Introduction to Quantum Computing — Lecturer: Tongyang Li, scribed by Rui Yang
Fall 2023, Peking University — *Date: November 17, 2022*

Lecture 13

Quantum walk & Hamiltonian simulation

- Element distinctness
- Hamiltonian simulation

1 Quantum walk algorithm (Ambainis)

Consider the Hamming graph $H(N, M)$.
Vertices: M-tuple of values from $\{1, 2, \cdots, N\}$ (so there are N^M vertices)
Edges: Two vertices are connected if and only if they differ in exactly one coordinate.

At each vertex, we store the values of the function at the corresponding inputs. In other words, the vertex $(x_1, x_2, \ldots, x_M) \in \{1, 2, \ldots, N\}^M$ is represented by the state

$$|x_1, x_2, \cdots, x_M, f(x_1), \cdots, f(x_M)\rangle .$$

Consider the search problem on this Hamming graph.

Marked vertex: Those containing some $x \neq y$ with $f(x) = f(y)$

Note that given the stored function values, we can check whether we are at a marked vertex with no additional queries.

In the case where the elements are not all distinct (say $f(a) = f(b)$), the total number of marked vertices is at least:

# of vertices: both a and b appear exactly one among all X_i.

$$\binom{M}{2} \cdot 2 \cdot (N-2)^{M-2} = M(M-1) \cdot (N-2)^{M-2}$$

$\uparrow$ $(a \quad b)$
places $(b \quad a)$

$\Rightarrow$ The fraction of marked vertices is at least $\varepsilon \geqslant \frac{M(M-1)(N-2)^{M-2}}{N^M}$.

To analyze the quantum walk, we also need the eigenvalues of the stochastic matrix. The adjacency matrix of the Hamming graph $H(N.M)$ is $A = \sum_{i=1}^{M} (J - i)^{(i)}$, where J denotes the $n \times n$ all-1 matrix, and the superscript (i) indicates that this matrix acts on the i^{th} coordinate.

Formally: $A = (J-I)\otimes I \otimes \ldots \otimes I + I \otimes (J-I) \otimes I \otimes \cdots \otimes I + \cdots + I \otimes I \otimes \cdots \otimes I \otimes (J-I)$.

The eigenvalues of J are N and 0 :

$$\begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = N \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

$$\mathrm{rank}(J) = 1, \mathrm{tr}(J) = N$$

$$\begin{aligned} (J - NI)X = 0 &\Rightarrow X_1 + \cdots + X_N = NX_1 = NX_2 = \cdots = NX_N \\ &\Rightarrow X_1 = \cdots \cdot X_N \end{aligned}$$

N is an eigenvalue with single multiplicity.

This implies that $J - I$ has eigenvalues $N-1$ and -1,

$$\text{eigenvectors } |\psi_0\rangle = \frac{1}{\sqrt{N}}(1, \ldots, 1)^\top \text{ and } |\psi_1\rangle, \cdots, |\psi_{N-1}\rangle$$

A direct calculation shows that the eigenvectors of A are all

$$|\psi_{i_1}\rangle, \ldots, |\psi_{i_M}\rangle : i_1, \ldots, i_M \in \{0, 1, \cdots, N-1\} \text{ with eigenvalue}$$

$$(N-1)\cdot(\#\text{ of } i = 0) - 1 \cdot (M - (\#\text{ of } i = 0)) = N \cdot (\#\text{ of } i = 0) - M$$

The largest eigenvalue of A is $NM - M$, the second largest is $N(M-1) - M$.

Note that the Hamming graph is regular with deg $= M(N-1)$ (M: coordinates, N-1: each has N-1 alternative choices).

$\Rightarrow$ The stochastic matrix $A/M(N-1)$

$\Rightarrow$ The spectral gap is $\delta = \frac{N}{M(N-1).}$

Initialization. Need to query the black box.

In particular, to prepare an initial superposition over vertices of this graph takes M queries.

Update: Move from one vertex to an adjacent vertex takes two queries:

$$\begin{aligned} |x_1, \ldots, x_m, f(x_1), \ldots, f(x_M)\rangle &\xmapsto{U^\dagger} |x_1, x_2, \ldots, x_M, 0, f(x_2), \ldots, f(x_M)\rangle \\ &\longmapsto |x_1', x_2, \ldots, x_M, 0, f(x_2), \ldots, f(x_M)\rangle \\ &\xmapsto{U} |x_1', x_2, \ldots, x_M, f(x_1'), f(x_2), \cdots, f(x_M)\rangle \end{aligned}$$

To replace x by y in a particular coordinate, we use one query (inverse unitary) to erase $f(x)$ to 0 and another query to compute $f(y)$.

Overall cost: The total quantum query complexity is

$$\begin{aligned}M + 2\cdot O\left(\frac{1}{\sqrt{\delta\varepsilon}}\right) &= M + O\left(\sqrt{\frac{M(N-1)}{N}}\cdot\sqrt{\frac{N^M}{M(M-1)(N-2)^{M-2}}}\right)\\ &= M + O\left(\sqrt{M}\cdot\sqrt{\frac{N^2}{M^2}}\right)\\ &= M + O\left(\frac{N}{\sqrt{M}}\right).\\ M + \frac{N}{2\sqrt{M}} + \frac{N}{2\sqrt{M}} &\geqslant 3\cdot\sqrt[3]{M\cdot\frac{N}{2\sqrt{M}}\cdot\frac{N}{2\sqrt{M}}} = \Omega(N^{2/3})\end{aligned}$$

We can set $M = N^{2/3} \Rightarrow$ overall quantum query complexity becomes $O\left(N^{2/3}\right)$.

Note that this gives an algorithms using classical random walks, with complexity $M + O(\frac{N^2}{M})$. This is optimal when $M = \Theta(N)$.

$\Rightarrow$ Classical query complexity is $\Theta(N)$.

Time complexity. Classical: $O(n\log n)$: After each query, maintain a sorted list of function values. $O(\log n)$ overhead in each iteration.

Quantum: $\widetilde{O}\left(N^{2/3}\right)$. $\widetilde{O}(f) = O(f\cdot$ poly $(\log f)$.

Need poly$(\log N)$ = poly(n) gates to implement the 1-qubit and 2-qubit gates. But is otherwise efficient. Algorithms based on similar ideas have wide applications.

In general: We have a setup cost S, a cost U to update the state after one step of the walk, and a cost C to check whether a vertex is marked.

In Ambainis's algorithm for element distinctness:

$S = M$ to query M positions

$U = 2$ to remove one of items and add another

$C = 0$ since the function values for the subset are stored

In general, there is a quantum algorithm to solve such a problem with cost $S + \frac{1}{\sqrt{\delta\varepsilon}}(U + C)$.

Research paper. Magniez, Nayak, Roland, Santha. Search via quantum walk. SIAM on Joural on Computing 2011, arXiv: quant-ph/0608026. $S + \frac{1}{\sqrt{\varepsilon}}\left(\frac{1}{\sqrt{\delta}}U + C\right)$.

Furthermore, it can be modified to find a marked item when one exists.

Remark 1.1. The element distinctness problem can be extended to the k-distinctness problem, i.e., determine whether there exists pairwise different $X_1, \ldots, X_k \in [N]$ *s.t.* $f(x_1) = \cdots = f(x_k)$.

Quantum query: $O\left(N^{\frac{3}{4}-\frac{1}{4}\frac{1}{2^k-1}}\right)$ Belovs, FOCS 2012. arxiv: 1205.1534.

$\tilde{O}\left(N^{\frac{3}{4}-\frac{1}{4}\frac{1}{2^k-1}}\right)$ Jeffery and Zur, STOC 2023. arxiv: 2208.13492.

2 Hamiltonian simulation

So far:

Techniques	Problems	Algorithm
Hadamard transform (QFT over $\mathbb{Z}_2^{\otimes n}$)	Boolean f, determine constant or balanced $f(x) = x \oplus s$, find s	Deutsch-Jozsa Simon
$\downarrow$		
Phase estimation Period finding (QFT over $\mathbb{Z}_{2^d}$)	$U\|\psi\rangle = e^{i\theta}\|\psi\rangle$, find θ integer factorization	Inverse QFT Shor
Amplitude amplification (total function)	Unstructured search	Grover
$\downarrow$		
Quantum walks (graphs in general)	Element distinctness	Quantum walk on Hamming graph

In most of these problems, we study functions with quantum inputs.

How about something else. especially something genuinely quantum?

One of the most well-known rules in quantum mechanics:

the Schrodinger equation: $i\hbar\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle$

$H(t)$ is the Hamiltonian, and $\hbar$ is Planck's constant. For conveinence, it's typical to choose units which $\hbar = 1$.

Goal: Can we efficiently simulate the evolution of the Schrodinger equation? In other words, given an initial state $|\psi(0)\rangle$, determine $|\psi(t)\rangle$ for any time t.

Actually, this was the first application of quantum computing when Richard Feynman initiated the field in the 1980s.

For H independent of time, the solution of the Schrodinger equation

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

$e^{-iHt} = I + (-iHt) + \frac{(-iHt)^2}{2!} + \cdots$

H: We consider cases where H is a time-independent n-qubit Hamiltonian:

$H \in \mathbb{C}^{2^n \times 2^n}$ and $H^\dagger = H$. $N = 2^n$.

We say that an n-qubit Hamiltonian H can be efficiently simulated if, for any $t > 0, \varepsilon > 0$, there is a quantum circuit U consisting of $\text{poly}(n, t, \frac{1}{\varepsilon})$ gates, s.t. $\left\|U - e^{-iHt}\right\| \le \varepsilon$. WLOG $\|H\| = \text{poly}(n)$.

We would like to understand: When H can be efficiently simulated.

Basic facts: H Hermitian $(H^\dagger = H) \Rightarrow$ eigenvalues $\lambda_1, \ldots, \lambda_N$ are real.

e^{-iHt}: eigenvalues $e^{-i\lambda_1 t}, e^{-i\lambda_2 t}, \cdots, e^{-i\lambda_1 t}$ have unit modulus

Can prove: If $H = H^\dagger, e^{-iHt}e^{iHt} = I \Rightarrow e^{-iHt}$ is a unitary.

Fact: We cannot simulate arbitrary Hamiltonians efficiently, just as we cannot hope to efficiently implement arbitrary unitaries.

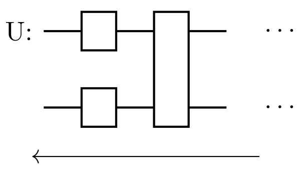
Instead: Study a few classes of Hamiltonians that can be efficiently simulated, and start from simple Hamiltonians, and combine them to more complicated ones.

Observation 1. H can be efficiently implemented if H acts nontrivially on a constant number of qubits.

Trivial by the Solovay-Kitaev Theorem: Any unitary evolution on a constant number of qubits can be approximated with error $\leqslant \varepsilon$ using poly $(\log 1/\varepsilon)$ one- and two-qubit gates.

Observation 2. If H can be efficiently simulated, so does cH for any $c = \mathrm{poly}(n)$. $e^{-iHt} = e^{-icH \cdot c^{-1} t}$.

- Simply take the original Hamiltonian and rescale t.
- This holds even if $c < 0$: just take $\dagger$ (conjugate transpose) for the quantum circuit.



$U^\dagger = \underline{\hspace{3cm}}$

$\longleftarrow X^\dagger = X, Y^\dagger = Y, Z^\dagger = Z$

$U = A_1 A_1 \cdots A_m$

$U^\dagger = A_m^\dagger A_{m-1}^\dagger \cdots A_2^\dagger A_1^\dagger$

Observation 3. If H can be efficiently simulated, and $U \in \mathbb{C}^{2^n \times 2^n}$ can be efficiently implemented, then $UHU^\dagger$ can be efficiently simulated.

$$\left(UHU^\dagger\right)^\dagger = \left(U^\dagger\right)^\dagger H^\dagger U^\dagger = UHU^\dagger.$$

This is because $e^{-iUHU^\dagger t} = U \cdot e^{-iHt} \cdot U^\dagger$.

Proof.

$$\left(UHU^\dagger\right)^k = \underbrace{UHU^\dagger UHU^\dagger \cdots UHU^\dagger}_{k \text{ times}} = UH^kU^\dagger.$$

$$e^{-iUHU^\dagger t} = I + \frac{-iUHU^\dagger t}{1!} + \frac{\left(-iUHU^\dagger t\right)^2}{2!} + \cdots = U\left(I + \frac{-iHt}{1!} + \frac{(-iHt)^2}{2!} + \cdots\right)U^\dagger = U \cdot e^{-iHt} \cdot U^\dagger.$$

■

Observation 4. If $H \in \mathbb{C}^{2^n \times 2^n}$ is diagonal in the computational basis, and diagonal element $d(k) = \langle k|H|k\rangle$ can be efficicutly computed for each $k \in [n]$, then H can be efficiently simulated.

$$H = \begin{bmatrix} d(1) & & & \\ & d(2) & & \\ & & \ddots & \\ & & & d(2^n) \end{bmatrix} \quad e^{-iHt} = \begin{bmatrix} e^{-id(1)t} & & \\ & \ddots & \\ & & e^{-id(2^n)t} \end{bmatrix}.$$

$$H^\dagger = H \Rightarrow \overline{d(k)} = d(k) \Rightarrow d(k) \in \mathbb{R}.$$

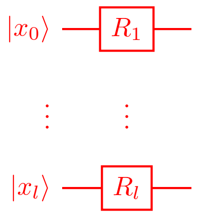
$$|k,0\rangle \overset{\text{compute}}{\overset{d}{\longmapsto}} |k,d(k)\rangle \longmapsto e^{-itd(k)}|k,d(k)\rangle \overset{\text{uncompute}}{\overset{d}{\longmapsto}} e^{-itd(k)}|k,0\rangle = e^{-iHt}|k\rangle|0\rangle \quad \forall k \in [2^n]$$

Because $|k\rangle, k \in [2^n]$ is complete, this process simulates H for any input $|\psi(0)\rangle$ for time t.

For the step $|k, d(k)\rangle \longmapsto e^{-itd(k)}|k, d(k)\rangle$:

$d(k)$ can be regarded as a binary number. Precision $\leqslant \varepsilon$ not important $\Rightarrow$ at most $\lceil \log_2 1/\varepsilon \rceil$ bits.

A series of R_k, where $R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{pmatrix}$, $\quad x = 0, x_1, \ldots, x_l,\ l \leqslant \lceil \log_2 1/\varepsilon \rceil$



output state $= e^{-i2\pi x}\,|x_1\rangle \cdots |x_l\rangle$

Remark 2.1. Observation 3 and 4 together give a way to simulate any Hamiltonian that can be efficiently diagonalized, and whose eigenvalues can be efficiently computed.