Quantum Computing

Vaughn Sohn

November 17, 2024

Contents

Chapter 1

Quantum Algorithms

Lecture 9

1.1 Introduction

7 Oct. 10:30

이번 챕터에서 우리는 *Quantum Algorithm*에 대해 다루고자한다. 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 다루고자하는 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 $f : \{0,1\} \to \{0,1\}$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 f에 대한 unitary operator U_f는 다음과 같이 정의된다.

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

$|x\rangle$는 f의 입력으로 사용되는 **oracle qubit**이며, $|y\rangle$는 1일때는 $f(x)$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_f를 한 번 호출함으로써 얻게 된 것이다! Classical computer에서 병렬연산은 서로 다른 컴퓨팅 자원을 사용할 뿐, f를 여러번 호출해야 하는 사실은 변하지 않지만, quantum computer에서의 병렬연산은 실제로 f를 한 번 호출하여 모든 연산을 수행할 수 있다.

이렇게 어떤 함수 f에 대응되는 unitary operator를 설계하는 것은 n-bit boolean function에 대해서 쉽게 일반화할 수 있다. n-bit boolean function $f : \{0,1\}^n \to \{0,1\}$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.

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

이 회로를 이용하면 한 번의 연산만으로 2^n개의 입력에 대한 출력을 동시에 얻을 수 있다. 그러나 한 가지 주의해야할 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 f이 **balance**인지 **constant**인지를 판단하는 문제를 해결한다.[1] Classical computer가 이 문제를 해결하기 위해서는 두 함숫값 $f(0), f(1)$을 비교하기 위해서 반드시 2번의 함수 호출이 필요하다. 그러나, 지금부터 우리는 quantum computer는 **단 한번**의 gate call만으로 이 문제를 해결할 수 있음을 보이고자한다.

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.[2]

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

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

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

[1] constant: $f(0) = f(1)$, balance: $f(0)\neq f(1)$

[2] $|f(x)\rangle - |1\oplus f(x)\rangle$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, f의 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 f가 balance인지 constant인지를 다음 규칙에 따라서 쉽게 판단할 수 있다. □

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

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 소개하고자한다.

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 $f : \{0,1\}^n \to \{0,1\}$이 **constant**인지 **balanced**인지를 판단하는 문제를 해결하는 알고리즘이다.[3] 마찬가지로 classical computer가 이 문제를 해결하기 위해서는 최대 $2^{n-1}+1$개의 function value를 비교해야하기 때문에 $2^{n-1}+1$번의 함수 호출을 필요로한다. 그러나, 지금부터 이런문제를 해결하려고 할 때도, quantum computer는 **단 한번**의 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 n개의 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^n개의 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)

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

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 *독립적*으로 적용한 결과이다.

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

이를 이용하여 H gate를 적용한 state를 표현하면, 다음과 같다.

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

[3] 여기서 말하는 balanced는 2^n개의 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 -1이라는 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 x가 어떤 값이 되던지간에 $x\cdot z$ [4]의 값이 0이 되기 때문에, 다음과 같이 표현할 수 있게 된다. $|0\rangle^{\otimes n}$의 amplitude의 square norm이 1이기 때문에, 100% 확률로 0^n을 측정할 수 있다.

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

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

$$\Big(\frac{1}{2^n}2^{n-1}|0\rangle^{\otimes n}\,|-\rangle\Big) + \Big(-\frac{1}{2^n}2^{n-1}|0\rangle^{\otimes n}\,|-\rangle\Big) = \boxed{0\,|0\rangle^{\otimes n}\,|-\rangle}$$

따라서, oracle qubit을 측정한 결과가 0^n인지 확인하여, f가 constant인지 balanced인지를 알 수 있다. □

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

1.3 Hamiltonian simulations

Hamiltonian simulation은 quantum computer가 고안된 핵심적인 이유 중 하나이다. 리처드 파인만의 말을 인용하자면 자연, 그중에서도 특히 미시세계는 고전역학을 따르지 않기때문에, 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 t에 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[a] Hamiltonian은 주어진 H가 n-qubit system에서 **최대 k개의 system**에 대해서만 동작하는 H_i들의 합으로 표현되는 Hamiltonian이다. (이때 L은 n에 대해 polynomial이다.)

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

[a] 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.

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

[4] $x\cdot z = x_1z_1 + x_2z_2 + \cdots + x_nz_n \mod 2$

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, $[H_j, H_k] = 0$[5]이 성립한다면 $e^{-iHt} = e^{-i\sum H_j t} = \prod e^{-iH_j t}$[6]가 성립하기 때문에, 각각의 H_i에 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_k에 대하여, 어떤 t에 대해서도 다음이 성립한다.[a]

$$e^{i(H_j+H_k)t} = \lim_{m\to\infty}\left(e^{iH_jt/m}e^{iH_kt/m}\right)^m$$

이는 H_j, H_k가 not-commute라 하더라도 항상 성립한다.

[a] 직관적으로, 주어진 time t 대신에, 매우 짧은 시간간격인 t/m에 대한 time evolution을 m번 반복하여 t에 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 $\tilde{U}(t) = e^{-iH_1t}e^{-iH_2t}$를 시간에 대해서 미분하면, 다음과 같이 전개할 수 있다. (with initial condition $\tilde{U}(0) = I$)

- Eq. (1.3): H_1, H_2는 matrix라서 순서를 바꿀 수 없기에, 다른 항을 더하여 항을 정리하고 다시 뺀다.
- Eq. (1.4): commutator를 사용하여 항을 정리한다.

$$\begin{aligned}
i\frac{d\tilde{U}(t)}{dt} &= H_1e^{-iH_1t}e^{-iH_2t} + e^{-iH_1t}H_2e^{-iH_2t} && (1.2)\\
&= (H_1+H_2)\,e^{-iH_1t}e^{-iH_2t} + e^{-iH_1t}H_2e^{-iH_2t} - H_2e^{-iH_1t}e^{-iH_2t} && (1.3)\\
&= He^{-iH_1t}e^{-iH_2t} + \left[e^{-iH_1t}, H_2\right]e^{-iH_2t} && (1.4)\\
&= H\tilde{U}(t) + \left[e^{-iH_1t}, H_2\right]e^{-iH_2t} && (1.5)
\end{aligned}$$

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

$$\tilde{U}(t) = U(t) - i\int_0^t e^{-iH(t-s)}\left[e^{-iH_1s}, H_2\right]e^{-iH_2s}ds \tag{1.6}$$

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 bound 되는지 분석해보자. Eq. (1.6)의 항을 이항시키면, 두 operator의 차이에 대한 norm의 upper bound를 구할 수 있다. [8]

$$\|\tilde{U}(t) - U(t)\| = \left\|-i\int_0^t e^{-iH(t-s)}\left[e^{-iH_1s}, H_2\right]e^{-iH_2s}ds\right\| \le \int_0^t \left\|\left[e^{-iH_1s}, H_2\right]\right\|ds \tag{1.7}$$

이때, norm은 다음과 같이 정의된다.

Definition 1.3.2.

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

이제 이 norm이 적절한 error rate ϵ에 의하여 bound됨을 보임으로써, 이 Hamiltonian simulation을 실제로 활용할 수 있을지 분석할 수 있다.

[5] $[H_j, H_k] = H_jH_k - H_kH_j$
[6] 지수함수의 성질이 성립하기 위해서 필요한 조건. 실수들은 항상 commutative하기 때문에 이 성질이 자명하게 정립되었던 것
[7] See https://en.wikipedia.org/wiki/Duhamel%27s_principle
[8] e^{-iH_it}는 unitary operator이기 때문에 norm을 변화시키지 않기 때문에 무시해도 된다.

이를 위하여 다음과 같은 새로운 연산자를 가정하자.

$$G(t) \triangleq [e^{-iH_1t}, H_2]e^{iH_1t} = e^{-iH_1t}H_2e^{iH_1t} - H_2, \text{ with } G(0) = 0 \tag{1.8}$$

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

$$i\frac{d}{dt}G(t) = e^{-iH_1t}\,[H_1, H_2]\,e^{iH_1t}$$

양변을 t에 대해서 적분하면 다음과 같다.

$$G(t) = G(0) - i\int_0^t e^{-iH_1s}\,[H_1, H_2]\,e^{iH_1s}ds$$

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

$$\|G(t)\| = \left\|-i\int_0^t e^{-iH_1s}\,[H_1, H_2]\,e^{iH_1s}ds\right\| \le \int_0^t \|\,[H_1, H_2]\,\|ds = t\|\,[H_1, H_2]\,\| \tag{1.9}$$

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

$$\left\|\left[e^{-iH_1t}, H_2\right]\right\| = \|G(t)\| \le t\,\|[H_1, H_2]\|$$

따라서 이를 Eq. (1.7)에 대입하면, 다음을 얻는다.

$$\|\tilde{U}(t) - U(t)\| \le \int_0^t \underbrace{\left\|\left[e^{-iH_1s}, H_2\right]\right\|}_{s\|[H_1,H_2]\|} ds \le \int_0^t s\,\|[H_1, H_2]\|\,ds$$

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

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

commutator를 전개한 뒤, triangle inequality와 norm의 성질[9]을 이용하면 다음과 같이 전개할 수 있다.

$$\begin{aligned}\|\tilde{U}(t) - U(t)\| \le \frac{t^2}{2}\,\|[H_1, H_2]\| &= \frac{t^2}{2}\,\|H_1H_2 + (-H_2H_1)\| \le \frac{t^2}{2}(\|H_1H_2\| + \|H_2H_1\|) \\ &\le \frac{t^2}{2}2\|H_1\|\|H_2\| \le t^2\max\{\|H_1\|, \|H_2\|\}^2\end{aligned}$$

정리하면, 다음과 같다.

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

따라서 시간간격 Δt에 대한 error는 다음과 같이 bound된다.

$$\left\|e^{-iH\Delta t} - \left(e^{-iH_1\Delta t}e^{-iH_2\Delta t}\right)\right\| \le (\Delta t)^2\max\{\|H_1\|, \|H_2\|\}^2 \tag{1.11}$$

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

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

이 bound를 사용하면, 우리가 원하는 target error rate ϵ을 달성하기 위해서 필요한 m의 값을 $m = O(t^2\epsilon^{-1})$으로 결정할 수 있다.[10]

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

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

[9] $\|AB\| \le \|A\|\|B\|$

[10] $\max\{\|H_1\|, \|H_2\|\}^2$ (i.e., $\|H_i\|$)의 값은 상수라서 무시하였다.

$\Delta t = t/m$로 가정하면 다음을 얻는다.

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

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

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

$m = O(Lt^2\epsilon^{-1})$로 설정하게되면 우리는 항상 target error rate ϵ을 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 t에 대한 quadratic overhead (i.e., error)를 가지고 simulation을 할 수 있음을 보여준다. 즉, simulation time이 증가하더라도 error rate은 polynomial하게 증가하게 된다. □

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

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

이를 이용하면, time t에 대하여 error는 다음과 같이 표현된다.

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

즉, $m = O(t^{3/2}\epsilon^{1/2})$으로 설정하면, target error rate ϵ을 달성할 수 있으며 이는 1st-Trotter formula를 사용한 simulation보다 더 효율적이다.[11] 2nd-Trotter 방법을 L개의 term을 갖는 Hamiltonian에 대하여 일반화하면 m은 다음과 같다.

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

더 나아가 pth order Trotter formula를 사용하면, m은 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.

[11] time t에 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.[a]

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

[a] $\mathbf{x} = x_0 \cdots x_{N-1}$, $\mathbf{y} = y_0 \cdots y_{N-1}$

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** $\{|0\rangle, \cdots, |N-1\rangle\}$에 대한 변환만이 정의되어 있다.

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

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

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

이때, y_k는 다음과 같다.[a]

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

[a] DFT와 똑같다.

즉, QFT가 하는일은 basis $\{|k\rangle\}$에 대한 linear combination으로 표현된 벡터 $|x\rangle$를 다른 basis $\{|j\rangle\}$에 대한 linear combination으로 나타내는 *basis transform*이다.[12]

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

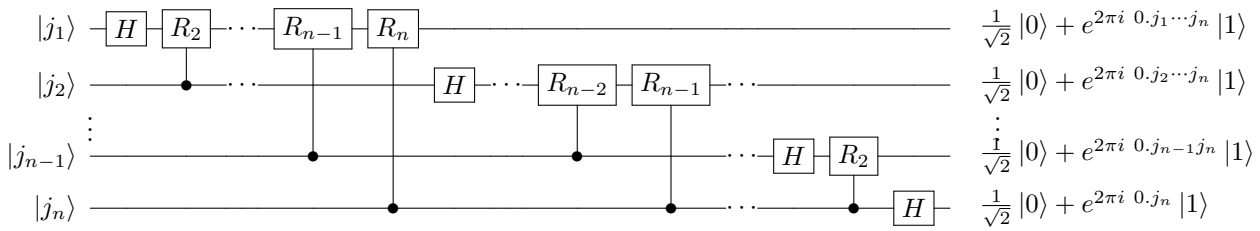


Figure 1.1: QFT circuit

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

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

또한, 소수도 다음과 같은 binary representation으로 표현할 수 있다.

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

[12] $\{|k\rangle\}$ basis에서의 amplitude는 x_i이며, $\{|j\rangle\}$ basis에서의 amplitude는 y_i이다.

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): $k = \sum k_l 2^{n-l}$이므로 $k/2^n = \sum k_l 2^{-l}$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. ($\sum_{k_1,k_2,\cdots,k_n\in\{0,1\}}$ 을 $\sum_{k_l\in\{0,1\}}$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.

$$|j\rangle \to \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n} |k\rangle \tag{1.12}$$

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

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

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

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

$$= \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \ldots j_n}|1\rangle\right)}{2^{n/2}} \tag{1.17}$$

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

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

본격적으로 quantum circuit을 만들기 위해서 rotation operator R_k를 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.

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

Fig. 1.1의 circuit의 연산을 단계별로 따라가보자.

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

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

2. apply Hadamard gate on the first qubit : 첫 번째 qubit; $|j_1\rangle$에 H를 적용하면, 다음을 얻는다. [13]

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

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

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

[13] 이때, $e^{i\pi} = e^{2i\pi\frac{1}{2}} = -1$ 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 i번째라면, R_i gate를 적용해야한다.

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

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.

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

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

6. n번째 qubit까지 이 과정을 수행하면 최종적으로 다음 state를 얻게된다.

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

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

따라서 우리는 1번의 QFT circuit을 수행하는 것으로 n개의 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 n개의 qubit에 대해 모두 필요한 gate의 개수는 다음과 같다.[14]

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

Some Remarks

- QFT는 exponential speed-up을 달성하는 것처럼 보이지만, 실제로는 값을 관측하게 되면 하나의 데이터에 대한 결과만 얻을 수 있다.
- 또한, input state $|j\rangle$를 준비하는 과정도 효과적이지 못하다.
- 따라서 QFT을 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.

Lecture 12

1.5.2 Performance

28 Oct. 10:30

Lecture 13

30 Oct. 10:30

[14] qubit의 순서를 바꾸기 위해 필요한 SWAP gate에 대한 gate complexity는 $O(n)$이므로 무시할 수 있다.

1.6 Applications of phase estimation

1.6.1 Order-finding algorithm

Order-finding

Uncomputation

The continued fraction expansion

Performance

1.6.2 Shor's algorithm: factoring

Lecture 14

1.7 Applications of the QFT

4 Nov. 10:30

1.7.1 Period-finding

1.7.2 Discrete logarithm

1.7.3 Hidden subgroup problem

1.8 Quantum search algorithms

1.8.1 Grover operator

1.8.2 Grover search algorithm

1.8.3 Performance

Lecture 15

1.8.4 Example: Classical circuit-SAT problem

6 Nov. 10:30

1.8.5 Amplitude amplification

1.9 Amplitude estimation algorithm (Quantum counting)

1.10 HHL (Harrow–Hassidim–Lloyd) algorithm

Lecture 16

1.11 Optimality of the quantum search algorithm

8 Nov. 17:00

Chapter 2

Introduction to Computational Complexity

Lecture 16

2.1 Introduction

8 Nov. 17:00

Lecture 17

2.2 The class NP: Reducibility and completeness

11 Nov. 17:00

2.2.1 P and NP problems

2.2.2 Reducibility and NP-completeness

2.2.3 Boolean formula and Cook-Levin theorem

2.3 Quantum complexity

2.3.1 Probabilistic algorithms

2.3.2 Quantum algorithms

2.3.3 BQP vs PSPACE

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,[a] hence we put the proof in appendix. ■

[a] 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

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

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

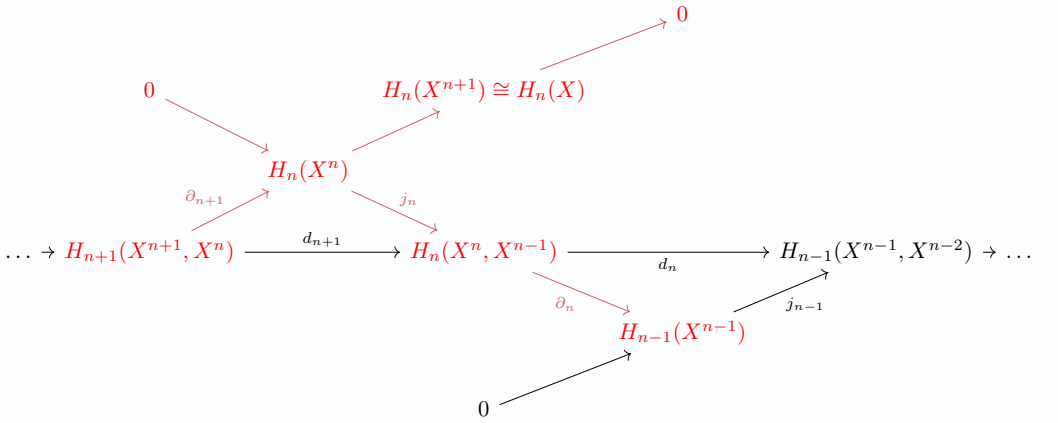
You should see all the common usages of internal links. Additionally, we can use citations as citation [NC01], which just link to the reference page!

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

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

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.[1]

[1] 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.

[Haa19] Jeongwan Haah. "Product decomposition of periodic functions in quantum signal processing". In: *Quantum* 3 (2019), p. 190.

[NC01] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Vol. 2. Cambridge university press Cambridge, 2001.