xrightarrow[110]{\text{假設觀測}}$ 塌縮成: $\frac{1}{\sqrt{2}}(|100> + |111>) \xrightarrow{\text{做}\boxed{H}} \frac{1}{2}(|000> + |011> - |100> - |111>) \xrightarrow[\text{方程式}]{\text{解}}$

$(0 \cdot s_1) \oplus (0 \cdot s_2) \oplus (0 \cdot s_3) = 0$
$(0 \cdot s_1) \oplus (1 \cdot s_2) \oplus (1 \cdot s_3) = 0$
$(1 \cdot s_1) \oplus (0 \cdot s_2) \oplus (0 \cdot s_3) = 0$
$(1 \cdot s_1) \oplus (1 \cdot s_2) \oplus (1 \cdot s_3) = 0$

$\Rightarrow s_2 \oplus s_3 = 0$, $s_1 = 0$, $s_1 \oplus s_2 \oplus s_3 = 0$ $\to$ ~~$s_1s_2s_3 = 000$~~ (不合), $s_1s_2s_3 = 011$

Note that x, viewed as a function from $[N]$ to $[N]$ is a 2-to-1 function, where the 2-to-1-ness is determined by the unknown *mask* s. The queries to the input here are slightly different from before: the input $x = (x_1, \ldots, x_N)$ now has variables x_i that themselves are n-bit strings, and one query gives such a string completely ($|i, 0^n\rangle \mapsto |i, x_i\rangle$). However, we can also view this problem as having $n2^n$ binary variables that we can query individually. Since we can simulate one x_i-query using only n binary queries (just query all n bits of x_i), this alternative view will not affect the number of queries very much.

3.2 The quantum algorithm

Simon's algorithm starts out very similar to Deutsch-Jozsa: start in a state of $2n$ zero qubits $|0^n\rangle|0^n\rangle$ and apply Hadamard transforms to the first n qubits, giving

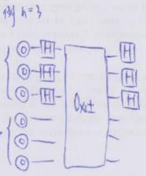
$$\frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i\rangle|0^n\rangle.$$

例 $n=3$

前n了

後n了

Oracle



The Deutsch-Jozsa problem showed an exponential quantum improvement over the best *deterministic* classical algorithms; the Bernstein-Vazirani problem shows a polynomial improvement over the best *randomized* classical algorithms that have error probability $\leq 1/3$. In this chapter we will combine these two features: we will see a problem where quantum computers are exponentially more efficient than bounded-error randomized algorithms.

3.1 The problem

Let $N = 2^n$, and $[N] = \{1, \ldots, N\}$, which we can identify with $\{0,1\}^n$. Let $j \oplus s$ be the n-bit string obtained by bitwise adding the n-bit strings j and s mod 2.

Simon's problem [101]:
For $N = 2^n$, we are given $x = (x_1, \ldots, x_N)$, with $x_i \in \{0,1\}^n$, with the property that there is some unknown non-zero $s \in \{0,1\}^n$ such that $x_i = x_j$ iff ($i = j$ or $i = j \oplus s$). The goal is to find s.

例: $n=3$ $\quad s = \begin{pmatrix}0\\1\\1\end{pmatrix}$ $\quad i = \begin{pmatrix}0\\0\\0\end{pmatrix}$, $j = \begin{pmatrix}0\\1\\1\end{pmatrix}$ $\quad i = \begin{pmatrix}0\\1\\0\end{pmatrix}$, $j = \begin{pmatrix}0\\0\\1\end{pmatrix}$

At this point, the second n-qubit register still holds only zeroes. A query turns this into

$$\frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i\rangle|x_i\rangle.$$

Now the algorithm measures the second n-bit register (this measurement is actually not necessary, but it facilitates analysis). The measurement outcome will be some value x_i and the first register will collapse to the superposition of the two indices having that x_i-value:

$$\frac{1}{\sqrt{2}}(|i\rangle + |i \oplus s\rangle)|x_i\rangle.$$

We will now ignore the second register and apply Hadamard transforms to the first n qubits. Using Equation (2.1) and the fact that $(i \oplus s) \cdot j = (i \cdot j) \oplus (s \cdot j)$, we can write the resulting state as

$$\frac{1}{\sqrt{2^{n+1}}}\left(\sum_{j \in \{0,1\}^n} (-1)^{i \cdot j}|j\rangle + \sum_{j \in \{0,1\}^n} (-1)^{(i \oplus s) \cdot j}|j\rangle\right) =$$

$$\frac{1}{\sqrt{2^{n+1}}}\left(\sum_{j \in \{0,1\}^n} (-1)^{i \cdot j}\left(1 + (-1)^{s \cdot j}\right)|j\rangle\right).$$

The Fourier transform

(1) F_N is a unitary matrix

- Define the (j, k) entry of matrix F_N by $\frac{1}{\sqrt{N}} w_N^{jk} = \frac{1}{\sqrt{N}}\left(\cos\left(\frac{2\pi j}{N}\right) + i * sin\left(\frac{2\pi k}{N}\right)\right)$
- F_2 is $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. The (1,1) entry is $-1 = \left(\cos\left(\frac{2\pi 1}{2}\right) + i * sin\left(\frac{2\pi 1}{2}\right)\right)$

(2) Fast Fourier transform (FFT) takes O(Nlog$N$)

- Recursively separate Fourier transform into half size of the even-numbered entries and half size of the odd-numbered entries.

$$\begin{aligned}
\hat{v}_j &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} v_k \\
&= \frac{1}{\sqrt{N}} \left(\sum_{\text{even } k} \omega_N^{jk} v_k + \omega_N^{j} \sum_{\text{odd } k} \omega_N^{j(k-1)} v_k \right) \\
&= \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{N/2}} \sum_{\text{even } k} \omega_{N/2}^{jk/2} v_k + \omega_N^{j} \frac{1}{\sqrt{N/2}} \sum_{\text{odd } k} \omega_{N/2}^{j(k-1)/2} v_k \right)
\end{aligned}$$

(3) Quantum Fourier transform (QFT)

- An example, $N = 8$ ($n = 3$) we have 3 qubits.

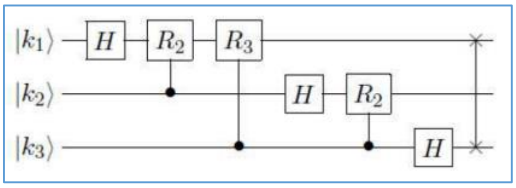
$F_8\,|k_1k_2k_3>$

$= \frac{1}{\sqrt{2}}(|0> + \boxed{e^{2\pi 0.k_3}}|1>) \otimes \frac{1}{\sqrt{2}}(|0> + e^{2\pi 0.k_2k_3}|1>) \otimes \frac{1}{\sqrt{2}}(|0> + e^{2\pi 0.k_1k_2k_3}|1>)$

l=1, $k_1k_2k_3 \Rightarrow k_1k_2.k_3$

$e^{k_1k_2.k_3} = e^{k_1k_2} * e^{0.k_3} = e^{0.k_3}$

$$\begin{aligned}
F_N|k\rangle &= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i jk/2^n} |j\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i (\sum_{\ell=1}^{n} j_\ell 2^{-\ell})k} |j_1 \ldots j_n\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \prod_{\ell=1}^{n} e^{2\pi i j_\ell k/2^\ell} |j_1 \ldots j_n\rangle \\
&= \bigotimes_{\ell=1}^{n} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i k/2^\ell} |1\rangle \right).
\end{aligned}$$



(4) Efficient circuit for QFT

Note: Rs gate = $\begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^s} \end{pmatrix}$

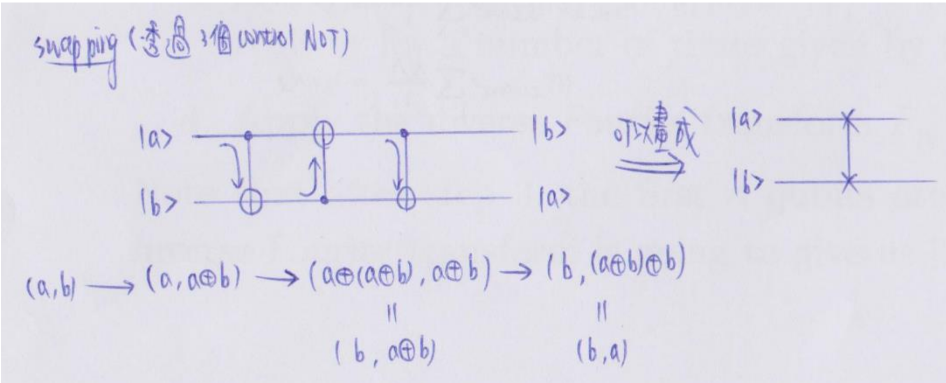
Note: $H\,|0> = \frac{1}{\sqrt{2}}(|0> + |1>)$, $H\,|1> = \frac{1}{\sqrt{2}}(|0> - |1>)$.

It can be generalized as $H\,|i> = \frac{1}{\sqrt{2}}(|0> + (-1)^i\,|1>)$

We only go through $|k_3>$, because $|k_1>$ and $|k_2>$ are in similar way.

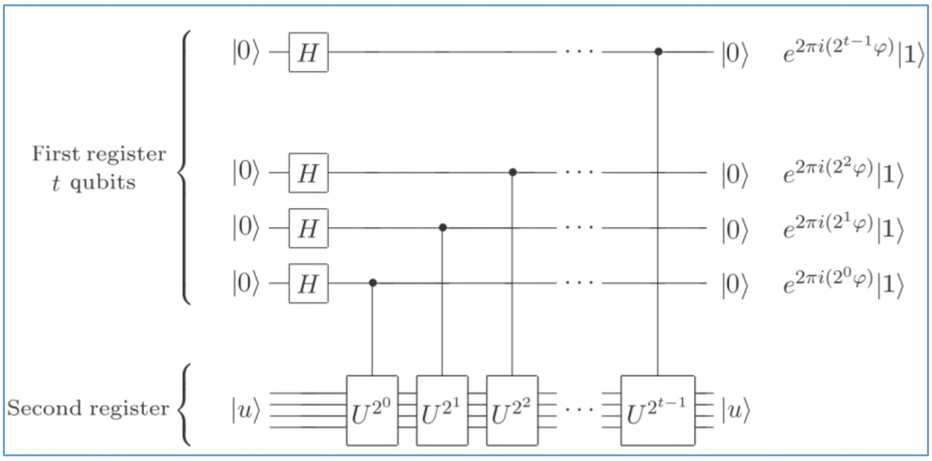
Applying H to $|k_1>$ obtains the state $\frac{1}{\sqrt{2}}(|0> + (-1)^{k1}\ |1>)$ and $(-1)^{k1} = e^{2\pi 0.k_1}$. We can rewrite as $\frac{1}{\sqrt{2}}(|0> + e^{2\pi 0.k_1}|1>)$. And then conditioned on $|k_2>$ applying R_2 obtains the state $\frac{1}{\sqrt{2}}(|0> + e^{2\pi 0.k_1k_2}|1>)$. Finally conditioned on $|k_3>$ applying R_3 obtains the state $\frac{1}{\sqrt{2}}(|0> + e^{2\pi 0.k_1k_2k_3}|1>)$. Swap gates are put at the end of the circuit. (because output qubits have the wrong order with respect to the proper QFT, i.e., output circuit qubit $|k_1>$ gets $\frac{1}{\sqrt{2}}(|0> + e^{2\pi 0.k_1k_2k_3}|1>)$, whereas the output QFT qubit $|k_1>$ gets $\frac{1}{\sqrt{2}}(|0> + e^{2\pi 0.k_3}|1>)$.)

(5) Swap circuit



(6)Phase estimation

- Suppose we apply U and are given an eigenvector $|\psi>$ of U ($U|\psi> = \lambda\ |\psi>$). We want to approximate eigenvalue $\lambda = e^{2\pi\phi}$, where $\phi \in [0,1)$, $\phi = 0.\phi_1\phi_2\phi_3\ldots\phi_n$.



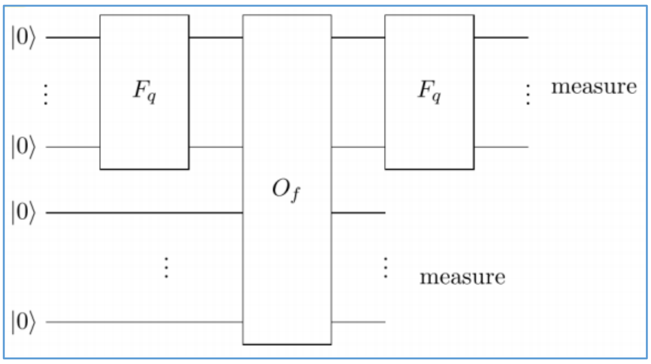
Shor's factoring algorithm

(1) The period-finding (order-finding) problem

- We want to find factors of the composite number $N>1$ and assume that N is odd and not a prime power. Given an integer $x \in \{0, 1, 2, \ldots, N-1\}$ which is coprime to N. Consider the sequence $x^0 \pmod{N}$, $x^1 \pmod{N}$, $x^1 \pmod{N}$,… The sequence will cycle with the period r, where $0< r \le N$ such that $x^r = 1 \pmod{N}$. The period r is even and $x^{r/2}+1$ and $x^{r/2}-1$ are not multiples of N. If we find r, then $\gcd(x^{r/2}+1, N)$ and $\gcd(x^{r/2}-1, N)$ are the factors of N.

  $x^r = 1 \pmod{N}$
  $\Rightarrow x^r - 1 = 0 \pmod{N}$
  $\Rightarrow (x^{r/2}+1)(x^{r/2}-1) = 0 \pmod{N}$
  $\Rightarrow (x^{r/2}+1)(x^{r/2}-1) = kN$ for some k.

- Given a function f and black box that maps $|a>|0^n> \rightarrow |a>|f(a)>$ where $f(a)=x^a \pmod{N}$. We want to find the period r.



Step 1: pick $q = 2^L$ such that $N^2 < q \le 2N^2$.
Step 2: the first (second) register consists of L ($\lceil logN \rceil$) qubits. (initial state $|0^n>$)
Step 3: apply QFT to the first register. (The second register is invariant)
Step 4: apply O_f to all registers. ($|a>|0^n> \rightarrow |a>| f(a)=x^a \pmod{N}>$)
Step 5: measure the second register, and the state will collapse to observed state.
Step 6: apply QFT to observed state.
Step 7: measure and compute the period r.

(2) Shor’s algo for factoring 15 (I gave another example for n=21)

e.g. $n=21$, $x=11$

① $n^2 = 441 < q = 2^l < 2n^2 = 882$ $\quad\therefore q = 512$

$\because 11^6 = 1 \pmod{21}$

$\Rightarrow 6 = \text{order}_{21}(11)$

② 將H做用在Reg1 $\Rightarrow \frac{1}{\sqrt{512}} \sum_{a=0}^{511} |a\rangle|0\rangle$

③ 將mod exponential 做用在Reg2 $\Rightarrow \frac{1}{\sqrt{512}} \sum_{a=0}^{511} |a\rangle|11^a \pmod{21}\rangle$

$= \frac{1}{\sqrt{512}} \left(|0\rangle|1\rangle + |1\rangle|11\rangle + |2\rangle|16\rangle \cdots\right)$

④ 任意觀測Reg2中的其中一個
就會得到Reg1坍塌後的結果
並對他們做QFT

⑤ 觀測Reg1. $\underbrace{|85\rangle, |171\rangle, |256\rangle, |341\rangle, |427\rangle, |512\rangle}_{|\frac{512}{6}\cdot i\rangle \text{ where } i=1\sim 6}$ 比例分別佔大約 $\frac{1}{6}$

⑥ 假設觀測427

$$\frac{c}{q} = \frac{427}{512} = 0 + \frac{1}{1+\frac{1}{5+\frac{1}{42+\frac{1}{2}}}} = [0,1,5,42,2] \approx \frac{5}{6}$$

$6 = r$

$$\therefore \gcd(x^{\frac{r}{2}} \pm 1, n) = \gcd(11^3 \pm 1, 21)$$

$$\Rightarrow \begin{cases} \gcd(7,21) = 7 \\ \gcd(9,21) = 3 \end{cases}$$

#

(3)Continued fractions

$$[a_0, a_1, \ldots, a_m] = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\cdots}}}$$

Hidden subgroup problem

(1) Hidden subgroup problem (HSP)

Given a known group G and a function f: $G \to S$ where S is some finite set. Suppose f has the property that there exists a subgroup $H \le G$ such that f is constant within each coset, and distinct on different cosets: $f(g) = f(g')$ if and only if $gH = g'H$. The goal is to find H.

(2) Reduce Simon's problem to HSP

G is the additive group $\mathbf{Z}_2{}^n = \{0, 1\}^n$ of size 2^n, $H = \{0, s\}$ for a hidden s $\in \{0, 1\}^n$, and satisfies $f(x) = f(y)$ if and only if $x - y \in H$. Therefore, finding the generator of H (finding s) solves Simon's problem.

(3) Reduce period-finding to HSP

f: $\mathbf{Z} \to \mathbf{Z}_n{}^*$ by $f(a) = x^a$ mod n, and find the period r of f. Since $<x>$ is a size-r subgroup of the group $\mathbf{Z}_n{}^*$, and the period r divides $|\mathbf{Z}_n{}^*| = \phi(N)$, we can restrict the domain of f to $\mathbf{Z}_{\phi(n)}$. Therefore, let $G = \mathbf{Z}_{\phi(N)}$, and consider its subgroup $H = <r>$ of all multiples of r up to $\phi(N)$ (i.e., $H = r\,\mathbf{Z}_{\phi(n)} = \{0 ,r , 2r,\ldots, \phi(N) - r\}$). To find generator of H (finding r) solves the period-finding problem.

(4) Reduce discrete logarithm to HSP

Take $G = \mathbf{Z}_N * \mathbf{Z}_N$, and define a function f: $G \to C$ by $f(x, y) = r^x A^{-y}$. For group elements $g_1 = (x_1, y_1)$, and $g_2 = (x_2, y_2) \in G$. We have $f(g_1) = f(g_2) \Leftrightarrow r^{x1-ay1} = r^{x2-ay2} \Leftrightarrow (x_1 - x_2) = a(y_1 - y_2)$ mod $N \Leftrightarrow g_1 - g_2 \in <(a, 1)>$. Let H be the subgroup of G generated by the element $(a, 1)$. To find the generator of H solves the discrete logarithm problem.

(5) Shor's algo for 4^k=10(mod 13)

- Given g, x, prime p, and the algorithm tries to find $k = \log_g x \pmod{p-1}$.

  Step 1: pick $q = 2^t$ such that $p \le q \le 2p$

  Step 2: initial three quantum registers with zeros. ($|\psi_0\rangle = |0\rangle|0\rangle|0\rangle$)

  Step 3: apply H to the first and second register. ($|\psi_1\rangle = \frac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}|a|b|0>$)

  Step 4: perform mod exponential. ($|\psi_2\rangle = \frac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}|a|b|g^a x^b \pmod p>$)

  Step 5: measure the third register. Suppose we observe m where $g^L = m \pmod p$.

  ($|\psi_3\rangle = \frac{1}{\sqrt{p-1}}\sum_{b=0}^{p-2}|L - br - k_b(p-1) > |b>$)

  Step 6: apply QFT to the first and second register.

$(|\psi_4> = \frac{\sqrt{p-1}}{q}\sum_{\mu=0}^{q-1} w_q^{L\mu}|\mu, \mu r>)$

Step 7: measure the first and second register, and get $(\mu, \mu r)$.

$k = \mu^{-1}\mu r (mod\ (p-1))$

Example 4.12. Let $g = 4, p = 13, x = 10$. We try to find

$$k \equiv \log_g 10 \pmod{12},$$

such that

$$10 \equiv 4^k \pmod{13}.$$

[1] Find a number q such that $\overset{p}{13} \le q = 2^4 = 16 < 2 \cdot 13$. $>p$

[2] Initialize the three quantum registers with zeroes:

$$|\Psi_0\rangle = |0, 0, 0\rangle.$$

[3] Perform a Hadamard transform on Reg1 and Reg2, we get

$$H : |\Psi_0\rangle \to |\Psi_1\rangle = \frac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}|a\rangle|b\rangle|0\rangle$$
$$= \frac{1}{12}\sum_{a=0}^{11}\sum_{b=0}^{11}|a\rangle|b\rangle|0\rangle.$$

[4] Perform the modular exponentiations, we get

$$U_f : |\Psi_1\rangle \to |\Psi_2\rangle = \frac{1}{12}\sum_{a=0}^{11}\sum_{b=0}^{11}|a\rangle|b\rangle|4^a \cdot 10^b \pmod{13}\rangle.$$

The relationship between $4^a \cdot 10^b \pmod{13}$ and a, b, shown in the following table:

	b											
a	0	1	2	3	4	5	6	7	8	9	10	11
0	1	10	9	12	3	4	1	10	9	12	3	4
1	4	1	10	9	12	3	4	1	10	9	12	3
2	3	4	1	10	9	12	3	4	1	10	9	12
3	12	3	4	1	10	9	12	3	4	1	10	9
4	9	12	3	4	1	10	9	12	3	4	1	10
5	10	9	12	3	4	1	10	9	12	3	4	1
6	1	10	9	12	3	4	1	10	9	12	3	4
7	4	1	10	9	12	3	4	1	10	9	12	3
8	3	4	1	10	9	12	3	4	1	10	9	12
9	12	3	4	1	10	9	12	3	4	1	10	9
10	9	12	3	4	1	10	9	12	3	4	1	10
11	10	9	12	3	4	1	10	9	12	3	4	1

[5] Measure Reg3, suppose we observe 4 satisfying $4^l \equiv 4 \pmod{13}$, where $0 \le l \le 11$. Now Reg1 and Reg2 are in the states

$$\frac{1}{\sqrt{12}}(|0\rangle|5\rangle + |0\rangle|11\rangle + |1\rangle|0\rangle + |1\rangle|6\rangle + |2\rangle|1\rangle + |2\rangle|7\rangle + |3\rangle|2\rangle + |3\rangle|8\rangle +$$
$$|4\rangle|3\rangle + |4\rangle|9\rangle + |5\rangle|4\rangle + |5\rangle|10\rangle + |6\rangle|5\rangle + |6\rangle|11\rangle + |7\rangle|0\rangle + |7\rangle|6\rangle +$$
$$|8\rangle|1\rangle + |8\rangle|7\rangle + |9\rangle|2\rangle + |9\rangle|8\rangle + |10\rangle|3\rangle + |10\rangle|9\rangle + |11\rangle|4\rangle + |11\rangle|10\rangle).$$

[6] Perform QFT on Reg1 and Reg2, we get

$$\frac{\sqrt{12}}{16}\sum_{\mu=0}^{15} w_{16}^{3\mu}|\mu, \mu r\rangle$$
$$= \frac{\sqrt{12}}{16}(|0\rangle|0\rangle + |1\rangle|5\rangle + |2\rangle|10\rangle + |3\rangle|15\rangle + |4\rangle|4\rangle + |5\rangle|1\rangle +$$
$$|6\rangle|6\rangle + |7\rangle|11\rangle + |8\rangle|4\rangle + |9\rangle|9\rangle + |10\rangle|2\rangle + |11\rangle|7\rangle + |12\rangle|0\rangle +$$
$$|13\rangle|5\rangle + |14\rangle|10\rangle + |15\rangle|3\rangle.$$

where $w_{16} = e^{\frac{2\pi i}{16}}$.

[7] Measure Reg1 and Reg2, we get $(13, 5)$, thus $r \equiv 13^{-1} \cdot 5 \pmod{12} \equiv 5$.

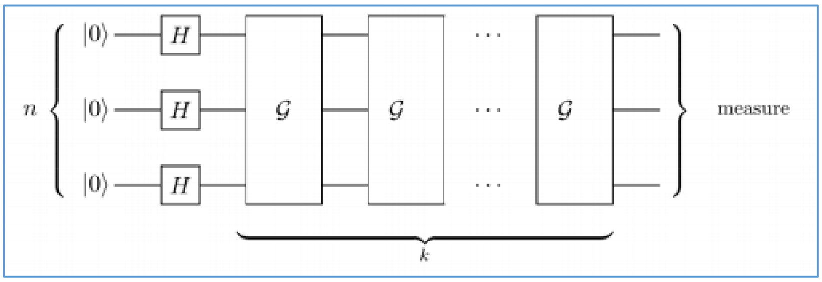
<u>(6) Pohlig-Hellman (略)</u>

Grover's search algorithm

(1) The search problem

- $N = 2^n$, given a function f: $\{0, 1\}^N \rightarrow \{0, 1\}$ To find x_i such that $f(x_i)=1$.
- The Grover's algorithm takes $O(\sqrt{N})$ Grover iterates.

  Grover iterate is $G = H^{\otimes n}RH^{\otimes n}O_{x,\pm} = (2|U\rangle\langle U| - I)$, where $R = (2|0^n\rangle\langle 0^n| - I)$.



  Step 1: apply H to n qubits, and obtain the state $|U\rangle$

  Step 2: $|U\rangle = \sin(\theta)|G\rangle + \cos(\theta)|B\rangle$, where $\theta = \sin^{-1}(\sqrt{t/N})$.

  $|G\rangle$ represents a good state, whereas $|B\rangle$ represents a bad state.

  t is the number of solution(s).

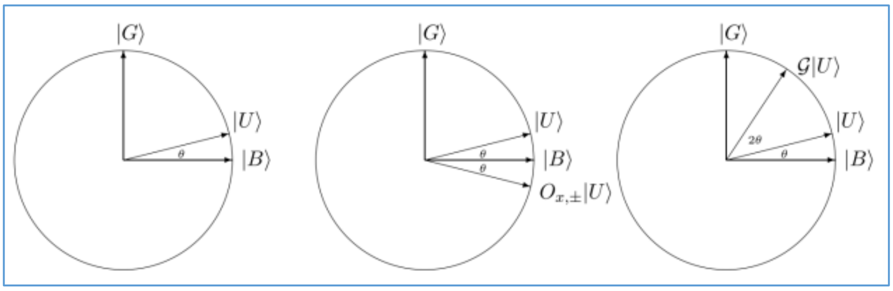
  Step 3: apply k Grover iterates and then measure.

(2) Geometric argument

- Grover iterate can be separated into two parts, which are $H^{\otimes n}RH^{\otimes n}$ and $O_{x,\pm}$.

  Reflect through (phase inversion) $|B\rangle$ (i.e., apply $O_{x,\pm}$.)

  Reflect through (inversion about mean) $|U\rangle$ (i.e., apply $H^{\otimes n}RH^{\otimes n}$)



(3) Algebraic argument

- After k Grover iterates, the state becomes $\sin((2k+1)\theta)|G\rangle + \cos((2k+1)\theta)|B\rangle$. If now we measure, then the probability of seeing the solution is $P_k = \sin((2k+1)\theta)^2$.
- We choose $(2k+1)\theta = \pi/2$, and therefore $P_k = \sin(\pi/2)^2 = 1$.

  $\because \theta \cong \sin\theta$ and $\sin\theta = \sqrt{t/N}$

  $\therefore$ $(2k+1)\theta = \pi/2$

  $\Rightarrow (2k+1)\sqrt{t/N} = \pi/2$

  $\Rightarrow (2k+1) = \pi/2\sqrt{N/t} \Rightarrow k = \pi/4\sqrt{N/t} - 1/2$

  One can see that if t increases, it involves less than k Grover iterate.

Lattice-based cryptography

(1) Congruential PKC (How is it related to a lattice problem?)

Alice	Bob
Key Creation	
Choose a large integer modulus q. Choose secret integers f and g with $f < \sqrt{q/2}$, $\sqrt{q/4} < g < \sqrt{q/2}$, and $\gcd(f, qg) = 1$. Compute $h \equiv f^{-1}g \pmod{q}$. Publish the public key (q, h).	
Encryption	
	Choose plaintext m with $m < \sqrt{q/4}$. Use Alice's public key (q, h) to compute $e \equiv rh + m \pmod{q}$. Send ciphertext e to Alice.
Decryption	
Compute $a \equiv fe \pmod{q}$ with $0 < a < q$. Compute $b \equiv f^{-1}a \pmod{g}$ with $0 < b < g$. Then b is the plaintext m.	

- An attacker can find the private key (f, g) from the known public key (q, h). The equation $h = f^{-1}g \pmod{q}$ can be rewritten as $fh = g \pmod{q}$. If an attacker can find any pair of positive integers F and G satisfying $Fh = G \pmod{q}$, $F = O(\sqrt{q})$, and $G = O(\sqrt{q})$, then (F, G) is likely to serve as a decryption key.
  $\because Fh = G \pmod{q}$
  $\therefore Fh = G + Rq$
  $(F, G) = F(1, h) - R(0, q)$, where $(1, h)$ and $(0, q)$ are the known vectors. It is related to a lattice problem. Notably, there is an extremely rapid method for finding short vectors in 2-dimesional lattices.

(2) Knapsack PKC (How is it related to a lattice problem?)

- Use a list $M = (M_1, M_2, \ldots, M_n)$ to encode a secret binary vector $x = (x_1, x_2, \ldots, x_n)$ into S, where $S = \sum_{i=1}^{n} x_i M_i$. However, it takes $O(2^{n/2+\varepsilon})$ to decode. If the receiver possesses some trapdoor information of M, then the solution is unique and allow he/she to find x easily.
- Merkle-Hellman subset-sum cryptosystem
  Step 1: generate a superincreasing sequence $r = (r_1, r_2, \ldots, r_n)$.
  $r_{i+1} >= r_i$, for $1 \le i \le n - 1$
  Step 2: choose A and B with $B > 2r_n$ and $\gcd(A, B) = 1$.
  Step 3: public key $M = Ar_i \pmod{B}$, for $1 \le i \le n$.

(encode) Step 4: use M to encode binary plaintext x into ciphertext S, where $S = xm$
(decode) Step 5: compute $S' = A^{-1}S \pmod{B}$ and solve the subset-sum problem S'.

- It is impractical because of the larger public and private key sizes.
- An attacker can reformulate the subset-sum problem using vectors. If he/she can

find a small nonzero vector in lattices, then he/she will be able to find *t*, and to recover plaintext *x*.

$$L = \{a_1\boldsymbol{v}_1 + a_2\boldsymbol{v}_2 + \cdots + a_n\boldsymbol{v}_n + a_{n+1}\boldsymbol{v}_{n+1} : a_1, a_2, \ldots, a_{n+1} \in \mathbb{Z}\}.$$

$$\begin{pmatrix} 2 & 0 & 0 & \cdots & 0 & m_1 \\ 0 & 2 & 0 & \cdots & 0 & m_2 \\ 0 & 0 & 2 & \cdots & 0 & m_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2 & m_n \\ 1 & 1 & 1 & \cdots & 1 & S \end{pmatrix}.$$

$$\begin{aligned} \boldsymbol{v}_1 &= (2, 0, 0, \ldots, 0, m_1), \\ \boldsymbol{v}_2 &= (0, 2, 0, \ldots, 0, m_2), \\ &\vdots \qquad\qquad \vdots \\ \boldsymbol{v}_n &= (0, 0, 0, \ldots, 2, m_n), \\ \boldsymbol{v}_{n+1} &= (1, 1, 1, \ldots, 1, S). \end{aligned}$$

$$\boldsymbol{t} = \sum_{i=1}^{n} x_i\boldsymbol{v}_i - \boldsymbol{v}_{n+1} = (2x_1 - 1, 2x_2 - 1, \ldots, 2x_n - 1, 0),$$

(3) Gram-Schmidt algorithm

- Given a basis $v_1, v_2, \ldots, v_n$ for a vector space $V \subset R^m$. The goal is to find an orthogonal basis $v^*_1, v^*_2, \ldots, v^*_n$ for V.

Set $\boldsymbol{v}_1^* = \boldsymbol{v}_1$.
Loop $i = 2, 3, \ldots, n$.
  Compute $\mu_{ij} = \boldsymbol{v}_i \cdot \boldsymbol{v}_j^* / \|\boldsymbol{v}_j^*\|^2$ for $1 \le j < i$.
  Set $\boldsymbol{v}_i^* = \boldsymbol{v}_i - \sum_{j=1}^{i-1} \mu_{ij}\boldsymbol{v}_j^*$.
End Loop

(4) A basis for lattice L

- $v_1, v_2, \ldots, v_n$ is a basis of the lattice *L*, and $w_1, w_2, \ldots, w_n$ is in L (w_i can be written as a linear combination of the basis vectors: $a_{i1}v_1 + a_{i2}v_2 + \ldots + a_{in}v_n$). If the matrix

  $A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$ have integer entries, and $\det(A) = \pm 1$, then $w_1, w_2, \ldots, w_n$

  is also a basis of the lattice *L*.

(5) Fundamental domain of lattice L corresponding to a basis

- $v_1, v_2, \ldots, v_n$ is a basis of the lattice *L*. The fundamental domain of *L* is the set $F(v_1, v_2, \ldots, v_n) = \{ t_1v_1 + t_2v_2 + \ldots + t_nv_n \mid 0 \le t_i < 1\}$.
- Every fundamental domain for *L* has the same volume.

(6) det(L)

- determinant of *L* (also called the covolume of *L*): $F(v_1, v_2, \ldots, v_n) = \det(L)$

(7) Hadamard's inequality

- Every basis $(v_1, v_2, \ldots, v_n)$ satisfies $\|v_1\|\ \|v_2\| \ldots \|v_n\| \geq \det(L)$.

(8) SVP, CVP, apprSVP, apprCVP

- SVP (shortest vector problem): find a shortest nonzero vector in lattice L.
- CVP (closet vector problem): find a vector $w \in R^m$ that is not in L, and it is closet to a vector v in L.
- apprSVP (approximate SVP): find a shortest nonzero vector v in L that is no more than $\psi(n)$ times longer than the shortest nonzero vector v_{shortest}, and satisfying $\|v\| \leq \psi(n)\ \|v_{\text{shortest}}\|$, where $\psi(n)$ is a function of n.
- apprCVP (approximate CVP): the same as apprSVP.

  Note: any vector in L is $\| a_1v_1 + a_2v_2 + \ldots + a_nv_n \|^2$

  $= a_1^2 \| v_1 \|^2 + a_2^2 \| v_2 \|^2 + \ldots + a_n^2 \| v_n \|^2$

  (vectors are pairwise orthogonal, i.e, $v_i \cdot v_j = 0$ for $i \neq j$)

  Therefore, the shortest vectors are in the set $\{\pm v_1, \pm v_2, \ldots, \pm v_n\}$.

  The closet vector $w = t_1v_1 + t_2v_2 + \ldots + t_nv_n$, with $t_1, t_2, \ldots, t_n \in R$.

  Distance between w and v is $\|w - v\|^2 = (a_1 - t_1)^2 \| v_1 \|^2 + \ldots + (a_n - t_n)^2 \| v_n \|^2$.

  The distance is minimized if we take each t_i to be integer closet to a_i.

(9) Hermite's theorem (proved by using Minkowski's theorem)

- Hermite's theorem:
  - ✓ Definition: Every lattice L of dimension n contains a nonzero vector v satisfying $\| v \| \leq \sqrt{n}\det(L)^{1/n}$.
- Minkowski's theorem
  - ✓ Definition: Let $L \subset R^n$ be a lattice of dimension n and let S be a bounded symmetric convex set whose volume satisfies $\text{Vol}(S) > 2^n\det(L)$. Then S contains a nonzero lattice vector. If S is also closed, then it suffices to take $\text{Vol}(S) \geq 2^n\det(L)$.
  - ✓ Proof of Hermite's theorem:

    I. Let $L \subset R^n$ be a lattice and let S be the hypercube in R^n, centered at 0, whose sides have length $2B$.

    $S = \{(x_1, x_2, \ldots, x_n) \in R^n : -B \leq x_i \leq B$ for all i $\leq i \leq n\}$

    II. The set S is symmetric, closed, and bounded, and its volume is $\text{Vol}(S) = (2B)^n$.

    III. Set $B = \det(L)^{1/n}$, then $\text{Vol(S)} = 2^n\det(L)$.

    IV. There is a vector $a \in S \cap L$,

$$\|a\| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2} \leq \sqrt{n}B = \sqrt{n}\det(L)^{1/n}$$

(10) Hermite's theorem can be improved (Gaussian expected shortest length)

- Let $B_R(a)$ be a ball of radius R in $\boldsymbol{R}^n$, then the volume is

  $\mathrm{Vol}\big(B_R(a)\big) = \frac{\pi^{n/2}R^n}{gamma(1+n/2)}$, where $gamma(s+1) = s * gamma(s)$

  For larger n, the volume is approximately given by

  $\mathrm{Vol}(B_R(a))^{1/n} \approx \sqrt{2\pi e/n} * R$
- By the definition of Minkowski's theorem: $\mathrm{Vol}(S) \geq 2^n \det(L)$

  $\leftrightarrow \mathrm{Vol}(S)^{1/n} \geq 2\det(L)^{1/n}$.

  $\therefore\ \sqrt{2\pi e/n} \cdot R\ \geq 2\det(L)^{1/n}$

  $\Rightarrow \| v \| \leq \sqrt{2n/\pi e} \cdot \det(L)^{1/n}$
- Let the L be the lattice of dimension n, the Gaussian expected shortest length

  $\sigma(L) = \sqrt{n/2\pi e} \cdot \det(L)^{1/n} \cong 0.484\sqrt{n}\ \det(L)^{1/n}$

  That is, $\| v \| \leq 0.484\sqrt{n}\det(L)^{1/n}$.

(11) Convolution polynomial rings

- Fix a positive integer N. The ring of convolution polynomials (of rank N) is the quotient ring $R = \frac{Z[x]}{x^N-1}$. For example, if we have a term $x^k = x^{iN+j}$, then it can be reduced as x^j.
- The ring of convolution polynomials (mod q) is the quotient ring $R_q = \frac{(Z/qZ)[x]}{x^N-1}$.

  For example, we work in ring R_{11}, and if have a term -13, then it can be reduced as 9 (because $9 = -13 \pmod{11}$).
- Let $a(x) \in R_q$. The center-lift of $a(x)$ to R is unique polynomial $a'(x) \in R$ satisfying $a'(x) \bmod q = a(x)$ whose coefficients are chosen in the interval $-q/2 < a'_i \leq q/2$. For example, $N = 5$, $q = 7$, and consider the polynomial $a(x) = 5 + 3x + 4x^2$. The center-lift of $a(x) = -2 + 3x - 3x^2$. (because 5 and 4 are not in the interval $(-3, 3]$)
- (Center-lift of a) * (Center-lift of b) $\neq$ (Center-lift of $a*b$)
- The multiplicative inverse. For example, $N = 5$, $q = 2$, the polynomial $1 + x + x^4$ has an inverse $1 + x^2 + x^3$. (because $(1 + x + x^4) * (1 + x^2 + x^3) = 1$)
- Let q be prime, then $a(x) \in R_q$ has a multiplicative inverse if and only if $\gcd(a(x), x^N-1) = 1$ in $(Z/qZ)[x]$

(12) NTRU PKC (How is it related to a lattice problem?)

Definition. For any positive integers d_1 and d_2, we let

$$\mathcal{T}(d_1, d_2) = \left\{ \boldsymbol{a}(x) \in R : \begin{array}{l} \boldsymbol{a}(x) \text{ has } d_1 \text{ coefficients equal to } 1, \\ \boldsymbol{a}(x) \text{ has } d_2 \text{ coefficients equal to } -1, \\ \boldsymbol{a}(x) \text{ has all other coefficients equal to } 0 \end{array} \right\}.$$

Alice	Bob
Public parameter creation	
A trusted party chooses public parameters (N, p, q, d) with N and p prime, $\gcd(p,q) = \gcd(N,q) = 1$, and $q > (6d+1)p$.	
Key creation	
Choose private $\boldsymbol{f} \in \mathcal{T}(d+1, d)$ that is invertible in R_q and R_p. Choose private $\boldsymbol{g} \in \mathcal{T}(d, d)$. Compute $\boldsymbol{F}_q$, the inverse of $\boldsymbol{f}$ in R_q. Compute $\boldsymbol{F}_p$, the inverse of $\boldsymbol{f}$ in R_p. Publish the public key $\boldsymbol{h} = \boldsymbol{F}_q \star \boldsymbol{g}$. (mod q)	
Encryption	
	Choose plaintext $\boldsymbol{m} \in R_p$. Choose a random $\boldsymbol{r} \in \mathcal{T}(d, d)$. Use Alice's public key $\boldsymbol{h}$ to compute $\boldsymbol{e} \equiv p\boldsymbol{r} \star \boldsymbol{h} + \boldsymbol{m} \pmod{q}$. Send ciphertext $\boldsymbol{e}$ to Alice.
Decryption	
Compute $\boldsymbol{f} \star \boldsymbol{e} \equiv p\boldsymbol{g} \star \boldsymbol{r} + \boldsymbol{f} \star \boldsymbol{m} \pmod{q}$. Center-lift to $\boldsymbol{a} \in R$ and compute $\boldsymbol{m} \equiv \boldsymbol{F}_p \star \boldsymbol{a} \pmod{p}$. ← Center-lift module p	

- In brute-force search, it takes much time. Note that the number of $T(d_1,d_2) = N!/[d_1!d_2!(N-d_1-d_2)!]$.
- The equation $h = f * g \pmod{q}$ has a hidden relationship $f * h = g \pmod{q}$.

  $\Rightarrow f * h = g + qu$

  We can establish NTRU lattice $M_h^{NTRU} = \begin{pmatrix} I & h \\ o & qI \end{pmatrix}$ to find privates key f and g.

  If we find out the vector $(f, -u)$, then we can recover privates key.

  (because $(f, -u)\begin{pmatrix} I & h \\ o & qI \end{pmatrix} = (f, g)$)

$$M_{\boldsymbol{h}}^{\mathrm{NTRU}} = \left(\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & 1 & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right)$$

Notice that $M_{\boldsymbol{h}}^{\mathrm{NTRU}}$ is composed of four N-by-N blocks:

Upper left block = Identity matrix,
Lower left block = Zero matrix,
Lower right block = q times the identity matrix,
Upper right block = Cyclical permutations of the coefficients of $\boldsymbol{h}(x)$.