

# Quantum Computing

Vaughn Sohn

November 25, 2024

# Contents

<b>1</b>	<b>Quantum Algorithms</b>	<b>2</b>
1.1	Introduction . . . . .	2
1.2	Elementary quantum algorithms using quantum parallelism . . . . .	2
1.3	Hamiltonian simulations . . . . .	5
1.4	Quantum Fourier transform . . . . .	8
1.5	Phase estimation . . . . .	11
1.6	Applications of phase estimation . . . . .	15
1.7	Applications of the QFT . . . . .	19
1.8	Quantum search algorithms . . . . .	22
1.9	Amplitude estimation algorithm (Quantum counting) . . . . .	26
1.10	HHL (Harrow–Hassidim–Lloyd) algorithm . . . . .	28
1.11	Optimality of the quantum search algorithm . . . . .	30
<b>2</b>	<b>Introduction to Computational Complexity</b>	<b>33</b>
2.1	Introduction . . . . .	33
2.2	The class NP: Reducibility and completeness . . . . .	33
2.3	Quantum complexity . . . . .	36
<b>A</b>	<b>Useful Environments for the Note</b>	<b>39</b>
A.1	Useful Environment . . . . .	39
A.2	Commutative Diagram . . . . .	40
A.3	Fancy Stuffs . . . . .	40

# Chapter 1

## Quantum Algorithms

Lecture 9

### 1.1 Introduction

7 Oct. 10:30

이번 챕터에서 우리는 *Quantum Algorithm*에 대해 다루고자 한다. Quantum algorithm은 quantum circuit이나 quantum computer에서 구현되는 알고리즘을 지칭한다. Classical computer가 어려운 문제를 해결하기 위하여 만들어진 것처럼, quantum algorithm에 대해서 공부하고 새로운 방식을 고안하는 것은 quantum computer의 동작방식과 quantum computer의 한계를 분석하기 위한 중요한 과제이다. Quantum computer라는 개념이 등장하고 나서부터 지금까지 많은 종류의 quantum algorithm들이 고안되어 왔다. 이번 강의에서 다루고자 하는 quantum algorithm은 다음과 같다.

- Elementary quantum algorithms
- Hamiltonian simulations
- Quantum Fourier transform
- Phase estimation
- Quantum search algorithm (Grover search algorithm)
- Amplitude amplification / estimation algorithms
- HHL algorithm

### 1.2 Elementary quantum algorithms using quantum parallelism

Quantum mechanics만의 특징을 이용할 수 있는 quantum computer는 *Quantum parallelism*이라는 특성을 가진다. 이는 quantum computer가 특정 oracle; function  $f(x)$ 에 대하여 동시에 여러개의 입력에 대한 결과를 병렬적으로 얻을 수 있다는 의미이다. 고전적인 개념인  $f(x)$ 를 quantum computer에서 실행하기 위해서는, quantum computer의 연산단위인 *unitary operator*로 함수를 표현해야하는 필요가 있다.

먼저, 간단한 one-bit boolean function  $f : \{0, 1\} \rightarrow \{0, 1\}$ 를 생각해보자. 직관적으로, 이 함수에 대응되는 operator는 다음과 같이 설계할 수 있다. 이 operator는  $f(0) = 1$ 이라면,  $U_f |0\rangle = |1\rangle$ 처럼 동작하도록 quantum gate들을 이용하여 구현된다.

$$|x\rangle \xrightarrow{U_f} |f(x)\rangle$$

그러나 이러한 방식으로 operator를 설계하게 되면, *non-invertible* 함수  $f(x)$ 에 대한 operator는 더이상 unitary 조건을 만족하지 못한다. 따라서 이 문제를 해결하기 위하여 control을 수행하는 추가적인 input qubit을 추가하여 unitary operator가 되도록 설계한다.

**Definition 1.2.1 (Unitary oracle).** 함수  $f$ 에 대한 unitary operator  $U_f$ 는 다음과 같이 정의된다.

$$|x\rangle |y\rangle \xrightarrow{U_f} |x\rangle |y \oplus f(x)\rangle$$

$|x\rangle$ 는  $f$ 의 입력으로 사용되는 **oracle qubit**이며,  $|y\rangle$ 는 1일 때는  $f(x)$ 의 결과를 flip 시키고 0일 때는  $f(x)$ 의 결과를 반환하는 역할을 수행한다. 그럼 이렇게 설계한 unitary operator에 *superposition state*를  $|x\rangle$ 로 제공하면 결과적으로 우리는 다음과 같은 two-qubit state를 얻게 된다.

$$|+\rangle|y\rangle \xrightarrow{U_f} \frac{1}{\sqrt{2}}(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle)$$

즉, one-bit boolean function에서 가능한 모든 입력 0, 1에 대한 출력을  $U_f$ 를 한 번 호출함으로써 얻게 된 것이다! Classical computer에서 병렬연산은 서로 다른 컴퓨팅 자원을 사용할 뿐,  $f$ 를 여러번 호출해야 하는 사실은 변하지 않지만, quantum computer에서의 병렬연산은 실제로  $f$ 를 한 번 호출하여 모든 연산을 수행할 수 있다.

이렇게 어떤 함수  $f$ 에 대응되는 unitary operator를 설계하는 것은 n-bit boolean function에 대해서 쉽게 일반화할 수 있다. n-bit boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ 에 대응되는 unitary operator는 Definition 1.2.1을 이용하여 쉽게 설계할 수 있으며, 더 나아가 n-qubit에 대한 superposition state를 입력으로 제공하면 다음 결과를 얻게된다.

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|0\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle$$

이 회로를 이용하면 한 번의 연산만으로  $2^n$ 개의 입력에 대한 출력을 동시에 얻을 수 있다. 그러나 한 가지 주의해야 할 점은, 우리가 결과를 측정하는 순간 여러개의 superposition output은 사라지고 확률에 따라서 단 하나의 결과만을 얻을 수 있다는 사실이다. 따라서 이러한 quantum parallelism만의 독특한 특징을 잘 활용하여 효과적인 연산을 수행할 수 있도록 알고리즘을 설계하는 것이 중요하다.

이 강의에서는 parallelism의 장점을 활용하는 대표적인 알고리즘들(e.g., Deutsch's algorithm, Deutsch-Josza algorithm, Simon's algorithm, and Bernstein-Vazirani algorithm)에 대해 소개한다.

### 1.2.1 Deutsch's algorithm

Deutsch's algorithm은 주어진 one-bit boolean function  $f$ 이 **balance**인지 **constant**인지를 판단하는 문제를 해결한다.<sup>1</sup> Classical computer가 이 문제를 해결하기 위해서는 두 합수값  $f(0), f(1)$ 을 비교하기 위해서 반드시 2번의 함수 호출이 필요하다. 그러나, 지금부터 우리는 quantum computer는 단 한번의 gate call만으로 이 문제를 해결할 수 있음을 보이고자 한다.

1. input state : 우리는 다음과 같은 state를 input으로 제공한다.

$$|\psi\rangle = |0\rangle|1\rangle$$

2. apply Hadamard gates on both qubits :  $H$  gate를 가하면, 다음과 같은 상태로 변화한다.

$$|\psi\rangle = |+\rangle|- \rangle$$

3. apply  $U_f$  : operator를 통과한 state는 다음과 같다.<sup>2</sup>

$$\begin{aligned} |\psi\rangle &= U_f \left( \frac{1}{2} (|00\rangle + |10\rangle - |01\rangle - |11\rangle) \right) \\ &= \frac{1}{2} (|0, f(0)\rangle|1, f(1)\rangle - |0, 1 \oplus f(0)\rangle - |1, 1 \oplus f(1)\rangle) \\ &= \frac{1}{2} \left( |0\rangle(-1)^{f(0)}(|0\rangle - |1\rangle) + |1\rangle(-1)^{f(1)}(|0\rangle - |1\rangle) \right) \\ &= \frac{1}{2} \sum_{x \in \{0,1\}} (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle) \end{aligned}$$

따라서 각 case에 대해 state는 다음의 2가지 모습을 띠게 된다.

$$|\psi\rangle = \begin{cases} \pm |+\rangle|- \rangle & \text{if } f(0) = f(1) \\ \pm |- \rangle|- \rangle & \text{if } f(0) \neq f(1) \end{cases}$$

<sup>1</sup>constant:  $f(0) = f(1)$ , balance:  $f(0) \neq f(1)$

<sup>2</sup> $|f(x)\rangle - |1 \oplus f(x)\rangle$ 는  $f(x)$ 의 값에 따라서,  $|0\rangle - |1\rangle$  ( $f(x) = 0$ ) 또는  $|1\rangle - |0\rangle$  ( $f(x) = 1$ )이 된다.

4. apply again Hadamard gates on oracle qubit : 마지막으로 oracle qubit에  $H$  gate를 가하면,  $f$ 의 종류에 따라서 다음과 같은 상태가 된다.

$$|\psi\rangle = \begin{cases} \pm |0\rangle |- \rangle & \text{if } f(0) = f(1) \\ \pm |1\rangle |- \rangle & \text{if } f(0) \neq f(1) \end{cases}$$

따라서, oracle qubit을 측정하면, 100%의 확률로 0 또는 1의 값을 얻게될 것이며, 그 값에 따라서 우리는  $f$ 가 balance인지 constant인지를 다음 규칙에 따라서 쉽게 판단할 수 있다.  $\square$

$$|\psi\rangle = \begin{cases} \text{constant} & \text{if } q_o = 0 \\ \text{balance} & \text{if } q_o = 1 \end{cases}$$

Deutsch's algorithm은 classical algorithm보다 quantum algorithm 알고리즘이 더 효과적임을 보인 첫 번째 알고리즘이다. 하지만, 그 효과는 단지 2번의 호출을 1번으로 줄일 뿐이다. 따라서 지금부터 더 효과적인 알고리즘들에 대해서 소개하고자 한다.

## Lecture 10

### 1.2.2 Deutsch-Jozsa algorithm

14 Oct. 10:30

Deutsch-Jozsa algorithm은 간단히 말하자면, one-bit boolean function에 대한 문제인 Deutsch-Jozsa algorithm을  $n$ -bit boolean function에 대한 문제로 일반화한 것이다. 즉,  $n$ -bit boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ 이 **constant**인지 **balanced**인지를 판단하는 문제를 해결하는 알고리즈다.<sup>3</sup> 마찬가지로 classical computer가 이 문제를 해결하기 위해서는 최대  $2^{n-1} + 1$ 개의 function value를 비교해야하기 때문에  $2^{n-1} + 1$  번의 함수 호출을 필요로 한다. 그러나, 지금부터 이런 문제를 해결하려고 할 때도, quantum computer는 단 한번의 gate call만으로 이 문제를 해결할 수 있음을 보일 것이다.

1. input state : 우리는 다음과 같은 state를 input으로 제공한다.  $\{0, 1\}^n$ 의 가능한 모든 input을 나타내기 위하여, oracle qubit은  $n$ 개의 qubit으로 구성된다.

$$|\psi\rangle = |0\rangle^{\otimes n} |1\rangle$$

2. apply Hadamard gates on both qubits: 이때,  $|+\rangle^{\otimes n}$ 은 가능한 모든  $2^n$ 개의  $n$ -bit string들의 superposition이기 때문에, 다음과 같이 표현할 수 있다.

$$\begin{aligned} |\psi\rangle &= |+\rangle^{\otimes n} |- \rangle \\ &= \sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^n}} |x\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \end{aligned}$$

3. apply  $U_f$  : operator를 통과한 state는 다음과 같다. (Deutsch's algorithm의 표현을 이용하자)

$$\begin{aligned} |\psi\rangle &= U_f \frac{1}{\sqrt{2^{n+1}}} \left( \sum_{x \in \{0,1\}^n} |x\rangle (|0\rangle - |1\rangle) \right) \\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle) \end{aligned}$$

4. apply again Hadamard gates on oracle qubit :  $H$  gate를 임의의  $n$ -qubit computational basis에 가한 결과는 다음과 같다. 이는 single qubit에 대한 동작을  $n$ -qubit에 대해 독립적으로 적용한 결과이다.

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{x \cdot z} |z\rangle \quad (1.1)$$

이를 이용하여  $H$  gate를 적용한 state를 표현하면, 다음과 같다.

$$|\psi\rangle = \frac{1}{2^n} \sum_{x, z \in \{0,1\}^n} (-1)^{f(x) + x \cdot z} |z\rangle |- \rangle$$

<sup>3</sup>여기서 말하는 balanced는  $2^n$ 개의 input중에서  $2^{n-1}$ 개의 input에 대한 결과가 0이고, 나머지  $2^{n-1}$ 개의 input에 대한 결과가 1인 경우를 의미한다.

만약  $f(x)$ 가 constant function이라면,  $\forall x$ 에 대해서  $f(x)$ 의 값은 항상 동일하기 때문에,  $(-1)^{f(x)}$ 의 값이  $+1$ , 또는  $-1$ 이라는 constant가 되어 다음과 같이 나타낼 수 있다.

$$|\psi\rangle = \pm \frac{1}{2^n} \sum_{x,z \in \{0,1\}^n} (-1)^{x \cdot z} |z\rangle |-\rangle$$

$z = 0^n$ 인 경우를 가정해보자. 이는  $x$ 가 어떤 값이 되던지간에  $x \cdot z$ <sup>4</sup>의 값이 0이 되기 때문에, 다음과 같이 표현할 수 있게 된다.  $|0\rangle^{\otimes n}$ 의 amplitude의 square norm이 1이기 때문에, 100% 확률로  $0^n$ 을 측정할 수 있다.

$$\pm \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot 0^n} |0\rangle^{\otimes n} |-\rangle = \pm \frac{1}{2^n} 2^n |0\rangle^{\otimes n} |-\rangle = |\pm |0\rangle^{\otimes n} |-\rangle|$$

반면,  $f(x)$ 가 balance function이라면,  $f(x)$ 의 값이 0인 경우와 1인 경우가 정확히  $1/2$ 씩 나타나기 때문에, 각 항들이 소거되면서  $z = 0^n$ 에 대한 amplitude가 0이 된다.

$$\left( \frac{1}{2^n} 2^{n-1} |0\rangle^{\otimes n} |-\rangle \right) + \left( -\frac{1}{2^n} 2^{n-1} |0\rangle^{\otimes n} |-\rangle \right) = |0|0\rangle^{\otimes n} |-\rangle|$$

따라서, oracle qubit을 측정한 결과가  $0^n$ 인지 확인하여,  $f$ 가 constant인지 balanced인지를 알 수 있다.  $\square$

$$|\psi\rangle = \begin{cases} \text{constant} & \text{if } q_o = 0^n \\ \text{balance} & \text{if } q_o \neq 0^n \end{cases}$$

## 1.3 Hamiltonian simulations

Hamiltonian simulation은 quantum computer가 고안된 핵심적인 이유 중 하나이다. 리처드 파인만의 말을 인용하자면 자연, 그중에서도 특히 미시세계는 고전역학을 따르지 않기 때문에, quantum mechanical을 따르는 simulation이 필요하다.

**Note.** *Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy.*

Hamiltonian simulation은 간단히 말해 quantum state의 time evolution을 구하는 것이다. Schrödinger equation에 의하면, initial state  $|\psi(0)\rangle$ 의 시간  $t$ 에 대한 time evolution은 다음과 같이 주어진다.

$$|\psi(t)\rangle = e^{-iHt} |\psi(0)\rangle$$

이 방정식을 곧바로 해결하여 문제를 풀 수도 있지만, Hamiltonian matrix의 크기가 exponential하게 증가하기 때문에 이를 classical algorithm으로 효과적으로 해결하기는 어렵다.

### 1.3.1 Solution of Hamiltonian simulations

우리는 k-local Hamiltonian을 이용하여 Hamiltonian simulation 문제를 해결하고자 한다.

**Definition 1.3.1.** k-local<sup>a</sup> Hamiltonian은 주어진  $H$ 가  $n$ -qubit system에서 최대  $k$ 개의 system에 대해서만 동작하는  $H_i$ 들의 합으로 표현되는 Hamiltonian이다. (이때  $L$ 은  $n$ 에 대해 polynomial이다.)

$$H = \sum_{i=1}^L H_i$$

<sup>a</sup>여기서 local은 geometrically local이 아니라 단순히 system의 '개수'를 나타내기 위해 사용된다.

**Example.** 예를 들어, 다음과 같이 정의되는 Hamiltonian은 2개의 system (2, 4)에 대해서만 동작한다.

$$H_i = I_1 \otimes Z_2 \otimes I_3 \otimes X_4 \otimes I_{\perp}$$

<sup>4</sup> $x \cdot z = x_1 z_1 + x_2 z_2 + \dots + x_n z_n \mod 2$

Hamiltonian이 k-local Hamiltonian이라고 가정할 때, 이를 이용하여 어떻게 문제를 해결할 수 있을까? 그 아이디어는 단순하다.  $e^{-iHt}$ 를 계산하는 것은 어렵지만,  $e^{-iH_it}$ 는 최대  $k << n$  subsystem에만 작용하기에 더 단순하며, 그 단순성 덕분에 quantum circuit을 사용하여 시뮬레이션 하기 쉽다.

만약,  $[H_j, H_k] = 0$ <sup>5</sup>이 성립한다면  $e^{-iHt} = e^{-i\sum H_j t} = \prod e^{-iH_j t}$ <sup>6</sup>가 성립하기 때문에, 각각의  $H_i$ 에 대한 time evolution을 독립적으로 계산한 결과를 곱하여 전체 time evolution을 쉽게 구할 수 있다. 그러나 일반적으로  $[H_j, H_k] = 0$ 은 성립하지 않기 때문에, 우리는 Trotter formula를 이용한다.

**Theorem 1.3.1 (Trotter formula).** Hermitian operator  $H_j, H_k$ 에 대하여, 어떤  $t$ 에 대해서도 다음이 성립한다.<sup>a</sup>

$$e^{i(H_j+H_k)t} = \lim_{m \rightarrow \infty} \left( e^{iH_j t/m} e^{iH_k t/m} \right)^m$$

이는  $H_j, H_k$ 가 not-commute라 하더라도 항상 성립한다.

<sup>a</sup>직관적으로, 주어진 time  $t$  대신에, 매우 짧은 시간간격인  $t/m$ 에 대한 time evolution을  $m$ 번 반복하여  $t$ 에 대한 time evolution을 근사할 수 있다는 것을 의미한다.

따라서  $L = 2$ 인 k-local Hamiltonian  $H = H_1 + H_2$ 에 대해, Hamiltonian simulation을 수행하는 propagator  $\tilde{U}(t) = e^{-iH_1 t} e^{-iH_2 t}$ 를 시간에 대해서 미분하면, 다음과 같이 전개할 수 있다. (with initial condition  $\tilde{U}(0) = I$ )

- Eq. (1.3):  $H_1, H_2$ 는 matrix라서 순서를 바꿀 수 없기에, 다른 항을 더하여 항을 정리하고 다시 뺀다.
- Eq. (1.4): commutator를 사용하여 항을 정리한다.

$$i \frac{d\tilde{U}(t)}{dt} = H_1 e^{-iH_1 t} e^{-iH_2 t} + e^{-iH_1 t} H_2 e^{-iH_2 t} \quad (1.2)$$

$$= (H_1 + H_2) e^{-iH_1 t} e^{-iH_2 t} + e^{-iH_1 t} H_2 e^{-iH_2 t} - H_2 e^{-iH_1 t} e^{-iH_2 t} \quad (1.3)$$

$$= H e^{-iH_1 t} e^{-iH_2 t} + [e^{-iH_1 t}, H_2] e^{-iH_2 t} \quad (1.4)$$

$$= H \tilde{U}(t) + [e^{-iH_1 t}, H_2] e^{-iH_2 t} \quad (1.5)$$

미분 방정식에 대한 Duhamel's formula<sup>7</sup>를 이용하면,  $\tilde{U}(t)$ 를 다음과 같이 구할 수 있다.

$$\tilde{U}(t) = U(t) - i \int_0^t e^{-iH(t-s)} [e^{-iH_1 s}, H_2] e^{-iH_2 s} ds \quad (1.6)$$

### 1.3.2 Performance

그렇다면, 이제 실제 Hamiltonian에 대한 time evolution  $U(t) = e^{-iHt}$ 와 Trotter formula를 이용하여 근사한  $\tilde{U}(t)$ 의 error가 어떻게 bound되는지 분석해보자. Eq. (1.6)의 항을 이항시키면, 두 operator의 차이에 대한 norm의 upper bound를 구할 수 있다.<sup>8</sup>

$$\|\tilde{U}(t) - U(t)\| = \left\| -i \int_0^t e^{-iH(t-s)} [e^{-iH_1 s}, H_2] e^{-iH_2 s} ds \right\| \leq \int_0^t \| [e^{-iH_1 s}, H_2] \| ds \quad (1.7)$$

이때, norm은 다음과 같이 정의된다.

**Definition 1.3.2.**

$$\|A\| \triangleq \max_{|\psi\rangle} \|A|\psi\rangle\|_2 = \max_{|v\rangle \neq 0} \frac{\|A|v\rangle\|_2}{\||v\rangle\|_2}$$

이제 이 norm이 적절한 error rate  $\epsilon$ 에 의하여 bound됨을 보임으로써, 이 Hamiltonian simulation을 실제로 활용할 수 있을지 분석할 수 있다.

<sup>5</sup>  $[H_j, H_k] = H_j H_k - H_k H_j$

<sup>6</sup> 지수함수의 성질이 성립하기 위해서 필요한 조건. 실수들은 항상 commutative하기 때문에 이 성질이 자명하게 정립되었던 것

<sup>7</sup> See [https://en.wikipedia.org/wiki/Duhamel%27s\\_principle](https://en.wikipedia.org/wiki/Duhamel%27s_principle)

<sup>8</sup>  $e^{-iH_i t}$ 는 unitary operator이기 때문에 norm을 변화시키지 않기 때문에 무시해도 된다.

이를 위하여 다음과 같은 새로운 연산자를 가정하자.

$$G(t) \triangleq [e^{-iH_1 t}, H_2] e^{iH_1 t} = e^{-iH_1 t} H_2 e^{iH_1 t} - H_2, \text{ with } G(0) = 0 \quad (1.8)$$

이 연산자에 대하여  $\tilde{U}(t)$ 처럼 시간에 대한 도함수를 구하면 다음과 같고,

$$i \frac{d}{dt} G(t) = e^{-iH_1 t} [H_1, H_2] e^{iH_1 t}$$

양변을  $t$ 에 대해서 적분하면 다음과 같다.

$$G(t) = G(0) - i \int_0^t e^{-iH_1 s} [H_1, H_2] e^{iH_1 s} ds$$

norm을 취하고 triangle inequality를 이용하면,  $\|G(t)\|$ 에 대한 upper bound를 얻는다.

$$\|G(t)\| = \left\| -i \int_0^t e^{-iH_1 s} [H_1, H_2] e^{iH_1 s} ds \right\| \leq \int_0^t \| [H_1, H_2] \| ds = t \| [H_1, H_2] \| \quad (1.9)$$

이때,  $G(t)$ 의 norm은 자기자신의 정의 (1.8)에 의하여 다음 관계가 성립하게 된다.

$$\|[e^{-iH_1 t}, H_2]\| = \|G(t)\| \leq t \| [H_1, H_2] \|$$

따라서 이를 Eq. (1.7)에 대입하면, 다음을 얻는다.

$$\|\tilde{U}(t) - U(t)\| \leq \int_0^t \underbrace{\|[e^{-iH_1 s}, H_2]\|}_{s \| [H_1, H_2] \|} ds \leq \int_0^t s \| [H_1, H_2] \| ds$$

$\|[H_1, H_2]\|$ 는  $s$ 에 관계없는 상수이기 때문에, 다음과 같이 정리할 수 있으며,

$$\|\tilde{U}(t) - U(t)\| \leq \frac{t^2}{2} \| [H_1, H_2] \|$$

commutator를 전개한 뒤, triangle inequality와 norm의 성질<sup>9</sup>을 이용하면 다음과 같이 전개할 수 있다.

$$\begin{aligned} \|\tilde{U}(t) - U(t)\| &\leq \frac{t^2}{2} \| [H_1, H_2] \| = \frac{t^2}{2} \| H_1 H_2 + (-H_2 H_1) \| \leq \frac{t^2}{2} (\| H_1 H_2 \| + \| H_2 H_1 \|) \\ &\leq \frac{t^2}{2} 2 \| H_1 \| \| H_2 \| \leq t^2 \max \{ \| H_1 \|, \| H_2 \| \}^2 \end{aligned}$$

정리하면, 다음과 같다.

$$\|\tilde{U}(t) - U(t)\| = \|U(t) - \tilde{U}(t)\| \leq t^2 \max \{ \| H_1 \|, \| H_2 \| \}^2 \quad (1.10)$$

따라서 시간간격  $\Delta t$ 에 대한 error는 다음과 같이 bound된다.

$$\| e^{-iH\Delta t} - (e^{-iH_1 \Delta t} e^{-iH_2 \Delta t}) \| \leq (\Delta t)^2 \max \{ \| H_1 \|, \| H_2 \| \}^2 \quad (1.11)$$

Trotter formula에 의하여 시간간격  $\Delta t = t/m$ 에 대하여  $m$ 단계 근사의 오차는  $\Delta t$ 에 대한 단일 단계 오차의 누적으로 생각할 수 있기 때문에, 다음을 얻는다.

$$\left\| e^{-iHt} - (e^{-iH_1 \Delta t} e^{-iH_2 \Delta t})^m \right\| \leq m \frac{t^2}{m^2} \max \{ \| H_1 \|, \| H_2 \| \}^2 = \frac{t^2}{m} \max \{ \| H_1 \|, \| H_2 \| \}^2$$

이 bound를 사용하면, 우리가 원하는 target error rate  $\epsilon$ 을 달성하기 위해서 필요한  $m$ 의 값을  $m = O(t^2 \epsilon^{-1})$ 으로 결정할 수 있다.<sup>10</sup>

이 과정을 더 많은 term을 가지는 k-local Hamiltonian에 대해서 일반화할 수 있다.

$$\left\| e^{-i \sum_{i=1}^L H_i \Delta t} - \prod_{j=1}^L e^{-i H_j \Delta t} \right\| = O \left( \frac{t^2}{L} \sum_{i < j} \| [H_i, H_j] \| \right).$$

<sup>9</sup>  $\|AB\| \leq \|A\| \|B\|$

<sup>10</sup>  $\max \{ \| H_1 \|, \| H_2 \| \}^2$  (i.e.,  $\| H_i \|$ )의 값은 상수라서 무시하였다.

$\Delta t = t/m$ 로 가정하면 다음을 얻는다.

$$\left\| e^{-i \sum_{i=1}^L H_i t} - \left( \prod_{j=1}^L e^{-i H_j \Delta t} \right)^m \right\| = O \left( \frac{m \Delta t^2}{L} \sum_{i < j} \| [H_i, H_j] \| \right) = O \left( \frac{t^2}{m L} \sum_{i < j} \| [H_i, H_j] \| \right)$$

이때,  $\|H_i\| = O(1)$ 이므로  $\sum \| [H_i, H_j] \| = L^2$ 이 되어 다음과 같이 bound된다.

$$\left\| e^{-i \sum_{i=1}^L H_i t} - \left( \prod_{j=1}^L e^{-i H_j \Delta t} \right)^m \right\| = O \left( \frac{Lt^2}{m} \right).$$

$m = O(Lt^2\epsilon^{-1})$ 로 설정하게되면 우리는 항상 target error rate  $\epsilon$ 을 upper bound로 가지는 근사 operator  $\tilde{U}(t)$ 를 구성할 수 있고 이를 이용하여 simulation 할 수 있다. 따라서, 이 solution은 주어진  $k$ -local Hamiltonian에 대하여, time  $t$ 에 대한 quadratic overhead (i.e., error)를 가지고 simulation을 할 수 있음을 보여준다. 즉, simulation time이 증가하더라도 error rate은 polynomial하게 증가하게 된다.  $\square$

반면, 2nd-Trotter formula를 이용하여 표현할 수도 있다.

$$e^{i(A+B)\Delta t} = e^{iA\Delta t/2} e^{iB\Delta t} e^{iA\Delta t/2} + O((\Delta t)^3)$$

이를 이용하면, time  $t$ 에 대하여 error는 다음과 같이 표현된다.

$$\left\| e^{-i(A+B)t} - \prod_{i=1}^m e^{-iA\Delta t/2} e^{-iB\Delta t} e^{-iA\Delta t/2} \right\| = O(m(\Delta t)^3) = O(t^3/m^2)$$

즉,  $m = O(t^{3/2}\epsilon^{1/2})$ 으로 설정하면, target error rate  $\epsilon$ 을 달성할 수 있으며 이는 1st-Trotter formula를 사용한 simulation보다 더 효율적이다.<sup>11</sup> 2nd-Trotter 방법을  $L$ 개의 term을 갖는 Hamiltonian에 대하여 일반화하면  $m$ 은 다음과 같다.

$$m = O \left( \frac{\left( \sum_{j=1}^L \|H_j\| t \right)^{3/2}}{\epsilon^{1/2}} \right)$$

더 나아가  $p$ th order Trotter formula를 사용하면,  $m$ 은 다음과 같다.

$$m = O \left( \frac{\left( \sum_{j=1}^L \|H_j\| t \right)^{1+1/p}}{\epsilon^{1/p}} \right)$$

Hamiltonian simulation을 위하여 다양한 연구들이 현재까지도 진행되고 있으며, 대표적인 알고리즘들은 다음을 참고하라.

- Higher-order product formula [Chi+21]
- Linear combination of unitary (LCU) [BCK15]
- Quantum signal processing (*optimal*) [Haa19]

## Lecture 11

### 1.4 Quantum Fourier transform

16 Oct. 10:30

#### 1.4.1 Quantum Fourier transform

Quantum Fourier transform은 Shor algorithm이나 HHL algorithm과 같은 다양한 quantum algorithm에서 사용되는 중요한 알고리즘이다. QFT에 대해서 소개하기에 앞서, 먼저 classical Fourier transform에 대해서 간단히 알아보자.

---

<sup>11</sup>time  $t$ 에 대한  $3/2$  overhead

**Definition 1.4.1 (Discrete Fourier transform).** Discrete Fourier transform은 vector  $\mathbf{x} \in \mathbb{C}^N$ 을 입력으로 받아서, 다음과 같이 정의되는 연산을 수행하여 output vector  $\mathbf{y} \in \mathbb{C}^N$ 으로 변환하는 과정이다.<sup>a</sup>

$$y_k \triangleq \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N}$$

<sup>a</sup> $\mathbf{x} = x_0 \cdots x_{N-1}$ ,  $\mathbf{y} = y_0 \cdots y_{N-1}$

**Definition 1.4.2 (Quantum Fourier transform).** Quantum Fourier transform은 DFT와 유사하게, quantum state vector  $|x\rangle = |x_0 \cdots x_{N-1}\rangle$ 을 입력으로 받아서, output quantum state vector  $|y\rangle = |y_0 \cdots y_{N-1}\rangle$ 으로 변환하는 과정이다. 단, DFT와는 다르게 computational basis  $\{|0\rangle, \dots, |N-1\rangle\}$ 에 대한 변환만이 정의되어 있다.

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle.$$

따라서 임의의 state vector  $|x\rangle$ 의 transform은 변환된 basis vector  $|k\rangle$ 들의 linear combination으로 표현하게 된다.

$$\sum_{j=0}^{N-1} x_j |j\rangle \rightarrow \sum_{j=0}^{N-1} x_j \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle = \sum_{k=0}^{N-1} y_k |k\rangle,$$

이때,  $y_k$ 는 다음과 같다.<sup>a</sup>

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N}.$$

<sup>a</sup>DFT와 똑같다.

즉, QFT가 하는 일은 basis  $\{|k\rangle\}$ 에 대한 linear combination으로 표현된 벡터  $|x\rangle$ 를 다른 basis  $\{|j\rangle\}$ 에 대한 linear combination으로 나타내는 basis transform이다.<sup>12</sup>

이제 QFT가 어떻게 구현되는지 알아보자.  $|j\rangle$ 에 대한 변환을 수행하는 QFT의 circuit은 다음과 같다.

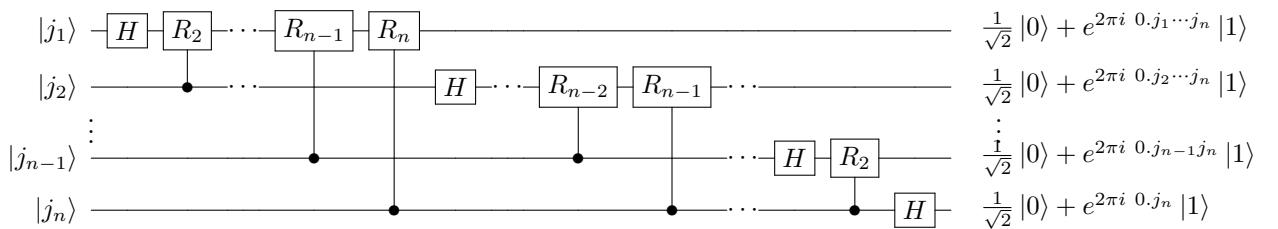


Figure 1.1: QFT circuit

QFT가  $n$ -qubit system에 대해서 computational basis를 변환시킬 때, 우리는 computational basis를 나타내기 위하여 binary representation을 도입하고자 한다. 어떤  $|j\rangle \in \{|0\rangle, \dots, |2^n - 1\rangle\}$ 에 대한 binary representation은 다음과 같다. ( $N = 2^n$ )

$$j = j_1 j_2 \dots j_n = j_1 2^{n-1} + \dots + j_n 2^0 = \sum_{k=1}^n j_k 2^{n-k}.$$

또한, 소수도 다음과 같은 binary representation으로 표현할 수 있다.

$$0.j_l j_{l+1} \dots j_m = j_l/2 + j_{l+1}/2^2 + \dots + j_m/2^{m-l+1} = \sum_{k=l}^m j_k / 2^{k-l+1}$$

<sup>12</sup>  $\{|k\rangle\}$  basis에서의 amplitude는  $x_i$ 이며,  $\{|j\rangle\}$  basis에서의 amplitude는  $y_i$ 이다.

Binary representation을 이용하면 QFT의 연산을 다음과 같이 분석할 수 있다.

- Eq. (1.12): Definition 1.4.2에 따라,  $|j\rangle$ 에 대한 QFT는 다음과 같이 표현된다.
- Eq. (1.13):  $k = \sum k_l 2^{n-l}$ 으로  $k/2^n = \sum k_l 2^{-l}$ 이다.
- Eq. (1.14):  $e^{a+b} = e^a e^b$ , 그리고  $|k\rangle$ 가  $n$ -qubit에 대해 tensor product로 표현됨을 이용한다.
- Eq. (1.15): 표현 단순화. ( $\sum_{k_1, k_2, \dots, k_n \in \{0,1\}} \equiv \sum_{k_l \in \{0,1\}}$ 로 표현)
- Eq. (1.16): (1)  $k_l = 0$ , then  $e^{2\pi i j k_l 2^{-l}} = e^0$ . (2)  $k_l = 1$ , then  $e^{2\pi i j k_l 2^{-l}} = e^{2\pi i j 2^{-l}}$
- Eq. (1.17): 모든 tensor product들을 전개한뒤 fraction을 binary representation으로 표현한다.

$$|j\rangle \rightarrow \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j k / 2^n} |k\rangle \quad (1.12)$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^1 \dots \sum_{k_n=0}^1 e^{2\pi i j (\sum_{l=1}^n k_l 2^{-l})} |k_1, \dots, k_n\rangle \quad (1.13)$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^1 \dots \sum_{k_n=0}^1 \bigotimes_{l=1}^n e^{2\pi i j k_l 2^{-l}} |k_l\rangle \quad (1.14)$$

$$= \frac{1}{2^{n/2}} \bigotimes_{l=1}^n \sum_{k_l=0}^1 e^{2\pi i j k_l 2^{-l}} |k_l\rangle \quad (1.15)$$

$$= \frac{1}{2^{n/2}} \bigotimes_{l=1}^n [ |0\rangle + e^{2\pi i j 2^{-l}} |1\rangle ] \quad (1.16)$$

$$= \frac{(|0\rangle + e^{2\pi i 0.j_n} |1\rangle) (|0\rangle + e^{2\pi i 0.j_{n-1}j_n} |1\rangle) \dots (|0\rangle + e^{2\pi i 0.j_1j_2\dots j_n} |1\rangle)}{2^{n/2}} \quad (1.17)$$

즉,  $|j\rangle$ 의 각 qubit  $|j_i\rangle$ 에 대해 독립적으로 특정 gate  $U$ 를 적용하여  $|0\rangle + e^{2\pi i 0.j_i\dots j_n}$ 가 되도록 quantum circuit을 설계하면, 그 결과가 QFT에 해당한다는 사실을 알아냈다.

$$U |j_i\rangle \rightarrow |0\rangle + e^{2\pi i 0.j_i\dots j_n} |1\rangle$$

본격적으로 quantum circuit을 만들기 위해서 rotation operator  $R_k$ 를 다음과 같이 정의하자. 이렇게 정의한 operator는  $|0\rangle$ 에 대해서는 아무것도 수행하지 않지만,  $|1\rangle$ 에 대해서는 phase  $e^{2\pi i / 2^k}$ 를 적용한다.

$$R_k \triangleq \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^k} \end{pmatrix}$$

Fig. 1.1의 연산을 단계별로 따라가보자.

1. input state : 다음과 같은 input state로 시작한다. 이는 computational basis state 중 하나이다.

$$|\psi\rangle = |j_1, \dots, j_n\rangle$$

2. apply Hadamard gate on the first qubit : 첫 번째 qubit;  $|j_1\rangle$ 에  $H$ 를 적용하면, 다음을 얻는다. <sup>13</sup>

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|0\rangle + (-1)^{j_1} |1\rangle) |j_2, \dots, j_n\rangle = \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i 0.j_1} |1\rangle) |j_2, \dots, j_n\rangle$$

3. apply controlled- $R_2$  gate with the first qubit as the target and the second qubit as control.

$$\begin{aligned} |\psi\rangle &= \begin{cases} \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i 0.j_1} |1\rangle) |j_2, \dots, j_n\rangle & \text{if } j_2 = 0, \\ \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i 0.01} e^{2\pi i 0.j_1} |1\rangle) |j_2, \dots, j_n\rangle & \text{if } j_2 = 1 \end{cases} \\ &= \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i 0.j_1 j_2} |1\rangle) |j_2, \dots, j_n\rangle. \end{aligned}$$

<sup>13</sup> 때,  $e^{i\pi} = e^{2i\pi \frac{1}{2}} = -1$  그리고  $e^0 = e^{2i\pi \frac{0}{2}} = 1$ 라는 사실을 이용한다.

4. apply controlled- $R_k$  gate consequently : 3번의 과정을 control qubit을 하나씩 증가시키면서 반복한다. 이때, control qubit의 순서가  $i$ 번째라면,  $R_i$  gate를 적용해야한다.

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle) |j_2, \dots, j_n\rangle$$

5. apply controlled- $R_k$  gate consequently with other *target state* : 3-4번 과정을 다른 control qubit에 대해서 반복한다.

예를 들어, second qubit을 target qubit으로서 가정하면, 다음과 같은 state를 얻게된다.

$$|\psi\rangle = \frac{1}{\sqrt{2^2}} (|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle) (|0\rangle + e^{2\pi i 0.j_2 j_3 \cdots j_n} |1\rangle) |j_3, \dots, j_n\rangle$$

일반화하면,  $j$ 번째 qubit에 대해,  $k(j < k)$ 번째 qubit을 control qubit으로 controlled- $R_{k-j}$  gate를 차례대로 적용하는 과정을 반복한다.

6.  $n$ 번째 qubit까지 이 과정을 수행하면 최종적으로 다음 state를 얻게된다.

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} (|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n} |1\rangle) (|0\rangle + e^{2\pi i 0.j_2 j_3 \cdots j_n} |1\rangle) \cdots (|0\rangle + e^{2\pi i 0.j_n} |1\rangle)$$

7. 따라서 SWAP gate를 적용하여 qubit의 순서를 반대로 뒤집으면, 우리가 구하고자하는 QFT의 결과 (see Eq. (1.17))를 얻을 수 있다.  $\square$

따라서 우리는 1번의 QFT circuit을 수행하는 것으로  $n$ 개의 data에 대한 QFT결과를 동시에 얻을 수 있다. (단, 관측하면 확률에 따라 하나의 결과만 얻게 된다.)

### 1.4.2 Performance

그렇다면, QFT가 가지는 gate complexity가 얼마인지 분석해보자. 알고리즘에 따라 첫 번째 qubit에 대해서 1개의  $H$ -gate, 그리고  $(n-1)$ 개의 controlled rotation gate를 필요로한다. 두 번째 qubit에 대해서는 1개의  $H$ -gate, 그리고  $(n-2)$ 개의 controlled rotation gate를 필요로한다. 따라서  $n$ 개의 qubit에 대해 모두 필요한 gate의 개수는 다음과 같다.<sup>14</sup>

$$n + n - 1 + \cdots + 1 = \frac{n(n+1)}{2} = \Theta(n^2)$$

### Some Remarks

- QFT는 exponential speed-up을 달성하는 것처럼 보이지만, 실제로는 값을 관측하게 되면 하나의 테이터에 대한 결과만 얻을 수 있다.
- 또한, input state  $|j\rangle$ 를 준비하는 과정도 효과적이지 못하다.
- 따라서 QFT를 활용하여 알고리즘을 설계하는 것은 쉽지 않다.

## 1.5 Phase estimation

### 1.5.1 Phase estimation

QFT를 활용하는 중요한 알고리즘 중 하나가 바로 *phase estimation*이다. phase estimation이 해결하고자 하는 문제에 대해서 먼저 소개한다. 다음의 관계를 만족하는 unitary operator  $U$ 에 대해, 우리가 알고 있는 것은 오직 eigenvector인  $|u\rangle$ 이고 eigenvalue는 알지 못한다고 하자.

$$U|u\rangle = e^{2\pi i \varphi}|u\rangle$$

Phase estimation의 목적은 eigenvalue에 대한 phase  $\varphi \in [0, 1]$ 를 구하는 것이다.

Eigenvector를 알고있을 때, eigenvalue를 구하는 문제에는 다양한 활용이 존재한다. 예를 들어, 주어진 Hamiltonian에 대해 ground state의 energy를 추정하는 상황을 가정하자. k-local Hamiltonian  $\hat{H}$ 와 ground state  $|\psi_0\rangle$ 에 대해 다음이 성립한다.

$$\hat{H}|\psi_0\rangle = E_0|\psi_0\rangle$$

따라서, 우리가  $E_0$ 를 estimate하기를 원한다는 것은  $\hat{H}$ 에 대한 eigenvector  $|\psi_0\rangle$ 이 알려져있을 때 eigenvalue를 구하는 것을 의미하며, 이는 phase estimation으로 해결할 수 있는 문제이다.<sup>15</sup>

<sup>14</sup>qubit의 순서를 바꾸기 위해 필요한 SWAP gate에 대한 gate complexity는  $O(n)$ 이므로 무시할 수 있다.

<sup>15</sup> $\hat{H}$ 를 unitary operator  $\hat{U} = e^{-i\hat{H}t}$ 로 구성하면,  $\hat{U}$ 에서 eigenvalue는  $e^{-iE_0 t}$ 가 된다. (with matrix exponential)

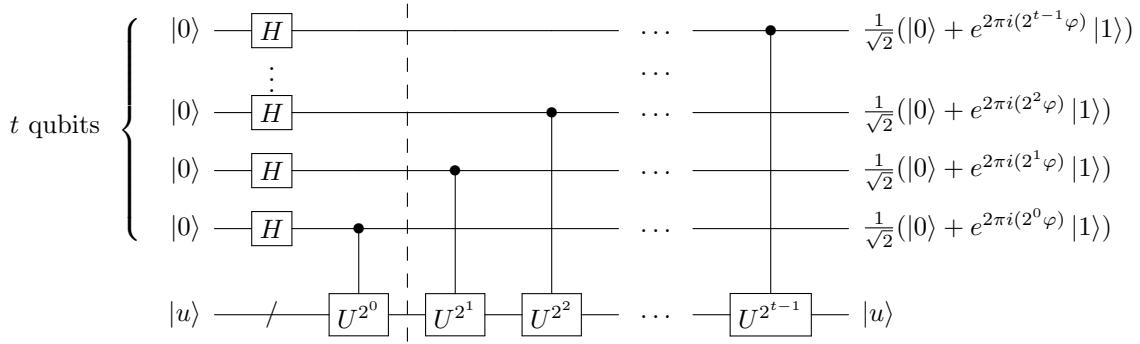


Figure 1.2: QPE circuit (first step)

Phase estimation을 설계하기 위해, 우리는 먼저 다음과 같은 oracle들을 가지고 있다고 가정한다.

- eigenvector  $|u\rangle$ 를 준비시킬 수 있는 oracle
- controlled- $U^{2^j}$  operation을 수행할 수 있는 oracle

Phase estimation algorithm은 크게 2가지 step으로 구성되는데, 첫 번째 step은 Fig. 1.2와 같이 구성되어 controlled-U operation을 차례대로 수행하는 과정이다. 두 번째 step은 첫 번째 register에 대해 inverse QFT를 수행하는 것이다. 각 step별로 어떻게 연산이 이루어지는지 따라가보자.

먼저 Fig. 1.2의 연산은 다음 단계를 따른다.

1. input state : phase estimation이 사용하는 2개의 register들을 각각 다음과 같이 초기화한다.<sup>16</sup>

$$|\psi\rangle = |0^{\otimes t}\rangle |u\rangle$$

2. apply  $H$ -gate : 첫 번째 register에 대해  $H$ -gate를 적용하면 다음과 같은 state를 얻게된다.

$$|\psi\rangle = \frac{1}{\sqrt{2^t}}(|0\rangle + |1\rangle)^{\otimes t} |u\rangle$$

3. apply controlled-U operation : 두 번째 register를 target으로 하고, 첫 번째 register의 각 qubit을 control로 하여 controlled-U operation을 가하는 과정을 control qubit을 바꿔가면서 수행한다.

예를 들어,  $t$  번째 qubit을 control qubit으로 하여 controlled- $U^{2^0}$ 를 가하면, 다음의 상태를 얻는다.<sup>17</sup>

$$\begin{aligned} |\psi\rangle &= \frac{1}{\sqrt{2^t}}(|0\rangle + |1\rangle)^{\otimes t-1}(|0\rangle |u\rangle + |1\rangle e^{2\pi i(2^0\varphi)} |u\rangle) \\ &= \frac{1}{\sqrt{2^t}}(|0\rangle + |1\rangle)^{\otimes t-1}(|0\rangle + e^{2\pi i(2^0\varphi)} |1\rangle) |u\rangle \end{aligned}$$

일반화하여,  $j$  번째 qubit을 target qubit으로 하여 두 번째 레지스터에 controlled- $U^{t-j}$  gate를 가하는 과정을  $t$  개의 qubit에 대해서 모두 수행하면 얻게되는 최종 state는 다음과 같다.

$$\frac{1}{2^{t/2}} \left( |0\rangle + e^{2\pi i 2^{t-1}\varphi} |1\rangle \right) \left( |0\rangle + e^{2\pi i 2^{t-2}\varphi} |1\rangle \right) \dots \left( |0\rangle + e^{2\pi i 2^0\varphi} |1\rangle \right) = \frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi i \varphi k} |k\rangle. \quad (1.18)$$

이렇게 얻은 state는 QFT를  $|N\psi\rangle$ 에 대해서 적용한 result state와 동일하다. (put  $j = N\varphi$ ,  $N = 2^t$ ) 따라서,  $\varphi$ 의 값을 얻기 위하여 Eq. (1.18)에 inverse QFT를 수행하여  $N\varphi$ 를 구할 수 있다.

앞에서 우리는  $\varphi \in [0, 1]$ 이라고 가정하였으므로,  $\varphi = 0.\varphi_1\varphi_2 \dots \varphi_t = \sum_{k=1}^t \varphi_k / 2^k$ 이라는 binary representation으로 표현할 수 있다. 따라서 inverse QFT 결과로 얻는  $N\varphi$ 는 다음과 같다.

$$N\varphi = 2^t \sum_{k=1}^t \varphi_k / 2^k = \sum_{k=1}^t \varphi_k 2^{t-k} = \varphi_1 \varphi_2 \dots \varphi_t$$

<sup>16</sup> 첫 번째 register의 qubit 개수  $t$ 는 QPE의 accuracy, success probability를 결정한다.

<sup>17</sup> target qubit은  $|u\rangle$ 이지만,  $U^{2^0}$ s는 phase만 변화시키기 때문에, control qubit에 대해 phase term을 옮겨서 표현할 수 있다.

즉, 최종적으로 QPE를 진행하면  $\varphi$ 의 binary representation에서 소수점 아래 자릿수를 값으로 갖는  $t$ 개의 qubit을 얻게되며, 각 qubit의 값은 0 또는 1이므로 computational basis에서 측정하게되면  $\varphi$ 의 근사값을 얻을 수 있다.  $\square$

## Lecture 12

### 1.5.2 Performance

28 Oct. 10:30

앞에서 우리는 phase  $\varphi$ 가  $t$ 개의 소수점 아랫수로 표현될 수 있는 값이라는 가정을 했었다. 그러나,  $\varphi = 0.\varphi_1\varphi_2 \dots \varphi_t\varphi_{t+1} \dots$ 처럼  $t+1$ 개 이상의 소수점 아랫자리수를 가지게 되면, QPE를 사용하여 얻은 결과  $\tilde{\varphi}$ 는 실제값의 근사치가 된다. 즉, error가 발생하게 된다.

error를 분석하기 위해서,  $t$ 개의 qubit로 나타낼 수 있는 이진수 표현중에서 가장  $\varphi$ 와 가까운  $\varphi$ 보다 크지 않은 수를 만드는 정수를  $b$ 라고 하자. ( $b/2^t = 0.b_1 \dots b_2$ )

$$b = \arg \min_{0 \leq b \leq 2^t - 1} \varphi - b/2^t, \quad (b/2^t \leq \varphi) \quad (1.19)$$

우리는 이제 QPE의 측정 결과가  $b$ 와 가까우며, 따라서  $\varphi$ 를 높은 확률로 추정할 수 있음을 보이고자한다. Eq. (1.18)에 IQFT를 적용한 결과는 다음과 같다.

$$\frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi i \varphi k} |k\rangle \xrightarrow{\text{IQFT}} \frac{1}{2^t} \sum_{k,l=0}^{2^t-1} e^{2\pi i \varphi k} \underbrace{e^{-2\pi i l k / 2^t}}_{|k\rangle} |l\rangle = \frac{1}{2^t} \sum_{k,l=0}^{2^t-1} \left( e^{2\pi i (\varphi - l/2^t)} \right)^k |l\rangle.$$

만약  $\varphi = l/2^t$ 라면,  $e^{2\pi i (\varphi - l/2^t)} = 1$ 이 2 $t$ 개의 항에 대해서 더해지기 때문에,  $|l\rangle$ 의 amplitude가 1이되어서 100% 확률로  $|l\rangle = |N\varphi\rangle$ 를 관측한다. 그러나  $\varphi$ 가  $l/2^t$ 와 유사하다면, 다른 항들도 관측될 확률을 가질 수 있고 이것이 바로 error를 발생시키는 원인이 된다.

**Definition 1.5.1 (Inverse Quantum Fourier transform).** Inverse QFT는 fourier basis  $|k\rangle$ 에 대하여, 다음 변환을 수행한다.<sup>a</sup>

$$|k\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-2\pi i j k / N} |j\rangle$$

<sup>a</sup>phase를 소거하기 위해 phase가 음수이다.

다른 항이 관측될 확률을 분석하기 위해서,  $\alpha_l$ 을  $|(b+l) \bmod 2^t\rangle$  state에 대한 amplitude라고 하자. 등비급수 공식을 사용하여 나타내면 다음과 같다. ( $\delta = \varphi - b/2^t$ ) (See Eq. (1.19))

$$\alpha_l \triangleq \frac{1}{2^t} \sum_{k=0}^{2^t-1} \left( e^{2\pi i (\varphi - (b+l)/2^t)} \right)^k = \frac{1}{2^t} \left( \frac{1 - e^{2\pi i (2^t \varphi - (b+l))}}{1 - e^{2\pi i (\varphi - (b+l)/2^t)}} \right) = \frac{1}{2^t} \left( \frac{1 - e^{2\pi i (2^t \delta - l)}}{1 - e^{2\pi i (\delta - l/2^t)}} \right)$$

이때,  $a_l$ 은  $b + l \bmod 2$  연산에 의해 결정되기 때문에, 서로 다른  $l$ 에 대해서 동일한  $\alpha_l$ 을 정의할 수 없도록  $l$ 은  $-2^{t-1} < l \leq 2^{t-1}$  범위로 제한된다.

이제 QPE의 측정 결과가  $m$ 이라고 가정하자. 우리는  $m$ 과  $b$ 의 차이  $|m - b|$ 가  $\beta$ 보다 클 확률을 계산하고자한다. 이는 우리가 허용할 수 있는 error  $\beta$ 보다 더 큰 error가 발생할 failure probability를 의미한다.

$$\Pr(|m - b| > \beta) = \sum_{\beta \leq |l|} |\alpha_l|^2 = \sum_{-2^{t-1} < l \leq -(\beta+1)} |\alpha_l|^2 + \sum_{\beta+1 \leq l \leq 2^{t-1}} |\alpha_l|^2 \quad (1.20)$$

이때,  $|1 - e^{i\theta}|$ 에 대한 다음의 2가지 bound를 적용하여 확률을 계산할 수 있다.

- $|1 - e^{i\theta}| \geq \frac{2|\theta|}{\pi}$ , for  $(-\pi \leq \theta \leq \pi)$
- $|1 - e^{i\theta}| \leq 2$ , for any  $\theta \in \mathbb{R}$ .

$l$ 의 범위에 대한 제한 덕분에,  $-\pi \leq 2\pi(\delta - l/2^t) \leq \pi$  조건 만족하여  $|\alpha_l|$ 의 upper bound를 구할 수 있다.

$$|\alpha_l| = \left| \frac{1}{2^t} \left( \frac{1 - e^{2\pi i (2^t \delta - l)}}{1 - e^{2\pi i (\delta - l/2^t)}} \right) \right| \leq \frac{2}{2^t |1 - e^{2\pi i (\delta - l/2^t)}|} \leq \frac{1}{2^{t+1} (\delta - l/2^t)} = \frac{1}{2} \left( \frac{1}{2^t \delta - l} \right) \quad (1.21)$$

따라서 Eq. (1.20)에 Eq. (1.21)을 대입하면, 다음의 upper bound를 얻는다.

$$\Pr(|m - b| > \beta) \leq \frac{1}{4} \left[ \sum_{-2^{t-1} < l \leq -(\beta+1)} \frac{1}{(l - 2^t \delta)^2} + \sum_{\beta+1 \leq l \leq 2^{t-1}} \frac{1}{(l - 2^t \delta)^2} \right] \quad (1.22)$$

$0 \leq 2^t \delta \leq 1$ 을 이용하여 간략히 표현하면, 다음을 얻는다.

- Eq. (1.23) : upper bound를 구하기 위해서 각 항의 값이 최대가 되도록 하는  $2^t \delta$ 를 대입한다.<sup>18</sup>
- Eq. (1.24) : 첫 번째 항을 보면  $l^2$ 에 대해서 연산하고 있기 때문에  $l$ 의 범위를 음수에서 양수로 바꾸어도 된다.
- Eq. (1.25) : 두 항의 하나로 나타낸 뒤<sup>19</sup>, 합을 적분으로 근사한다.

따라서 정리하자면, QPE의 failure probability는  $\beta$ 에 대해  $\frac{1}{2(\beta-1)}$ 로 bound 된다.

$$\Pr(|m - b| > \beta) \leq \frac{1}{4} \left[ \sum_{-2^{t-1} < l \leq -(\beta+1)} \frac{1}{l^2} + \sum_{\beta+1 \leq l \leq 2^{t-1}} \frac{1}{(l-1)^2} \right] \quad (1.23)$$

$$= \frac{1}{4} \left[ \sum_{\beta+1 \leq l < 2^{t-1}} \frac{1}{l^2} + \sum_{\beta+1 \leq l \leq 2^{t-1}} \frac{1}{(l-1)^2} \right] \quad (1.24)$$

$$\leq \frac{1}{2} \sum_{l=\beta}^{2^{t-1}-1} \frac{1}{l^2} \leq \frac{1}{2} \int_{\beta-1}^{2^{t-1}-1} \frac{1}{l^2} dl \leq \frac{1}{2(\beta-1)}. \quad (1.25)$$

만약 우리가  $\varphi$ 를 정확도  $2^{-n}$ 으로 추정하기를 원한다면,<sup>20</sup>  $\beta = 2^{t-n} - 1$ 으로 선택한다. 이때,  $t$ 는 QPE에 사용하는 qubit의 개수이다. ( $p = t - n$ ) 그럼 이  $\beta$ 에 대해, failure probability는  $1/2(2^{t-n} - 2) = 1/2(2^p - 2)$ 로 bound되기 때문에, QPE의 정확도는 적어도  $1 - 1/(2(2^p - 1))$  이상이다.

따라서,  $n$  bits로 표현된  $\varphi$ 를 success probability가  $1 - \delta$  이상이 되도록 QPE를 수행하기 위해서는, 첫 번째 레지스터의 개수  $t$ 를 다음과 같이 결정해야한다.

$$t = n + \left\lceil \log \left( 2 + \frac{1}{2\delta} \right) \right\rceil$$

또한, 앞에서 우리는 eigenvector의 state  $|u\rangle$ 로 준비할 수 있는 oracle을 가정했다. 그러나, 실제 상황에서 는 이것이 불가능할 수도 있다. 만약 우리가 다른 state  $|\psi\rangle$ 를  $|u\rangle$  대신에 사용했다고 하자.  $|\psi\rangle$ 는 eigenbasis에 대해 다음과 같은 linear combination으로 표현된다.

$$|\psi\rangle = \sum_u c_u |u\rangle$$

각 eigenstate의 eigenvalue가  $e^{2\pi i \varphi_u}$ 라면,  $|\psi\rangle$ 에 대해 phase estimation을 수행한 결과는 다음과 같이 각각의 eigenstate에 대해 phase estimation을 수행한 결과들의 superposition state가 될 것이다.

$$|0^{\otimes n}\rangle |\psi\rangle \xrightarrow{\text{QPE}} \sum_u c_u |\varphi_u\rangle |u\rangle,$$

따라서 이러한 output state를 측정하게 되면, probability  $|c_u|^2$ 에 따라서 랜덤하게 어떤 eigenstate  $|u\rangle$ 에 대한 phase  $|\varphi_u\rangle$ 를 얻게된다. 즉, 우리가 실제로 얻고자하는 eigenstate  $|u\rangle$ 에 대한 amplitude  $c_u$ 가 충분히 크다면, QPE를 여러번 반복하여 우리가 원하는 eigenvalue를 얻을 수 있다.

마지막으로, QPE의 resource requirement를 분석해보자.

- (space complexity) accuracy  $\epsilon$ 을 달성할 probability가  $1 - \delta$ 가 되도록 필요한 qubit의 개수 :

$$t = \log(1/\epsilon) + \left\lceil \log \left( 2 + \frac{1}{2\delta} \right) \right\rceil = O \left( \log \frac{1}{\delta\epsilon} \right).$$

<sup>18</sup> 첫 번째 항은  $|l\rangle$  음수이기 때문에  $2^t \delta$ 가 0이 되도록하고, 두 번째 항은  $|l\rangle$  양수이기 때문에  $2^t \delta$ 가 1이 되도록한다.

<sup>19</sup> using  $\frac{1}{l^2} + \frac{1}{(l-1)^2} \leq \frac{2}{l^2}$ .

<sup>20</sup> n-bit 이진수에서 n번째 자릿수 이하에서 발생하는 차이값

- (time complexity)

- (step 1) QPE를 실행하기 위해서는 controlled unitary를  $2^t$ 번 가해야하므로, gate complexity를 qubit의 개수  $t$ 에 대해서 표현하면  $O(1/(\delta\epsilon))$ 이 된다.
- (step 2)  $t$ 개의 qubit에 대해 IQFT를 가하기 때문에 IQFT의 complexity는  $\Theta(t^2)$ 이 된다.

따라서 QPE의 total time complexity는 다음과 같다.

$$O\left(\frac{1}{\delta\epsilon}\right) + \Theta\left(\left(\log \frac{1}{\delta\epsilon}\right)^2\right)$$

## Lecture 13

### 1.6 Applications of phase estimation

30 Oct. 10:30

#### 1.6.1 Ground state energy estimation

앞에서 phase estimation의 대표적인 예시로 사용했던 ground state energy estimation에 대해 조금 더 자세히 살펴보자.  $H$ 가 다음의 eigendecomposition을 가지는 Hamiltonian operator라고 하자.<sup>21</sup>

$$H |\psi_j\rangle = \lambda_j |\psi_j\rangle, \quad (\text{suppose } 0 < \lambda_0 < \lambda_1 \leq \dots \leq \lambda_{N-1} < \frac{1}{2})$$

우리가 ground state  $|\psi_0\rangle$ 의 정확한 state를 알고 있다면, QPE를 적용해서 높은 확률로 ground state energy를 추정할 수 있다. 반면, 다음과 같은 eigenstate의 approximation state를 이용한다고 하자.

$$|\psi\rangle = \sum_{k \in \{0, \dots, N-1\}} c_k |\psi_k\rangle$$

이 state를 관측했을 때, ground state를 얻을 확률은  $p_0 = |\langle\psi|\psi_0\rangle|^2$  이고,  $p_0$ 이 충분히 크다면 QPE를 여러번 반복하여 ground state energy를 추정할 수 있다. (Poly)

Ground state energy estimation은  $\epsilon$ 의 정확도로 ground state energy  $\lambda_0$ 를 추정하고자 한다. ( $\epsilon < \lambda_0$ ) 이 문제는 quantum many-body physics, quantum chemistry, optimization 등 많은 분야에서 중요한 문제이다. 일반적으로, QPE를 활용하기 위해서 Hamiltonian에 대한 unitary operation을 이용할 수 있다고 가정한다.

$$U = e^{2\pi i H}$$

이 unitary에 대해 ground state energy는 phase에 위치하게 되며, 따라서 QPE를 적용할 수 있다.

$$U |\psi_0\rangle = e^{2\pi i \lambda_0} |\psi_0\rangle.$$

#### 1.6.2 Order-finding algorithm

##### Prerequisite: Uncomputation

먼저, order-finding algorithm을 설명하기 전에 uncomputation technique에 대해 먼저 이야기해보자. Classical circuit  $f$ 에 대응되는 quantum circuit  $O_f$ 를 구성하는 과정을 *uncomputation*이라고 한다.<sup>22</sup>

$$O_f |x, b\rangle = |x, b \oplus f(x)\rangle$$

구체적으로 uncomputation 과정을 어떻게 수행하는지 단계별로 살펴보자.

1. classical circuit을 *reversible*하도록  $n$ 개의 ancilla qubit을 추가한다. ( $g(x)$ 는 임의의 garbage 값)

$$\tilde{O}_f |x\rangle |0^n\rangle |b\rangle \rightarrow |g(x)\rangle |f(x)\rangle |b\rangle$$

2.  $|b\rangle$ 와 ancilla qubit에 대해 controlled-X gate를 적용한다.

$$|g(x)\rangle C(X) |f(x)\rangle |b\rangle \rightarrow |g(x)\rangle |f(x)\rangle |b \oplus f(x)\rangle$$

3.  $O_f^\dagger$ 를 적용하고 ancilla qubit을 무시하면,  $O_f = \tilde{O}_f^\dagger C(X) \tilde{O}_f$ 는 우리가 원하는 연산을 수행한다.

$$\tilde{O}_f^{-\dagger} |g(x)\rangle |f(x)\rangle |b \oplus f(x)\rangle \rightarrow |x\rangle \underbrace{|0^n\rangle}_{\text{ancilla}} |b \oplus f(x)\rangle$$

<sup>21</sup>다음 가정은 non-degenerate case; 중복된 고윳값이 없는 상태를 다룬다는 것을 의미한다.

<sup>22</sup>Deutsch's algorithm에서 사용했던 oracle과 동일한 개념이다.

### Order-finding

이제 order-finding problem이 무엇인지 소개하려고 한다. order-finding problem은 다음과 같이 표현된다. <sup>23</sup>

- input: integer  $(x, N)$ ,  $(x < N, \gcd(x, N) = 1)$
- output: the order of  $x \bmod N$ ,  $r$ .

**Definition 1.6.1 (Order).** 다음을 만족하는 가장 작은 양의 정수  $r$ 를 *order of  $x$  modulo  $N$* 이라고 한다.

$$x^r \equiv 1 \pmod{N}$$

Classical algorithm으로 이 문제를 input의 크기  $L$ 에 대하여 polynomial time에 해결할 수 있는 방법은 현재까지 없다고 알려져 있다. 이때  $L$ 은  $N$ 과  $N$ 보다 작은 수  $x$ 를 표현하는데 필요한 bit의 개수를 의미하기 때문에  $L \triangleq \lceil \log N \rceil$ 이다.

Quantum algorithm으로 이 문제를 해결하기 위해서 우리는 *phase estimation*을 활용한다. Phase estimation을 적용하기 위해서는 unitary operator  $U$ 와 그에 대응되는 eigenvector  $|u\rangle$ , eigenvalue  $\varphi$ 의 형태로 문제를 정의해야 한다. 따라서 이를 위해 computational basis  $|y\rangle \in \{|0\rangle, \dots, |2^L - 1\rangle\}$ 에 대하여 다음과 같이 동작하는 unitary를 정의하자.

$$U|y\rangle \triangleq \begin{cases} |xy \bmod N\rangle & 0 \leq y < N, \\ |y\rangle & N \leq y < 2^L \end{cases}$$

이 unitary는  $N$ 보다 큰 basis들의 subspace에 대해서는 identity matrix처럼 행동하지만, 그보다 작은 subspace에 대해서는 주어진 input  $x$ 에 대해  $xy \bmod N$  연산 결과가 되도록 만든다. 이렇게 정의한 unitary가 정말로 valid한지를 확인하기 위해 subspace에 대해 작용하는 unitary  $|U\rangle$ 가 valid한지를 확인해보자.

$$\langle y|\tilde{U}^\dagger \tilde{U}|y'\rangle = \langle xy \bmod N|xy' \bmod N\rangle = \langle y|y'\rangle$$

연산 전후 norm을 보존하기 때문에 우리가 정의한 unitary는 valid하다. <sup>23</sup> □

이제 다음과 같이 정의한 state가  $U$ 의 eigenvector임을 보인다면, phase estimation을 통해 어떤 임의의 eigenvector에 대해서도 항상 eigenvalue  $r$ 을 얻을 수 있다는 사실을 입증할 수 있다.

$$|u_s\rangle \triangleq \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i s k}{r}\right) |x^k \bmod N\rangle$$

어떤 정수  $0 \leq s \leq r-1$ 에 대해서도  $|u_s\rangle$ 가 eigenvector임을 보이기 위해 definition을 이용하자.

- Eq. (1.26) :  $x^k \bmod N$ 은 modulo 연산 때문에 항상 그 결과가  $\{0, \dots, N-1\}$  범위에 존재하여 unitary operation이  $|x^{k+1} \bmod N\rangle$ 으로 state를 변화시킨다.
- Eq. (1.27) :  $k = k+1$ 로 재정의하면, sigma의 적용 범위가  $[0, r-1]$ 에서  $[1, r]$ 로 변화하지만, modulo  $N$  연산 때문에 연산 범위를 동일하게 생각할 수 있다.
- Eq. (1.29) : 따라서  $U$ 를  $|u_s\rangle$ 에 적용한 결과는 단순히  $|u_s\rangle$ 에 phase  $e^{2\pi i s / r}$ 을 가한 결과와 동일하므로  $|u_s\rangle$ 는  $U$ 의 eigenvector이다. □

$$U|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i s k}{r}\right) |x^{k+1} \bmod N\rangle \quad (1.26)$$

$$= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i s(k-1)}{r}\right) |x^k \bmod N\rangle \quad (1.27)$$

$$= \frac{1}{\sqrt{r}} \exp\left(\frac{2\pi i s}{r}\right) \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i s k}{r}\right) |x^k \bmod N\rangle \quad (1.28)$$

$$= \exp\left(\frac{2\pi i s}{r}\right) |u_s\rangle. \quad (1.29)$$

따라서, 어떤  $s$ 에 대해서도 phase estimation을 적용하면  $s/r$ 을 얻을 수 있다.

<sup>23</sup> 또는  $\gcd(x, N) = 1$ 이라면, modulo  $N$ 에서  $x$ 의 역원은 항상 존재한다는 성질을 활용해서 증명할 수도 있다.

단, phase estimation을 수행하기 위해서는 아직 2가지의 요소가 더 필요하다.

- 다음 연산을 구현하는 방법 ( $t$ 개의 qubit를 이용할 때)

$$|z\rangle U^{z_t 2^{t-1}} \cdots U^{z_1 2^0} |y\rangle = |z\rangle |x^z y \mod N\rangle.$$

- $U$ 의 eigenvector  $|u_s\rangle$ 를 <sup>24</sup> 없이 준비하는 방법.

(1) 의 구현은 앞에서 설명한 *uncomputation* 기법을 활용하여 생각할 수 있다. 다음 gate는  $f(z) = x^z \mod N$ 을 수행하는 classical function을 quantum circuit으로 구현한 것이다.<sup>25</sup>

$$|z\rangle |y\rangle |0\rangle \xrightarrow{\tilde{O}_f} |z\rangle |y\rangle |x^z \mod N\rangle \quad (1.30)$$

만약  $f(z) = x^z \mod N$ 을 계산하는 classical circuit이 있다면, uncomputation을 사용하여 Eq. (1.30)과 같은 연산을 quantum computer에서도 수행할 수 있다. *modular exponentiation*<sup>26</sup>에 의하여,  $f(z) = x^z \mod N$ 은 다음과 같이 계산할 수 있다. ( $z = z_t z_{t-1} \cdots z_1$ )

$$x^z \mod N = (x^{z_t 2^{t-1}} \mod N)(x^{z_{t-1} 2^{t-2}} \mod N) \cdots (x^{z_1 2^0} \mod N)$$

따라서 이 아이디어를 이용하여 uncomputation 과정을 적용하면 다음의 circuit을 만들 수 있다.

1. *modular exponentiation*을 사용하면, Eq. (1.30)을 다음과 같이 계산할 수 있다.

$$|z\rangle |y\rangle |0\rangle \xrightarrow{\tilde{O}_f} |z\rangle |y\rangle |(x^{z_t 2^{t-1}} \mod N)(x^{z_{t-1} 2^{t-2}} \mod N) \cdots (x^{z_1 2^0} \mod N)\rangle$$

2.  $|q_3\rangle$ 과 ancilla qubit  $|q_2\rangle$ 을 곱한다. ( $|q_1 q_2 q_3\rangle$ )

$$|z\rangle |y\rangle |x^z \mod N\rangle \rightarrow |z\rangle |x^z y \mod N\rangle |x^z \mod N\rangle$$

3. inverse operation  $\tilde{O}_f^\dagger$ 를 취한다.

$$|z\rangle |x^z y \mod N\rangle |x^z \mod N\rangle \xrightarrow{\tilde{O}_f^\dagger} |z\rangle |x^z y \mod N\rangle |0\rangle$$

$|q_3\rangle$ 을 무시하면,  $O_f = \tilde{O}_f^\dagger U \tilde{O}_f$ 는 우리가 원하는 연산을 수행한다. 만약  $|x^z y \mod N\rangle$ 을 나타내기 위해서 사용하는 qubit의 개수가  $L$ 개라면, 이 연산의 time complexity는  $O(L^3)$ 이다.

- QPE에 사용하는 qubit의 개수  $t$ 는 다음과 같이 설정된다.<sup>27</sup>

$$t = 2L + 1 + \lceil \log(2 + 1/(2\delta)) \rceil = O(L)$$

- $0 \leq j < t$  범위의 모든  $j$ 에 대해 이를 계산해야하므로,  $O(t)$ 번 연산을 수행한다.

- 매 단계마다,  $x^z y \mod N$  값을 갱신하기 위해서 이전 단계까지의 누적 계산 결과  $x^{z_1 2^0} \cdots x^{z_{i-1} 2^{i-1}}$ 에 이번 단계에 얻은 결과인  $x^{z_i 2^i}$ 를 곱한 뒤  $\mod N$ 을 취한다.

–  $L$  bit 수의 곱셈 연산의 complexity:  $O(L^2)$

–  $\mod N$  연산의 complexity:  $O(L^2)$

반면, (2) 를 구현하는 것은 조금 더 까다롭다. 이 문제를 해결하기 위해서  $|u_s\rangle$ 들의 superposition state를 대신 입력으로 제공하는 방법을 사용할 수 있다. 만약  $r$ 개의 qubit를 이용하여  $|u_s\rangle$ 를 준비한다고 가정하면, superposition state는 다음과 같이 정리할 수 있다.<sup>28</sup>

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = \frac{1}{r} \sum_{s,k=0}^{r-1} \exp\left[-\frac{2\pi i s k}{r}\right] |x^k \mod N\rangle = \sum_{k=0}^{r-1} \delta_{k0} |x^k \mod N\rangle = |1\rangle.$$

따라서  $|1\rangle$ 을 입력으로 사용하면, 확률에 따라서 임의의  $s$ 에 대해  $\varphi \approx s/r$ 를 적어도  $(1 - \delta)/r$  확률로  $2L + 1$  bit 정확도를 달성할 수 있다.

<sup>24</sup>eigenvector의 정의에 이미 우리가 얻고자하는 값인  $r$ 이 들어있다

<sup>25</sup> $0 \oplus f(z)$

<sup>26</sup> $ab \mod N = (a \mod N)(b \mod N)$

<sup>27</sup>우리가 조절할 수 있는 parameter인  $L$ 에 의해, QPE의 accuracy가 조절된다.

<sup>28</sup>complex exponential의 orthogonality에 의하여,  $\sum_{s=0}^{r-1} e^{(2\pi i s/r)(k-0)} = \begin{cases} 0, & \text{for } k \neq 0 \\ r, & \text{for } k = 0 \end{cases}$

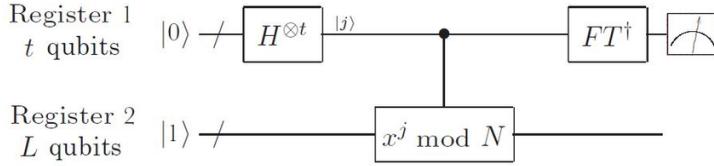


Figure 1.3: Order-finding algorithm [Dha15]

### The continued fraction expansion

우리가 앞의 알고리즘을 통해서 얻은 결과는  $s/r$ 이다. 따라서 이 결과로부터  $r$ 의 값을 얻어내는 추가적인 후처리가 필요하다. 우리가 얻은  $\varphi \approx s/r$ 은 임의의  $2L+1$  bit 소수이므로 임의의 실수이지만, 실제로 우리가 얻고자 하는 값은  $s/r$ 은 분명히 유리수이다. 이 특징을 활용하기 위해 continued fraction이라는 개념을 도입한다. continued fraction은 주어진 실수를 연분수 형태로 표현한 것이다.<sup>29</sup>

$$[a_0, \dots, a_M] \triangleq a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \dots}}}$$

주어진 convergent를 연분수 형태로 변환하는 알고리즘을 *continued fraction algorithm*이라고 한다.

**Theorem 1.6.1.**  $s/r$ 이 유리수이고,  $\varphi$ 가 QPE로 얻은 근사값일 때 두 값의 차이가 다음과 같은 upper bound를 만족한다면, continued fraction algorithm을 적용하여 얻은 결과가 바로  $s/r$ 이다.

$$\left| \frac{s}{r} - \varphi \right| \leq \frac{1}{2r^2}$$

Order-finding problem에서는  $(1-\delta)/r$ 의 높은 확률로  $2L+1$  bit 정확도를 달성하기에, 다음을 만족한다.

$$\left| \frac{s}{r} - \varphi \right| \leq \frac{1}{2^{2L+1}} \leq \frac{1}{2r^2}$$

따라서 continued fraction algorithm을 적용할 수 있으므로,  $O(L^3)$ 의 time complexity로  $s/r$ 을 구할 수 있고  $\gcd(s, r) = 1$ <sup>30</sup>이라면  $r$ 을 구할 수 있다.

### Performance

Order-finding algorithm의 정확도를 분석하기 위해서, 이 알고리즘이 실패하는 경우를 분석해보자.

- QPE가 실패하는 경우 : 아주 작은 확률  $\epsilon$ 로 실패할 수 있다. 이 경우 qubit의 개수를 늘려서 QPE의 정확도를 높이거나 QPE를 여러번 반복하여 각 outcome의 확률을 비교하여 해결할 수 있다.
- $\gcd(s, r) \neq 1$ 인  $s$ 를 얻는 경우 :  $0 \leq s < r - 1$  범위에서 두 수가 서로소일 확률은 매우 높기 때문에, 알고리즘을  $2 \log N$ 번 반복하면 이 문제를 해결할 수 있다.

마지막으로 Order-finding algorithm의 circuit이 필요로하는 자원을 분석해보자.

- $O(L)$  : Register 1에  $H$  변환을 가하는데 필요한 circuit의 개수
- $O(L^2)$  : IQFT를 적용하는데 필요한 circuit의 개수
- $O(L^3)$  : Controlled unitary; modular exponentiation에 필요한 circuit의 개수
- $O(L^3)$  : continued fraction algorithm이 사용하는 time complexity
- $O(\log N)$  : 오류가 발생했을 때, 다시 알고리즘을 실행하는 횟수

<sup>29</sup>이때  $[a_0, \dots, a_M]$ 을  $M$ th convergent라고 한다.

<sup>30</sup>만약 두 수 사이에 공약수가 존재하면, continued fraction algorithm이 반환하는 값이 통분한 값이 된다

### 1.6.3 Shor's algorithm: factoring

이제 마지막으로 그 유명한 Shor's algorithm에 소개하고자 한다. Classical computer에서 polynomial time에 해결하는 효과적인 알고리즘이 아직 존재하지 않는 *factoring*을 quantum computer를 사용하여 효율적으로 수행할 수 있다는 사실을 보여주는 중요한 알고리즘이다.

Prime factorization problem은 다음과 같이 정의된다.

- **input:** natural number  $N$
- **output:** list of prime numbers  $\{p_1, p_2, \dots, p_k\}$  such that  $N = p_1^{e_1} \cdot p_2^{e_2} \cdots \cdot p_k^{e_k}$

이를 해결하는 Shor's algorithm은 다음과 같다.

1. **if**  $N \mid 2$ , **return** 2
2. **if**  $N \mid p$ 에 대해서  $N = p^k$ 라면, **return**  $p$ <sup>31</sup>
3.  $1 \leq x \leq N - 1$  범위에서 랜덤하게  $x$ 를 선택한다.

- (a) **if**  $\text{gcd}(N, x) > 1$ <sup>32</sup>, **return**  $\text{gcd}(N, x)$
- (b) **else**, **order-finding algorithm**을 사용하여 order  $r$ 을 찾는다.

4. **if**  $r \mid \phi(N)$ 라면, **goto** step 3.

5. **else**,

- (a) 다음을 계산한다.

$$x^r - 1 \equiv (x^{r/2} - 1)(x^{r/2} + 1) \equiv 0 \pmod{N}$$

- (b) 계산 결과로부터 얻은 2개의 항에 대해, 다음의 조건들을 확인한다.

- i. **if**  $N \mid x^{r/2} - 1$ , **error**<sup>33</sup>
- ii. **elif**  $N \mid x^{r/2} + 1$ , **goto** step 3.
- iii. **neither**,  $N \mid (x^{r/2} - 1)(x^{r/2} + 1)$ 을 동시에 나눌 수 없으므로 각각의 항이  $N$ 의 인수를 포함하고 있다는 의미이다.  

$$\boxed{\text{return } [\text{gcd}(N, x^{r/2} + 1), \text{gcd}(N, x^{r/2} - 1)]}$$

Shor's algorithm이 성공하기 위해서는 order-finding algorithm으로 얻은  $r$ 이 짹수여야하며, **neither** case에 포함되어야 한다. 정수론의 개념들을 도입하면, 성공확률은  $1/2$  이상임을 증명할 수 있으며, 여기에서 증명을 생략한다.

## Lecture 14

### 1.7 Applications of the QFT

4 Nov. 10:30

이번 section에서는 QFT를 활용하는 문제들에 대해서 소개하고자 한다. QFT를 활용하여 효과적으로 해결할 수 있는 문제들은 모두 *Hidden subgroup problem*으로 일반화할 수 있다.

#### 1.7.1 Period-finding

Hidden subgroup problem을 소개하기 전에, 먼저 hidden subgroup problem의 특수한 경우들에 대해 이야기하고자 한다. 첫 번째로 소개하는 문제는 *period-finding problem*이다.

- **input:** period function  $f$  such that  $f(x) = f(x + r)$  for some unknown integer  $r$ , ( $0 < r < 2^L$ )
- **output:** least integer  $r > 0$  such that  $f(x) = f(x + r)$

주어진 period function은 **uncomputation**을 사용하여 다음과 같은 unitary operator로 표현할 수 있다.

$$U_f |x\rangle |y\rangle \rightarrow |x\rangle |y \oplus f(x)\rangle$$

<sup>31</sup> 이를 판단하는 효과적인 classical algorithm이 존재

<sup>32</sup> using Euclidean algorithm

<sup>33</sup> order  $r \mid \phi(N)$  이를 만족하는 가장 작은 수이기 때문에 이런 경우는 일어나지 않는다.

이  $U_f$ 가 주어지면, 다음의 과정을 따라서 period  $r$ 을 찾을 수 있다.

1. input state :  $|0\rangle|0\rangle$ 으로 state를 초기화한다.<sup>34</sup>

$$|\psi\rangle = |0\rangle|0\rangle$$

2. apply  $H$  gate on 1st register

$$|\psi\rangle = \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|0\rangle$$

3. apply  $U_f$  :  $|y\rangle = |0\rangle$ 으로 2nd register에  $f(x)$ 가 저장된다.

$$|\psi\rangle = \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|f(x)\rangle$$

4. apply QFT on 2nd register

QFT를 사용하여 함수  $f(x)$ 를 표현하면, 다음과 같다. (See Definition. 1.4.2)

$$f(x) \rightarrow \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} e^{2\pi i l x / r} |\hat{f}(l)\rangle, \quad |\hat{f}(l)\rangle \triangleq \frac{1}{\sqrt{r}} \sum_{x=0}^{r-1} e^{-2\pi i l x / r} |f(x)\rangle$$

따라서 이를 이용하면, state를 다음과 같이 표현할 수 있다.

$$|\psi\rangle = \frac{1}{\sqrt{r}2^t} \sum_{l=0}^{r-1} \sum_{x=0}^{2^t-1} e^{2\pi i l x / r} |x\rangle|\hat{f}(l)\rangle$$

5. apply IQFT on 1st register<sup>35</sup>

$$|\psi\rangle = \frac{1}{\sqrt{r}2^t} \sum_{l=0}^{r-1} \sum_{x,y=0}^{2^t-1} e^{2\pi i l x / r} e^{-2\pi i x y / 2^t} |y\rangle|\hat{f}(l)\rangle = \frac{1}{\sqrt{r}2^t} \sum_{l=0}^{r-1} \sum_{x,y=0}^{2^t-1} e^{(2\pi i x)(l/r - y/2^t)} |y\rangle|\hat{f}(l)\rangle \quad (1.31)$$

final state의 coefficient를 분석하기 위해  $r$ 이  $2^t$ 를 나누는가에 대한 cases를 분석할 수 있다.

- $r \nmid 2^t$  : 일반적으로는  $r$ 이  $2^t$ 를 나누지 않는 경우가 더 많다.<sup>36</sup> 그러나 이 경우에도 최종 outcome이 높은 확률로  $nl$ 이 된다.
- $r \mid 2^t$  :  $nr = 2^t$ 가 되게 만드는 positive integer가 존재한다. 따라서 이를 이용하면 state가 다음과 같이 단순화 된다. (Eq. (1.31)에  $r = 2^t/n$  대입)<sup>37</sup>

$$\frac{1}{2^t} \sum_{x,y=0}^{2^t-1} e^{(2\pi i x / 2^t)(nl-y)} |y\rangle = \sum_{y=0}^{2^t-1} \delta_{y,nl} |y\rangle = |nl\rangle = \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} |nl\rangle|\hat{f}(l)\rangle$$

따라서 1st register를 측정한 결과를  $2^t$ 로 나누면  $nl/2^t = l/r$ 이므로  $r$ 을 구할 수 있다.  $\square$

### 1.7.2 Discrete logarithm

이번에 소개하려는 문제는 *Discrete logarithm problem*; 이산 로그 문제이다. 이 문제는 다음과 같이 정의된다.

- **input:**  $x \in G$ , 이때  $G = \langle g \rangle$ 는  $g$ 로부터 만들어지는 cyclic group<sup>38</sup><sup>39</sup>이다.
- **output:** minimum  $\alpha$  such that  $g^\alpha = x$  (or equivalently,  $\alpha = \log_g x$ )

<sup>34</sup>(1st register) 합수값을 저장하기 위해  $t = O(L + \log(1/\epsilon))$  개의 qubit, (2nd register) ancilla qubit

<sup>35</sup>2nd register가  $x, l$ 간의 QFT를 이용했다면, 1st register는  $l$ 이 아닌 다른 변수  $y$ 에 대해 IQFT를 수행하게 된다.

<sup>36</sup> $2^t$ 의 약수; 2의 거듭제곱이 아닌 이상 나눌 수 없음

<sup>37</sup> $nl - y$ 에 대해, complex exponential의 orthogonality를 사용하여 정리한다.

<sup>38</sup>See [https://en.wikipedia.org/wiki/Cyclic\\_group](https://en.wikipedia.org/wiki/Cyclic_group)

<sup>39</sup>대표적인 cyclic group으로 modular multiplication이 있다.

다음은 다양한 cyclic group들의 discrete log에 대한 예시이다.

- (Property of logarithm) For any  $G = \langle g \rangle$ ,  $\log_g 1 = 0$ ,
- For  $G = \mathbb{Z}_7^\times$ ,  $\log_3 2 = 2$ ,
- For  $G = \mathbb{Z}_{541}^\times$ ,  $\log_{126} 282 = 101$ .

이 문제를 해결하기 위해, Discrete logarithm problem을 *period finding* problem의 형태로 변환할 수 있다. 먼저 다음 integer function  $f$ 를 가정하자. 여기서  $\gcd(a, N) = 1$ 이며, modulo  $N$ 에서  $a$  order가  $r$ 이라고 하자.

$$f(x_1, x_2) = a^{sx_1+x_2} \bmod N$$

이렇게 정의한 함수는 어떤  $l$ 에 대해  $(l, -ls)$ 라는 period를 가진다.

$$f(x_1 + l, x_2 - ls) = a^{sx_1+sl+x_2-ls} \bmod N \equiv a^{sx_1+x_2} \bmod N = f(x_1, x_2)$$

Uncomputation을 사용하면 이  $f$ 에 대응되는 unitary를 구성할 수 있다.

$$U|x_1\rangle|x_2\rangle|y\rangle = |x_1\rangle|x_2\rangle|y \oplus f(x)\rangle$$

우리는 discrete logarithm problem을 다시 다음과 같이 표현할 수 있다.<sup>40</sup>

- input:  $b \in \mathbb{Z}_N^\times$  such that  $b \equiv a^s \pmod{N}$ ,  $a \in \mathbb{Z}$
- output: integer  $s$

따라서 period function  $f$ 를 이용하면 문제를 해결할 수 있다.

1. input state : 3개의 register를 사용한다.

$$|\psi\rangle = |0\rangle|0\rangle|0\rangle$$

2. apply  $H$ -gate on 1st and 2nd register

$$|\psi\rangle = \frac{1}{2^t} \sum_{x_1, x_2=0}^{2^t-1} |x_1\rangle|x_2\rangle|0\rangle$$

3. apply  $U_f$  :  $f(x_1, x_2)$ 를 3rd register에 저장한다.

$$|\psi\rangle = \frac{1}{2^t} \sum_{x_1, x_2=0}^{2^t-1} |x_1\rangle|x_2\rangle|f(x_1, x_2)\rangle$$

4. apply QFT on 3rd register

QFT를 사용하여 함수  $f(x_1, x_2)$ 를 표현하면 다음과 같다. (See Definition. 1.4.2)

$$|f(x_1, x_2)\rangle \rightarrow \frac{1}{r} \sum_{l_1, l_2=0}^{r-1} e^{2\pi i(l_1 x_1 + l_2 x_2)/r} |\hat{f}(l_1, l_2)\rangle$$

$$|\hat{f}(l_1, l_2)\rangle \triangleq \frac{1}{r} \sum_{x_1, x_2=0}^{r-1} e^{-2\pi i(l_1 x_1 + l_2 x_2)/r} |f(x_1, x_2)\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} e^{-2\pi i l_2 j / r} |f(0, j)\rangle$$

따라서 이를 이용하면, state를 다음과 같이 표현할 수 있다.<sup>41</sup>

$$\begin{aligned} |\psi\rangle &= \frac{1}{2^t \sqrt{r}} \sum_{l_2=0}^{r-1} \sum_{x_1, x_2=0}^{2^t-1} e^{2\pi i(sl_2 x_1 + l_2 x_2)/r} |x_1\rangle|x_2\rangle|\hat{f}(sl_2, l_2)\rangle \\ &= \frac{1}{2^t \sqrt{r}} \sum_{l_2=0}^{r-1} \left[ \sum_{x_1=0}^{2^t-1} e^{2\pi i(sl_2 x_1)/r} |x_1\rangle \right] \left[ \sum_{x_2=0}^{2^t-1} e^{2\pi i(l_2 x_2)/r} |x_2\rangle \right] |\hat{f}(sl_2, l_2)\rangle \end{aligned}$$

<sup>40</sup>  $a \nmid g$ ,  $s \nmid \alpha$ 에 대응된다.

<sup>41</sup> put  $l_1 = l_2, l_2 = sl_2$

5. apply IQFT on 1st and 2nd register

$$|\psi\rangle = \frac{1}{\sqrt{r}} \sum_{l_2=0}^{r-1} |sl_2/r\rangle |l_2/r\rangle |\hat{f}(sl_2, l_2)\rangle$$

따라서 1st, 2nd register를 측정하면 각각  $sl/r$ , 그리고  $l/r$ 이라는 값을 얻게 되는데,  $sl/r \times r/l = s$ 이므로  $s$ 를 구할 수 있다. □

### 1.7.3 Hidden subgroup problem

이제 Hidden subgroup problem에 대해 이야기할 시간이 찾아왔다.<sup>42</sup>

**Definition 1.7.1 (Hidden subgroup problem).** Let  $f$  be a function from finitely generated group  $G$  to some finite set  $X$ . s.t.  $f$  is constant on the cosets of a subgroup  $K$ , and distinct on each coset.<sup>a b</sup>

$$f : G \rightarrow X \quad \longrightarrow \quad U_f |g\rangle |h\rangle = |g\rangle |h \oplus f(g)\rangle \quad (\text{for } g \in G, h \in X)$$

Find a generating set for  $K$ .

<sup>a</sup> $K$ 가 주어졌을 때, coset은  $G$ 의 원소에 대해서  $gK = \{gk : k \in K\}$ 로 정의된다.

<sup>b</sup>같은 coset이라면  $f$ 의 값은 매우 동일하지만, 서로 다른 coset에 속한다면  $f$ 의 값은 서로 다르다.

앞에서 우리가 다루었던 order-finding problem, period-finding, discrete logarithm과 같은 문제들이 Hidden subgroup problem의 특수한 예시이다. 예를 들어, order-finding problem이 왜 Hidden subgroup problem으로 표현될 수 있는지 알아보자. Hidden subgroup problem에서 각각을 다음과 같이 정의하자.

- $G = \mathbb{Z}$
- $K = r\mathbb{Z}$
- (coset)  $K_g = \{rg\}$  for  $g \in \mathbb{Z}$  (e.g.,  $K_1 = \{\dots, -2, -1, 0, \dots, 1, 2\}$ ,  $K_3 = \{\dots, -6, -3, 0, \dots, 3, 6\}$ )
- $f(k) = a^k \pmod N$

이때, (1)  $f(k) = f(k+r)$ 의 주기를 가지고 (2)  $0 \leq k \leq r-1$  범위에서 각각의  $f(k)$ 의 값을 모두 구분된다. 이 두 가지 성질은  $f$ 가 동일한 coset에 속하는 원소들에 대해서 constant하고, 서로 다른 coset에 속한 원소들에 대해서는 distinct한 값을 가지는 것을 의미한다. 따라서 order-finding problem은 Hidden subgroup problem의 특수한 경우로 생각할 수 있다.

$G$ 가 finite abelian group<sup>43</sup>이라면, QFT를 적용하여 Hidden subgroup problem을 해결할 수 있다. 그러나, non-abelian group에 대해서는 아직 이 문제를 효과적으로 푸는 방법이 알려지지 않았다. (non-abelian group에 대한 대표적인 문제가 바로 *Graph isomorphism*)

## 1.8 Quantum search algorithms

Quantum search algorithm에 대해서 알아보자. 정렬되어있거나 특정한 규칙이 있는 데이터에 대한 search algorithm은 classical computer에서도 이미 충분히 효과적으로 수행할 수 있다. 그러나, 여기에서 다룰 search problem은 unstructured한 데이터를 다룬다. 이를 위해 먼저 하나의 oracle을 가정하자.

$N$ 개의 항목이 있는 unstructured 데이터에 대해서,  $f(x)$ 는  $x$ 번째 항목이 우리가 찾고자 하는 항목인지 를 판단하는 oracle이다. 따라서  $f(x) = 1$ 이면,  $x$ 가 solution이 되고  $f(x) = 0$ 이면,  $x$ 가 solution이 아니다. 편의를 위해서  $N = 2^n$ 이며, 찾고자 하는 항목의 개수가  $M$  ( $1 \leq M \leq N$ )이라고 가정하자.

$$f : \{0, 1\}^n \rightarrow \{0, 1\}$$

이 문제를 어떻게 해결할 수 있을까? 단순하게 생각하면 랜덤한  $x$ 를 선택하여  $f(x)$ 의 값을 확인한 뒤  $x$  가 solution인지 아닌지를 판단할 수 있으며, 선택한  $x$ 가 solution일 확률은  $M/N$ 이다. 만약  $N$ 이  $M$ 에 비해 exponential하게 크다면, success probability는 exponential하게 줄어들기에 이는 효율적이지 않다.

Quantum computer에서  $f$ 에 대한 unitary는 다음과 같이 정의된다.

$$|x\rangle |q\rangle \xrightarrow{O} |x\rangle |q \oplus f(x)\rangle$$

<sup>42</sup>See [https://en.wikipedia.org/wiki/Hidden\\_subgroup\\_problem](https://en.wikipedia.org/wiki/Hidden_subgroup_problem)

<sup>43</sup>교환법칙이 성립하는 군

$x$ 는  $N$ 개의 항목에 대한 인덱스를 나타내기 위해  $n$ 개의 qubit으로 이루어지며, query qubit은 1개의 qubit으로 이루어진다. 즉, 만약  $f(x) = 1$ ;  $x$ 가 solution이라면 query qubit의 값이 bit-flip된다.

만약  $|q\rangle = |-\rangle$ 으로 설정하면, 다음과 같은 일이 벌어진다.

$$|x\rangle \left( \frac{|0\rangle - |1\rangle}{2} \right) \xrightarrow{O} \frac{|x\rangle |f(x)\rangle - |x\rangle |1 \oplus f(x)\rangle}{2} = (-1)^{f(x)} |x\rangle \left( \frac{|0\rangle - |1\rangle}{2} \right)$$

$x$ 가 solution이라면 bit-flip이 일어나서 phase가 바뀌게 되고, solution이 아닌 경우에는 phase가 바뀌지 않는다. 따라서 이 oracle  $O$ 를 다음과 같이 단순하게 쓸 수 있다.

$$|x\rangle \xrightarrow{O} (-1)^{f(x)} |x\rangle.$$

따라서 이 oracle을 활용하면, classical computer와 동일한 randomized algorithm을 적용하여도  $O(\sqrt{N/M})$ 의 complexity를 가진다.

### 1.8.1 Grover operator

Unstructured search problem을  $\Theta(\sqrt{N})$  복잡도로 해결할 수 있는 Grover search algorithm을 설명하기 위해 먼저 Grover operator  $G$ 를 정의한다.<sup>44</sup>

$$G \triangleq (2|\Psi\rangle\langle\Psi| - I)O$$

이때  $|\Psi\rangle$ 는 다음과 같이 정의되며<sup>45</sup>,  $O$ 는 oracle이다.

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

우리는 이제  $G$ 가 superposition of solutions state  $|\beta\rangle$ , superposition of non-solutions ketalpha<sup>46</sup> span하는 two-dimensional space에서 회전 연산을 수행한다는 것을 보일 것이다.

$$|\alpha\rangle \triangleq \frac{1}{\sqrt{N-M}} \sum_{x:f(x)=0} |x\rangle, \quad |\beta\rangle \triangleq \frac{1}{\sqrt{M}} \sum_{x:f(x)=1} |x\rangle. \quad (1.32)$$

$|\Psi\rangle$ 를 다음과 같이  $|\alpha\rangle, |\beta\rangle$ 의 linear combination으로 표현할 수 있기 때문에,  $|\Psi\rangle$ 는  $\text{span}\{|\alpha\rangle, |\beta\rangle\}$  vector space에 속한다.

$$|\Psi\rangle = \sqrt{\frac{N-M}{N}} |\alpha\rangle + \sqrt{\frac{M}{N}} |\beta\rangle$$

$G$ 가 수행하는 연산을 이해하기 위해서, 각 단계마다 gate가 수행하는 역할을 분석해보자.

- Oracle  $O$ 가  $\text{span}\{|\alpha\rangle, |\beta\rangle\}$ 에 속하는 임의의 vector에 대해 다음과 같이 동작한다.

$$O(a|\alpha\rangle + b|\beta\rangle) = a|\alpha\rangle - b|\beta\rangle$$

$\Rightarrow |\alpha\rangle$  축에 대한 reflection

- ( $2|\Psi\rangle\langle\Psi| - I$ ) 연산은 eigenvalue  $\pm 1$ 에 대해 spectral decomposition을 할 수 있다.

$$2|\Psi\rangle\langle\Psi| - I = |\Psi\rangle\langle\Psi| - (I - |\Psi\rangle\langle\Psi|)$$

$|\Psi\rangle\langle\Psi|$ 는  $|\Psi\rangle$ 에 대한 projector이고 반대로  $I - |\Psi\rangle\langle\Psi|$ 는  $|\Psi\rangle$ 의 orthogonal subspace에 대한 projector이다. 따라서 이 연산은  $\{|\Psi\rangle, |\Psi\rangle^\perp\}$  vector space에서  $|\Psi\rangle$  축에 대한 reflection을 수행하게 된다.

$\Rightarrow |\Psi\rangle$  축에 대한 reflection

따라서 2개의 reflection operator들의 곱으로 정의되는 Grover operator는 rotation operator가 된다. 이는  $G^k|\Psi\rangle$ 를 수행하면  $\text{span}\{|\alpha\rangle, |\beta\rangle\}$  vector space에 속하는 모든 vector를 만들어 낼 수 있다는 의미이다!

좀 더 자세한 설명을 위해  $|\Psi\rangle$ 를  $\theta$ 를 사용하여 표현하자.<sup>46</sup>

$$|\Psi\rangle = \cos \frac{\theta}{2} |\alpha\rangle + \sin \frac{\theta}{2} |\beta\rangle$$

<sup>44</sup>  $G = (H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n})O$

<sup>45</sup>  $N$ 까지의 모든 가능한 입력들의 superposition

<sup>46</sup>  $(\sqrt{(N-M)/N})^2 + (\sqrt{M/nN})^2 = 1$ 을 이용하면 cos, sin function을 이용할 수 있다.

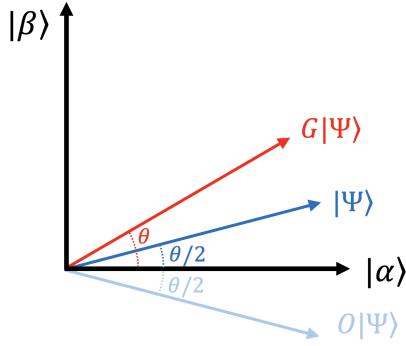


Figure 1.4: Grover operator

그럼,  $G$ 가  $|\Psi\rangle$ 에 대해 가해지면, state는 다음과 같이 변한다.

$\Rightarrow$  즉,  $G$ 는  $|\Psi\rangle$ 를  $\theta$ 만큼 rotation 시킨다.

$$G|\Psi\rangle = \cos \frac{3\theta}{2} |\alpha\rangle + \sin \frac{3\theta}{2} |\beta\rangle \quad (1.33)$$

따라서  $G$  gate는  $\{|\alpha\rangle, |\beta\rangle\}$  basis에 대해 다음과 같은 matrix notation으로 표현할 수 있다.

$$G = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

$G$ 를  $k$ 번 적용하면, 다음 상태를 얻게된다.

$$G^k|\Psi\rangle = \cos \left( \frac{2k+1}{2}\theta \right) |\alpha\rangle + \sin \left( \frac{2k+1}{2}\theta \right) |\beta\rangle$$

만약  $(2k+1/2)\theta$ 가  $\pi/2$ 에 충분히 가깝다면,  $\cos((2k+1/2)\theta)$ 가 0에 가까워지기 때문에  $G^k|\Psi\rangle$ 를 측정하면,  $|\beta\rangle$ 를 얻을 확률이 높아진다. 이 아이디어를 기반으로 Grover search algorithm이 제안되었다.

### 1.8.2 Grover search algorithm

Grover search algorithm은 앞에서 도입한 Grover operator를 사용하여 unstructured search problem을 해결하는 알고리즘으로, 다음과 같은 과정을 따른다.

1. input state

$$|\psi\rangle = |0\rangle^{\otimes n}$$

2. apply Hadamard gate :  $|\Psi\rangle$ 를 만들기 위해 Hadamard gate를 적용한다.

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

3. (Grover iteration;  $G = (H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n})O$ ) 다음을  $k$ 번 반복한다.

- (a) apply the oracle  $O$
- (b) apply the Hadamard gate  $H^{\otimes n}$
- (c) perform a conditional phase shift ( $0\circ$  아닌 모든 basis는 phase shift가 된다.)

$$|x\rangle \rightarrow -(-1)^{\delta_{x0}} |x\rangle$$

- (d) apply the Hadamard gate  $H^{\otimes n}$

4. measure the state : 우리가 찾고자하는  $M$ 개의 항목중에서 하나의 index를 확률에 따라 얻을 수 있다.  
(See Eq. (1.32))

$$|\psi\rangle = \cos \left( \frac{2k+1}{2}\theta \right) |\alpha\rangle + \sin \left( \frac{2k+1}{2}\theta \right) |\beta\rangle \longrightarrow |\beta\rangle$$

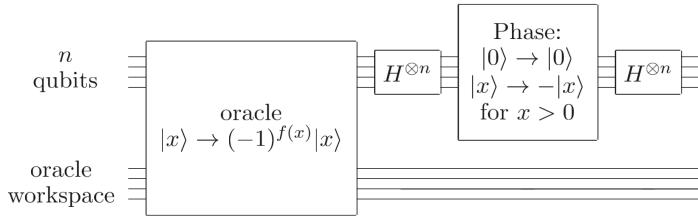


Figure 1.5: Grover operator circuit

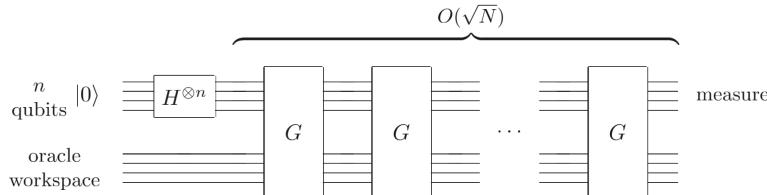


Figure 1.6: Grover search algorithm circuit

### 1.8.3 Performance

그렇다면, Grover algorithm을 사용하여 solution state  $|\beta\rangle$ 에 다가가기 위해서는 총 몇 번의 iteration이 필요할까? Iteration을  $k$ 번 했을 때, state는 다음과 같다.

$$G^k |\Psi\rangle = \cos\left(\frac{2k+1}{2}\theta\right) |\alpha\rangle + \sin\left(\frac{2k+1}{2}\theta\right) |\beta\rangle$$

$k$ 를 다음과 같이 설정하면  $|\alpha\rangle$ 의 amplitude가 0이 되므로, state가  $|\beta\rangle$ 가 된다.

$$\frac{2k+1}{2}\theta = \frac{\pi}{2}$$

위 조건에 가장 가까운  $k$ 를 선택했을 때, 얻을 수 있는 최종 state  $|\psi\rangle$ 의 각도는  $\theta/2 \leq \pi/4$ 로 bound 된다. 따라서 최종 state가  $\beta$ 가 될 확률은 적어도 절반이 넘는다.<sup>47</sup>

좀 더 구체적으로 나타내면 다음과 같다.<sup>48</sup>

$$k = \frac{\pi}{2\theta} - \frac{1}{2} = \frac{\pi}{4 \arccos \sqrt{(N-M)/N}} - \frac{1}{2} = \frac{\pi}{4} \sqrt{\frac{N}{M}} - \frac{1}{2} - O\left(\sqrt{\frac{M}{N}}\right)$$

따라서 Grover search algorithm의 complexity;  $k$ 는  $O(\sqrt{N/M})$ 이다. 이는 랜덤하게  $M$ 을 선택하여 문제를 해결하는 방법보다 훨씬 효율적이다. ( $O(N/M)$ )<sup>49</sup>

한 가지 남아있는 문제점은,  $k$ 를 선택하기 위해서는  $M$ 의 값을 알아야한다는 것이다. 만약 문제에서 우리가 찾으려는 항목이 있는 곳의 개수  $M$ 을 제공하지 않는다면,  $k$ 를 결정할 수 없다. 이를 해결하는 2가지 방법이 존재한다.

- $k$ 를 랜덤하게 선택하여 탐색한다. (이 경우에도 비슷한 수준의 성능을 낼 수 있음이 알려져 있다.)
- $M$ 을 추정하는 알고리즘을 사용한다. (by [amplitude estimation algorithm](#))

### Lecture 15

#### 1.8.4 Example: Classical circuit-SAT problem

6 Nov. 10:30

이번에는 Grover search algorithm을 이용하여 classical circuit-SAT problem을 해결하는 예시를 살펴보자.

**Problem** Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ 을 계산하는 Poly( $n$ ) 크기의 classical circuit이 존재한다고 가정하자. 이때,  $f(x) = 1$ 이 되도록 만드는  $x$ 가 존재하는가? 존재한다면  $x$ 는 몇 개인가?

<sup>47</sup>  $\sin(\pi/4) = \cos(\pi/4) = 1/\sqrt{2}$

<sup>48</sup>  $\arccos \sqrt{1-x} \approx \sqrt{x} + O(x^{3/2})$  임을 이용한다.

<sup>49</sup> 성공확률이  $M/N$ 이므로, complexity는 그 역수

랜덤하게  $x$ 를 결정하여  $f(x)$ 의 값을 확인하는 방식은  $O(2^n)$ 이라는 complexity를 가진다. 그러나 Grover search algorithm을 사용하면,  $O(2^{2/n})$ 이라는 complexity만으로도 이 문제를 해결할 수 있다.

먼저 uncomputation을 사용하여 주어진  $f$ 에 대한  $O_f$ 를 설계하자.

$$O_f|x, b\rangle = |x, b \oplus f(x)\rangle$$

이 oracle은  $N$ 개의 항목중에서  $M$ 개의 항목을 찾기 위해 각 항목의 위치가 우리가 찾기를 바라던 값인지 를 확인하는 search algorithm의 oracle과 동일하다. 따라서 Grover search algorithm을 동일한 방식으로 적용하면 주어진 문제를 쉽게 해결할 수 있다.

### 1.8.5 Example: Amplitude amplification

Grover search algorithm은 *amplitude amplification*을 수행하기 위해 사용할 수 있다. Amplitude amplification은 bad state  $|\alpha\rangle$ 에 대한 amplitude는 감소시키고, good state  $|\beta\rangle$ 에 대한 amplitude는 증가시키는 것을 목적으로 한다. 먼저 다음과 같은 unitary를 가정하자.<sup>50</sup>

$$|\psi_0\rangle \triangleq U|0^n\rangle = \sqrt{p_0}|\beta\rangle + \sqrt{1-p_0}|\alpha\rangle = \sin \frac{\theta}{2}|\beta\rangle + \cos \frac{\theta}{2}|\alpha\rangle, \quad s.t. \langle\alpha|\beta\rangle = 0, p_0 \ll 1$$

이 문제에서는 unstructured search problem에서 요구하는  $|\alpha\rangle, |\beta\rangle$ 가 각각 특정 computational basis의 superposition state여야한다는 조건이 없기 때문에,  $|\alpha\rangle$ 에 대한 reflector를 설계해야한다.

$U, U^\dagger$  그리고 bad state  $|\alpha\rangle$ 의 orthogonal subspace에 대한 projector  $I - |\alpha\rangle\langle\alpha|$ 가 주어졌을 때, 다음과 같이 initial state  $|\psi_0\rangle$ 의 orthogonal subspace에 대한 projector  $I - |\psi_0\rangle\langle\psi_0|$ 를 만들어낼 수 있다.

$$2|\psi_0\rangle\langle\psi_0| - I = U(2|0^n\rangle\langle 0^n| - I)U^\dagger.$$

따라서 다음 연산을 수행하면,  $G$  operator와 유사하게,  $|\alpha\rangle$ 에 대한 reflection,  $|\psi_0\rangle$ 에 대한 reflection 일어나서  $|b\rangle$ 에 가깝게 회전하게 된다.

$$G \triangleq (2|\psi_0\rangle\langle\psi_0| - I)(2|\alpha\rangle\langle\alpha| - I) = U(2|0^n\rangle\langle 0^n| - I)U^\dagger(2|\alpha\rangle\langle\alpha| - I)$$

## 1.9 Amplitude estimation algorithm (Quantum counting)

Amplitude estimation algorithm; Quantum counting algorithm은 Grover search problem에서 solution의 개수  $M$ 을 추정하는 알고리즘이다. 이 알고리즘을 먼저 사용하게 되면,  $M$ 이 얼마인지 추정할 수 있을뿐더러 만약 주어진 데이터셋에 solution이 존재하지 않는 경우,  $M = 0$ 임을 확인해줄 수 있다.

이를 이용하면, search problem의 해가 있는지 없는지를 판단하는 문제의 형태로 변환할 수 있는 NP-complete 문제를 푸는데 도움이 될 수 있다.

Grover operator  $G$ 를  $\{|\beta\rangle, |\alpha\rangle\}$ 에 대해서 matrix 표현으로 나타내면, 다음과 같다.

$$\begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}$$

$G$ 를 spectral decomposition하면 다음과 같이 나타낼 수 있고

$$G = \cos \theta |\beta\rangle\langle\beta| + \cos \theta |\alpha\rangle\langle\alpha| + \sin \theta |\beta\rangle\langle\alpha| - \sin \theta |\alpha\rangle\langle\beta|.$$

따라서  $G$ 의 eigenvalue, eigenvector는 각각 다음과 같다.

- eigenvalue :  $e^{\pm i\theta}$
- eigenvector :  $|\psi_\pm\rangle = \frac{1}{\sqrt{2}}(|\beta\rangle \pm i|\alpha\rangle)$

만약 우리가  $e^{\pm i\theta}$ 에서  $\theta$ 의 값을 알아낸다면  $\sin(\theta/2) = \sqrt{M/N}$ 이므로  $M$ 이 얼마인지 추정할 수 있다.  $\theta$ 의 값을 알아내기 위해 phase estimation을 도입할 수 있다. 그런데  $G$ 의 eigenvector를 준비하려면,  $|\alpha\rangle, |\beta\rangle$ 의 상태를 알아야한다는 문제가 존재한다.

<sup>50</sup>amplitude amplification은 사실 unstructured search problem에 대한 일반화된 형태이다.

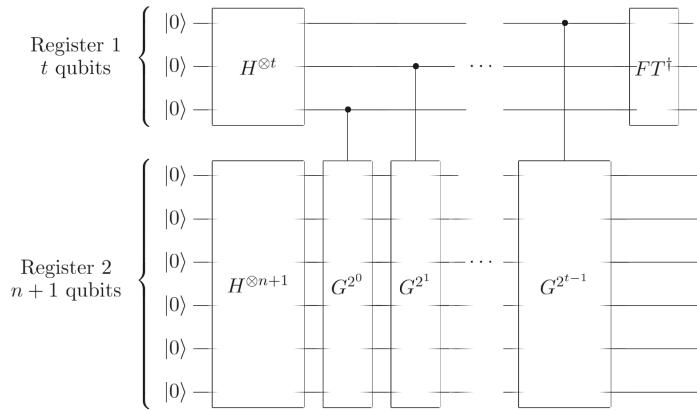


Figure 1.7: Quantum counting circuit

이를 해결하기 위해 initial state와  $G$ 의 eigenvector가 다음의 관계를 가짐을 이용하자.

$$|\langle \psi_0 | \psi_+ \rangle|^2 = \frac{1}{2} \left| \sin \frac{\theta}{2} + i \cos \frac{\theta}{2} \right|^2 = \frac{1}{2} = |\langle \psi_0 | \psi_- \rangle|^2.$$

즉, eigenvector 대신에 initial state를 이용하면, 각각  $1/2$ 의 확률로 eigenstate 둘 중 하나가 되기 때문에 이를 이용하여  $\theta$ 를 추정할 수 있다. Fig. 1.7에서 Register 2를  $|\psi_0\rangle$ 으로 초기화하기 위해  $H$ -gate를 사용하고 있는 것을 확인할 수 있다.

Phase estimation의 성능은 측정에 사용하는 register 1의 qubit 개수  $t$ 에 의존한다.  $t = m + \lceil \log(2 + 1/2\delta) \rceil$  으로 선택하면, 적어도  $1 - \delta$ 의 확률로  $2^{-m}$  accuracy를 달성할 수 있다.

$$|\Delta\theta| = |\theta - \tilde{\theta}| \leq 2^{-m}$$

그렇다면,  $\tilde{\theta} \approx \theta$ 로부터 추정한  $\tilde{M}$ 의 error  $|\Delta M| = |\tilde{M} - M|$ 의 upper bound는 얼마인가? 다음의 inequality<sup>51</sup>을 Eq. (1.35)에 적용하면,

$$\left| \sin\left(\frac{\theta + \Delta\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) \right| \leq \frac{|\Delta\theta|}{2}, \quad \left| \sin\left(\frac{\theta + \Delta\theta}{2}\right) \right| \leq \sin\left(\frac{\theta}{2}\right) + \frac{|\Delta\theta|}{2}$$

$|\Delta M|/N$ 에 대해서 다음과 같이 정리할 수 있으며

$$\frac{|\Delta M|}{N} = \left| \sin^2\left(\frac{\theta + \Delta\theta}{2}\right) - \sin^2\left(\frac{\theta}{2}\right) \right| \quad (1.34)$$

$$= \left( \sin\left(\frac{\theta + \Delta\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) \right) \left| \sin\left(\frac{\theta + \Delta\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) \right| \quad (1.35)$$

$$\leq \left( 2 \sin\left(\frac{\theta}{2}\right) + \frac{|\Delta\theta|}{2} \right) \frac{|\Delta\theta|}{2} \quad (1.36)$$

따라서 Eq. (1.36)를  $|\Delta M|$ 에 대해 정리하면, 다음의 upper bound를 얻을 수 있다.

$$|\Delta M| \leq \underbrace{\left( 2N \sin\left(\frac{\theta}{2}\right) + \frac{N|\Delta\theta|}{2} \right)}_{\sqrt{M/N}} \frac{|\Delta\theta|}{2} = \left( 2\sqrt{MN} + \frac{N}{2^{m+1}} \right) \frac{1}{2^{m+1}}$$

즉,  $N \approx 2^{2m}$ 을 만족하도록  $m$ 을 선택하면,  $|\Delta M|$ 의 upper bound는  $O(\sqrt{M})$ 이 된다.

마지막으로 Quantum counting의 complexity를 계산해보자. Quantum counting algorithm은  $t$ 개의 qubit에 모두 controlled- $G^{2^i}$  gate를 가하기 때문에  $O(2^t)$ 의 complexity를 가진다. 이때,  $t = m + \lceil \log(2 + 1/2\delta) \rceil$ 로 선택했기 때문에,  $O(2^m)$ 이 되며, 이는  $O(\sqrt{N})$ 과 동일하다.

<sup>51</sup>  $\sin A - \sin B = 2 \cos\left(\frac{A+B}{2}\right) \sin\left(\frac{A-B}{2}\right)$

## 1.10 HHL (Harrow–Hassidim–Lloyd) algorithm

### Problem Setup

마지막으로 다룰 알고리즘은 HHL algorithm[HHL09] 이다. HHL algorithm은 다음 linear equation의 solution  $x$ 를 구하는 quantum algorithm이다.

$$Ax = b$$

이때,  $A$ 는 nonsingular matrix이고  $A \in \mathbb{C}^{N \times N}$ ,  $b \in \mathbb{C}^N$ 이다.<sup>52</sup>

Linear algebra에 따르면 다음 관계를 만족하기 때문에,  $A$ 의 역행렬을 구해서 solution vector  $x$ 를 구할 수 있다. 그러나 문제는,  $A$ 의 역행렬을 계산하는데 exponential complexity가 요구된다는 것이다.

$$x = A^{-1}b$$

따라서 quantum computer를 사용하여 linear equation을 더 효과적으로 해결하기 위해 HHL algorithm이 고안되었다. 이를 위해서 먼저  $A$ 가 Hermitian;  $A = A^\dagger$ 이라고 가정하자. 만약, 주어진  $A$ 가 Hermitian이 아니라면 다음과 같이 새로운 Hermitian 행렬  $\tilde{A}$ 로 재구성하여

$$\tilde{A} = \begin{pmatrix} 0 & A^\dagger \\ A & 0 \end{pmatrix} = |1\rangle\langle 0| \otimes A + |0\rangle\langle 1| \otimes A^\dagger$$

다음과 같이 변환한 문제를 풀어서  $b$ 를 구할 수 있기 때문에 이 가정은 일관성을 잃지 않는다.

$$\tilde{A}|0, x\rangle = |1, b\rangle$$

다음으로  $b$ 를 quantum state vector로 표현할 수 있도록  $\|b\| = 1$ 라고 가정하자. 더 염밀하게 표현하자면, quantum state를  $b$ 로 초기화하는 unitary  $U_f$ 가 존재한다고 하자.

$$|b\rangle = U_b |0^n\rangle$$

마지막으로,  $x$ 를 quantum state vector로 손실 없이 표현할 수 있으려면  $\|x\| = 1$ 을 만족해야한다. 그러나,  $b$ 가 unit vector라고 해서  $x = A^{-1}b$ 의 결과로 얻어지는  $x$  또한 unit vector일 것이라는 보장은 존재하지 않는다. 따라서 quantum version의 linear equation을 해결하는 알고리즘의 목표는,  $|\hat{x}\rangle$ <sup>53</sup>와 차이가 매우 작은 state  $|\tilde{x}\rangle$ 를 얻는 것이다.<sup>54</sup>

$$\|\hat{x}\rangle - |\tilde{x}\rangle\| \leq \epsilon, \quad |\hat{x}\rangle = \frac{\hat{x}}{\|\hat{x}\|} = \frac{A^{-1}b}{\|A^{-1}b\|}.$$

알고리즘으로 얻은 output은 normalize된 solution이므로, 우리가 실제로 얻고 싶은 solution  $x$ 를 구하기 위해서는  $\|\tilde{x}\|$ 의 값까지 알고리즘이 output으로 반환해야 한다.

지금까지 설명한 내용을 정리하면, HHL algorithm의 problem은 다음과 같이 정의된다.

- input: Hermitian matrix  $A \in \mathbb{C}^{N \times N}$ ,  $b \in \mathbb{C}^N$
- output:  $\tilde{x} \in \mathbb{C}^N$  s.t.  $|\hat{x} - \tilde{x}\rangle \leq \epsilon$  where  $Ax = b$  and norm of solution  $\|\tilde{x}\|$

### Algorithm description

먼저  $A$ 가 다음과 같은 eigenvalue, eigenvector를 가진다고 하자. ( $0 \leq j < N$ )

$$A|v_j\rangle = \lambda_j|v_j\rangle$$

이때,  $A$ 의 eigenvalue들이 다음을 따르며,  $d$ -bit representation으로 표현할 수 있다고 하자. (즉,  $A$ 는 positive matrix이다.)<sup>55</sup>

$$0 < \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{N-1} < 1$$

$A$ 는 Hermitian이면서 동시에 positive matrix이므로 Hamiltonian simulation을 이용하면 다음과 같은 unitary를 만들 수 있다.<sup>56</sup>

$$U = e^{i2\pi A}$$

<sup>52</sup>without generality

<sup>53</sup>정답  $x$ 를 normalize한 state

<sup>54</sup>solution이  $\|x\| \neq 1$ 이라면, quantum state vector로 표현 불가능하기에 우리가 QC를 통해 얻고자하는 결과는  $\hat{x}$ 이다. 따라서 알고리즘의 연산 결과로 얻은 state가 normalize된  $x$ 와 최대한 가깝게 만들고자 한다.

<sup>55</sup>eigenvalue가 negative value를 가지는 경우에도 HHL algorithm을 적용할 수 있지만, 추후 사용할 QPE결과를 그에 맞춰서 수정해야 한다.

<sup>56</sup>unitary로 설계되었기 때문에, 이제 quantum computer에서 사용할 수 있다!

이를 이용하면, HHL algorithm은 다음과 같이 동작한다.

1. input state :  $H$ 에 대한 eigenvector들의 superposition state인  $|b\rangle$ 로 2nd register를 초기화 한다.

$$|\psi\rangle = |0^d\rangle \sum_j \beta_j |v_j\rangle = |0^d\rangle |b\rangle$$

2. perform QPE : QPE를 수행하여,  $U$ 의 eigenvalue  $\lambda_j$ 들의 superposition state가 1st register에 저장되게 한다.

$$|\psi\rangle = \sum_j \beta_j |\lambda_j\rangle |v_j\rangle$$

**Note (Idea).** 이때, solution  $x$ 를 다음과 같이 표현할 수 있음을 이용하자.

$$A^{-1}|b\rangle = \left( \sum_j \lambda_j^{-1} |v_j\rangle \langle v_j| \right) |b\rangle = \sum_j \frac{\beta_j}{\lambda_j} |v_j\rangle$$

따라서 앞으로 우리가 할 일은 1st register에 대한 controlled rotation을 수행하여 각  $\beta_j$ 에  $\lambda_j^{-1}$ 을 곱하여 solution  $x$ 와 유사한 상태로 만드는 것이다.

3. apply controlled rotation unitary :  $U_{CR}$ 은 다음과 같이 정의된다. 즉,  $|\lambda_j\rangle$ 의 값에 따라서 첫 번째 qubit이  $|1\rangle$  방향으로 회전하게 된다.

$$U_{CR}|0\rangle |\lambda_j\rangle = \left( \sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right) |\lambda_j\rangle$$

따라서 이렇게 정의된 gate를  $|\psi\rangle$ 에 가하면, 다음과 같은 상태가 된다.

$$|\psi\rangle = |0\rangle \sum_j \beta_j |\lambda_j\rangle |v_j\rangle \xrightarrow{U_{CR}} \sum_j \left( \sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right) \beta_j |\lambda_j\rangle |v_j\rangle$$

4. perform  $QPE^\dagger$  : QPE를 uncomputation하여 1st register<sup>57</sup>를 초기상태  $|0\rangle^d$ 로 되돌린다. 따라서, ancilla bits들의 표현을 생략하면 최종적으로 얻게되는 state는 다음과 같다.

$$|\psi\rangle = \sum_j \left( \sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right) \beta_j |v_j\rangle.$$

5. measuring the single qubit

첫 번째 qubit의 값을 측정했을 때, outcome이 1이라면 post-state는 다음과 같으며,

$$\tilde{x} = \sum_j \frac{C\beta_j}{\tilde{\lambda}_j} |v_j\rangle$$

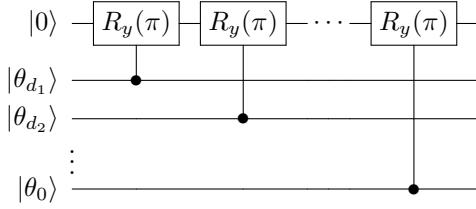
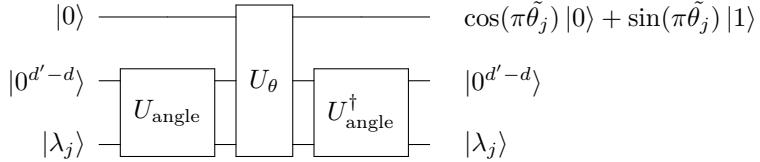
이를 normalize한 값; 즉 우리가 원하는 결과가 qubit에 저장되게 된다.

$$|\tilde{x}\rangle = \frac{\tilde{x}}{\|\tilde{x}\|} \approx |x\rangle$$

6. estimate  $p(1)$ <sup>58</sup>

unnormalize solution  $x$ 를 얻기 위해서 필요한 값,  $\|\tilde{x}\|$ 은 outcome이 1일 확률과 동일하다.

$$p(1) = \left\| \sum_j \frac{C\beta_j}{\tilde{\lambda}_j} |v_j\rangle \right\|^2 = \|\tilde{x}\|^2 \approx C^2 \|A^{-1}|b\rangle\|^2$$


 Figure 1.8:  $U_{\text{angle}}$  circuit

 Figure 1.9:  $U_{CR}$  circuit

$p(1)$ 은  $C$ 가 커질수록 증가하기 때문에,  $C$  값의 upper bound인  $\lambda_0$ 이 가능한 커야 HHL algorithm<sup>57</sup>을 효과적으로 동작할 수 있다. 또는 HHL을 적용하기 전에 **amplitude amplification**을 수행하여 최대한  $p(1)$ 이 커지도록 만들 수 있다.

마지막으로, 어떻게  $U_{CR}$ 을 설계할 수 있는지에 대해 이야기 해보자.  $U_{CR}$ 은 d-bit representation으로 표현되는  $0 \leq \lambda_j < 1$  값에 따라 computational basis에서 rotation을 수행한다. 이를  $0 \leq \theta < 1$ ,  $\theta = 0.\theta_{d-1} \dots \theta_0$ 에 대해 표현하면, 다음과 같이 나타낼 수 있다.

$$U_\theta |0\rangle |\theta\rangle \rightarrow (\cos \pi\theta |0\rangle + \sin \pi\theta |1\rangle) |\theta\rangle$$

이 연산은 Y축에 대한 rotation gate와 유사하게 동작한다.

$$e^{-i\tau \hat{Y}} = \begin{pmatrix} \cos \tau & -\sin \tau \\ \sin \tau & \cos \tau \end{pmatrix} \equiv R_y(2\tau)$$

따라서  $\tau = \pi(0.j_{d-1} \dots j_0)$ 으로 선택하여  $R_Y(2\tau)$ 를 이용하면,  $U_\theta$ 를 구현할 수 있다. Fig. 1.8를 보면 우리가 QPE에서 사용했던 controlled operation과 다르게,  $\lambda_j$ 의 각 자릿수의 값에 따라 single qubit이 회전하는 구조로 이루어져 있음을 알 수 있다.

$$U_\theta = \sum_{j \in [2^d]} \exp(-i\pi(0 \cdot j_{d-1} \dots j_0) \hat{Y}) \otimes |j\rangle \langle j|$$

$\theta_j$ 를 다음과 같이 설정하면  $|\lambda_i\rangle$ 에 대한 controlled-rotation gate를 만들 수 있다.

$$\theta_j = \frac{1}{\pi} \sin^{-1}(C/\lambda_j)$$

$\theta$  상태로 준비시키는 연산은 classical computer가 쉽게 할 수 있으므로 quantum computer도 uncomputation과 같은 기법을 사용하면, initial state를  $|\theta\rangle$ 로 전환하는 다음의 unitary를 만들 수 있다.

$$U_{\text{angle}} |0^{d'-d}\rangle |\lambda_j\rangle = |\theta_j\rangle$$

## Lecture 16

### 1.11 Optimality of the quantum search algorithm

8 Nov. 17:00

이번 챕터에서는, quantum search algorithm의 optimality에 대해 다루어보고자 한다. Unstructured search problem<sup>58</sup>의 single solution  $x$ 를 가진다고 하자. 그럼 이 문제를 해결하기 위해서 initial state  $|\psi\rangle$ 에 대해 oracle  $O_x$ 를  $k$ 번 적용하여 solution state에 가까워지도록 만들 수 있다.  $O_x = I - 2|x\rangle \langle x|$ 는  $|x\rangle$ 에 대해서 phase  $-1$ 을 가하고 다른 state는 변화시키지 않는다.

<sup>57</sup>ancilla

<sup>58</sup> $p(1)$ 을 empirical distribution으로부터 구할 수도 있고, 또는 amplitude estimation을 이용할 수도 있다. 만약 추정한  $p(1)$ 이 너무 낮다면 HHL을 사용하지 않는 결정을 내릴 수도 있다.

$k$ 번 oracle을 적용하는 동안, unitary operation  $U_1 \cdots U_k$ 를 각 연산의 사이에 적용한다고 하면, 다음과 같이 state evolution을 표현할 수 있다.

$$\begin{aligned} |\psi_k^x\rangle &\triangleq U_k O_x U_{k-1} O_x \cdots U_1 O_x |\psi\rangle \\ |\psi_k\rangle &\triangleq U_k U_{k-1} \cdots U_1 |\psi\rangle \end{aligned}$$

우리의 목표는 다음의 값에 대한 bound를 찾는 것이다.

$$D_k \triangleq \sum_x \| |\psi_k^x\rangle - |\psi_k\rangle \|^2$$

직관적으로,  $D_k$ 는 oracle을 적용하지 않았을 때의 state evolution  $|\psi_k\rangle$ 를 기준으로,  $k$ 번 oracle을 적용했을 때 발생하는 편차를 나타낸다. 만약  $D_k$ 의 값이 작다면, 어떠한  $x$ 에 대해서도 그 state가 유사하다는 것이므로 높은 확률로 solution을 찾기 힘들 것이다.

증명은, 다음 과정을 따라 진행된다. (1)  $D_k$ 가  $O(k^2)$ 의 upper bound를 가진다는 것을 보인다. (2)  $N$ 개의 item을 구분하기 위해서는  $D_k$ 는  $\Omega(N)$ 의 lower bound를 가져야 한다는 것을 보인다.  
먼저  $D_k \leq 4k^2$ 를 보이자.<sup>59</sup>  $k = 0$ 일 때는 자명하게 성립한다.  $k$ 일 때  $D_k \leq 4k^2$ 가 성립한다고 가정하면,  $k + 1$ 일 때는 다음과 같이 쓸 수 있다.

$$D_{k+1} = \sum_x \|U_{k+1} O_x |\psi_k^x\rangle - U_{k+1} |\psi_k\rangle\|^2 = \sum_x \|O_x |\psi_k^x\rangle - |\psi_k\rangle\|^2 = \sum_x \|O_x (|\psi_k^x\rangle - |\psi_k\rangle) + (O_x - I) |\psi_k\rangle\|^2.$$

Norm에 대한 성질  $\|b + c\|^2 \leq \|b\|^2 + 2\|b\|\|c\| + \|c\|^2$ 을 적용하면, 다음과 같은 upper bound를 얻는다.

- Eq. (1.37) : Property of norm ( $b \triangleq O_x (|\psi_k^x\rangle - |\psi_k\rangle)$  and  $c \triangleq (O_x - I) |\psi_k\rangle = -2 \langle x | \psi_k \rangle |x\rangle$ )
- Eq. (1.38) : Cauchy-Schwarz inequality
- Eq. (1.40) : Inductive hypothesis  $D_k \leq 4k^2$

$$D_{k+1} \leq \sum_x \left( \| |\psi_k^x\rangle - |\psi_k\rangle \|^2 + 4 \| |\psi_k^x\rangle - |\psi_k\rangle \| \| \langle x | \psi_k \rangle | + 4 | \langle \psi_k | x \rangle |^2 \right) \quad (1.37)$$

$$\leq D_k + 4 \left( \sum_x \| |\psi_k^x\rangle - |\psi_k\rangle \|^2 \right)^{1/2} \left( \sum_{x'} | \langle \psi_k | x' \rangle |^2 \right)^{1/2} + 4 \quad (1.38)$$

$$= D_k + 4 \sqrt{D_k} + 4 \quad (1.39)$$

$$\leq 4k^2 + 8k + 4 = 4(k + 1)^2. \quad (1.40)$$

이제,  $N$ 개의 item을 구분하기 위해  $D_k = \Omega(N)$ 이 necessary condition임을 보이자.

증명을 위해서 먼저,  $|\langle x | \psi_k^x \rangle|^2 \geq 1/2$ 가 모든  $x$ 에 대해서 성립한다고 가정하자. 즉, 이는 어떤 solution  $x$ 가 주어지더라도  $1/2$  이상의 확률로 solution을 구할 수 있다는 의미이다. (i.e.,  $N$ 개의 item을 구분할 수 있다) 가정으로부터, 다음을 얻을 수 있다.

$$\| |\psi_k^x\rangle - |x\rangle \|^2 = 2 - 2 | \langle x | \psi_k^x \rangle | \leq 2 - \sqrt{2}$$

따라서 solution  $|x\rangle$ 와  $k$ 번 oracle을 적용하여 얻은 state  $|\psi_k^x\rangle$ 의 차이를 error  $E_k$ 로 정의하면, 다음의 upper bound를 얻는다.<sup>60</sup>

$$E_k \triangleq \sum_x \| |\psi_k^x\rangle - |x\rangle \|^2 \leq (2 - \sqrt{2})N. \quad (1.41)$$

Solution  $|x\rangle$ 와 oracle을 적용하지 않았을 때의 state evolution  $|\psi_k\rangle$ 의 차이를  $F_k$ 로 정의하자.

$$F_k \triangleq \sum_x \| |x\rangle - |\psi_k\rangle \|^2$$

그렇다면,  $D_k$ 와  $F_k$ 에 대해서 다음과 같은 관계를 유도할 수 있다.

<sup>59</sup>by mathematical induction

<sup>60</sup> $N$ 개의 item이 있으므로 가능한  $x$ 의 개수는  $N$ 개이다.

- Eq. (1.42) : By definition of  $D_k$
- Eq. (1.43) : Property of norm ( $b \triangleq |\psi_k^x\rangle - |x\rangle$  and  $c \triangleq |x\rangle - |\psi_k\rangle$ )
- Eq. (1.44) : By definition of  $D_k$ ,  $F_k$
- Eq. (1.45) : Property of norm

$$D_k = \sum_x \| |\psi_k^x\rangle - |x\rangle + |x\rangle - |\psi_k\rangle \|^2 \quad (1.42)$$

$$\geq \sum_x \| |\psi_k^x\rangle - |x\rangle \|^2 - 2 \sum_x \| |\psi_k^x\rangle - |x\rangle \| \| |x\rangle - |\psi_k\rangle \| + \sum_x \| |x\rangle - |\psi_k\rangle \|^2 \quad (1.43)$$

$$= E_k + F_k - 2 \sum_x \| |\psi_k^x\rangle - |x\rangle \| \| |x\rangle - |\psi_k\rangle \| \quad (1.44)$$

$$\geq E_k + F_k - 2\sqrt{E_k F_k} \quad (1.45)$$

$$= \left( \sqrt{E_k} - \sqrt{F_k} \right)^2 \quad (1.46)$$

Cauchy-Schwarz inequality를 이용하면, 다음이 성립함을 보일 수 있고

$$F_k \geq 2N - 2\sqrt{N} \quad (1.47)$$

$E_k \leq (2 - \sqrt{2})N$ 임을 이용하면 (see Eq. (1.41)), 충분히 큰  $N$ 에 대해서  $c \gtrsim (\sqrt{2} - \sqrt{2 - \sqrt{2}})^2 \approx 0.42$ 보다 작을 때 다음이 성립함을 보일 수 있다.

$$D_k \geq cN$$

앞에서 우리가 보인 증명에 의하면,  $D_k \leq 4k^2$ 이므로 다음을 얻는다.

$$k \geq \sqrt{cN/4}$$

따라서 정리하면,  $N$ 개의 item을 구분할 때, success probability가 적어도  $1/2$  이상인 search algorithm의 optimal complexity는  $\Omega(\sqrt{N})$ 이다.  $\square$

# Chapter 2

## Introduction to Computational Complexity

### Lecture 16

#### 2.1 Introduction

8 Nov. 17:00

**Computational complexity**는 computational problem을 풀기 위해서 필요로하는 자원의 종류인 time, space의 요구량을 분석하는 연구이다. Computational complexity의 주요 연구는 어떤 문제를 풀기 위한 best algorithm이 요구하는 자원의 양이 input size에 대해 어떤 lower bound를 가지는 것을 증명하는 것이다. 이는 심지어 그 문제를 해결하는 알고리즘이 구체적으로 무엇인지 알지 못하더라도 논의될 수 있다.

*Strong Church-Turing thesis*는 모든 computational model<sup>1</sup> probabilistic Turing machine에서 Polynomial time으로 시뮬레이션 할 수 있다고 주장한다. 양자 컴퓨터가 주목받고 있는 이유 중 하나로는 양자 컴퓨터가 classical computer가 polynomial time에 효과적으로 해결할 수 없으리라고 여겨졌던 문제들을 효과적으로 해결하는 경우가 발생하고 있기 때문이다. 이러한 결과들은 Strong Church-Turing thesis에 대해 의문을 제기하게 만든다.

Computational complexity를 다룰 때 있어서 주의해야 할 사항은 어떤 문제를 해결하는데 exponential resource가 요구된다는 것을 "엄밀하게" 증명하는 것은 매우 까다롭다는 사실이다. (e.g., NP problem) 이는 quantum computer에 대해서도 중요한 문제 중 하나인데 만약 양자컴퓨터가 polynomial time에 해결할 수 있는 NP problem이 사실은 P class에 속한다는 것이 증명된다면 quantum computer로서 얻을 수 있는 계산적 이점이 사라지기 때문이다.

#### 2.2 The class NP: Reducibility and completeness

##### 2.2.1 P and NP problems

우리는 지금부터 computational complexity를 다루기 위해 "YES / NO"라는 답변을 하는 *decision problem*에 대해서만 집중하고자 한다. Output이 2가지라는 측면에서, 이러한 problem은 output<sup>1</sup> one-bit은 boolean function  $f : \{0, 1\}^* \rightarrow \{0, 1\}$ 로 나타낼 수 있다.

Turing machine<sup>1</sup> 어떤 language  $L \subset \{0, 1\}^*$ 에 속하는 모든 word를 decide할 수 있을 때, 그 machine이 함수  $f_L : \{0, 1\}^* \rightarrow \{0, 1\}$ ,  $f_L(x) = 1 \Leftrightarrow x \in L$ 를 계산한다고 한다.

**Definition 2.2.1.** 모든  $x \in \{0, 1\}^*$ 에 대해 polynomial time에  $f_L$ 을 계산할 수 있는 turing machine  $M$ 이 존재한다면, language  $L \subset \{0, 1\}^*$ 이 P class에 속한다고 한다.

$$x \in L \Leftrightarrow M(x) = 1, \quad x \notin L \Leftrightarrow M(x) = 0.$$

P는 직접적으로 주어진 문제를 해결하는 것과 관련 있다. 그러나 어떤 경우에는, 직접 문제를 푸는 것보다 적절한 hint가 주어졌을 때 해답을 검증하는 것이 더 효율적일 수 있다.<sup>1</sup>

"효율적으로 검증 가능한 문제"를 엄밀하게 다루기 위해서 먼저 NP class를 정의하자.

<sup>1</sup>efficiently verifiable solutions

**Definition 2.2.2.** 모든  $x \in \{0,1\}^*$ 에 대해 다음을 만족하는 polynomial time TM  $M$ 과 polynomial  $p : \mathbb{N} \rightarrow \mathbb{N}$ 가 존재한다면, language  $L \subset \{0,1\}^*$ 이 NP class에 속한다고 한다.

$$\begin{aligned} x \in L &\iff \exists u \in \{0,1\}^{p(|x|)} \text{ s.t. } M(x, u) = 1 \\ x \notin L &\iff \forall u \in \{0,1\}^{p(|x|)} \text{ s.t. } M(x, u) = 0 \end{aligned}$$

$x \in L$ 에 대해  $M(x, u) = 1$ 가 되게 하는  $u \in \{0,1\}^{p(|x|)}$  를 *witness*(or *certificate*)라고 부른다.

a.i.e.,  $M$ 이 polynomial time에  $x$ 가 solution인지 아닌지를 확인할 수 있게 만드는 hint  $u$

여기서 한 가지 주목할 점은 NP를 정의할 때, 그 문제를 풀기 위해서 요구되는 자원의 양이 얼마나 많이 필요한지는 정의하지 않았다는 점이다. NP는 단지 효율적으로 검증할 수 있는 문제들의 집합이다.

다음은 NP의 정의를 만족하는 decision problem의 예시들이다.

- Independent set problem: 그래프  $G$ 와  $k$ 가 주어졌을 때,  $k$ -size의 independent set이 존재하는지 확인하는 문제. → (*witness*) independent set이 존재한다면, 그 set의 원소들을 제공한다.
- Traveling salesman problem: 그래프  $G$ 와 각 edge의 가중치가 주어졌을 때, 가중치의 합이  $k$ 를 넘지 않으며 모든 노드를 정확히 한번씩만 방문하는 closed path가 존재하는지 확인하는 문제. → (*witness*) 최단 경로가 존재한다면, 그 경로를 제공한다.
- Subset sum problem:  $n$ 개의 정수  $A_1, \dots, A_n$ 과 정수  $T$ 가 주어졌을 때, 원소들의 합이  $T$ 가 되는 subset이 존재하는지 확인하는 문제. → (*witness*) subset이 존재한다면, 그 subset의 원소들을 제공한다.

이렇게 P, NP를 정의하면 각 complexity class간의 관계를 생각해볼 수 있다.

- $P \subset NP^2$  : 만약  $L \in P$ 가 TM  $N$ 을 사용하여 poly-time에 decide될 때, TM  $N$ 을 zero-polynomial witness  $p(x)$  (i.e., empty)와 함께 사용하여  $x$ 를 poly-time에 decide할 수 있으므로  $L \in NP$ 이다.
- $NP \subset EXP$  : 만약  $L \in NP$ 가 TM  $M$ , witness  $p()$ 를 사용하여 poly-time에 decide될 때, TM  $M$ 을 사용하여 모든 가능한 witness를 대입해보면서  $x$ 가 solution인지 아닌지 decide할 수 있으므로  $L \in EXP$ 이다. ( $2^{O(p(n))} \approx O(2^{n^c})$  time<sup>3</sup>)

Nondeterministic turing machine을 이용하면, NP를 다르게 정의할 수도 있다. NDTM은 매 step마다 2가지 서로 다른 transition function  $\delta_0, \delta_1$  중에서 하나를 선택할 수 있다. 모든  $x$ 에 대해 accept state에 도달하게 만드는 선택의 조합이 존재할 때,  $M(x) = 1$ 이며 그렇지 않을 경우  $M(x) = 0$ 이다. NDTM에서의 witness는 이러한 선택의 조합을 제공하는 것이다.

**Definition 2.2.3.** 모든  $x \in \{0,1\}^*$ 에 대해 다음을 만족하는 polynomial time NDTM  $M$ 과 polynomial  $p : \mathbb{N} \rightarrow \mathbb{N}$ 가 존재한다면, language  $L \subset \{0,1\}^*$ 이 NP class에 속한다고 한다.

$$\begin{aligned} x \in L &\iff M(x) = 1 \\ x \notin L &\iff M(x) = 0. \end{aligned}$$

## 2.2.2 Reducibility and NP-completeness

특정 class에 속하는 문제들중에서 가장 어려운 문제는 어떻게 정의할 수 있을까? Computational complexity에서는 이를 위해 reduction을 사용한다.

**Definition 2.2.4.** 다음을 만족하는 polynomial time computable function  $f : \{0,1\}^* \rightarrow \{0,1\}^*$ 이 존재한다면, language  $L \subset \{0,1\}^*$ 이  $L' \subset \{0,1\}^*$ 에 polynomial time reducible이라고 한다.

$$\forall x \in \{0,1\}^*, \quad x \in L \quad if \text{ and only if } \quad f(x) \in L'.$$

직관적으로, reduction을 이용한다면 language  $L$ 에 속하는 word  $x$ 를 decide하는 대신, poly-time에  $f(x)$ 를 계산하여 language  $L'$ 에 속하는  $f(x)$ 를 decide하는 문제로 바꿔서 생각할 수 있다. 따라서  $L$ 의 complexity는  $L'$ 의 complexity보다 복잡할 수 없다.<sup>4</sup>

<sup>2</sup>  $P \subset NP$ 인지는 아직 밝혀지지 않았다.

<sup>3</sup> Since  $O(p(n)) = O(n^c)$  for some constant  $c$

<sup>4</sup>  $L$ 을 푸는 문제를  $L'$ 을 푸는 문제로 바꾸는데 걸리는 복잡도가 polynomial이므로

**Definition 2.2.5.** NP에 속하는 어떤 문제도  $L'$ 로 polynomial time reduction될 때,  $L'$ 을 *NP-hard*라고 하며, NP-hard이면서 동시에 NP에 속하는 문제를 *NP-complete*이라고 한다.<sup>a</sup>

<sup>a</sup>NP-complete = NP-hard  $\cap$  NP

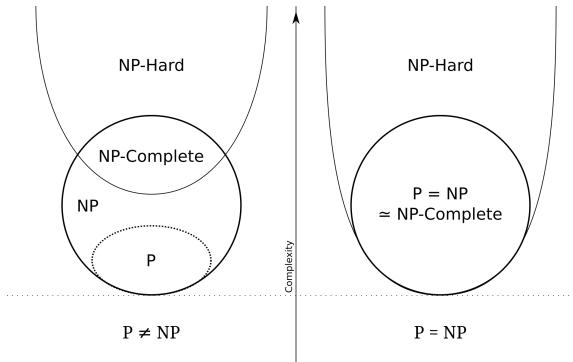


Figure 2.1: P and NP class [Np]

## Lecture 17

### 2.2.3 Boolean formula and Cook-Levin theorem

11 Nov. 17:00

앞에서 우리는 NP에 속하는 여러 문제들의 예시를 살펴보았다. 이제 NP-complete에 속하는 문제 3-SAT problem를 소개하려고 한다. 이를 위해서 먼저 boolean formula에 대해 알아보자.

Boolean formula는 boolean variable  $u_1, \dots, u_n$ 과 logical operator NOT( $\neg$ ), OR( $\vee$ ), AND( $\wedge$ )로 구성되는 수식이다. 어떤 boolean formula를 나타내는 변수를  $\varphi$ 라고 정의할 때, boolean formula를 이루는 각각의 변수에 어떤 값을 대입하였는지를  $z \in \{0, 1\}^n$ 으로 표현하자. 이때,  $\varphi(z) = 1$ 이 되도록 하는  $z$ 가 존재한다면  $\varphi$ 는 *satisfiable*이다. (e.g.,  $\varphi = u_1 \vee \neg u_2$  is satisfiable for  $z = (1, 0)$ )

특히 어떤 boolean formula가 *conjunctive normal form*이라는 의미는 변수들의 OR의 AND로 이루어진 수식이라는 의미이다. 예를 들어,

$$(u_1 \vee \bar{u}_2 \vee u_3) \wedge (u_2 \vee \bar{u}_3 \vee u_4) \wedge (\bar{u}_1 \vee u_3 \vee \bar{u}_4),$$

또는 일반화하여 나타내면 다음과 같다.

$$\bigwedge_i \left( \bigvee_j v_{i,j} \right)$$

이제 우리는 language SAT을 다음과 같이 정의할 수 있다.

**Definition 2.2.6.** Language **SAT**은 모든 가능한 satisfiable CNF formula들의 집합이다.<sup>a</sup> 즉, SAT problem은 주어진 boolean formula  $x$ 가 satisfiable한지 결정하는 문제이다.

<sup>a</sup>변수가 3개인 satisfiable CNF formula들의 집합을 3-SAT이라고 한다.

**Theorem 2.2.1 (Cook-Levin theorem).** 1. SAT is NP-complete, and 2. 3-SAT is NP-complete.

만약 주어진 boolean formula가 satisfiable하다면,  $\varphi(z) = 1$ 이 되도록 하는  $z$ 를 witness로 제공하면 문제를 풀 수 있기 때문에 SAT이 NP에 속한다는 것은 쉽게 증명할 수 있다. NP-hard에 속한다는 것을 증명하기 위해 NP에 속하는 모든 문제가 SAT로 reduction될 수 있음을 증명하면 된다.

이 수업에서는 SAT로의 reduction을 이용하는 대표적인 문제인 *Classical Hamiltonian problem*에 대해서 살펴보자 한다. 다음과 같은 3-local Hamiltonian을 가정하자.

$$H = \sum_{c=1}^m H_c(x_{c_1}, x_{c_2}, x_{c_3})$$

Hamiltonian problem은 ground state의 energy를 계산하는 문제이므로 다음과 같이 표현할 수 있다.

$$H_c(x_{c_1}, x_{c_2}, x_{c_3}) = 0, \quad x \in \{-1, 1\}$$

즉, boolean function의 값이 1이 되도록하는 3-SAT problem과 유사한 형태를 보이는 것을 알 수 있다.

$$f_c(u_{c_1}, u_{c_2}, u_{c_3}) = u_{c_1} \vee \bar{u}_{c_2} \vee u_{c_3} = 1, \quad u \in \{0, 1\}$$

이를 이용하면, SAT problem의 결과로부터 Hamiltonian problem의 결과를 쉽게 계산할 수 있다. 다음과 같이 Hamiltonian을 계산하면,  $f_c = 1$ 이 되도록하는 조합에 대해, Hamiltonian의 energy가 0이 되는 것을 알 수 있다. (e.g., (1, 0, 0) for  $f_c = 1$ , then  $H_c = 0$ .)

$$\frac{1 - u_{c_1}}{2} \frac{1 + u_{c_2}}{2} \frac{1 - u_{c_3}}{2}$$

## 2.3 Quantum complexity

### 2.3.1 Probabilistic algorithms

Quantum computer의 complexity를 정의하기 전에 먼저, 우리에게 익숙한 deterministic algorithm에서 벗어나 probabilistic algorithm에 대해 살펴보자. Probabilistic algorithm을 수행하는 probabilistic Turing machine은 NDTM처럼 2개의 서로 다른 transition function  $\delta_0, \delta_1$ 을 가지며, 각각의 step마다 두 함수 중 하나를 1/2의 확률로 선택하여 수행한다. 확률에 따라서 동작하기 때문에  $x \in L$ 이 input으로 주어지더라도 TM이 종료되었을 때 reject state에 도달하게 되면 reject이라고 답한다.

**Definition 2.3.1.** 모든  $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 polynomial time PTM  $M$ 이 존재한다면, language  $L \subset \{0, 1\}^*$ 이  $BPP^a$  class에 속한다고 한다.<sup>b</sup>

$$\Pr[M(x) = L(x)] \geq \frac{2}{3}$$

<sup>a</sup>bounded-error probabilistic polynomial

<sup>b</sup> $M(x)$ 의 결과가 틀리지 않을 확률이 2/3이상

**Definition 2.3.2.** 모든  $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 polynomial time TM  $M$ 과 polynomial  $p : \mathbb{N} \rightarrow \mathbb{N}$ 가 존재한다면, language  $L \subset \{0, 1\}^*$ 이  $BPP$  class에 속한다고 한다.<sup>a</sup>

$$\Pr_{r \in \{0, 1\}^{p(|x|)}}[M(x, r) = L(x)] \geq \frac{2}{3}$$

<sup>a</sup>이때  $r$ 은 TM  $M$ 에 random하게 추가되는 random bits.

PTM에 대한 NP problem은 다음과 같이 정의된다.

**Definition 2.3.3.** 모든  $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 polynomial time TM  $M$ 과 polynomial  $p : \mathbb{N} \rightarrow \mathbb{N}$ 가 존재한다면, language  $L \subset \{0, 1\}^*$ 이  $MA^a$  class에 속한다고 한다.

$$\begin{aligned} x \in L &\iff \exists u \in \{0, 1\}^{p(|x|)} \text{ s.t. } \Pr_r[M(x, u, r) = 1] \geq \frac{2}{3} \\ x \notin L &\iff \forall u \in \{0, 1\}^{p(|x|)} \text{ s.t. } \Pr_r[M(x, u, r) = 0] \geq \frac{2}{3}. \end{aligned}$$

<sup>a</sup>Merlin-Arthur

### 2.3.2 Quantum algorithms

이제 PTM에 대한 complexity class를 확장하여 Quantum computer에 대한 class를 정의해보자.

**Definition 2.3.4.** 모든  $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 uniform family of polynomial time quantum circuits  $\{Q_n : n \in \mathbb{N}\}$ 가 존재한다면, language  $L \subset \{0, 1\}^*$ 이  $BQP^a$  class에 속한다고 한다.

$$\Pr[Q_{|x|}(x) = L(x)] \geq \frac{2}{3}$$

<sup>a</sup>bounded-error quantum polynomial

BQP class는 Quantum circuit으로 문제를 풀었을 때, 오류가 발생할 확률이  $1/3$  이하인 문제들의 집합이다. BQP class에 속하면서 P class에 속하지 않을 것이라고 믿는 문제들이 바로 quantum algorithm으로 해결하기 효과적인 문제들이다. (e.g., prime factorization)  $Q(x)$ 를 실행한 뒤 첫 번째 qubit을 측정하여 그 값이 0인지 1인지를 기준으로 decidable problem을 해결할 수 있다.

**Definition 2.3.5.** 모든  $x \in \{0, 1\}^*$ 에 대해 다음을 만족하는 uniform family of polynomial time quantum circuits  $\{Q_n : n \in \mathbb{N}\}$ 과 quantum state  $|\psi\rangle$ 이 존재한다면, language  $L \subset \{0, 1\}^*$ 이 QMA class에 속한다고 한다.

$$\Pr [Q_{|x|}(x, |\psi\rangle) = L(x)] \geq \frac{2}{3}$$

<sup>a</sup>classical algorithm의 hint 역할

QMA에 속하는 대표적인 예시로  $k$ -local Hamiltonian problem<sup>o</sup> 있다. 다음과 같이 주어진  $k$ -local Hamiltonian<sup>o</sup> 있을 때,

$$H = \sum_{i=1}^L H_i, \quad (\|H_i\|_1 \leq 1)$$

Hamiltonian의 ground state energy가 특정한 threshold  $b$  이상인지, 또는  $a$  이하인지를 결정하는 문제이다. ( $b - a = \Omega(n^{-\alpha})$ )  $k$ -local Hamiltonian problem<sup>o</sup> QMA에 속하는 이유는, phase estimation algorithm을 이용하여 eigen-value를 추정하고 그 값이  $b \geq, a \leq$ 인지를 확인하는 것으로 문제를 해결할 수 있기 때문이다.

더 나아가,  $k$ -local Hamiltonian problem은 QMA-complete이기 때문에 SAT problem처럼 QMA에 속하는 모든 문제들은  $k$ -local Hamiltonian problem으로 reduction 될 수 있다.

### 2.3.3 BQP vs PSPACE

마지막으로, BQP class와 PSPACE class의 관계를 살펴보면서 complexity class간의 관계를 정리해보자.

**Definition 2.3.6.** TM<sup>o</sup> input size<sup>o</sup> 대해 polynomial space와 arbitrary time을 사용하여 문제를 해결할 수 있을 때, language  $L \subset \{0, 1\}^*$  PSPACE class에 속한다고 한다.

$BQP \neq PSPACE$  대해서는 아직 모르지만,  $BQP \subset PSPACE$  관계에 대해서는 증명할 수 있다.

$U = U_T \cdots U_1$ 인 unitary gate를 가정하자. ( $T = \text{poly}(n)$ )<sup>5</sup> 이때 각 gate는  $\{C(X), \text{single-qubit gate}\}$  중 하나이며, 따라서 각 gate는 최대 2개의 qubit에만 작용할 수 있다. Input qubit  $n$ 개에 추가로 중간 계산을 위해 ancilla qubit  $n$ 개를 사용한다고 가정하자. 그러면  $U$ 를 적용한 뒤, 첫 번째 qubit을 관측했을 때 그 결과가 0이 될 확률은 amplitude를 계산해서 얻을 수 있다. ( $|\mathbf{y}\rangle = |0\rangle |\mathbf{y}'\rangle$ )

$$\langle \mathbf{y} | U_T \cdots U_1 | x^n, 0^n \rangle = \sum_{x_1, \dots, x_{T-1} \in \{0, 1\}^n} \langle \mathbf{y} | U_T | x_{T-1} \rangle \cdots \langle x_2 | U_2 | x_1 \rangle \langle x_1 | U_1 | x^n, 0^n \rangle.$$

이 때,  $\Sigma$ 를 이루는 각각의 term을 계산하기 위해서 필요한 space는 poly-space<sup>o</sup>므로 확률을 구하는 문제는 PSPACE에 속한다. 따라서  $BQP \subset PSPACE$ 이다.□

**Note (Summary).**  $P \subset BPP \subset BQP \subset PSPACE$

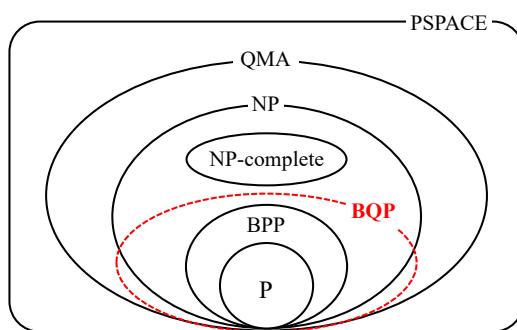


Figure 2.2: Computational complexity class

<sup>5</sup> $T$ 가 input qubit에 대해 polynomial이므로, 이 gate  $U$ 는 BQP에 속한다.

# Appendix

# Appendix A

## Useful Environments for the Note

### A.1 Useful Environment

We now see some common environment you'll need to complete your note.

**Definition A.1.1** (Natural number). We denote the set of *natural numbers* as  $\mathbb{N}$ .

**Lemma A.1.1** (Useful lemma). Given the axioms of *natural numbers*  $\mathbb{N}$ , we have

$$0 \neq 1.$$

**An obvious proof.** Obvious. ■

**Proposition A.1.1** (Useful proposition). From [Lemma A.1.1](#), we have

$$0 < 1.$$

**Exercise.** Prove that  $1 < 2$ .

**Answer.** We note the following.

**Note.** We have [Proposition A.1.1](#)! We can use it iteratively!

With the help of [Lemma A.1.1](#), this holds trivially. ⊗

**Example.** We now can have  $a < b$  for  $a < b$ !

**Proof.** Iteratively apply the exercise we did above. ⊗

**Remark.** We see that [Proposition A.1.1](#) is really powerful. We now give an immediate application of it.

**Theorem A.1.1** (Mass-energy equivalence). Given [Proposition A.1.1](#), we then have

$$E = mc^2.$$

**Proof.** The blank left for me is too small,<sup>a</sup> hence we put the proof in appendix. ■

<sup>a</sup>[https://en.wikipedia.org/wiki/Richard\\_Feynman](https://en.wikipedia.org/wiki/Richard_Feynman)

From [Theorem A.1.1](#), we then have the following.

**Corollary A.1.1** (Riemann hypothesis). The real part of every nontrivial zero of the Riemann zeta function is  $\frac{1}{2}$ , where the Riemann zeta function is just

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \frac{1}{1^s} + \frac{1}{2^s} + \frac{1}{3^s} + \dots$$

**Proof.** The proof should be trivial, we left it to you. ■

TODO  
mark

As previously seen. We see that [Lemma A.1.1](#) is really helpful in the proof!

### Internal Link

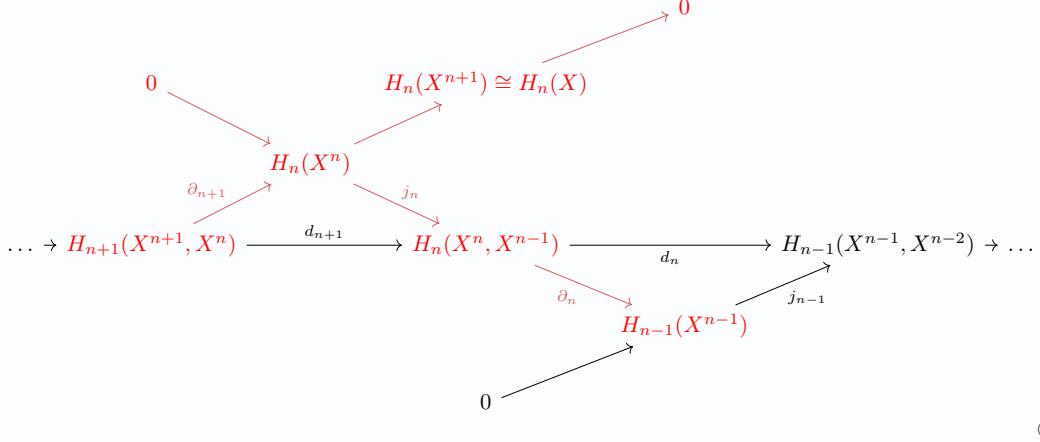
You should see all the common usages of internal links. Additionally, we can use citations as citation [\[NC01\]](#), which just link to the reference page!

## A.2 Commutative Diagram

We can use the package `tikz-cd` to draw some commutative diagram.

**Example.** The cellular homology agrees with singular homology.

**Proof.** The following commutative diagram shows everything.



⊗

## A.3 Fancy Stuffs

With this header, you can achieve some cool things. For example, we can have multiple definitions under a parent environment, while maintains the numbering of definition. This is achieved by `definition*` environment with `definition` inside. For example, we can have the following.

**Definition.** We have the following number system.

**Definition A.3.1 (Rational number).** The set of *rational number*, denote as  $\mathbb{Q}$ .

**Definition A.3.2 (Real number).** The set of *real number*, denote as  $\mathbb{R}$ .

**Definition A.3.3 (Complex number).** The set of *complex number*, denote as  $\mathbb{C}$ .

**Note.** And indeed, we can still reference them correctly. For instance, we can use [rational numbers](#) to define [real numbers](#) and then further use it to define [complex numbers](#).

Furthermore, we can completely control the name of our environments. We already saw we can name definition, lemma, proposition, corollary and theorem environment. In fact, we can also name remark, note, example and proof as follows.

**Example (Interesting Example).** We note that  $1 \neq 2!$

**Note (Important note).** As a consequence,  $2 \neq 3$  also.

**Remark (Easy observation).** We see that from here, we easily have the following theorem.

**Theorem A.3.1 (Lebesgue Differentiation Theorem).** Let  $f \in L^1$ , then

$$\lim_{r \rightarrow 0} \frac{1}{m(B(x, r))} \int_{B(x, r)} |f(y) - f(x)| dy = 0$$

for a.e.  $x$ .

**An obvious proof of Theorem A.3.1.** Obvious. ■

As we can see, specifically for the proof environment, we allow `autoref` and `hyperref`. One can actually allow all example, note and remark environment's name to use reference, but I think that is overkilled. But this can be achieved by modify the header in an obvious way.<sup>1</sup>

---

<sup>1</sup>This time I mean it!

# Bibliography

- [BCK15] Dominic W Berry, Andrew M Childs, and Robin Kothari. “Hamiltonian simulation with nearly optimal dependence on all parameters”. In: *arXiv preprint arXiv:1501.01715* (2015).
- [Chi+21] Andrew M Childs et al. “Theory of trotter error with commutator scaling”. In: *Physical Review X* 11.1 (2021), p. 011020.
- [Dha15] Sayandip Dhara. *Quantum Order Finding and Factorization*. July 2015. doi: [10.13140/RG.2.1.4954.9925](https://doi.org/10.13140/RG.2.1.4954.9925).
- [Haa19] Jeongwan Haah. “Product decomposition of periodic functions in quantum signal processing”. In: *Quantum* 3 (2019), p. 190.
- [HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. “Quantum algorithm for linear systems of equations”. In: *Physical review letters* 103.15 (2009), p. 150502.
- [NC01] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Vol. 2. Cambridge university press Cambridge, 2001.
- [Np ] *NP(complexity)*. URL: [https://en.wikipedia.org/wiki/NP\\_\(complexity\)](https://en.wikipedia.org/wiki/NP_(complexity)).