CPSC 536W (Topics in Quantum Computation) Notes

Rio Weil

This document was typeset on April 16, 2024

Introduction:
This is a set of lecture notes taken from UBC's CPSC 536W (Topics in Quantum Computation - Quantum Algorithms) course, taught by Dr. Daochen Wang. The course covers algorithms by Simon/Shor/Grover, quantum walks, polynomial/adversary/polynomial methods, quantum query complexity, quantum signal processing, quantum communication complexity, non-local games, and dequantization of quantum algorithms. The primary course reference is Quantum Algorithms (lecture notes) by Andrew Childs. If any errors are found in the notes, feel free to email me at ryoheiweil@phas.ubc.ca.

Contents

1 Classical Query Complexity

Course Logistics

The instructor for the course is Daochen Wang. The TA for the course is Xingyu Zhou. Drop deadline for the course is Jan$\dot{2}$2nd. Assessment is based on 4 homework assignments (the first of which is due on the drop deadline). Office hours are Friday 2-3pm in ICICS X553.

Motivation

Quantum computing - using quantum mechanics to solve computational problems.

Wrong popsci explanation - n qubits can be in 2^n different states of the classical bits (e.g. the 8 states $000, 001, 010, 100, 011, 101, 110, 111$ for $n = 3$) at the same time. A quantum computer can "try all these at the same time".

Why is this wrong? One way to see it is if I have n randombits/rbits, then the possible states of such bits are also one of the 2^n states. We will see later on that randomized computing is a subset of QC and in some computational models that QC is more powerful.

In particular, in this course (or a majority of it) we will be looking at the query model of quantum computation. This is not the same as the usual computation model considered, namely the Turing model. In some sense, the Turing model is a real-world model while the query model is an idealized model.

Why the query model? There are two reasons:

1. It is possible to (mathematically rigorously) prove quantum speedups in this model, i.e. that a quantum computation takes less resources compared to a classical computation.
2. In the Turing model, proving classical lower bounds is notoriously difficult, but this is necessary for proving quantum speedup. In fact in the Turing model there is no proof of QC speedup. However - in the query model has translated (historically) to "apparent" speedups in the Turing model; that is, it beats any known classical algorithm (Example: Shor's factoring algorithm for factoring an n-digit number in time $O(n^2)$ vs. best known classical algorithm - generalized number sieve - which takes $O(2^{n^{1/3}})$. Historical note that Shor devised this algorithm under the context of thinking about a problem in the query model, namely Simon's problem).

Introduction to the Query model

Denote by $\mathbb{N}$ the set of positive integers. "Alphabet" is defined to be a finite nonempty set.

Let $n, m \in \mathbb{N}$ and $\Sigma = \{0, 1, \ldots, m-1\}$ (often $m = 2$, i.e. bits) and Γ be an alphabet. The main character of query complexity if a function $f : D \subseteq \Sigma^n \to \Gamma$, where Γ is WLOG often taken to be $\Gamma = \{0, 1\}$.

The main question of query complexity is as follows; given $x \in D$, how many positions of x do we need to read to compute $f(x)$? To be concrete, let's consider an example. Take:

$$\begin{array}{rccc} f & : & \{0,1\}^3 & \longrightarrow & \{0,1\} \\ & & (x_1, x_2, x_3) & \longmapsto & (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3) \end{array} \tag{1.1}$$

How many bits do we need to read to compute this function? We only need to read two bits; if we read x_1, its either 0 or 1. If $x_1 = 0$, $x_1 \wedge x_2 = 0$ and $\neg x_1 \wedge x_3 = x_3$ so we read x_3. If $x_1 = 1$, then $\neg x_1 \wedge x_3 = 0$ and $x_1 \wedge x_2 = x_2$ so we read x_2.

There are two types of classical query complexity; deterministic $D(f)$ and randomized $R(f)$. Here we have shown that $D(f) \leq 2$ (and in fact it is exactly equal). There is also a quantum complexity $Q(f)$. A quantum speedup in the query model is defined as $R(f) > Q(f)$.

Another example is the function:

$$\begin{array}{rccc} OR_n & : & \{0,1\}^n & \longrightarrow & \{0,1\} \\ & & (x_1, x_2, \ldots, x_n) & \longmapsto & x_1 \vee x_2 \vee \ldots \vee x_n \end{array} \tag{1.2}$$

Some facts about this function:

1. $D(OR_n) \geq n$
2. $R(OR_n) \geq \Omega(n)$
3. $Q(OR_n) \leq O(\sqrt{n})$ (a la Grover).

A review of asymptotic notation below:

Definition: Asymptotic notation

1. Take $g : \mathbb{N} \to \mathbb{R}, h : \mathbb{N} \to \mathbb{R}$. Then $g(n) = O(h(n))$ if $\exists x > 0, n_0 \in \mathbb{N}$ such that $\forall n \geq n_0$ $g(n) \leq ch(n)$.
2. $g(n) = \Omega(h(n))$ is the same, with $\geq$ instead.
3. $g(n) = \Theta(h(n))$ if $g(n) = O(h(n))$ and $g(n) = \Omega(h(n))$.

Definition: Deterministic decision tree/query algorithm

A deterministic decision tree (DDT) is an m-ary tree T with a unique vertex labelled as "root" with additional data:

1. Each leaf of T is labelled by an element in Γ.
2. Each non-leaf (internal) vertex of T is labelled by an element in $[n] = \{1, 2, \ldots, n\}$.
3. For all non-leaf (internal) vertices v, the m edges between v and its children are each labelled by a unique element in $\{0, 1, \ldots, m-1\}$.

This is the mathematical object we work with. How does it do computation? In the way you expect; start at the root and follow the edges based on the labels. Mathematically:

Definition: Deterministic query computation

Let T be a DDT and $x \in D$. We define $T(x) \in \Gamma$ by the following procedure:

1. Set v_{current} to be the root vertex.
2. Repeat the following until the label of a leaf is output:
   (i) If v_{current} is a leaf, then output its label.
   (ii) Otherwise, let $i \in [n]$ be the label of v_{current}, and let v be the child of v_{current} such that the edge (v, v_{current}) is labelled by x_i. Set $v_{\text{current}} = v$.

We say that T computes f if and only if $\forall x \in D, T(x) = f(x)$.

Note: the $\forall$ in the above definition is very important! We are working in the "worst case". It has to work for all possible inputs.

Let's work through the above example with $f(x) = (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)$.

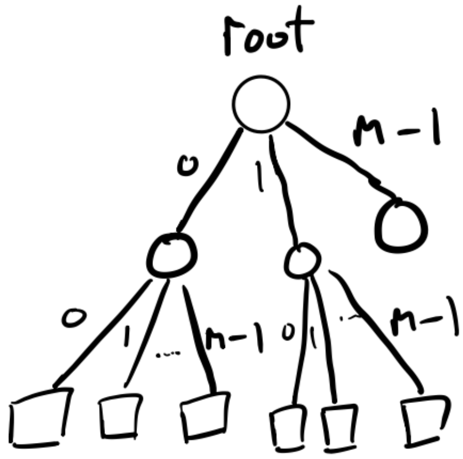

Figure 1.1: *m*-ary tree T

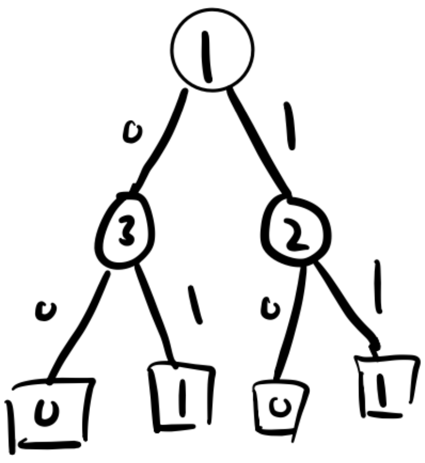

Figure 1.2: Minimal depth decision tree that computes f

Definition: Deterministic query complexity

Given a DDT T, its depth, denote $\text{depth}(T)$ as the maximum length of a root-to-leaf path. Then:

$$D(f) := \min_{\text{DDT } T \text{ computing } f} \text{depth}(T) \tag{1.3}$$

Definition: Randomized decision tree/query algorithm

A randomized decision tree (RDT) is a probability distribution over DDTs $\tau = \{(p_1, T_1), \ldots (p_n, T_k)\}$ with $p_i \in [0,1]$ wiht $\sum_i p_i = 1$.

Definition: Randomized query computation

Given $x \in D$ and an RDT $\mathcal{T}$, with $\tau(x)$ for the random variable on Γ defined by $\forall i \in \Gamma \Pr[\tau(x) = i] = \Pr[T(x) = i | T \leftarrow \tau] = \sum_{j \in [k, T_j(x) = i]} p_j$. Let $\epsilon \in (0, 1/2)$. We say that τ computes f with bounded error ϵ if the following holds:

$$\forall x \in D, \Pr[\tau(x) = f(x)] \geq 1 - \epsilon = \sum_{j \in [k], T_j(x) = f(x)} p_j \tag{1.4}$$

Definition: Randomized query complexity

Given a RDT τ, its depth $\text{depth}(\tau) = \max_{j \in [k], p_j > 0}(\text{depth}(T_j))$. Then, let $\epsilon \in (0, 1/2)$. Then:

$$R_\epsilon(f) = \min_{\tau \text{ computes } f \text{ with bounded error } \epsilon} \text{depth}(\tau) \tag{1.5}$$

It is standard to write $R(f) = R_{1/3}(f)$.

2 OR, Dirac Notation

Classical Query complexity of OR

Proposition

$D(OR_n) = n$.

Proof. $n \geq D(OR_n)$ is obvious (in fact is obvious for any function on n-bits that n is an upper bound). Since $D(f)$ is a minimum over decision trees computing f, simply take the tree which checks every bit in the input, outputting 1 if any of the bits are 1, and outputting 0 if all of the bits are 0 (the worst case/depth of the tree is checking all of the bits and finding all are zero).

$n \leq D(OR_n)$ is not quite as simple. We develop an "adversary argument" for this purpose.

In general, we imagine a two-player game between the Algorithm and an Adversary based on a Boolean function f. The game is played as follows:

1. The Adversary maintains a bag of strings S containing the domain, usually $\{0,1\}^n$.

2. At each round, the Algorithm is allowed to query a new bit, say, the ith, and te Adversary answers wit $x_i \in \{y_i, \exists y \in S\}$, i.e. the Adversary chooses an output that describes an element in S. The set S is updated to remove all y such that $y_i \neq x_i$.

3. The game ends if $f(y)$ takes the same value for all $y \in S$

The length of any such game is a lower bound on $D(f)$. Informally - if we fix a decision tree for f, the Algorithm in the game asks the Adversaries queries following the decision tree, using the responses to navigate down the tree. S corresponds to all strings that would have lead the algorithm to the current node, and the game ends if all leaf nodes under the current node have the same level (because the Algorithm can output the value of f and does not need to query any new bits).The game proceeding to k rounds therefore implies the depth of the decision tree is at least k.

Having discussed the technique, we describe the procedure for OR_n. Fix an algorithm/decision tree. Consider the adversary that returns every query with 0. After any $n-1$ queries, there is some index $i \in [n]$ not queried yet (suppose WLOG $i = 1$). At this point, the Adversary's bag contains at least 2 strings evaluating to different values, namely 0^n and 10^{n-1}. Thus, the game proceeds to the nth round, giving $D(OR_n) \geq n$. □

Proposition

$R_\epsilon(OR_n) \geq (1-2\epsilon)n$

Proof. Suppose $\exists$ RDT τ of depth $k \in \{1, 2, \ldots, n\}$ that computes OR_n with bounded error ϵ. Then, $\forall x \in \{0,1\}^n$, we have:

$$Pr^T_{T \leftarrow \tau}[T(x) = OR_n(x)] \geq 1 - \epsilon \tag{2.1}$$

Then the LHS is equivalent to:

$$\sum_{TDDT} \mathbb{1}[T(x) = OR_n(x)] Pr[T \leftarrow \tau] \geq 1 - \epsilon \tag{2.2}$$

Consider a probability distribution on $\{0,1\}^n$, then take $\mathbb{E}_{x \leftarrow \mu}[(*)]$. Using the linearity of expectation and hitting the indicator function with the expectyation, we get:

$$\sum_{TDDT} Pr_{x \leftarrow \mu}[T(x) = OR_n(x)] Pr[T \leftarrow \tau] \geq 1 - \epsilon \tag{2.3}$$

Now, there exists T^* a DDT such that:

$$Pr_{x \leftarrow \mu}[T^*(x) = OR_n(x)] \geq 1 - \epsilon \tag{2.4}$$

There are many ways to see this. One way; suppose for every T^* did not hold. Then, the $Pr_{x \leftarrow \mu}[T(x) = OR_n(x)]$ is less than $1 - \epsilon$, which violates the equality in Eq. (2.3) because $Pr[T \leftarrow \tau] < 1$ (And sums to one taken over the entire $TDDT$). This is a fairly standard technique when proving bounds with randomized algorithms.

Now, lets define μ as follows:

$$\mu(x) = \begin{cases} \frac{1}{2n} & \text{if } x \text{ is of Hamming weight 1.} \\ \frac{1}{2} & \text{if } x = 0^n \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

Note the Hamming weight is just the number of ones in the bit string, i.e. for $x \in \{0,1\}^n$ we have $|x| = \sum_{i=1}^n x_i$.

Suppose that when the decision tree sees all zeroes, i.e. x is the all zero string, it does the reasonable thing and outputs 0 (Checking the other case where it outputs 1 will be your homework!). This has probability $Pr_{x \leftarrow \mu}[T^*(x) = OR_n(x)] = \frac{1}{2} \cdot 1 = \frac{1}{2}$. Next, how many bit strings are of Hamming weight 1 such that its $i_1 \dots i_k$ are all zero. This happens to be k bit strings (i_{k+1} to i_n could be 1). So we have $n-k$ paths where the DDT spits out the wrong thing (i.e. spits out 0) while the OR_n is 1, and k where it gives the correct answer. The probability of a Hamming weight 1 bitstring is $\frac{1}{2n}$. So, we get:

$$Pr_{x \leftarrow \mu}[T^*(x) = OR_n(x)] = \frac{1}{2} \cdot 1 + \frac{k}{2n} \geq 1 - \epsilon \tag{2.6}$$

Then:

$$\frac{k}{2n} \geq \frac{1}{2} - \epsilon \implies k \geq (1 - 2\epsilon)n \tag{2.7}$$

which proves the claim. □

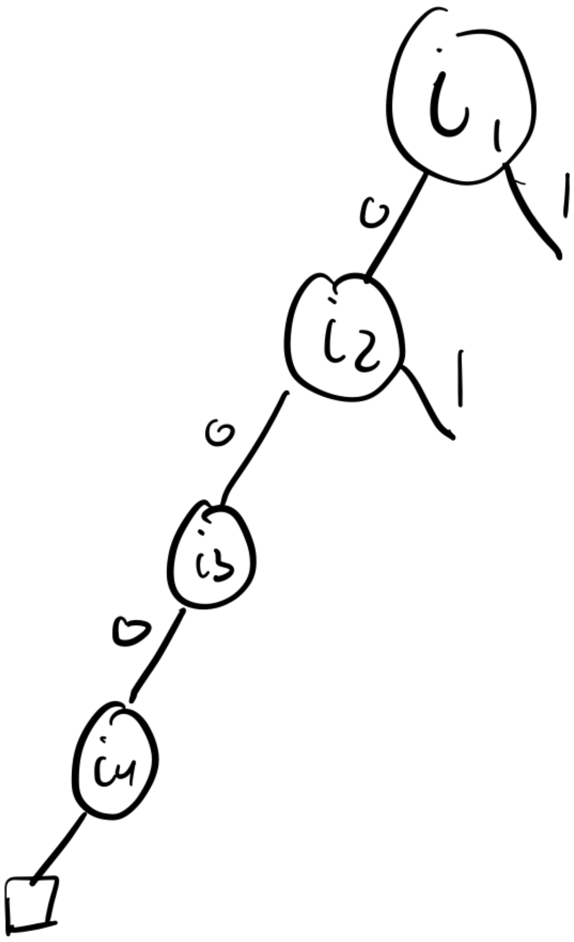

Figure 2.1: Visualization of above argument

Two notes:

1. This seemingly magical approach where we fix a decision tree and then choosing a distribution usually works for proving lower bounds. The answer actually turns out to be yes. The steps at the

beginning of this proof can always be used to give the optimal lower bound on randomized query complexy - this is Yao's principle.

2. The $OR_n : \{0,1\}^n \to \{0,1\}$ function has been considered. We can consider a restriction of the domain $OR_n^{0,1} : \{0^n\} \cup \left\{\text{Hamming weight} | \left\{strings\right\}\right\} \to \{0,1\}$ - note that $R_\epsilon(OR_n^{0,1}) \geq (1-2\epsilon)n$ still holds.

Dirac notation

For $n \in \mathbb{N}$, an d-dimensional quantum state is a unit vector $v \in \mathbb{C}^d$, written as $|v\rangle$ ("ket v"), where unit refers to the vector having an $l2$-norm of 1, i.e. $\sum_{i=1}^d |v_i|^2 = 1$.

We can then define $\langle v|$ ("bra v") as the complex conjugate transpose of v.

$\langle u|v\rangle$ is just the inner product of u and v, $\langle u|v\rangle = \sum_{i=1}^d u_i^* v_i = \langle u, v\rangle$. This is the "bracket"!

We can also put things together as $|u\rangle\langle v|$, which is a matrix/outer product:

$$|v\rangle\langle u| = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \begin{pmatrix} u_1^* & u_2^* & u_3^* \end{pmatrix} = \begin{pmatrix} v_1u_1^* & v_1u_2^* & v_1u_3^* \\ v_2u_1^* & v_2u_2^* & v_2u_3^* \\ v_3u_1^* & v_3u_2^* & v_3u_3^* \end{pmatrix} \tag{2.8}$$

Given $|v_1\rangle \in \mathbb{C}^{d_1}, |v_2\rangle \in \mathbb{C}^{d_2}$, we denote $|v_1\rangle|v_2\rangle := |v_1\rangle \otimes |v_n\rangle \in \mathbb{C}^{d_1 \cdot d_2}$. This is known as the tensor, or Kronecker product. Explicitly:

$$|v\rangle \otimes |u\rangle = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \otimes \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} v_1u_1 \\ v_1u_2 \\ v_1u_3 \\ v_2u_1 \\ v_2u_2 \\ v_2u_3 \end{pmatrix}. \tag{2.9}$$

The "computational basis" of $\mathbb{C}^d$ is the basis (shown below for $d = 4$, but easily generalizes):

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, |2\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, |3\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.10}$$

so $|v\rangle = \sum_{i=1}^d v_i|i\rangle$.

An n-qubit quantum state v is a vector in $\mathbb{C}^{2n}$. Then, any such $|v\rangle$ can be expanded as follows:

$$|v\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x_1\rangle|x_2\rangle|x_3\rangle \dots |x_n\rangle \tag{2.11}$$

where $\alpha_x \in \mathbb{C}$. E.g. for $n = 3$, we have:

$$|0\rangle|1\rangle|1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.12}$$

We can also take tensor products of matrices:

$$\begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{pmatrix} \otimes V = \begin{pmatrix} u_{11}V & u_{12}V \\ u_{21}V & u_{22}V \end{pmatrix}. \tag{2.13}$$

Definition: Projective measurement

Let Γ be an alphabet. An Γ-outcome projective measurement on $\mathbb{C}^d$ is a set of $|\Gamma|$ matrices $\Pi_1, \Pi_2, \ldots \Pi_{|\Gamma|} \in \mathbb{C}^{d\times d}$ such that the $\{\Pi_i\}_i$ are a set of orthogonal projectors, i.e. $\forall i, j$ we have $\Pi_i\Pi_j = \delta_{ij}\Pi_i$ and $\sum_{i=1}^{|\Gamma|} \Pi_i = \mathbb{I}_d$.

Definition: Performing a measurement

Let $\mathcal{M}$ be a Γ-outcome projective measurement on $\mathbb{C}^d$, i.e. $\mathcal{M} = \left\{\Pi_1, \ldots, \Pi_{|\Gamma|}\right\}$ and let $|\psi\rangle \in \mathbb{C}^d$. Then to measure $\mathcal{M}$ produces the following:

1. Output $i \in [|\Gamma|]$ with probability:

$$p(i) = \left\|\Pi_i|\psi\rangle\right\|^2 = \langle\psi|\Pi_i|\psi\rangle \tag{2.14}$$

2. The state becomes:

$$|\psi\rangle = \frac{\Pi_i|\psi\rangle}{\left\|\Pi_i|\psi\rangle\right\|} \tag{2.15}$$

where the denominator appears so it remains normalized.

Definition: Computational Basis measurement

The computational basis measurement on $\mathbb{C}^d$ is defined by the following $[d]$-outcome measurement $|0\rangle\langle 0|, |1\rangle\langle 1|, \ldots |d-1\rangle\langle d-1|$.

An example of projecting into a subspace

Consider the Hilbert space $\mathbb{C}^2 \otimes \mathbb{C}^2 \cong \mathbb{C}^4$ of two qubits. Suppose we want to measure the first qubit, but not the second qubit. The projectors corresponding to this measurement are:

$$\Pi_0 = |0\rangle\langle 0| \otimes \mathbb{I}_2, \quad \Pi_1 = |1\rangle\langle 1| \otimes \mathbb{I}_2. \tag{2.16}$$

which have the action of projecting the first qubit onto one of the two computational basis states, and doing nothing to the second qubit. We can verify that they obey the conditions for being a set of orthogonal projectors:

$$\Pi_0^2 = (|0\rangle\langle 0|)^2 \otimes (\mathbb{I}_2)^2 = |0\rangle\langle 0| \otimes \mathbb{I}_2 = \Pi_0 \tag{2.17}$$

$$\Pi_1^2 = (|1\rangle\langle 1|)^2 \otimes (\mathbb{I}_2)^2 = |1\rangle\langle 1| \otimes \mathbb{I}_2 = \Pi_1 \tag{2.18}$$

$$\Pi_0\Pi_1 = (|0\rangle\langle 0|1\rangle\langle 1|) \otimes \mathbb{I}_2 = 0 \tag{2.19}$$

$$\Pi_0 + \Pi_1 = (|0\rangle\langle 0| + |1\rangle\langle 1|) \otimes \mathbb{I}_2 = \mathbb{I}_2 \otimes \mathbb{I}_2 = \mathbb{I}_4 \tag{2.20}$$

but notably, the number of projectors in the set is strictly less than the dimension of the underlying Hilbert space, i.e. this corresponds to a measurement of a subspace.

Quantum Query Algorithm

Definition: Quantum query algorithm

A quantum query algorithm of depth $d \in \mathbb{N}$ is specified by the following data:

1. $w \in \mathbb{N}$
2. $d+1$ unitary matrices $U_0, U_1, \dots U_d$ acting on $\mathbb{C}^n \otimes \mathbb{C}^m \otimes \mathbb{C}^w$.
3. A Γ-outcome projective measurement on $\mathbb{C}^{nmw}$.

Definition: Quantum oracle

For $x \in \{0, 1, \dots, m-1\}^n$, the "quantum oracle" of x is the unitary matrix $O_x \in \mathbb{C}^{nm \times nm}$ defined by $O_x|i\rangle|j\rangle = |i\rangle|j + x_i \mod m\rangle$, for all $i \in \{0, \dots, n-1\}$ and $j \in \{0, 1, \dots m-1\}$.
In particular, we often deal with the case where $m = 2$, where:

$$O_x|i\rangle|b\rangle = |i\rangle|b \oplus x_i\rangle \tag{2.21}$$

3 Quantum query complexity

Definition: Quantum query computation

Given $x \in D$ and a quantum query algorithm $\mathcal{A}$, we write $\mathcal{A}(x)$ for the random variable taking values in Γ defined by $(\forall j \in \Gamma)$:

$$Pr[\mathcal{A}(x) = j] = \left\| \Pi_j U_d (O_x \otimes \mathbb{I}_w) U_{d-1} (O_x \otimes \mathbb{I}_w) \dots U_1 (O_x \otimes \mathbb{I}_w) U_0 |0\rangle \right\|^2 \tag{3.1}$$

Note there are $d+1$ unitaries U_i and d queries to the quantum oracle.
Let $\epsilon \in (0, 1/2)$, we say $\mathcal{A}$ computes f with bouned error ϵ if $\forall x \in D$, $Pr[\mathcal{A}(x) = f(x)] \geq 1 - \epsilon$.

Definition: Quantum query complexity

For $\epsilon \in (0, 1/2)$, $Q_\epsilon(f)$ is defined to be the minimum depth of any quantum query algorithm that computes f with bounded error ϵ.

We now move to the (quantum) Grover algorithm. The upper bound on the quantum query complexity of the Grover algorithm turns out to be $O(\sqrt{n})$, a smaller exponent than the lower bound of the classical query complexity $O(n)$. For $t \in \mathbb{N}$, consider $OR_n^{o,t} : \{x \in \{0,1\}^n \, ||x| = 0 \text{ or } |x| = t\} \to \{0, 1\}$.

Proposition: Grover's Algorithm

For all $n \in \mathbb{N}$, $t \in \mathbb{N}$ such that $t \leq \frac{n}{3}$, we have $Q(OR_n^{0,t}) \leq \frac{\pi}{4}\sqrt{\frac{n}{t}}$.

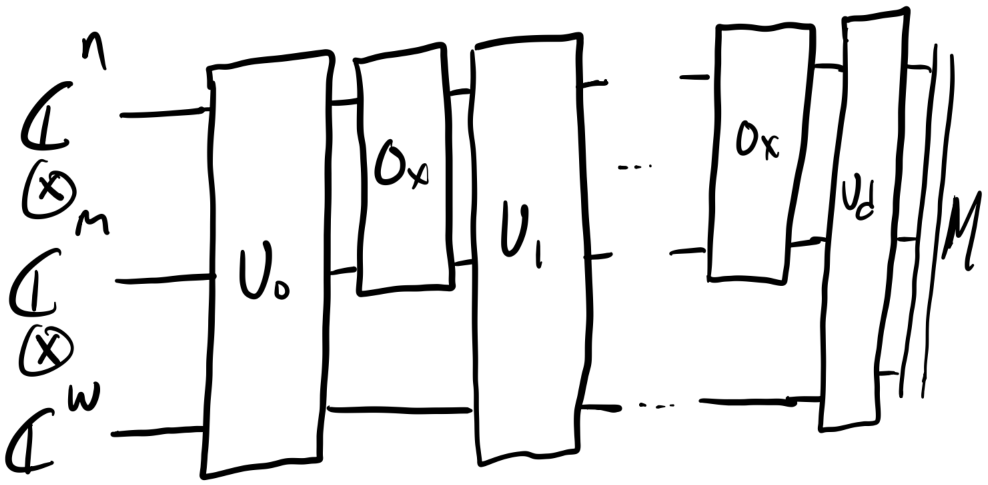

Figure 3.1: Circuit picture of quantum query computation

Definition: Quantum phase oracle

For $x \in \{0,1\}^n$, the quantum phase oracle of x is the unitary matrix $U_x \in \mathbb{C}^{2n \times 2n}$ defined by:

$$U_x|i\rangle|b\rangle = (-1)^{x_{i+1}b}|i\rangle|b\rangle \tag{3.2}$$

where $i \in \{0,1,\ldots,n-1\}$ and $b \in \{0,1\}$.

Lemma: Phase kickback trick

For all $x \in \{0,1\}^n$:

$$U_x = (\mathbb{I}_n \otimes H)O_x(\mathbb{I}_n \otimes H) \tag{3.3}$$

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

Proof. First, notice that for $b \in \{0,1\}$, we have:

$$H|b\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^b|1\rangle) \tag{3.4}$$

Then, it suffices to show that RHS$|i\rangle|b\rangle$ = LHS$|i\rangle|b\rangle$. We have:

$$\begin{aligned}
\text{RHS}|i\rangle|b\rangle &= (\mathbb{I}_n \otimes H)O_x|i\rangle H|b\rangle \\
&= (\mathbb{I}_n \otimes H)O_x|i\rangle \frac{1}{\sqrt{2}}(|0\rangle + (-1)^b|1\rangle) \\
&= \frac{1}{\sqrt{2}}(\mathbb{I}_n \otimes H)(|i\rangle|x_{i+1}\rangle + |i\rangle(-1)^b|1 \oplus x_{i+1}\rangle) \\
&= \frac{1}{\sqrt{2}}\left(|i\rangle\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_{i+1}}|1\rangle) + |i\rangle\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_{i+1}\oplus 1}|1\rangle)(-1)^b\right) \\
&= \frac{1}{2}|i\rangle\left((1+(-1)^b)|0\rangle + ((-1)^{x_{i+1}} - (-1)^b(-1)^{x_{i+1}})|1\rangle\right) \\
&= \begin{cases} |i\rangle|0\rangle & b=0 \\ (-1)^{x_{i+1}}|1\rangle & b=1 \end{cases} \\
&= (-1)^{bx_{i+1}}|i\rangle|b\rangle
\end{aligned}$$

□

A quick note - H is unitary, hence invertible - in fact $H^\dagger = H^{-1} = H$ so $H^2 = \mathbb{I}_2$ and as a result we find that:

$$O_x = (\mathbb{I}_n \otimes H)U_x(\mathbb{I}_n \otimes H) \tag{3.5}$$

The quantum query complexity only cares about calls to the oracle O_x. So, the query complexity does not change if we change the oracle to the phase oracle (the difference can be absorbed into the sequence $U_0, \ldots U_d$).

We now prove the Grover proposition.

Proof. Let $|\psi\rangle$ denote the n-dimensional state:

$$|\psi\rangle = \frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle. \tag{3.6}$$

For $x \in \{0,1\}^n$, let $V_x \in \mathbb{C}^{n\times n}$ be:

$$V_x := \sum_{i=0}^{n-1}(-1)^{x_{i+1}}|i\rangle\langle i| = \mathbb{I}_n - 2\sum_{i|x_{i+1}=1}|i\rangle\langle i| \tag{3.7}$$

Side note; if $b = 1$ then, $U_x|i\rangle b = (-1)^{x_{i+1}b}|i\rangle|b\rangle = (-1)^{x_{i+1}}|i\rangle|1\rangle$. Define $G \in \mathbb{C}^{n\times n}$ as:

$$G := \mathbb{I}_n - 2|\psi\rangle\langle\psi|. \tag{3.8}$$

Finally, let:

$$\Pi_0 := |\psi\rangle\langle\psi|. \tag{3.9}$$

Our measurement is $\mathcal{M} := \{\Pi_0, \mathbb{I}_n - \Pi_0\}$. Then for $k \in \mathbb{N}$, consider:

$$p_x := \left\|\Pi_0(GU_x)^k|\psi\rangle\right\|^2 \tag{3.10}$$

This can be thought as the probability that a k-query quantum algorithm outputs zero. In this setting U_0 is the unitary such that $U_0|0\rangle^n = |\psi\rangle$, and $U_1, \ldots U_k = G$.

There are two cases. We have restricted the domain of the OR to be Hamming weight 0 (i.e. O^n) and with Hamming weight t.

1. If $x = 0^n$, then $V_x = \mathbb{I}_n$, $G = 1 - 2|\psi\rangle\langle\psi|$. Then:

$$(GV_x)|\psi\rangle = G|\psi\rangle(\mathbb{I}_n - 2|\psi\rangle\langle\psi|)|\psi\rangle = |\psi\rangle - 2|\psi\rangle = -|\psi\rangle \implies (GV_x)^k|\psi\rangle = (-1)^k|\psi\rangle \tag{3.11}$$

and so $p_x = \left\| |\psi\rangle\langle\psi|(-1)^k|\psi\rangle \right\|^2 = 1$.

2. If x has Hamming weight t, then define:

$$|\psi_0\rangle = \frac{1}{\sqrt{n-t}} \sum_{i|x_{i+1}=0} |i\rangle, \quad |\psi_1\rangle = \frac{1}{\sqrt{t}} \sum_{i|x_{i+1}=1} |i\rangle \tag{3.12}$$

Then, by inspection:

$$|\psi\rangle = \sqrt{1 - \frac{t}{n}}|\psi_0\rangle + \sqrt{\frac{t}{n}}|\psi_1\rangle = \cos\vartheta|\psi_0\rangle + \sin\vartheta|\psi_1\rangle \tag{3.13}$$

where $\vartheta := \arcsin(\sqrt{t/n}) \in [0, \pi/2]$. We have:

$$GV_x|\psi_0\rangle = G|\psi_0\rangle = |\psi_0\rangle - 2|\psi\rangle\langle\psi|\psi_0\rangle = |\psi_0\rangle - 2\cos\vartheta|\psi\rangle = -\cos 2\vartheta|\psi_0\rangle - \sin 2\vartheta|\psi_1\rangle \tag{3.14}$$

$$GV_x|\psi_1\rangle = -G|\psi_1\rangle = -\sin(2\vartheta)|\psi_0\rangle - \cos(2\vartheta)|\psi_1\rangle \tag{3.15}$$

So - we can analyze the entire algorithm in the 2-dimensional subspace spanned by $|\psi_0\rangle, |\psi_1\rangle$ (which we note are orthogonal). Within the subspace span $\{|\psi_0\rangle, |\psi_1\rangle\}$, we can write GV_x as:

$$-GV_x \cong \begin{pmatrix} \cos(2\vartheta) & -\sin(2\vartheta) \\ \sin(2\vartheta) & \cos(2\vartheta) \end{pmatrix} \tag{3.16}$$

We then have (informally by composition of rotations - formally via diagonalization):

$$(-GV_x)^k = \begin{pmatrix} \cos(2k\vartheta) & -\sin(2k\vartheta) \\ \sin(2k\vartheta) & \cos(2k\vartheta) \end{pmatrix} \tag{3.17}$$

Therefore:

$$(GV_x)^k|\psi\rangle = (-1)^k \left(\cos((2k+1)\vartheta)|\psi_0\rangle + \sin((2k+1)\vartheta)|\psi_1\rangle\right) \tag{3.18}$$

and so:

$$p_x = \left\| \Pi_0 (GV_x)^k |\psi\rangle \right\|^2 = \cos^2(2k\vartheta) \tag{3.19}$$

k is the number of queries, so what do we choose? We choose it such that we can distinguish it from the all 0 case. I.e. that $p_x = 0$ and so $k = \frac{\pi}{4\vartheta}$. $\vartheta = \arcsin(\sqrt{t/n}) \sim \sqrt{t/n}$ so we choose $k = \frac{\pi}{4}\sqrt{\frac{n}{t}}$.

□

4 Basic Design Principles for Quantum Algorithms

We have now shown that $R(OR_n^{0,1}) \geq n/3$ but $Q(OR_n^{0,1}) \leq \frac{\pi}{4}\sqrt{n} + \frac{1}{2}$ - our first rigorous proof of a (quadratic) quantum speedup in terms of n within the query model.

In this lecture, we explore two very useful principles of quantum algorithm design given as two items in Fact 4 below. We apply these principles to show how the quantum query complexity of OR_n (without any domain restrictions on domain) is also $O(\sqrt{n})$. In later lectures, we will take these principles for granted.

Fact: Quantum query algorithms to simulate randomized query algorithms

Quantum (query) algorithms can efficiently simulate randomized (query) algorithms. In particular, $Q(f) \le R(f)$ for any f. See Section 2.3.3 of de Wolf's PhD Thesis for details.

Proof. (Sketch) We will see how a quantum query algorithm can simulate a DDT first by way of an example: consider the obvious depth-2 DDT T that computes $(\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_3)$ with 1 labelling the root. For this example, use the following fact:

Fact: Permutation unitary

Suppose $g : \{0, 1, \dots, a-1\} \to \{0, 1, \dots, b-1\}$, then there exists a unitary U_g (in fact a permutation matrix) acting on the space $\mathbb{C}^a \otimes \mathbb{C}^b = \mathbb{C}^{ab} (U_g \in \mathbb{C}^{ab \times ab})$ such that:

$$U_g |i\rangle|0\rangle = |i\rangle|g(i)\rangle \tag{4.1}$$

for all $i \in \{0, 1, \dots, a-1\}$.

Armed with this, we proceed as follows; let $I : \{0,1\} \to \{2,3\}$ be defined by $I(0) = 2$ and $I(1) = 3$. I maps the bit value of x_1 to the index that is queried next. Let $I - 1$ denote the function that first applies I and then subtracts 1. Let $h : \{0,1\} \times \{0,1,2\} \times \{0,1\} \to \{0,1\}$ by:

$$h(0, 2-1, 0) = 0, h(0, 2-1, 1) = 1 h(1, 3-1, 0) = 1, h(1, 3-1, 1) = 0. \tag{4.2}$$

We have defined h such that $h(a, I-1, b)$ is defined to be the value that T outputs if $x_1 = a$, I is the index of the variable queried next, and $x_I = b$.

Our register has dimensions $\mathbb{C}^3 \otimes \mathbb{C}^2 \otimes \mathbb{C}^3 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$. Where in $|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle$ the first two are query registers and the last three are workspace registers.

$$\begin{aligned}
&|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle \\
&\mapsto^{O_x} |0\rangle|x_1\rangle|0\rangle|0\rangle|0\rangle \\
&\mapsto^{U_{I-1}} |0\rangle|x_1\rangle|I(x_1)-1\rangle|0\rangle|0\rangle \\
&\mapsto^{O_x} |0\rangle|x_1\rangle|I(x_1)-1\rangle|x_{I(x_1)}\rangle|0\rangle \\
&\mapsto^{U_h} |0\rangle|x_1\rangle|I(x_1)-1\rangle|x_{I(x_1)}|h(x_1, I(x_1) - 1 m x_{I(x_1)})\rangle\rangle \\
&= |0\rangle|x_1\rangle|I(x_1)-1\rangle|x_{I(x_1)}\rangle|T(x)\rangle
\end{aligned}$$

Where $\mapsto^A$ denotes the application of A (tensored appropriately with identities) and the last line uses the definition of h. Then, measuring using $\{\Pi_0 := \mathbb{I}_{36} \otimes |0\rangle\langle 0|, \Pi_1 := \mathbb{I}_{36} \otimes |1\rangle\langle 1|\}$ gives outcome $T(x)$ with probability 1. This concludes the example for DDTs.

What about *RDT*s? Recall an RDT is a distribution $\{(p_i, T_i)\}_{i=0}^{K-1}$ over *DDT*s. We have seen how T_i can be simulated by a quantum query algorithm $\mathcal{A}_i$ for each i. Suppose $\mathcal{A}_i$ is specified by unitaries $\left\{U_j^i\right\}_{j=0,\dots,d}$. Then the RDT can be simulated by a quantum query algorithm $\mathcal{A}$ that starts with the state:

$$|\psi_0\rangle := \sum_{i=0}^{K-1} U_0^i |0\rangle \otimes \sqrt{p_i}|i\rangle. \tag{4.3}$$

(More precisely, we can define the U_0 of $\mathcal{A}$ such that $U_0|0\rangle = |\psi_0\rangle$). Then for $j \in \{1, \dots, d\}$, U_j of $\mathcal{A}$ is defined to be:

$$U_j := \sum_{i=0}^{K-1} U_j^1 \otimes |i\rangle\langle i|. \tag{4.4}$$

The measurement of $\mathcal{A}$ is still $\{\Pi_0 := |0\rangle\langle 0|, \Pi_1 := |1\rangle\langle 1|\}$ (appropriately tensored with identities) so the Πs only act non-trivially on the $T_i(x)$ register. $\square$

In our definition of quantum query complexity, there is one measurement coming at the end. But in fact, could have also allowed "intermediate measurements". The principle of deferred measurement says that such measurements can always be simulated by a measurement at the end.

Fact: Principle of Deferred Measurement

Suppose we make a measurement $\mathcal{M} := \{\Pi_1, \ldots, \Pi_k\}$ on a state $|\psi\rangle$ and if the measurement outcome is $i \in [k]$, we apply unitary U_i to another state $|\psi'\rangle$ (Comment: In Simon's problem, we need $|\psi'\rangle$ to be the post-measurement state of $|\psi\rangle$, but the proof is the same.) Then the effect of this procedure is that with probability $\left\|\Pi_i|\psi\rangle\right\|^2$, we end up with the final state $U_i|\psi'\rangle$.
Now, consider the following simulation; we apply the unitary:

$$U := \sum_{i=1}^{n} \Pi_i \otimes U_i \tag{4.5}$$

to the state $|\psi\rangle|\psi'\rangle$ and then measure the first register using $\mathcal{M}$. (Note that it is unitary by virtue of the orthogonality and completeness of the projectors, and the unitary of U).
Then, the probability of observing outcome $i \in [k]$ is:

$$\left\|(\Pi_i \otimes \mathbb{I})U|\psi\rangle|\psi'\rangle\right\|^2 = \left\|\Pi_i|\psi\rangle \otimes U_i|\psi'\rangle\right\|^2 = \left\|\Pi_i|\psi\rangle\right\|^2. \tag{4.6}$$

where the second last equality uses the fact that $\|u \otimes v\| = \|u\|\|v\|$ and that $\|Vu\| = \|u\|$ for unitary V. And the state on the second register becomes $U_i|\psi'\rangle$. This is precisely the same effect as the original procedure where the measurement comes first.

Using these two design principles, we can show the following:

Proposition: General quadratic upper bound on $Q(OR_n)$

$\exists c > 0$ such that for all $n \in \mathbb{N}$ we have $Q(OR_n) \le c\sqrt{n}$.

Proof. First, we may assume that $|x| \le 0.01n$. Else, if we randomly query 10000 indices of x, we'll not find a 1 (i.e. fail to distinguish the input from 0^n) with probability at most

$$\left(1 - \frac{0.01n}{n}\right)^{10000} \le e^{-100} \tag{4.7}$$

which is negligeble compared to the bounded error $1/3$ we care about (formally we would need to consider all failure probabilities and then use Boole's inequality). The inequality uses that $1 - x \le e^{-x}$ for all $x \ge 0$.
From the previous analysis, we see that, on input $x \in \{0,1\}^n$ using k queries we can get the probability of outputting 0 to be:

$$p_x(k) = \cos^2(2\theta_x k) = \frac{1 + \cos(4\theta_x k)}{2}. \tag{4.8}$$

where $\theta_x = \arcsin(\sqrt{|x|/n})$. Plot the graph of $p_x(k)$ as a function of k; note that its period T_x satisfies:

$$15 \le \frac{\pi}{2\arcsin\sqrt{0.01}} \le T_x := \frac{\pi}{2\theta_x} \le \frac{\pi}{2}\sqrt{n}, \tag{4.9}$$

where the first inequality uses the fact that $|x| \le 0.01n$ and the last inequality uses $|x| \ge 1$ (together with the monotonicity of $\arcsin(a)$ for $a \in [0,1]$ and $\arcsin(a) \ge a$ for $a \in [0,1]$).

Therefore, in the interval $[1, \lceil \frac{\pi}{2}\sqrt{n} \rceil]$, $p_x(k)$ runs over at least one period and each period must span over at least 15 positive integers (by the first inequality of Eq. (4.9)).

The last step of the algorithm is:

- Repeat the following 10000 times:
  1. Choose $k \in \mathbb{N}$ uniformly at random between 1 and $2\sqrt{n}$
  2. Run Grover's quantum query algorithm which has $p_k(x)$ probability of outputting 0 (i.e. the measurement outcome being 0).
  3. If the output is 1, return 1. If all repeats give output 0, return 0.

the intuition for why this works is that if we choose an integer k uniformly at random from $[1, \lceil \frac{\pi}{2}\sqrt{n} \rceil]$ then Eq. (4.9) shows that $p_x(k)$ is a constant away from 1 with constant probability (over the randomness of the choice of k) (think pictorially).

This means that the quantum query algorithm will output 1 with constant probability. (Recall $p_x(k)$ is the probability of the quantum algorithm outputting 0.) Since we would never see 1 when $x = 0^n$, we can just repeat this a large number of times and output 1 if and only if the quantum query algorithm outputs a 1 in any of those repeats. This allows us to suppress the error probability to be negligible. $\square$

Some remarks:

1. To see that the query algorithm described in the proof is a bonafide quantum query algorithm according to our definition, we need to use both facts that we established earlier, i.e., quantum can simulate randomized and principle of deferred measurement. The first fact allows us to convert the randomized query algorithm doing the preprocessing to a quantum query algorithm. But this quantum query algorithm could continue running if its output is not 1, and recall a quantum query algorithm's output always arises from a measurement. However, by the second fact, we can defer this measurement to the end. The second fact also allows us to defer the measurements made in each of the repeat loops to the end.

2. The exposition here expands a little on Scott Aaronson's lecture notes on Grover search (top of page 8).

3. A somewhat different algorithm, along the lines of what Nick suggested in class of exponentially increasing k from 1 to $O(n)$, is analyzed in detail in Section 4 of this paper.

4. In fact, there's yet another algorithm for computing ORn using a "fully quantum strategy" (i.e., very unlike the two algorithms mentioned above that are essentially Grover + classical ideas) called "fixed-point amplitude amplification". See this paper. Maybe we'll have time to discuss this when we talk about quantum signal processing.

Proposition: Error suppression/Chernoff bound

Let $\epsilon \in (0, 1/3)$. Let $f : D \subseteq \{0, 1, \ldots, m-1\}^n \to \Gamma$. Then $R_\epsilon(f) \le R(f)\lceil 18 \ln(1/\epsilon) \rceil$ and $Q_\epsilon(f) \le Q(f)\lceil 18 \ln(1/\epsilon) \rceil$.

Proof. Will prove the randomized case. Same idea also works in the quantum case via the principle of deferred measurement.

Suppose $\mathcal{T}$ is a RDT that computes f with bounded error 1/3. Take $k \in \mathbb{N}$ copies of $\mathcal{T}$ and output the modal output of the k copies. For a given $x \in D$, let X denote the number of copies that ouput the correct answer on x, the probability that each copy outputs the correct answer $p = \frac{1}{2} + \delta$, where $\delta \ge 1/6$ and the probability that each copy outputs the incorrect answer is $q = 1 - p = \frac{1}{2} - \delta \leq \frac{1}{3}$. Then, we are correct if and only if $X > k/2$, So, the probability that we are incorrect is:

$$\begin{aligned}
Pr[X \leq k/2] &= \sum_{i=0}^{k/2} Pr[X = i] = \sum_{i=0}^{k} /2 \binom{k}{i} p^i q^{k-i} \\
&\leq \sum_{i=0}^{k/2} \binom{k}{i} p^{k/2} q^{k/2} \\
&\leq 2^k (pq)^{k/2} \\
&= 2^k \left(\frac{1}{2} + \delta\right)^{k/2} \left(\frac{1}{2} - \delta\right)^{k/2} \\
&= 2^k \left(\frac{1}{4} - \delta^2\right)^{k/2} \\
&= (1 - 4\delta^2)^{k/2} \\
&\leq e^{-2k\delta^2}
\end{aligned}$$

So if we pick $k \geq \ln(1/\epsilon)/(2\delta^2)$, we have $pr[X \leq k/2] \leq \epsilon$. Since $\delta \geq 1/6$, it suffices to pick $k \geq 18 \ln(1/\epsilon)$. Hence the proposition. □

Remark: We have shown that given k i.i.d. random variables $X_1, \ldots, X_k$ taking variables in $\{0,1\}$ such that $\exists \delta \in [0, 1/2], \forall i, Pr[X_i = 1] = \frac{1}{2} + \delta$. Then, $Pr[\sum_{i=1}^{k} X_i \leq k/2] \leq e^{-2k\delta^2}$. This type of bound is known as a Chernoff bound, there are more sophisticated variants with more sophisticated proofs. The rough-and-ready proof given here is taken from Nielsen and Chuang, Box 3.4.

5 Time Complexity

We introduce machinery to define time complexity of decision problems.

Definition: Decision Problem

A decision problem is a set of functions $\mathcal{P} = \left\{P_n : \{0,1\}^n \to \{0,1\}, n \in \mathbb{N}\right\}$.

Two remarks:

1. This intuitively defines the problem of an input of $x \in \{0,1\}^n$ and the desired output is $P_n(x)$.
2. Given a language $L = \{0,1\}^* := \bigcup_{n \in \mathbb{N}} \{0,1\}^n$, we note the correspondence $\mathcal{P} \leftrightarrow L := \bigcup_{n \in \mathbb{N}} P_n^{-1}(1)$.

Defining quantum time complexity in terms of Turing machines is difficult, but in the picture of circuits it is intuitive.

Definition: Classical and Quantum Circuits

A classical (Boolean/cit) circuit is a directed acyclic graph, with $a \in \mathbb{N}$ vertices uniquely labelled as $1, \dots, a$ with no incoming edges (a "input bits"), $b \in \mathbb{N}$ vertices uniquely labelled by $1', \dots, b$ with no outgoing edges (b "output bits") with all other vertices labelled by:

$$cGATES := \{FANOUT, AND, OR, NOT\} \tag{5.1}$$

where AND, OR have 2 incoming edges and 1 outgoing edge, $FANOUT$ has 1 incoming edge and 2 outgoing edges, and NOT has 1 incoming edge and 1 outgoing edge.
A quantum (Boolean/qubit) circuit is a directed acyclic graph with a input bits and b output bits, where the other vertices are labelled by:

$$qGATES := \{H, T, Toffoli\} \tag{5.2}$$

where T, H have 1 incoming edge and 1 outgoing edge, and the $Toffoli$ has 3 incoming edges and 3 outgoing edges.

Note that you could have non-directed cycles in a circuit, e.g. $FANOUT$ going into an AND. There are also other universal gate sets we could choose, e.g. $cGATES = \{NAND\}$, or all $qGATES$ as all 1-qubit gates and any 2-qubit entangling gate.

Definition: Computation using classical and quantum circuits

In the classical case, we consider an input $\{0,1\}^a$. We put each of $x_1, \ldots, x_a$ into the input vertices. When we see AND we compute the AND of the bits, when we see OR we compute OR of the bits, when we see NOT we compute the NOT of the bit, and when we see $FANOUT$ we clone the bit.
In the quantum case, we note that the input and outputs of each gate are the same and so $a = b$. We identify the input $x_1x_2 \ldots x_a$ as $|x_1x_2 \ldots x_a\rangle$. The gates are:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{5.3}$$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \tag{5.4}$$

$$Toffoli = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{5.5}$$

with identies tensored appropriately on the qubits on which the gates do not act.
Finally, we make the measurement of:

$$\left\{ \Pi_z := |z\rangle\langle z| : z \in \{0,1\}^b \right\} \tag{5.6}$$

A classical circuit can be "described" by a string $y \in \overline{cGATES}^*$ where $\overline{cGATES} = cGATES \cup \{0, 1, Blank\}$. Similarly for quantum circuits with $y \in \overline{qGATES}^*$. This is analogous to how we can encode a Turing machine as a string.

Definition: Deterministic/Randomized/Quantum time cmplexity of decision problems

1. We say $\mathcal{P}$ can be solved in deterministic time T ("solved by a deterministic algorithm in time T") if $\exists$ a Turing machine $\mathcal{A}$ that for all $N \in \mathbb{N}$ satisfies the following. $\forall y \in \{0,1\}^N$, $\mathcal{A}$ runs in $O(T(n))$ steps and outputs the description of a classical circuit C_y such that $C_y(0^a) = P_N(y)$.

2. We say that $\mathcal{P}$ can be solved in randomized time T ("solved by a randomized algorithm in time T") if $\exists$ a Turing machine that for all $N \in \mathbb{N}$ satisdies the following. For all $y \in \{0,1\}^N$, $\mathcal{A}$ outputs the description of a classical circuit C_y on a input bits and 1 output bit such that:

$$Pr[C_y(r) = P_N(y) | r \leftarrow \{0,1\}^a] \geq \frac{2}{3} \tag{5.7}$$

3. We say that $\mathcal{P}$ can be solved in quantum time T ("solved by a quantum algorithm in time T") if $\exists$ a Turing machine $\mathcal{A}$ that for all $N \in \mathbb{N}$ satisfies the following. For all $y \in \{0,1\}^N$, $\mathcal{A}$ outputs the description of a quantum circuit C_y on a input bits and 1 output bit such that:

$$Pr[z_1 = P_N(y) | z \leftarrow \text{meas. outcome of } C_y(|0\rangle^a)] \geq \frac{2}{3} \tag{5.8}$$

Definition: ***P/BPP/BQP***

P is defined as the set:

$$P := \{\mathcal{P} | \exists c \in \mathbb{N} \text{ and } T : \mathbb{N} \to \mathbb{N} \text{ with } T(n) = O(n^c) \text{ s.t. } \mathcal{P} \text{ can be solved in deterministic time } T.\} \tag{5.9}$$

The definitions of BPP, BQP are analogous, replacing deterministic with randomized/quantum.

The key point between the query model and the circuit model is the following motto. In the query model we have $f : \{0,1\}^n \to \{0,1\}$. The $x \in \{0,1\}^n$ corresponds to $x : [n] \to \{0,1\}$ where $x_(i) = x_i$. The query model has relevance to time complexity if given the input $y \in \{0,1\}^N$ to P_N can be used to compute a circuit for x efficiently.

As an example; $y = (u_1 \wedge \neg u_2 \wedge u_3) \vee \dots$ for $u_1, \dots, u_l$ (3-SAT) can be used to efficiently construct a classical circuit for $x : \{0,1\}^l \to \{0,1\}$ such that $x(u) = y$ evaluated at u. It is then a fact that there is a Turing machine which can efficiently construct a classical circuit O_x.

Note in the case of the SAT problem, it is equivalent to finding $OR(x)$.

Recall in Grover's algorithm that we also had unitaries in addition to oracles. The unitaries can also be efficiently generated from the gate set.

6 Complexity continued

Correction: We defined a quantum circuit using acylic graphs and the Toffoli gate. But the Toffoli gate is asymmetric, so if the graph is acylic then we do not have data about the ordering. So the definition needs to be refined slightly.

The motto that we wrote last time was that if in the Query model we try to compute $f : \{0,1\}^n \to \{0,1\}$ and the input to f is an n-bit string, this relates to the time complexity if

1. $x : [n] \to \{0,1\}$ has an "efficiently describable" circuit.

2. The unitary operations for the quantum query algorithm have an "efficiently describable" quantum circuit.

A query algorithm has a representation in terms of a circuit diagram of interleaving oracles U_0 and O_x.

As an example, we have the k-SAT problem - the problem of does there exist a choice of $u_1, \ldots u_n$ $\{0,1\}$ such that y evaluated on the formula is equal to 1.

On the classical side, it is known that k-SAT problem (where l is the number of variables, c is the number of clauses, k is the number of terms per clause) has:

1. For $k = 2$, the deterministic time is $\leq \text{poly}(l, c)$
2. For $k > 2$, the randomized time is $\leq 2^{l(1-1/k)}\text{poly}(l, c)$.

Although there is no proof (as proving lower bounds in the Turing model is hard), there is a conjecture known as the *(Classical) Strong exponential time hypothesis*. For every $\epsilon > 0$, there exists $k \in \mathbb{N}$ such that no $O(2^{(1-\epsilon)l}\text{poly}(l, c))$ randomized algorithm can solve k-SAT. This has been a long-standing conjecture in classical complexity theory. This is violated in the quantum setting due to Grover search.

We described the query version of Grover search, but we should describe it in terms of bona fide quantum gates. Consider our input formula y. We have $x : \{0,1\}^l \to \{0,1\}$ (where $x \in \{0,1\}^{2^l}$) where $x(u_1, \ldots, u_l)$ is whatever y evaluates to on $u_1, \ldots, u_l$.

Important fact: Given the description of a classical circuit for x of size s, it outputs the classical description of a quantum circuit for O_x in time $O(s)$. Left as an exercise to prove. It remains to see how we efficiently construct the unitaries, but that will be a homework exercise.

Collision problem: $\text{Collision}_n : D_0 \cup D_1 \subseteq \{0, 1, \ldots, n-1\}^n \to \{0,1\}$ where:

$$D_0 = \left\{x \in \{0, 1, \ldots, n-1\}^n \,|\forall i, j \in [n], i \neq j \implies x_i \neq x_j\right\} \tag{6.1}$$

$$D_1 = \left\{x \in \{0, 1, \ldots, n-1\}^n \,|\forall i \in [n] \exists! j \in [n], j \neq i \text{ s. t. } x_i = x_j\right\} \tag{6.2}$$

Where D_0 are n-digit strings with no repeats (x is 1-1), and D_1 are n-digit strings where each character is repeated once (x is 2-1). The definition of collision is $\text{Collision}_n(D_0) = \{0\}$ and $\text{Collision}_n(D_1) = \{1\}$.

Proposition: Randomized complexity of Collision$_n$

$R(\text{Collision}_n) = O(\sqrt{n})$.

Proof. Follows by a birthday paradox argument. Choose a random size-k subset of $[n]$ uniformly at random. Output 1 iff you see a collision.

Case 1. $x \in D_1$. Then, we can pair up the x_i and x_j. There are $\binom{n}{k}$ size k subsets. What is the probability of a bad situation? In general, let's think about the number of k-size subsets that don't contain a collision. For the first point, we randomly choose a single point, so there is no chance of a collision, so we have n possible choices. For the second point we have $n-1$ choices left of which $n-2$ cause no collision. For the third we have $n-4$ viable choices, and so on until:

$$\frac{n(n-2)(n-4)\ldots(n-2(k-1))}{k!} \tag{6.3}$$

where we divide out by $k!$ to remove the redundancy from the ordering of the subsets. So the probability of not seeing a collision is going to be:

$$\frac{n(n-2)(n-4)\ldots(n-2(k-1))/k!}{n(n-1)(n-2)\ldots(n-k+1)/k!} = 1\left(1 - \frac{1}{n-1}\right)\left(1 - \frac{2}{n-2}\right)\ldots\left(1 - \frac{k}{n-k}\right) \tag{6.4}$$

we can upper bound $1 - x \leq e^{-x}$ so:

$$Pr[\text{Failure}] \leq 1 \cdot e^{-\frac{1}{n-1}}e^{-\frac{2}{n-2}} = e^{-\sum_{i=1}^{k-1}\frac{i}{n-i}} \tag{6.5}$$

Let's say $k \leq \frac{n}{2}$. Then, $n - i \geq \frac{n}{2}$ so:

$$\sum_{i=1}^{k-1} \frac{i}{n-i} \geq \frac{\sum_{i=1}^{k-1} i}{n} = \frac{k(k-1)}{2n} \geq \frac{(k-1)^2}{2n} \tag{6.6}$$

So then:

$$Pr[\text{Failure}] \leq e^{-\frac{(k-1)^2}{2n}} \leq \epsilon = \frac{1}{3} \tag{6.7}$$

so it suffices to choose $k = O(\sqrt{n}\log(1/\epsilon))$. □

Proposition: Quantum complexity of Collision$_n$

$Q(\text{Collision}_n) = O(n^{1/3})$.

Proof. We input $x_1, \dots x_k$. First, we classically query $x_1, \dots, x_k$.

1. If we query the above and we already find a collision, then we are done.
2. If no collisions, then quantumly Grover search for the k distinct symbols among $x_{k+1}, \dots, x_n$. For how many of these symbols will we get a hit for what we queried? Among the remaining symbols, there will be exactly k that match with the already queried symbols. On the $n-k$ symbols, we are not just trying to compute the OR function, but rather OR with a promise that there are no hits, or that there are k hits with th remaining symbols, i.e. $OR^{0,k}_{n-k}(\tilde{x}_{k+1}, \dots, \tilde{x}_n)$, which has complexity $O(\sqrt{n/k})$. The overall query complexity is $k + O(\sqrt{n/k})$ which is minimized at $k = \sqrt{n/k}$ so $k = n^{1/3}$ wherein the overall complexity is $O(k^{1/3})$.

□

There is an extra step in pre-processing the $x_{k+1}, \dots x_n$. For query complexity, this extra step of converting into bit strings cost only 2 oracle calls, $O(1)$. But there is also the function we need to apply that uses what we already queried. This doesn't use any queries, but may create some time issue complexities. There are more examples attached to this be relevant time-complexity wise; we need to assume something called QRAM which is outside the Turing model. But this is very controversial.

In the remaining 15 minutes, Daocheng will give an open question: Directed st-connectivity in hypercube problem. See Ambainis et al, SODA '18.

When $n = 3$, the hypercube graph looks like a cube with each with vertex a 3-bit string, and each edge has a bitstring x_k which indicates whether the edge exists or not in the hypercube. Does there exist a directed path from s to t that goes through edges that are available? (in the direction of increasing Hamming weight)?

Each vertex in the hypercube has n edges corresponding to the n bit flips. There are 2^n vertices, but avoiding the double counting we have a total of $n2^{n-1}$ edges. So the maximal query complexity is $n2^{n-1}$ (because if we query all the edges, we know there to be a directed path for sure). The question is can we do better.

Things that don't work:

1. Look at all the directed paths from 0^n and 1^n and Grover search over them. But, there are n^n directed paths, so Grover gives $\sqrt{n^n} = n^{n/2}$ which is much larger than the trivial upper bound.
2. Look at the "column" in the middle which are bit strings with Hamming weight $n/2$. Look at a specific vertex x on that column, and look at the subcube between 0^n and x, i.e $\{y \in \{0,1\}^n \, | y \leq x\}$. The number of edges in the subcube is $\sim 2^{n/2}$. In the subcube, we can naively query all of them to decide if 0^n is connected to x, taking $O(2^{n/2})$ queries. Can we then take all x in the middle, of which there are $2^n/\sqrt{n}$. If we Grover search over all of them, we have $\sqrt{2^n/\sqrt{n}}2^{n/2} = 2^n/n^{1/4}$. But the actual algorithm in the paper gives the upper bound $2^{0.8n}$... the best known lower bound is $\sqrt{2^n}$.

7 Analysis of the hypercube and Simon's problem

The hypercube problem continued

The hypercube problem continued. To review, if we consider the $n!$ paths (n choices of which bit to flip in the first step, then $n-1$, and so on) from 0^n to 1^n, and then consider $OR(\tilde{p}_1 \dots \tilde{p}_{n!})$ where $\tilde{p}_i = 1$ if the path is present and 0 otherwise. If we do a Grover search over all directed paths, then we have $O(\sqrt{n!}n) \sim O(n^{n/2})$ complexity (via the Sterling approximation) which is worse than the trivial upper bound. Better method:

1. We now consider if we go from 0^n to 1^n (and arrange a pictoral diagram going from left to right in order of increasing Hamming weight), we can classically all edges in the shaded areas i.e. from 0 up to a certain Hamming weight αn, and then from Hamming weight $n - \alpha n$ line to 1^n. The lines are the sets of points $\{z \in \{0,1\}^n \, ||z| = \alpha n\}$. Then we can classically query all edges in the shaded areas, which takes $2\sum_{k=0}^{\alpha n} \binom{n}{k} \cdot \frac{n}{2}$ queries.

2. Now if we try to decide whether $z^{(1)}$ is connected to $z^{(2)}$ via a directed path. We then have a sub-hypercube $\left\{z \in \{0,1\}^n \, |z^{(1)} \leq z \leq z^{(2)}\right\}$. The dimension of this hypercube is $n/2 - \alpha n$. So the number of edges inside of it is $(n/2 - \alpha/n)2^{n/2-\alpha n}/2$. So by investing $O(2^{n/2-\alpha n})$ (hiding the polynomial in the $*$) his number of (classical) queries we can decide if $z^{(1)}$ is connected to $z^{(2)}$

3. Now we ask how $z^{(2)}$ could possibly connected to 0^n. We want to OR over the points on the $|z| = \alpha n$ line, so we Grover search over $\sqrt{\binom{n}{\alpha n}} O^*(2^{n/2-\alpha n})$ possibilities.

4. So then in total we have $\sqrt{\binom{n}{n/2}}\sqrt{\binom{n}{\alpha n}}O^*(2^{n/2-\alpha n})$ so the overall cost is:

$$O^*(\sum_{k=0}^{\alpha n}\binom{n}{k} + \sqrt{\binom{n}{n/2}}\sqrt{\binom{n}{\alpha n}}2^{n/2-\alpha n}) \tag{7.1}$$

then we choose α optimally. For this we use a Lemma:

Lemma: Binomial Coefficient Bounds

Let $n \in \mathbb{N}$. Then:

$$\begin{cases} \forall k \in \mathbb{N}, 1 \leq k \leq \frac{n}{2}, \binom{n}{\leq k} := \sum_{i=0}^{k}\binom{n}{i} \leq 2^{h(k/n)n} \\ \forall l \in \mathbb{N}, 1 \leq l \leq n, \binom{n}{l} \leq 2^{h(l/n)n} \end{cases} \tag{7.2}$$

where $h : [0,1] \to [0,1]$ is the binary entropy defined by:

$$h(p) := -p\log_2(p) - (1-p)\log_2(1-p) \tag{7.3}$$

upper bounding using the Lemma:

$$\sum_{k=0}^{\alpha n}\binom{n}{k} + \sqrt{\binom{n}{n/2}}\sqrt{\binom{n}{\alpha n}}2^{n/2-\alpha n} \leq 2^{nh(\alpha n/n)} + \sqrt{2^{nh(1/2)}}\sqrt{2^{nh(2\alpha)}}2^{n/2-\alpha n} \tag{7.4}$$

$$= 2^{n\ln(\alpha)} + 2^{n(1+\frac{1}{2}h(2\alpha)-\alpha)} \tag{7.5}$$

so we optimize by setting the exponents to being the same. We can do this using our favourite software device to find $\alpha = 0.397$. This implies the quantum query complexity is $2^{nh(0.397)} = 2^{0.969n}$. Mo

Can we push this to extremes with more layers/recusive averaging? Currently we have $O^*(2^{0.8615n}$. The best known quantum lower bound is $\Omega^*(2^{0.5n})$ and there has been a 5 year open question to close the gap. Another question si whether we can remove the assumption of QRAM.

Simon's problem

Consider $\text{Simon}_n : D \coloneqq D_0 \cup D_1 \subseteq \{0,1,\ldots,n-1\}^n \to \{0,1\}$. The setting is similar to the collision problem. Here the disjoint sets D_0, D_1 are defined as:

$$D_0 \coloneqq \{x \in \{0,1,\ldots,n-1\}^n \,|\forall s \neq t, x(s) \neq x(t)\} \tag{7.6}$$

$$D_1 \coloneqq \left\{x : \{0,1\}^k \to \{0,1,\ldots,n-1\} \,|\exists a \in \{0,1\}^k - \left\{0^k\right\} \forall s,t \in \{0,1\}^k, x(s) = x(t) \iff s = t \oplus a\right\} \tag{7.7}$$

where $\text{Simon}_n(D_0) = \{0\}$ and $\text{Simon}_n(D_1) = \{1\}$.

Proposition: Quantum/randomized complexities of Simon's problem

$Q(\text{Simon}_n) = O(\log n)$ and $R(\text{Simon}_n) = \Omega(\{n\})$.

Lemma: Hadamard identity

Let $x \in \{0,1\}^k$ and the corresponding state $|x\rangle = |x_1\rangle|x_2\rangle\ldots|x_k\rangle \in \mathbb{C}^{2^k}$. let $H^{\otimes k}$ be the Hadamard tensored with itself k times where $H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$. Then:

$$H|x\rangle = \frac{1}{\sqrt{2^k}} \sum_{y\in\{0,1\}^k} (-1)^{x\cdot y}|y\rangle \tag{7.8}$$

Proof. Left to the reader - we already did $k = 1$ when proving the phase kickback trick. □

Lemma

Let $k, K \in \mathbb{N}$. Suppose $z_1, z_2, \ldots, z_K \leftarrow \mathbb{F}_2^k$ (where $\mathbb{F}_2$ is the field of two elements, i.e $\{0,1\}$ with XOR addition). Then, the probability that the dimension of the span of the z_is, i.e. the dimension of:

$$V \coloneqq \{a_1 z_1 + \ldots + a_K z_K | \forall i a_i \in \mathbb{F}_2\} \subseteq \mathbb{F}_2^k \tag{7.9}$$

is k at least $1 - 2^{k-K}$.

Sanity check: when $K \leq k$, $1 - 2^{k-K}$ is non-positive and the bound yields no information. Special case; if $K = K + 1$ then the probability is at least $1/2$

8 Simon's Problem continued

$D = D_0 \cup D_1 \subseteq \{0,1,\ldots,n-1\}^n$ (disjoint union) where $n = 2^k$. $x \in D_0 \iff x$ is a permutation of $\{0,1,\ldots,n-1\}$ while $x \in D_1 \iff \exists a \in \{0,1\}^k \setminus \left\{0^k\right\}$ such that $x(t) = x(s) \iff t = s \oplus a$. We've identified $[n] = \{0,1\}^k$. In the $k = 3$ case, we can identify $0, \ldots, 7$ by their binary representation, and then (example, which I missed TODO)

Let us return to the proof of the Lemma from last class.

Proof. Let $A \in \mathbb{F}_2^{K\times k}$ defined by:

$$A = \begin{pmatrix} -z_1- \\ -z_2- \\ \vdots \\ -z_K- \end{pmatrix} \tag{8.1}$$

the dimension of the span of $z_1, \dots z_k$ is (by definition) equal to the row-rank of A. But row-rank is equal to column rank. Then, by the rank-nullity theorem, column rank of $A = k$ implies that $\ker(A) = \left\{0^k\right\}$ (where we recall that the kernel is defined as $\left\{z \in \mathbb{F}_2^k | Az = 0\right\}$). So the question reduces to "what is the probability that the kernel of A is trivial?":

$$Pr[\ker(A) = \left\{0^k\right\}] = Pr[\exists z \neq 0^k | Az = 0] \leq \sum_{z \in \mathbb{F}_2^k \setminus \left\{0^k\right\}} Pr[Az = 0] \tag{8.2}$$

where we have applied the union bound $Pr(\bigcup_i A_i) \leq \sum_i P(A_i)$. Now, WLOG suppose $z_k = 1$. Then, $Az = (z_1 a_1 + z_2 a_2 + \dots) + z_k a_k$ where now in the brackets is just a uniformly random vector in $\mathbb{F}_2^k$, i.e.:

$$Pr[\ker(A) = \left\{0^k\right\}] \leq (2^k - 1)\frac{1}{2^K} \leq \frac{2^k}{2^K} \tag{8.3}$$

so $Pr[\ker(A) = \left\{0^k\right\}] \geq 1 - 2^{k-K}$. □

Lemma

Let $K \in \mathbb{N}$ and $0 \neq a \in \mathbb{F}_2^k$. Let $z_1, \dots, z_K \in \mathbb{F}_2^k$ be arbitrary such that $\forall i \in [K]$, $a \cdot z_i = 0$. Then, $\dim \operatorname{span}_{\mathbb{F}_2} \{z_1, \dots, z_k\} \leq k - 1$.

Proposition: Quantum query complexity of Simon's algorithm

$Q(\text{Simon}_n) = O(\log n)$

Proof. Create the following state using 1 query to x:

$$\frac{1}{\sqrt{2^k}} \sum_{s \in \{0,1\}^k} |s\rangle|0\rangle \mapsto^{O_x} \frac{1}{\sqrt{2^k}} \sum_{s \in \{0,1\}^k} |s\rangle|x(s)\rangle \tag{8.4}$$

where $|x(s)\rangle \in \mathbb{C}^n$. Let us measure the second register in the computational basis.

Case 1; $x \in D_0$: Say the outcome is $i \in \{0, \dots, n-1\}$, then the resulting state is by definition:

$$\mathbb{I} \otimes |i\rangle\langle i| \left(\frac{1}{\sqrt{2^k}} \sum_{s \in \{0,1\}^k} |s\rangle|x(s)\rangle \right) = \frac{1}{\sqrt{2^k}} \sum_{s \in \{0,1\}^k} |s\rangle|i\rangle \delta_{i,x(s)} \propto |x(s_0)\rangle \tag{8.5}$$

for some s_0 such that $x(s_0) = i$.

Case 2; $x \in D_1$: Say the outcome is $i \in \{0, 1, \ldots, n-1\}$. Then the resulting state (as before) is:

$$\frac{1}{\sqrt{2^k}} \sum_{s \in \{0,1\}^k} |s\rangle|i\rangle\delta_{i,x(s)} \propto (|s_0\rangle + |s_0 \oplus a\rangle)|i\rangle \tag{8.6}$$

where $x(s_0) = i$.

Now apply $H^{\otimes k}$ to the first register.

Case 1; $x \in D_0$: we have $|s_0\rangle|x(s_0)\rangle$ and applying the Hadamard lemma from last class we get:

$$\frac{1}{\sqrt{2^k}} \sum_{y \in \{0,1\}^k} (-1)^{s_0 \cdot y}|y\rangle|x(s_0)\rangle \tag{8.7}$$

Case 2; $x \in D_1$: we have $\frac{1}{\sqrt{2}}(|s_0\rangle + |s_0 \oplus a\rangle)|x(s_0)\rangle$ so applying the Hadamard lemma we get:

$$\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2^k}}\left(\sum_y (-1)^{s_0 \cdot y}|y\rangle + \sum_y (-1)^{(s_0+a)\cdot y}|y\rangle\right)\right)|x(s_0)\rangle \tag{8.8}$$

$$= \frac{1}{\sqrt{2^{k+1}}} \sum_y (-1)^{s_0 \cdot y}(1 + (-1)^{a \cdot y})|y\rangle \tag{8.9}$$

Now, measure the first register in the computational basis.

Case 1; $x \in D_0$: Consider $\left\{|z\rangle\langle z| | z \in \{0,1\}^k\right\}$, which just picks out one term from the sum, so:

$$\left\| |z\rangle\langle z| \frac{1}{\sqrt{2^k}} \sum_{y \in \{0,1\}^k} (-1)^{s_0 \cdot y}|y\rangle \right\|^2 = \frac{1}{2^k} \tag{8.10}$$

where the post-measurement state is z.

Case 2; $x \in D_1$: We have:

$$\left\| |z\rangle\langle z| \frac{1}{\sqrt{2^{k+1}}} \sum_y (-1)^{s_0 \cdot y}(1 + (-1)^{a \cdot y})|y\rangle \right\|^2 \tag{8.11}$$

$$= \frac{1}{2^{k+1}} \left\| \sum_y (-1)^{s_0 \cdot y}(1 + (-1)^{a \cdot y})\delta_{yz} \right\|^2 \tag{8.12}$$

$$= \frac{1}{2^{k+1}} \left\| (1 + (-1)^{a \cdot z}) \right\|^2 \tag{8.13}$$

$$= \begin{cases} \frac{1}{2^{k+1}} \cdot 4 = \frac{1}{2^{k-1}} & z \cdot a = 0 \\ 0 & z \cdot a = 1 \end{cases} \tag{8.14}$$

The first Lemma we have the measurement outcome uniformly random in $z \in \mathbb{F}_2^k$. In the second lemma we have the measurement outcome is uniformly random $z \in \mathbb{F}_2^k$ such that $z \cdot a = 0$.

We now repeat the process $K \in \mathbb{N}$ times (K-query algorithm). We choose $K = k + 1000$. In the first case, we have the probability:

$$Pr[\dim \operatorname{span}(z_1, \ldots, z_k) = k] \geq 1 - 2^{k-K} = 1 - 2^{-1000} \tag{8.15}$$

In the second case the probability is zero, and we are done. □

Some remarks:

1. A slight modification of the algorithm allows you to retrieve the a in the case of $x \in D_1$, by using the equation $z \cdot a = 0$ and solving the linear system of $Aa = 0$.

2. We consider the modification $\text{Simon}'_n : D' \subseteq (\mathbb{F}_2^k)^{\mathbb{F}_2^k} \to \mathbb{F}_2^k$. In fact this generalizes widely to algebraic groups (which leads to Shor, as we will discuss next time). Where $x \in F'$ iff $\exists a \in \mathbb{F}_2^k \setminus \left\{0^k\right\}$ such that $\forall s, t \in \mathbb{F}_2^k$, $x(s) = x(t) \iff s = t$ and $\text{Simon}'_n(x) =$ the "a" associated with x. By the remark above, $Q(\text{Simon}'_n) = O(k)$.

Proposition: Randomized query complexity of Simon's problem

$R(\text{Simon}_n) \geq \Omega(\sqrt{n})$.

Lemma

Suppose $f : D = D_0 \cup D_1 \subseteq \Gamma^n \to \{0,1\}$ such that $f(D_0) = \{0\}$ and $f(D_1) = \{1\}$. Suppose μ_0 is a (probability) distribution supported on D_0, and μ_1 a distribution supported on D_1. Let μ denote the distribution:

1. $b \leftarrow \{0,1\}$
2. $x \leftarrow \mu_b$.

Let $P \subseteq D_1$. Suppose that for all $b \in \{0,1\}$:

$$Pr[T(x) = b | x \leftarrow \mu_0] = Pr[T(x) = b | x \in P, x \leftarrow \mu_1] \tag{8.16}$$

then

$$Pr[T(x) = f(x) | x \leftarrow \mu] \leq \frac{1}{2} + \frac{1}{2} Pr[x \notin P | x \leftarrow \mu_1]. \tag{8.17}$$

Case to consider for intuition; if P is the entirety of D_1, T cannot distinguish between μ_0 and μ_1, so has a hard time computing f.

9 Randomized Query Complexity of Simon's Problem

Proof.

$$\begin{aligned} Pr[T(x) = f(x) | x \leftarrow \mu] &= \frac{1}{2} Pr[T(x) = f(x) | x \leftarrow \mu_0] + \frac{1}{2} Pr[T(x) = f(x) | x \leftarrow \mu_1] \\ &= \frac{1}{2} Pr[T(x) | x \leftarrow \mu_0] + \frac{1}{2}(Pr[T(x) = 1 | x \in P_1, x \leftarrow \mu_1) Pr[x \in P_1 | x \leftarrow \mu_1] \\ &+ Pr[T(x) = 1 | x \notin P_1, x \leftarrow \mu_1) Pr[x \notin P_1 | x \leftarrow \mu_1]) \\ &\leq \frac{1}{2} Pr[T(x) = 0 | x \leftarrow \mu_0] + \frac{1}{2} Pr[T(x) = 1 | x \leftarrow \mu_0] + \frac{1}{2} Pr[x \notin P_1 | x \leftarrow \mu_1] \\ &= \frac{1}{2} + \frac{1}{2} Pr[x \notin P_1 | x \leftarrow \mu_1] \end{aligned}$$

□

We now move to the proof of the proposition.

Proof. By averaging argument (Yao's principle), Suppose there's an RDT τ of depth d, then $\forall \mu$ distribution on D, $\exists$ DDT T such that:

$$Pr[T(x) = f(x) | x \leftarrow \mu] \geq 1 - \epsilon \tag{9.1}$$

where $\epsilon = \frac{1}{3}$.

Define μ_0, μ_1 where μ_b is supported on D_b and define μ as in the Lemma. μ_0 is uniformly randomly choosing a permutation on $\{0, 1, \ldots, n-1\}$ and μ_1 is uniformly randomly choosing $a \leftarrow \{0,1\}^k \setminus \{0\}$.

Now, choose a uniformly random element for each pair $\{x, x \oplus a\}$ from $\{0, 1, \ldots, n-1\}$.

WLOG, assume that T never queries the same variable twice, and that it is abalanced tree of depth D.

Let's consider the sequence of d responses that T gives, i.e. the labels of the edges of the decision tree. If $x \leftarrow \mu_0$, then we obtain uniformly random sequence of d distinct elements in $\{0, 1, \ldots, n-1\}$.

If instead $x \leftarrow \mu_1$: Let $t \in \{1, \ldots, d\}$, and $v_1, \ldots v_{t-1} \in \{0, 1, \ldots, n-1\}$ distinct. Let $s_1, \ldots s_t$ denote the sequence of indices that T queries on x, given $x(s_i) = v_i$ for $i \in \{1, \ldots, n-1\}$. We will say tha the sequence $x(s_1), x(s_2), \ldots x(s_t)$ is good if all values are distinct.

$$Pr[x(s_1), \ldots, x(s_t) \text{ is good} | x(s_1) = v_1, x(s_2) = v_s, \quad x(s_{t-1}) = v_{t-1}] \tag{9.2}$$
$$= Pr[x(s_t) \notin \{x(s_1), \ldots, x(s_t)\} |''] \tag{9.3}$$
$$= Pr[a(x) \notin \{s_1 \oplus s_2, s_1 \oplus s_3, \ldots, s_1 \oplus s_{t-1}, s_2 \oplus s_3, \ldots, s_{t-2} \oplus s_{t-1}\} |''] \tag{9.4}$$

How many elements in the above set? It is at most $\binom{t-1}{2}$. We choose randomly from $\{0,1\}^k \setminus \{0\}$ and want to avoid a set of size $\binom{t-1}{2}$, so the probability is:

$$Pr[x(s_1), \ldots, x(s_t) \text{ is good} | x(s_1) = v_1, x(s_2) = v_s, \quad x(s_{t-1}) = v_{t-1}] \tag{9.5}$$
$$\geq 1 - \frac{t-1}{2^k - 1 - \binom{t-1}{2}} \tag{9.6}$$

which can be obtained by looking at the complement event and bounding it via the union bound. Since the above analysis holds for any choice of distinct $v_1, \ldots v_{t-1}$, for $1 \leq k \leq d$ we have:

$$Pr[x \text{ is } t \text{ good} | x \text{ is } t-1 \text{ good}] \geq 1 - \frac{t-1}{2^k - 1 - \binom{t-1}{2}} \tag{9.7}$$

Therefore:

$$
\begin{aligned}
&Pr[x \text{ is } d\text{-good}] \\
&= Pr[x \text{ is } d\text{-good} | x \text{ is } (d-1) \text{ good}] Pr[x \text{ is } (d-1) \text{ good}] + Pr[x \text{ is } d\text{-good} | x \text{ is not } (d-1) \text{ good}] Pr[x \text{ is not } (d-1) \text{ good}] \\
&= Pr[x \text{ is } d\text{-good} | x \text{ is } (d-1) \text{ good}] Pr[x \text{ is } (d-1) \text{ good}] \\
&= Pr[x \text{ is } d\text{-good} | x \text{ is } (d-1) \text{ good}] Pr[x \text{ is } (d-1) \text{ good}] Pr[x \text{ is } d-1\text{-good} | x \text{ is } (d-2) \text{ good}] Pr[x \text{ is } (d-2) \text{ good}] \\
&= \ldots \\
&\geq \left(1 - \frac{d-1}{2} 2^k - 1 - \binom{d-1}{2}\right) \left(1 - \frac{d-2}{2^k - 1 - \binom{d-2}{2}}\right) \ldots \left(1 - \frac{d-(d-2)}{2^k - 1 - \binom{d-(d-2)}{2}}\right) \cdot 1 \\
&\geq 1 - \sum_{j=1}^{d} \frac{j-1}{2^k - 1 - \binom{j-1}{2}}
\end{aligned}
$$

Now assume WLOG that $1 + \binom{d-1}{2} \leq \frac{2^k}{2}$ or else we're done. Then:

$$Pr[x \text{ is } d\text{-good}] \geq 1 - \frac{1}{2^k - 1 - \binom{d-1}{2}} \sum_{j=1}^{d} (j-1) \geq 1 - \frac{2}{2^k} \frac{1}{2} d(d-1) \geq 1 - \frac{d^2}{2k} \tag{9.8}$$

Conditioned on x being d-good, the sequence of d responses to the d queries that T makes is a uniformly random sequence of d distinct elements. Therefore, if we let $P_1 := \{x \in D_1 | x \text{ is } d\text{-good}\}$ then:

$$Pr[T(x) = b | x \leftarrow \mu_0] = Pr[T(x) = b | x \leftarrow \mu_1, x \in P_1]. \tag{9.9}$$

So by our Lemma:

$$Pr[T(x) = \text{Simon}_n(x)] \leq \frac{1}{2} + \frac{1}{2}\frac{d^2}{2k} \tag{9.10}$$

in order to be $\leq 1 - \epsilon$ we have that $d \geq \sqrt{\frac{2k}{3}} = \Omega(\sqrt{n})$ and we are (finally) done. □

Remark: Simon's problem is in some sense very similar to the collision problem. A research question of interest to Daochen; can we define intermediate problems between Simon's problem and the collision problem? More precisely, $D_0^{\text{Collision}} = D_0^{\text{Simon}}$ and $D_1^{\text{Collision}} \supseteq D_1^{\text{Simon}}$. We've seen that $Q(\text{Collision}) = O(n^{1/3})$ and $R(\text{Collision}) = \Omega(\sqrt{n})$ (It turns out $Q(\text{Collision}) = \Omega(n^{1/3})$). We also showed that $Q(\text{Simon}_n) \leq O(\log n)$ and $R(\text{Simon}_n) \geq \Omega(\sqrt{n})$. Question; in QC we often have quadratic or superpolynomial speedups. Can we go in between by defining particular subsets for intermediate speedups?

10 Period Finding

Definition: Period Finding Problem

Let $N \in \mathbb{N}$. Then, we define the period finding problem as:

$$\text{Period}_N : D \subseteq \mathbb{Z}_N^{\mathbb{Z}} \to \{1, 2, \ldots, N\} \tag{10.1}$$

such that $x \in D$ if and only if there exists $r \in \mathbb{N}$ such that $x(s + r) = x(s)$ for all $s \in \mathbb{Z}$.

Note the similarity with Simon's problem; we've replaced $\mathbb{F}_2^N$ with $\mathbb{Z}_N$. Note that $\mathbb{Z}_N$ are the integers modulo N, and we must have $r \leq N$.

Lemma

For $r \in \mathbb{N}$ such that $r > 100$, the number of elements in the set $\{0, 1, \ldots, r-1\}$ that are coprime to r is at least $\frac{r}{5\ln(\ln(r))}$. Two numbers are coprime if the only common factors of the two numbers are 1.

Definition: Quantum Fourier Transform

Let $M \in \mathbb{N}$. Th quantum fourier transform QFT_M on $\mathbb{C}^M$ is the unitary defined by:

$$|j\rangle = \frac{1}{\sqrt{M}} \sum_{k=0}^{M} \omega_M^k |k\rangle \tag{10.2}$$

where $\omega_M := e^{2\pi i/M}$.

Question where did the complex numbers come from? We didn't need it in Simon's. The short answer is that it's not necessarily the complex numbers that is important, but rather it is the minus sign that is important. $\{H, \text{Toffoli}\}$ is universal as a gate set, for example.

Proposition

$Q(\text{Period}_N) \leq O(\ln \ln N)$.

Proof. Let $n := \lceil 2 \log N \rceil + 1$ so that $2^n \geq N^2$. Write $2^N - 1 = Br + b$, where $B \in \{0, 1, 2, \dots\}$ with $0 \leq b \leq r - 1$. Now, create teh state:

$$\frac{1}{\sqrt{2^N}} \sum_{s=0}^{2^n-1} |s\rangle |x(s)\rangle \tag{10.3}$$

using one query (in the same way as we do for Simon's algorithm). Now, we measure the second register in the computational basis via $\Pi_i = \mathbb{I}_{2^N} \otimes |i\rangle\langle i|$ where $i \in \{0, 1, \dots, N-1\}$.

We are promises that x is periodic with period r. So, suppose that the outcome is $k \in \mathbb{Z}_N$. Suppose the outcome is $k \in \mathbb{Z}_N$, let $s_0 \in \{0, 1, \dots, r-1\}$ be such that $x(s_0) = k$. Then, the state of the first register after the measurement becomes $\frac{1}{\sqrt{A+1}} \sum_{k=0}^{A} |s_0 + kr\rangle$ where $A = B$ if $s_0 = b$ and $A = B - 1$ if $s_0 > b$ (this is just a technicality in case s_0 falls into not the last full period).

Pictorially, if we look at the amplitudes of the state, we have positive amplitudes at $s_0 + nr$, i.e. the quantum state has period of length r. If we now apply QFT_{2^n}, we get:

$$\frac{1}{\sqrt{2^n(A+1)}} \sum_{y=0}^{2^n-1} \omega^{s_0 \cdot y} \sum_{k=0}^{A} \omega^{kry} |y\rangle \tag{10.4}$$

where we shorthand $\omega = \omega_{2^n} = \exp(2\pi i / 2^n)$.

We then have:

$$\sum_{k=0}^{A} \omega^{kry} = 1 + \omega^{ry} + \omega^{2ry} + \dots \omega^{Ary} \tag{10.5}$$

As y goes from $0, 1, \dots, 2^n - 1$, $ry/2^n$ goes from $0, \dots, r, \frac{2^n-1}{2^n} \in (r-1, r)$. Now, for each $j \in \{0, 1, \dots, r-1\}$, let $y_j \in \{0, 1, \dots, 2^n - 1\}$ be such that $ry_j/2^n$ is closest to j. Then we have:

$$|\frac{ry_j}{2^n} - j| \leq \frac{1}{2}\frac{r}{2^n} \tag{10.6}$$

Side comment: Generally in textbooks the Kitaev version of the period finding algorithm is discussed. This is the Shor version, and we discuss it this way because its similar to Simon's problem.

And now we can write:

$$\frac{ry_j}{2^n} = j + \eta_j \tag{10.7}$$

where $|\eta_j| \leq \frac{r}{2^n+1} \leq \frac{N}{2N^2} = \frac{1}{2N}$ where we use that $r \leq N$ and $2^n > N^2$ (the intuition is more peaks = easier to extract period).

Then:

$$S_j = \sum_{k=0}^{A} \omega^{kry_j} = \sum_{k=0}^{A} \omega^{k2^n(j+\eta_j)} = \sum_{k=0}^{A} \omega^{k_j 2^n} \cdot \omega^{k2^n\eta_j} = \sum_{k=0}^{A} \omega^{k2^n\eta_j} = \sum_{k=0}^{A} \exp(2\pi i k \eta_j) \tag{10.8}$$

Two cases:

1. $\eta_j = 0$. Then $S_j = A + 1$.

2. $\eta_j \neq 0$. Then $|S_j|^2 = |\frac{1-\exp(2\pi i(A+1)\eta_j)}{1-\exp(2\pi i \eta_j)}|^2 = \frac{\sin^2(\pi(A+1)\eta_j)}{\sin^2(\pi\eta_j)} \geq \frac{\sin^2(\pi(A+1)\eta_j)}{\pi^2\eta_j^2}$

Now, $|\pi(A+1)\eta_j| = \pi(A+1)\eta_j \leq \pi(B+1)\frac{r}{2^{n+1}} \leq \frac{\pi}{2} + \frac{\pi r}{2^{n+1}} < \frac{\pi}{2} + \frac{\pi N}{2N^2} = \frac{\pi}{2} + \frac{\pi}{2N}$.

If we now assume $N > 100$, $|\pi(A+1)\eta_j| \leq 0.505\pi$, but $\sin^2\theta \geq \theta^2/3$ for $\theta \in [-0.505\pi, 0.505\pi]$ and so:

$$|S_j|^2 \geq \frac{\pi^2(A+1)^2\eta_j^2}{\pi^2\eta_j^2}\frac{1}{3} = \frac{(A+1)^2}{3}. \tag{10.9}$$

so in both cases $|S_j|^2 \geq \frac{(A+1)^2}{3}$. Therefore, if we measure the state:

$$\frac{1}{\sqrt{2^n(A+1)}}\sum_{j=0}^{2^n-1}\omega^{x_0 y}\sum_{k=0}^{A}\omega^{kry}|y\rangle \tag{10.10}$$

in the computational basis, the probability of the outcome being y_j for some j such that $j \in \{0,1,\ldots,r-1\}$ and j is coprime for r is at least:

$$\frac{r}{5\ln\ln r}\frac{1}{2^n(A+1)}\frac{(A+1)^2}{3} \geq \frac{1}{\ln\ln r}\cdot 0.05 \tag{10.11}$$

The final step of this algorithm is as follows; let $z \in \{0,1,\ldots,2^n-1\}$ be the outcome of the measurement. We compute $r' \in \mathbb{N}$, $r' \leq N$ and $j' \in \{0,1,\ldots,r'-1\}$ that is coprime to r' such that:

$$|\frac{z}{2^n} - \frac{j'}{r'}| \leq \frac{1}{2}\cdot\frac{1}{2^n} \tag{10.12}$$

intuition; if j', r' coprime then we have a unique fractional form. Two cases:

1. If no such r', j' exist, repeat the entire procedure
2. If they do exist, verify r' is in fact the period. Rand $r_{\text{candidate}}$ to be r' if $r' \leq r_{\text{candidate}}$.

Repeat this $10000\ln\ln r$ times and output $r_{\text{candidate}}$. With probability ≥ 0.99, one of the outputs will correspond to the j that is coprime to r. (the probability of j being coprime to r is at most $0.05\frac{1}{\ln\ln r}) \geq 0.05\frac{1}{\ln\ln N}$ so the probability of all outputs j not coprime to r is $\leq \left(1 - 0.05\frac{1}{\ln\ln r}\right)^{10000\ln\ln N} \leq e^{-500} \leq \frac{1}{3}$).

Claim: If j coprime to r, then the procedure described above will output an r' such that $r' = r$.

□

11 Period Finding Continued, Factoring

Review/Finishing the period finding problem

Consider $N \in \mathbb{N}$, a nd a function $x : \mathbb{Z} \to \mathbb{Z}_N$. There exists $r \in \mathbb{N}$ such that $x(s) = x(t) \iff s = t + r$.

1. $n = \lceil 2\log_2 \rceil + 1$ so $2^n > N^2$.
2. Consider $\frac{1}{\sqrt{2^n}}\sum_{s=0}^{2^n-1}|s\rangle$.
3. Apply O_x to this to get $\frac{1}{\sqrt{2^n}}\sum_{s=0}^{2^n-1}|s\rangle|x(s)\rangle$.
4. Measure the 2nd register in the computational basis. Let the outcome be $x_D \in \mathbb{Z}_N$:

$$\mathbb{I}_{2^n} \otimes |x_0\rangle\langle x_0|\frac{1}{\sqrt{2^n}}\sum_{s=0}^{2^n-1}|s\rangle|x(s)\rangle = \frac{1}{\sqrt{2^n}}\sum_{s=0}^{2^n-1}|s\rangle|x_0\rangle\delta_{x_0,x(s)} \tag{11.1}$$

Pictorially we have peaks at $s_0 + nr$. Last time, a lot of the detail was that we did not assume $r|2^n$. But if we did, the analysis would become much simpler, and we get:

$$\frac{1}{\sqrt{B}} \sum_{k=0}^{B-1} |s_0 + kr\rangle \tag{11.2}$$

where $2^n - 1 = Br - 1$ and $B \in \{0, 1, 2, \ldots\}$.

5. Now, to recover r we take the QFT_{2^n} of the first register. This yields:

$$\rightarrow \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |j\frac{2^n}{r}\rangle \tag{11.3}$$

note that the above expression only makes sense if $r|2^n$, but for this simplified analysis we do assume it (the analysis from the previous lecture is more sophisticated, but also more general). There is also technically a complex phase factor on each of the terms in the sum above, but when we measure it in the computational basis (in the next step) since these have unit absolute value these will not factor into the analysis.

Note the QFT is necessary because before we apply it, the state depends on s_0 (which depends the measurement outcome x_0) so when we repeat this procedure generically we get different results and cannot extract the r. But the QFT removes this dependence.

6. Measure this state in the computational basis. If we do so, our outcome will be of the form $\frac{j}{r}2^n$.

7. Continuing from last time (with the more generic analysis); the probability that we measure some y_j such that:

   (a) $|y_j - \frac{2^n j}{r}| \leq \frac{1}{2}\frac{1}{2^n}$
   (b) j is coprime to r

   is at least $0.05\frac{1}{\ln \ln r} \geq 0.05\frac{1}{\ln \ln N}$. Of course in the simplified case we have that the first condition is trivially satisfied because the y_js are literally $\frac{j}{r}2^n$.

8. Repeat the above steps $10000 \ln \ln N$ times. Each time, let z be the measurement outcome. Compute some $r' \leq N$ ($r' \in \mathbb{N}$) and $j' \in \{0, 1, \ldots, r' - 1\}$ such that $|z - \frac{2^n j'}{r'}| \leq \frac{1}{2}\frac{1}{2^n}$. If r', j' exist, and $r' \leq r_{\text{candidate}}$, set $r_{\text{candidate}} = r'$ (Note we should initialize $r_{\text{candidate}} = N + 1$). At the end, output $r_{\text{candidate}}$.

   Claim: Suppose the measurement outcome $z = y_j$ where $j \in \{0, 1, \ldots, r - 1\}$ is coprime to r. Then, the r' we output will be exactly equal to r.

Proof. We have $|z - \frac{2^n j}{r}| \leq \frac{1}{2}\frac{1}{2^n}$. Now suppose $j' \in \{0, 1, \ldots, r' - 1\}$ or $r' \in \mathbb{N}$, $r' \leq N$ such that $|z - \frac{2^n j'}{r'}| \leq \frac{1}{2}\frac{1}{2^n}$. We make a uniqueness argument. We have by the triangle inequality:

$$|\frac{2^n j}{r} - \frac{2^n j'}{r'}| \leq \frac{1}{2^n} \tag{11.4}$$

Then:

$$|\frac{jr' - j'r}{rr'}| \leq \frac{1}{2}\frac{1}{2^n} < \frac{1}{2N^2} \tag{11.5}$$

But $rr' \leq N^2$, so if $jr' \neq j'r$, then the LHS is greater than $\frac{1}{N^2}$, a contradiction. So, $jr' = j'r$. So $r|jr'$ and $r'|j'r$, but the first implies $r|r'$ as j and r are coprime, and the second implies that $r'|r$ as r' is coprime to j' by how it was computed. So then $r = r'$. □

The probability that one of the repeats catches the correct r is high, and so we are done.

Order Finding and Factoring

So the above period finding algorithm is $O(\ln \ln N)$, and it is a key component of Shor's factoring algorithm. Note that we did a query model analysis, and to actually do the quantum algorithm we would need to find the corresponding quantum gates to actually perform it. That is to say, we need to describe a circuit for the unitaries U_j and the operator O_x.

One concrete problem we can establish these for is the order finding problem. Here, the input is $a, N \in \mathbb{N}$ with $a \leq N$ and a coprime to N. The desired output is the minimal $r \in \mathbb{N}$ such that $a^r = 1 \mod N$.

Note such an r exists because we have $a^1, a^2, \ldots, a^{N+1} \mod N$ that take on at most N values, so two must be the same, hence $a^i = a^j \mod N$ for some i, j. Then using a coprime to N, $a^{j-i} = 1 \mod N$.

Proposition

$\exists$ a quantum algorithm that solves this problem in time $O((\log N)^3 \text{poly}(\log \log N))$.

How does this relate to factoring? Let us input the N we are trying to factor. We choose some a uniformly at random from $\{0, 1, \ldots, N-1\}$.

1. If a not coprime to N, we are done - just can do the Euclidean algorithm.
2. If a coprime to N, then run the order finding algorithm to find r such that $a^r = 1 \mod N$. Let's assume r is even so $a^{r/2} + 1 \neq 0 \mod N$. Then $a^r - 1 = 0 \mod N$ so $(a^{r/2} - 1)(a^{r/2} + 1) = 0 \mod N$. So then $N | (a^{r/2} - 1)(a^{r/2} + 1)$, and there should be nontrivial factors of N in the two parts. Then we can take the GCD.

   Note that the probability that the assumption in the last step is satisfied is some constant, by number theoretic arguments.

This is why we care about the order finding problem - because there is a fully classical reduction from it to factoring.

Proof. (Sketch) Consider $x : \mathbb{Z} \to \mathbb{Z}_N$ defined by $x(k) = a^k \mod N$. Then $x(s) = x(t) \iff s = t + \text{ord}_N(s)$. If correct, $a^s = a^t \mod N$ so then $a^{s-t} = 1 \mod N$ tso then $\text{ord}_N(a) | s - t$. $\square$

How does this relate to period finding? Consider the function $x : \mathbb{Z} \to \mathbb{Z}_N$ defined by

12 Time Complexity of Shor's Algorithm

A remark; a quantum state is denoted as $|v\rangle \in \mathbb{C}^d$. When we think about this, we shouldn't think about this vector as written down on a piece of paper, but rather as a complex probability distribution. If we think about the probabilistic setting - if we have n fair coins, then the state of the system can be described as a (uniform) vector on $\mathbb{R}^{2n}$.

In Shor's algorithm for example, we deal with states $\in \mathbb{C}^{2n}$ where $n \sim 2 \log N$ where N is the factor we are trying to factor. The proof of Shoir's algorithm is that it scales poly$n$.

As a reminder - in the order finding problem, given $a, N \in \mathbb{N}$ with $a < N$ coprime to N the order of a mod N is the least $r \in \mathbb{N}$ such that $a^r = 1 \mod N$. The goal is to find r.

Proposition: Time Complexity of order finding

$\exists$ a quantum algorithm that solves the order finding problem in time $O(\log^3 N \log \log N)$.

Proof. 1. Consider the function $x : \mathbb{Z} \to \mathbb{Z}_N$ defined by $x(s) = a^s \mod N$. This x satisfies $x(s) = x(t) \iff s - t \in r\mathbb{Z}$.

$\boxed{\Longleftarrow}$ $s = t + rk$, $k \in \mathbb{Z}$. So, $x(s) = a^{t+rk} = a^t(a^r)^k \mod N = a^t \mod N = x(t)$.

$\boxed{\Longrightarrow}$ Write $s - t = qr + c$ where $q \in \mathbb{Z}$ and $c \in \{0, 1, \dots, r-1\}$. Then:

$$\begin{aligned}
&x(s) = x(t)\\
\Longrightarrow\ &a^s = a^t \mod N\\
\Longrightarrow\ &a^{s-t} = 1 \mod N\\
\Longrightarrow\ &a^{qr+c} = 1 \mod N\\
\Longrightarrow\ &a^c = 1 \mod N\\
\Longrightarrow\ &c = 0 \text{ by minimality of period } r\\
\Longrightarrow\ &s - t \in r \in \mathbb{Z}
\end{aligned}$$

The cost of computing $x(s)$ - the maximum s is 2^n where $n = \lceil 2\log_2 N \rceil + 1$. Consider repeated squaring to compute a^k:

$$a \mod N \to a^2 \mod N \to (a^2)^2 \mod N \to a^8 \to \dots \to a^k \tag{12.1}$$

We then have $(\log N)^2 \log k \le (\log N)^2 n \le O((\log N)^3)$. So, we have the time complexity of computing x, which corresponds exactly to the time complexity of instantiating the quantum oracle for x - i.e. the O_x part of the algorithm.

2. Now, we need to account for the non-O_x part of the query algorithm.

   (a) We need to consider the Hadamard transform that takes us from $|0\rangle^{\otimes n}$ to $\frac{1}{\sqrt{2^n}} \sum_{s=0}^{2^n-1} |s\rangle$ where the number of steps is n (we apply n Hadamard transforms).

   (b) QFT: There exists a turing machine that on input $n \in \mathbb{N}$ outputs the description of quantum circuit that implements a unitary approximating QFT_{2^n} to error ϵ in operator norm distance (i.e. $\|U - QFT_{2^n}\| \le \epsilon$) in order $O(n^2 \text{poly}(\log(n/\epsilon)))$ steps. Reference: Wikipedia + Solovay-Kitaev theorem (wikipedia uses non-standard gates, and SK theorem allows for approximation of said non-standard gates with logarithmic cost in terms of ϵ). Apply with $\epsilon = \frac{10^{-10}}{\log\log N} = O(\frac{1}{\log n})$. So total number of steps to describe the QFTS is $O(n^2 \text{poly}(\log n))$.

   (c) There is a final step; generate $z \in \{0,1\}^n$ such that with probability $0.01 \frac{1}{\log\log N}$ we have $|\frac{z}{2^n} - \frac{j}{r}| < \frac{1}{2 \cdot 2^n}$ where j coprime to r. Then extract r by unmerating $r' \in \mathbb{N}, r \le N$, $j' \in \{0, 1, \dots, r'-1\}$ coprime to r'. But this costs N. So naively we can do no better than the classical algorithm. But we can do better! Consider the **continued fractions algorithm**: There exists a Turing machine that on input $b_1 \dots b_n \in \{0,1\}^n \ne 0^n$ outputs all of the convergents(?) (p_i, q_i) of $\phi = 0b_1b_2\dots b_n \in (0,1)$ in $O(n^3)$ steps, where $p_i, q_i \in \mathbb{N}, p_i/q_i \le 1$ and p_i is coprime to q_i for all i. Moreoever, suppose there exists coprime $p < q \in \mathbb{N}$ usch that:

$$|\phi - \frac{p}{q}| \le \frac{1}{2q^2} \tag{12.2}$$

   then there exists $a \in [k]$ such that $p = p_a$ and $q = q_a$. Recall $|\frac{z}{2^n} - \frac{j}{r}| \le \frac{1}{2 \cdot 2^n} < \frac{1}{2N^2} \le \frac{1}{2r^2}$. As an example, cosnider $0.10010 \to \frac{1}{2} + \frac{1}{16} = \frac{9}{16}$. The convergents are:

$$\frac{9}{16} = \frac{1}{\frac{16}{9}} = \frac{1}{1 + \frac{7}{9}} = \frac{1}{1 + \frac{1}{\frac{9}{7}}} = \frac{1}{1 + \frac{1}{1 + \frac{2}{7}}} = \frac{1}{1 + \frac{1}{1 + \frac{1}{3 + \frac{1}{2}}}} \tag{12.3}$$

To get the convergents, we just draw circles; we have $\frac{1}{1} = \frac{p_1}{q_1}$ in the first step, then $\frac{1}{1+\frac{1}{1}} = \frac{1}{2} = \frac{p_2}{q_2}$ in the second step, and $\frac{1}{1+\frac{1}{1+\frac{1}{3}}} = \frac{4}{7} = \frac{p_3}{p_4}$ in the third step, and so on. Then in the last step we have $\frac{p_4}{q_4} = \frac{9}{16}$. This is the algorithm for computing the p_i, q_is. The guarantee is that in the good case of the zs, the $p/q = r$.
So, the cost of each repeat is:

$$n + (\log N)^2 + 1 + n^2 \mathrm{poly}(\log n) + n^3 \tag{12.4}$$

where the first term is the Hadamard, the second term is the oracle, the third term is the computational basis measurement, the fourth term is the QFT, and the last term is the continued fractions. The total number of repeats is $\log\log(N)$, so then we are dominated by the last term and have a total cost of $O(n^3 \log n)$.

□

Algorithm: Quantum Factoring (Shor)

Input: $N \in \mathbb{N}$ (represented in binary)

1. Check if N is even. If even, output 2.
2. Check if N is a kth power of a natural number $2, \ldots, \lceil \log N \rceil$.
3. Choose $a \leftarrow \{1, \ldots, N-1\}$. Compute $b = \gcd(a, N)$ by Euclid's algorithm. If $b > 1$, then output b, else continue
4. Compute $r = \mathrm{ord}_N(a)$.
5. Compute $d = \gcd(a^{r/2} - 1 \mod N, N)$. If $d > 1$. If $d > 1$, output d, else output "don't know". We are guaranteed that the algorithm doesn't output "don't know' with probability at least $1/2$, in which case it is constant.

Anecdote - Daochen thought he had broken Shor when he tried to code this up without the $\mod N$. Important to have! Though left out by many experts…

13 Finishing Up Shor, The Hidden Subgroup Problem

Final Remarks on Shor's Algorithm

We looked at the order finding algorithm last time. The runtime was $\tilde{O}((\log N)^2)$ for the classical part and $\tilde{O}((\log N)^3)$ for the quantum part (naively). This came from the repeated squaring of $x(s) = a^s \mod N$. But we can actually do this in $(\log N)^2$ time if we do a Fourier transform instead of doing the "schoolboy" multiplication. Then, the complexity of each squaring is just $\tilde{O}(\log N)$ and so all of the squarings takes $\tilde{O}((\log N)^2)$ time.

But last year, Oded Regev actually found a way to do this in $\tilde{O}((\log N)^{1.5})$ time. He reduced the complexity of the quantum part!

Idea: What if instead of choosing a, we chose a, b, c, d. Consider $x(s_1, s_2, s_3, \ldots s_k) = a_1^{s_1} a_2^{s_2} \ldots a_k^{s_k} \mod N$ where k is to be optimized. This has a period in a higher dimensional space of $\mathbb{Z}_k$. For each of the s_i, we don't have to go as far to find the period; in a higher-dimensional space the period is shorter, so we can raise the numbers to a smaller power and so this gives us some savings.

Basic Group Theory

Definition: Groups

A group $G = (S, \alpha)$ is defined by a set S and a function $\alpha : S \times S \to S$ such that:

1. (Identity) $\exists e \in S$ such that $\forall g \in S$, $\alpha(g, e) = g = \alpha(e, g)$
2. (Inverses) $\forall g \in S$, $\exists h \in S$ such that $gh = e = hg$.
3. (Associativity) $\forall g, h, k \in S$ such that $\alpha(\alpha(g, h), k) = \alpha(g, \alpha(h, k))$.

We didn't include it in the axiom, but the inverse is unique. The *order* of the group is $|S|$. Sometimes the notation is abused and we write $|G|$. A group G is Abelian if the α operation commutes i.e. $\forall g, h \in S$ we have $\alpha(g, h) = \alpha(h, g)$.

Some examples:

1. $G = \mathbb{F}_2^n$, with $\alpha =$ componentwise addition.
2. $G = \mathbb{Z}$, with $\alpha =$ addition.
3. $G = GL_n(\mathbb{C})$ the set of invertible $n \times n$ matrices with entries in $\mathbb{C}$ (with $\alpha =$ matrix multiplication.)
4. $G = D_{2n}$, the set of symmetries of a regular n-gon. A regular 3-gon is a triangle, a regular 4-gon is a square, a regular 5-gon is a pentagon, and so on. Let's take the square/4-gon as an example. The symmetries are the rotations and reflections $\left\{0^\circ = e, 90^\circ = \sigma, 180^\circ = \sigma^2, 270^\circ = \sigma^3, \tau, \tau\sigma, \tau\sigma^2, \tau\sigma^3\right\}$. The group operation is composition of the symmetries. D_{2n} can be thus written as $\{\sigma^s \tau^a | s \in \mathbb{Z}_n, a \in \mathbb{Z}_2\}$. It will later be convenient to use the slightly different representation of the element. We can write the elements as (s, a) where $(s, a) \cdot (t, b) = (s + (-1)^a t, a + b)$.

1, 2 are examples of Abelian groups, while 3,4 are not.

Definition: Subgroups

Let $G = (S, \alpha)$ be a group. We say $T \subseteq S$ form s a subgroup of G if:

1. T contiains the e of G
2. T is closed under α, that is $\forall g, h \in T$ we have $\alpha(g, h) \in T$.
3. T is inverse closed, i.e. $\forall g \in T$ we have $g^{-1} \in T$.

This definition defines a group $(T, \alpha|_T)$ (where $\alpha|_T$ denotes the restriction of the α operation onto $T \times T$ (rather than $S \times S$)).

Examples:

1. Any group G has the trivial subgroups $\{e\}, G$.
2. Suppose $G = \mathbb{F}_2^n$. Then, $\{a, (0, \dots, 0)\}$ where $0^n \neq a \in \mathbb{F}_2^n$. This follows from the fact that $a0^n = a$, $0^n 0^n = 0^n$, $aa = 0^n$ and so the group is closed, has the identity element, and contains all inverses, as required.
3. Take $G = \mathbb{Z}$. Then, $r\mathbb{Z} = \{r \cdot z | z \in \mathbb{Z}\}$ for $0 \neq r \in \mathbb{Z}$ is a subgroup.
4. For $y \in \mathbb{Z}_n$, $T_y := \{(0, 0), (y, 1)\}$ is a subgroup of D_{2n}, as $(y, 1) \cdot (y, 1) = (0, 0)$.

Definition: Hidden Subgroup Problem

Let G be a group and $\mathcal{H}$ denote a set of subgroups of G. Let Σ be an alphabet with $|\Sigma| \geq |G|$. The HSP$(G, \mathcal{H}, \Sigma)$ problem is to compute the function $f : D \subseteq \Sigma^G \to \mathcal{H}$ where $x \in D$ if and only if there exists $H \in \mathcal{H}$ with $x(g) = x(h) \iff gH = hH$ where $gH =:= \{gh | h \in H\}$

Proposition: Quantum query complexity of HSP

Let G be a finite group. Let $\mathcal{H}$ be the set of all subgroups of G and let be Σ be an alphabet such that $|\Sigma| \geq |G|$. Then, $Q(\text{HSP}(G, \mathcal{H}, \Sigma)) = O((\log|G|)^2)$.

Example: Simon's problem, with $G = \mathbb{F}_2^n \to O(n^2)$.

Interlude: Mixed Quantum States

So far:

1. A quantum state is a vector $|\psi\rangle \in \mathbb{C}^d$
2. A Γ-outcome measurement on $\mathbb{C}^d$ applied to $|\psi\rangle$ results in $i \in \Gamma$ according to some probability p_i. The state becomes $|\psi_i\rangle$ if $i =$ measurement outcome.

In Simon's/Period finding problems:

1. Either we did not care about the measurement outcome (first measurement)
2. Or we did care about the distribution on the measurement outcome, but not the distribution on the state it ends up in $|\psi_i\rangle$ (second measurement)

In either case, we didn't care about the distribution on the post-measurement state. We have post measurement state $|\psi_i\rangle, p_i$. There are two ways of doing analysis.

1. Naive but hard way: Do a for loop; for each $i \in \Gamma$, analyze the post-processing effect on the $|\psi_i\rangle$, then "average over the effects".
2. Sophisticated (but easier) way: define a single objkect that captures the distribution on the post-measurement states, and analyze the effects on the single object.

What is the mystery single object? Is is the density matrix $\rho \in \mathbb{C}^{d \times d}$, defined as:

$$\rho = \sum_{i \in \Gamma} p_i |\psi_i\rangle\langle\psi_i| \tag{13.1}$$

14 The Hidden Subgroup Problem Continued

Definition: Mixed quantum state

Let $d \in \mathbb{N}$. A mixed quantum state of dimension d is a matrix $\rho \in \mathbb{C}^{d \times d}$ such that:

1. H is positive semidefinite
2. $\text{Tr}(\rho) = 1$.

E.g. $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ where $|\psi_i\rangle$ are pure states. $\text{Tr}(\rho) = \sum_i p_i = 1$ and $\sum_i p_i |\langle\psi_i|\psi_i\rangle|^2 \geq 0$.

Definition: Effect of measurement on mixed quantum state

Let $d \in \mathbb{N}$, ρ be a mixed quantum state and $\mathcal{M} = \{\Pi_i | i \in \Gamma\}$ a measurement on $\mathbb{C}^d$. To measure ρ with $\mathcal{M}$ refers to a process that:

1. Outputs $i \in \Gamma$ wit probability $\mathrm{Tr}(\Pi_i \rho)$
2. The resulting state becomes $\frac{\Pi_i \rho \Pi_i}{\mathrm{Tr}(\Pi_i \rho)}$.

Definition: Schatten p-norm

Let $p \in [1, \infty)$ and $d \in \mathbb{N}$. The Schatten p-norm of a matrix $A \in \mathbb{C}^{d \times d}$ is defined to be:

$$\|A\|_p := \mathrm{Tr}((A^\dagger A)^{p/2})^{1/p} \tag{14.1}$$

(Recall the definition of functions of matrices, where if $A = \sum_i \lambda_i |i\rangle\langle i|$ then $f(A) = \sum_i f(\lambda_i)|i\rangle\langle i|$.) Also, when $p = \infty$, $\|A\|_\infty$ is the spectral norm of A.

Definition: Fidelity of quantum states

Let $\rho, \sigma \in \mathbb{C}^{d \times d}$ be mixed quantum states. The fidelity between ρ, σ is defined as:

$$F(\rho, \sigma) := \left\| \sqrt{\rho}\sqrt{\sigma} \right\|_1 \tag{14.2}$$

We observe that $F(\rho, \sigma) = F(\sigma, \rho)$ (symmetric) and $0 \leq F(\rho, \sigma) \leq 1$.

Lemma: Pretty good measurement

Let $N, d \in \mathbb{N}$. Let $\rho_1, \rho_2, \ldots, \rho_N \in \mathbb{C}^{d \times d}$ be mixed quantum states. Then there exists an $[N]-$outcome measurement $\mathcal{M} = \{\Pi_i | i \in [N]\}$ on $\mathbb{C}^d \otimes \mathbb{C}^N$ such that:

$$\mathrm{Tr}(\Pi_i(\rho_i \otimes |0\rangle\langle 0|)) \geq 1 - N\sqrt{\max_{i \neq j} F(\rho_i, \rho_j)} \tag{14.3}$$

Lemma: Holder's inequality for Schatten p-norms

Let $p \in [1, \infty]$, $d \in \mathbb{N}$, $A, B \in \mathbb{C}^{d \times d}$. Then:

$$\|AB\|_1 \leq \|A\|_p \|B\|_{p^*} \tag{14.4}$$

where $\frac{1}{p} + \frac{1}{p^*} = 1$.

Proof. Combine Von Neumann's Tracing inequality with the usual Holder's inequality for L^p norms. $\square$

Lemma

Let ρ, σ be mixed quantum states and Π_ρ be the projector onto the support of ρ. Then:

$$F(\rho,\sigma) \leq \sqrt{\mathrm{Tr}(\Pi_\rho \sigma)} \tag{14.5}$$

Proof.

$$\begin{aligned} F(\rho,\sigma) &= \left\| \sqrt{\rho}\sqrt{\sigma} \right\|_1 \\ &= \left\| \Pi_\rho \sqrt{\rho} \Pi_\rho \sqrt{\sigma} \right\|_1 \\ &\leq \left\| \Pi_\rho \sqrt{\rho} \right\|_2 \left\| \Pi_\rho \sqrt{\sigma} \right\|_2 \\ &= \sqrt{\mathrm{Tr}((\Pi_\rho\sqrt{\rho})^\dagger(\Pi_\rho\sqrt{\rho}))}\sqrt{\mathrm{Tr}((\Pi_\rho\sqrt{\sigma})^\dagger(\Pi_\rho\sqrt{\sigma}))} \\ &= \sqrt{\mathrm{Tr}(\sqrt{\rho}\Pi_\rho\Pi_\rho\sqrt{\rho})}\sqrt{\mathrm{Tr}(\sqrt{\sigma}\Pi_\rho\Pi_\rho\sqrt{\sigma})} \\ &= \sqrt{\mathrm{Tr}(\rho)}\sqrt{\mathrm{Tr}(\Pi_\rho\sigma)} \end{aligned}$$

□

Lemma

Let G be a finite group, $H, H' \leq G$. Then:

$$|gH \cap g'H'| = \begin{cases} |H \cap H'| & \text{if } g^{-1}g' \in HH' = \left\{hh' | h \in H, h' \in H'\right\} \\ 0 & \text{otherwise} \end{cases} \tag{14.6}$$

Moreover, $|HH'| = \frac{|H||H'|}{|H \cap H'|}$.

Lemma

Let G be a finite group. Let N be the number of subgroups of G. Then, $N \leq (|G|+1)^{\log_2|G|}$.

Proof. Consider a subgroup $H \leq G$. H can be specified by $\leq \log_2|H|$ independent generators. Start with $\langle e \rangle$, then $\langle e, h_2 \rangle = \left\{e, h_2, h_2^2, \dots\right\}$, then continue on and define h_{i+1} recursively by an element that can't be generated as a word of the previous elements. So, the number of subgroups is at most $\sum_{i=0}^{\log_2|G|}\binom{|G|}{i} \leq (|G|+1)^{\log_2|G|}$ □

We are now ready to prove the quantum query complexity of HSP proposition from last lecture.

Proof.

1. Create $\frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle|x(g)\rangle$. (1 query)
2. Measure the second register in the computational basis.

3. Consider the mixed state:

$$\rho = \frac{1}{|G|/|H|}\sum_{i=1}^{k}|g_iH\rangle\langle g_iH| \text{ where } |g_iH\rangle = \frac{1}{\sqrt{|H|}}\sum_{g\in g_iH}|g\rangle \tag{14.7}$$

Suppose we do this $l \in \mathbb{N}$ times. Then, $\rho_H^{\otimes l}$. And we consider:

$$\rho_{H_1}^{\otimes l}, \dots \rho_{H_N}^{\otimes l} \tag{14.8}$$

with $N = O(\log_2|G|)$. Use pretty good measurement lemma to find:

$$\mathrm{Tr}(\Pi_i(\rho_i \otimes |0\rangle\langle 0|)) \geq 1 - N\sqrt{\max_{i\neq j} F(\rho_i, \rho_j)} \geq \frac{2}{3} \tag{14.9}$$

so then:

$$1 - N\sqrt{\max_{i\neq j} F(\rho_{H_i}^{\otimes l}, \rho_{H_j}^{\otimes l})} \geq \frac{2}{3} \iff l \geq \Omega(\frac{\log_2 N}{\log(\max_{i\neq j}\frac{1}{F(\rho_{H_i},\rho_{H_j})})}) \tag{14.10}$$

It now suffices to show:

$$F(\rho_{H_i}, \rho_{H_j}) \leq \text{ Constant} \tag{14.11}$$

(Note: This last part takes place one lecture later) To this end we use the Lemma, and we can instead bound $\mathrm{Tr}(\Pi_{\rho_H}\rho'_H)$.

$$\begin{aligned}
\mathrm{Tr}(\Pi_{\rho_H}\rho'_H) &= \mathrm{Tr}(\frac{1}{|H|}\sum_{g} \in G|gH\rangle\langle gH| \cdot \frac{1}{|G|}\sum_{g'\in G}|g'H\rangle\langle g'H|) \\
&= \frac{1}{|H||G|}\sum_{g,g'\in G}\mathrm{Tr}(|gH\rangle\langle gH||g'H\rangle\langle g'H|) \\
&= |\frac{1}{\sqrt{H}}\sum_{h\in H}\langle gh|\frac{1}{\sqrt{H'}}\sum_{h'\in H}|g'h'\rangle|^2 \\
&= \frac{1}{|H||H'|}|\sum_{h\in H, h'\in H'}\langle gh|g'h'\rangle|^2 \\
&= \frac{1}{|H|^2|H'||G|}\sum_{g,g'\in G}|gH\cap g'H'|^2
\end{aligned}$$

The sum is equal to $|H\cap H'|^2$ if $g^{-1}g' \in HH'$ and 0 otherwise, so:

$$\begin{aligned}
\mathrm{Tr}(\Pi_{\rho_H}\rho'_H) &= \frac{1}{|H|^2|H'||G|}|G||Hh'| \\
&= \frac{|H\cap H'|}{|H|}
\end{aligned}$$

Now we want to bound this by a half. The fidelity is symmetric and we also have that it is less than $\frac{|H\cap H'|}{|H'|}$, from which we obtain that it must be less than a half (by using Lagrange's Theorem). The claim follows. □

15 Dihedral HSP

We have thus shown that the Quantum query complexity of HSP is at most $O((\log|G|)^2)$. However, we use a sophisticated measurement in the Lemma, and in general has no efficient description in terms of quantum circuits (unlike a computational basis measurement). So now, we will think about the HSP for the Dihedral group D_{2n}, which we will see that the measurement has a nontrivial construction. The difficulty boils down to non-Abelian, non-normal subgroups.

We recall $D_{2n} = \left\{\sigma^a \tau^b | a \in \mathbb{Z}_n, b \in \mathbb{Z}_2\right\}$. Where σ are rotational symmetries and τ are reflection symmetries of the regular n-gon. For convenience, we wrote this as $D_{2n} = \left\{(a,b) | a \in \mathbb{Z}_n, b \in \mathbb{Z}_2\right\}$. As a concrete example, $D_8 = \left\{e, \sigma, \sigma^2, \sigma^3, \tau, \sigma\tau, \sigma^2\tau, \sigma^3\tau\right\}$. Consider the subgroup:

$$\mathcal{H} = \left\{\langle (a,1) \rangle \, | a \in \mathbb{Z}_n\right\} \tag{15.1}$$

We focus on this particular set of subgroups due to its relevance in breaking lattice cryptography. The HSP only gets harder as we increase the size of $\mathcal{H}$.

Proposition

$T_{\text{non-oracular}}(HSP(D_{2n}, \mathcal{H}, \Sigma)) = 2^{O(\sqrt{\log n})}$.

Note: there are no complexity theoretic barriers to this (we don't solve NP problems in polynomial time, for example). If we got this to poly$(\log n)$, then we would break a lot of lattice-based cryptographic protocols.

Proof. Input is x which hides the subgroup $\langle (s,1) \rangle$.

1. Input $\frac{1}{\sqrt{2^n}} \sum_{a \in \mathbb{Z}_n, b \in \mathbb{Z}_2} |a,b\rangle |x(a,b)\rangle$

2. Measure the second register in the computational basis. Suppose we get $x_0 \in \Sigma$, then the state of the first register becomes:

$$\frac{1}{\sqrt{2}} \left(|t_0', a_0\rangle + |t_0' + (-1)^{a_0} s, a_0 + 1\rangle\right) \tag{15.2}$$

where $x(t_0', a_0) = x_0$. This state can be alternatively written as:

$$\frac{1}{\sqrt{2}} (|t_0, 0\rangle + |t_0 + s, 1\rangle) \tag{15.3}$$

for some $t_0 \in \mathbb{Z}_n$. Now, we apply $QFT_n \otimes \mathbb{I}_2$ on this state. This gives:

$$\frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{n}} \sum_{y \in \mathbb{Z}_n} \omega^{t_0 y} |y\rangle |0\rangle + \frac{1}{\sqrt{n}} \sum_{y' \in \mathbb{Z}_n} \omega^{(t_0+s)y'} |y'\rangle |1\rangle \right) \tag{15.4}$$

where $\omega = \exp(2\pi i/n)$. Now, by grouping the same ys together we have:

$$\frac{1}{\sqrt{2n}} \sum_{y \in \mathbb{Z}_n} \omega^{t_0 \cdot y} \left(|0\rangle + \omega^{s \cdot y} |1\rangle\right) |y\rangle \tag{15.5}$$

Now measure the second register of the state in the computational basis.

(a) Obtain a $y \in \mathbb{Z}_n$ uniformly at random

(b) Post-measurement state $\frac{1}{\sqrt{2}} \left(|0\rangle + \omega^{s \cdot y} |1\rangle\right)$

(c) Sketch of how to obtain s, from Kuperberg; Assume for simplicity that $n = 2^m$. $y \in \mathbb{Z}_n$ can be represented as an n-bit string. If $y = |10\dots 0 \to 2^{m-1}1\rangle$, then:

$$\frac{1}{\sqrt{2}}\left(|0\rangle + \exp(\frac{2\pi i}{m}2^{m-1}s)|1\rangle\right) = \frac{1}{\sqrt{2}}(|0\rangle + \exp(\pi i s)|1\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{s_0}|1\rangle) \qquad (15.6)$$

The idea is to press this button a few times, and d some clever post-processing. For a given y, z, we can combine the two:

$$|\psi_y\rangle \otimes |\psi_z\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \omega^{y\cdot s}|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + \omega^{z\cdot s}|1\rangle) \qquad (15.7)$$

We then use a CNOT gate to entangle the two, which gives:

$$CNOT(|\psi_y\rangle \otimes |\psi_z\rangle) = \frac{1}{2}(|00\rangle + \omega^{zs}|01\rangle + \omega^{ys}|10\rangle + \omega^{(y+z)s}|11\rangle) \qquad (15.8)$$

if we factor:

$$\frac{1}{2}\left((|0\rangle + \omega^{(y+z)s}|1\rangle)|0\rangle + \omega^{zs}(|0\rangle + \omega^{(y+z)s}|1\rangle)|1\rangle\right) \qquad (15.9)$$

we can then measure the second register in the computationa basis; we get 0 with $p = 1/2$ and get $|\psi_{y+z}\rangle$. We get 1 with $p = 1/2$ and get $|\psi_{y-z}\rangle$. The algorithm gives a sieving procedure. We want y to be 1 with a bunch of trailing zeros. This post-processing allows us to cancel out parts where y, z are the same. This is more or less a classical ability at this point - we can create a y with high probability. Note this is a superpolynomial speedup of $2^{\sqrt{\log n}}$.

□

16 Time Complexity of DHSP, Intro to Element Distinctness Problem

Time Complexity of DHSP

We consider the Dihedral group D_{2n} and a hidden subgroup $\{(0,0),(s,1)\}$ where s is the secret. We take $n = 2^m$ and consider $|\psi_k\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \omega^{ks}|1\rangle)$ where $\omega = \exp(2\pi i/n)$. It is good if $k = 000000\dots 0 \to 2^{m-1} = n/2 \in \mathbb{Z}_k$. I.e. the first part of k is all zero. So, then $s = s_{m-1}S_{m-2}s_{m-3}\dots s_0 = 2^{m-1}s_{m-1} + \dots + s_0 \in \mathbb{Z}_n$. Then $|\psi_k\rangle$ becomes:

$$|\psi_k\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \exp(\frac{2\pi i}{n}\frac{n}{2}s)|1\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + \exp(\pi i s_0)|1\rangle\right) \qquad (16.1)$$

why is it good? Because we can perform a hadamard gate to get $|s_0\rangle$, the secret we are trying to find.

Last time, given two states $|\psi_p\rangle, |\psi_q\rangle$ we discussed a procedure that outputs $0, |\psi_{p-q}\rangle$ and $1, |\psi_{p+q}\rangle$ with probability 1/2 each - the *Kuperburg sieve*. Let's review it and see how long it takes.

Proposition: Time Complexity of Kuperberg sieve

The Kuperberg sieve has time complexity $2^{O(\sqrt{logn})}$.

Proof. Let us review the algorithm.

1. Start off with 2^l states of the form $|\psi_p\rangle$. Preparing these states costs unit time.

2. For $k = 0, 1, \ldots, \frac{m}{a} - 1$. Note l, a are to be optimized for later. Pair the states that agree on the next a least significant bits, and do the combination operation hoping for the $-$ case (i.e. $|\psi_{p-q}\rangle$ rather than $|\psi_{p+q}\rangle$).

3. End up with a uniform mixture of $|\psi_0\rangle$ and $|\psi_{n/2}\rangle$

Analysis: At each stage, how many states do we expect to lose? we expect to retain $1/8$ of the states at each stage provided the number of states at the start of the stage is $\geq 2 \cdot 2^a$. Why $1/8$? 3 factors of a half:

(a) The combining operation takes in two states and outputs one.

(b) The combining operation succeeds with probability $1/2$.

(c) There are 2^a possible bit string sets we study (we want the next a least significant bits to agree). In each set, we can have at most one state that we cannot combine. So, if we have $2 \cdot 2^a$ states at the beginning of the stage, the maximum possible fraction of states that cannot be paired is $1/2$.

So we start with l states, and at each stage we lose $1/8 = 1/2^3$ states, so at the last stage $j = \frac{m}{a} - 1$, we have $\frac{2^l}{2^{3(m/a-1)}}$ states and we require this to be $\geq 2 \cdot 2^a$ for the procedure to succeed. So then:

$$2^{l-3(\frac{m}{a}-1)} \geq 2^{a+1} \implies l - 3\frac{m}{a} + 3 \geq a + 1 \implies l \geq 3\frac{m}{a} + a - 2 \tag{16.2}$$

Minimizing, we have $a = \sqrt{m}$ so $l = O(\sqrt{m})$ and so with 2^l states and $n = 2^m$ the time complexity is $2^{O(\sqrt{\log n})}$. □

Thus concludes the proof. If we are able to come up with an algorithm that is $\text{poly}(\log n)$ then you would be the next Peter Shor, because this would break the current "post-quantum" crypto systems.

Another approach

Another possible approach. Consider $k \leftarrow \mathbb{Z}_n$ with $|\psi_k\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \omega^{ks}|1\rangle)$. Consider measuring $|\psi_k\rangle$ using the following measurement; the Hadamard basis measurement $\mathcal{M} = \{|+\rangle\langle+|, |-\rangle\langle-|\}$ where $|\pm\rangle = \frac{|0\rangle \pm |1\rangle}{\sqrt{2}}$. The probability of getting $+$ is:

$$p(+) = \left\| |+\rangle\langle+| \frac{1}{\sqrt{2}}(|0\rangle + \omega^{ks}|1\rangle) \right\|^2 = |\frac{1}{2}(1 + \exp(\frac{2\pi iks}{n}))|^2 = \cos^2(\frac{\pi ks}{m}) \tag{16.3}$$

So - we have a random variable K chosen randomly from $\mathbb{Z}_n$ and $B \in \{+, -\}$. What is then $Pr[K = k|B = +]$? By Bayes' rule, we have:

$$Pr[K = k|B = +] = \frac{Pr[B = +|K = k]Pr[K = k]}{Pr[B = +]} = \frac{\cos^2(\frac{\pi ks}{m})\frac{1}{n}}{\frac{1}{n}\sum_l \cos^2(\frac{\pi ls}{n})} = \begin{cases} \frac{1}{n} & \text{if } s = 0 \\ \frac{2}{n}\cos^2(\frac{\pi ks}{n}) & \text{if } s \neq 0 \end{cases} \tag{16.4}$$

Probability of getting $+$ is $\frac{1}{2}$. We've reduced the problem to the following.

- We have unknown $s \in \mathbb{Z}_n$.
- Have ability to press a button at unit time cost to generate a sample $k \in \mathbb{Z}_n$ with probability given in Eq. (16.4). Either a constant or a cosine wave (which oscillates faster with increasing s). These distributions are sufficiently far apart that with only logarithmic presses to discriminate the two distributions. However, a time-efficient algorithm to do this is not known (research question to the classical algorithms students in the crowd). If you use MLE, this requires $\log n$ presses of the button but is not $\text{poly}(\log n)$ to run.

This concludes our discussion of HSP, though we note that it is still an open research area. DHSP has motivation to break lattice cryptography. For the symmetric group, HSP is relevant as it connects to solving the graph isomorphism problem. There is also HSP over $\mathbb{R}$, which has relevance to solving number-theoretic equations.

Element distinctness Problem

We are back in polynomial speedup land!

Definition: Element Distinctness Problem and Amplitude Amplification

Consider $n \in \mathbb{N}$, and the function

$$\begin{aligned} f \quad : \quad [n]^n &\longrightarrow \{0,1\} \\ x &\longmapsto \begin{cases} 1 & \text{if } x \text{ contains duplicates} \\ 0 & \text{if } x \text{ does not contain duplicates} \end{cases} \end{aligned} \tag{16.5}$$

which defines ED_n.

Example: Let $n = 5$. Then if $x = 12453$ then $f(x) = 0$ and if $x = 11453$ then $f(x) = 1$.

As a quick remark, $R(ED_n) = \Omega(n)$, which is obtained via the lower bound for OR_n. What is the quantum query complexity of this?

Proposition: Quantum query complexity of ED_n

$Q(ED_n) = O(n^{2/3})$.

There is also a weaker result of $Q(ED_n) = O(n^{3/4})$. To get this we use the widely used technique of amplitude amplification - which morally is glorified Grover search.

Proposition: Amplitude Amplification

Let $d \in \mathbb{N}$ and $\theta \in [0, \pi/2]$. Let $|\psi_0\rangle, |\psi_1\rangle$ be d-dimensional quantum states such that $|\psi\rangle = \cos\theta|0\rangle|\psi_0\rangle + \sin\theta|1\rangle|\psi_1\rangle$. Let $G = \mathbb{I}_{2d} - 2|\psi\rangle\langle\psi|$ and $U = \mathbb{I}_{2d} - 2|1\rangle\langle 1| \otimes \mathbb{I}_d$. Then for all $k \in \mathbb{N}$, we find:

$$(GU)^k|\psi\rangle = (-1)^k \left(\cos((2k+1)\theta)|0\rangle|\psi_0\rangle + \sin((2k+1)\theta)|1\rangle|\psi_1\rangle\right) \tag{16.6}$$

Proof. Same as the steps leading to the lecture 3 result (with generalized θ). □

Morally we have a procedure that succeeds if we measure $|1\rangle$, for which we can amplitude can be amplified to succeed with a constant probability (and quantum does better at classical at such an amplification) - $\frac{1}{\sqrt{p}}$ as opposed to $\frac{1}{p}$ in the randomized case.

In particular, writing $\sin(\theta) = \sqrt{p}$ the probability of measuring $|\psi\rangle$ (the initial state) in the computational basis on the first register is p (think of it as small). If we take k to be:

$$\frac{\pi}{4\theta} - \frac{1}{2} \in [\frac{\pi}{4\theta} - 1, \frac{\pi}{4\theta}]; \quad k \leq \frac{\pi}{4}\frac{1}{\arcsin(\sqrt{p})} \leq \frac{\pi}{4}\frac{1}{\sqrt{p}} \tag{16.7}$$

to obtain $\sin((2k+1)\theta) \in [\sin(\frac{\pi}{2} - \theta), 1] = [\sqrt{1-p}, 1]$ (the probability of success/measuring 1 is $\sin^2((2k+1)\theta)$, which for small p is a constant).

Typical application: Have a unitary A such that $A|0\rangle = |\psi\rangle$ so $G = A(1 - 2|0\rangle\langle 0|)A^\dagger$. So, $(GU)^k$ uses k applications of A and $A^{-1} = A^\dagger$ (with same/similar costs) If A succeeds with probability p then $O(\frac{1}{\sqrt{p}})$ applications of A and $A^\dagger$ boosts success to a constant.

17 Amplitude Amplification in Element Distinctness, Quantum Walk

Missed this lecture - notes based on provided references.

Grover Method for ED Problem

A classical algorithm takes $\Omega(n)$ to solve ED as it is at least as hard as unstructured search, and therefore the quantum lower bound is $\Omega(\sqrt{n})$.

By using Grover recursively, we can improve on the trivial $O(n)$ running time in the quantum case. Consider the subgroutine - query f in l randomly chosen places, and check whether one of those l belongs to a piar of inputs belongs to a pair by doing Grover on the remaining $n-l$. The initial setuip takes l queries and Grover takes $O(\sqrt{n-l}) = O(\sqrt{n})$ queries for a total of $l + O(\sqrt{n})$. This fails mst of the time, but succeeds with probability at least l/n. To boost it, we can use amplitude amplification (see last lecture) which takes $O(\sqrt{n/l})$ steps to boost the success probability ot a constant. Overall, we obtain a success probability of $\Omega(1)$ with:

$$(l+\sqrt{n})\sqrt{n/l} = \sqrt{nl} + n/\sqrt{l} \tag{17.1}$$

queries. To optimize the query complexity, we set the two terms to be equal, so $l = \sqrt{n}$ and so we obtain the claimed $Q(ED_n) = O(n^{3/4})$.

Note that the time complexity analysis of this algorithm would contain extra logarithmic factors, as the inner recursive use of Grocer must check an element against l queried function values, which is done in $O(\log l)$ time.

The lower bound is improved to $\Omega(n^{2/3})$ by using the $\Omega(n^{1/3})$ lower bound for collision by Aaronson and Shi. The quantum walk algorithm will give us an upper bound of $O(n^{2/3})$, showing that $Q(ED_n) = \Theta(n^{2/3})$. Let us thus move to discussion of the quantum walk.

Discrete-time quantum walks

A simple example of a dicrete-time random walk on a graph G is where we move from any vertex to each of its neighbours with equal probability. Thus the walk is governed by the $|V| \times |V|$ matrix M with entries:

$$M_{jk} = \begin{cases} 1/\deg(k) & (j,k) \in e \\ 0 & \text{otherwise} \end{cases} \tag{17.2}$$

for $j, k \in V$: an initial probability distribution p over the vertices evolves to $p' = Mp$ after one step of the walk.

To define a quantum analog, we want to specify a unitary U such that given an input state $|j\rangle$ corresponding to $j \in V$ it evolves into a superposition of the neighbours. We want this to happen the same way at every vertex, so it is tempting to write:

$$|j\rangle \mapsto |\partial_j\rangle := \frac{1}{\sqrt{\deg(j)}} \sum_{k:(j,k)\in E} |k\rangle \tag{17.3}$$

but the above does not define a unitary transformation, since the orthogonal states $|j\rangle, |k\rangle$ corresponding to adjacent j, k with common neighbour l evolve to non-orthogonal states. We could potentially avoid this using phases, but this sacrifices the idea of having the operator act identically one each site (and for some graphs it simply is not possible).

We get around this if we enlarge the Hilbert space, an idea proposed by Watrous as part of a logarithmic-space quantum algorithm for deciding whether two vertices are connected in a graph. Let the Hilbert space consist of $|j,k\rangle$ where $(j,k) \in E$. We can think of the walk as taking place on the (directed) edges of the graph; the state $|j,k\rangle$ represents a walker at vertex j that will move towards vertex k. Each step of the walk consists of two operations. sents a walker at vertex j that will move toward vertex k. Each step of the walk consists of two operations. First, we apply a unitary transformation that operates on the second register conditional on the first register. This transformation is sometimes referred to as a "coin flip", as it modifies the next destination of the walker. A common choice is the Grover diffusion operator over the neighbors of j, namely:

$$C := \sum_{j\in V} |j\rangle\langle j| \otimes (2|\partial_j\rangle\langle\partial_j| - I) \tag{17.4}$$

Next, the walker is moved to the vertex indicated in the second register. Since this must be unitary, we have to swap the registers using:

$$S := \sum_{(j,k)\in E} |j,k\rangle\langle k,j| \tag{17.5}$$

Overall, one step of the discrete-time quantum walk is described by SC.

In principle this can define a DTQW on any graph, but in practice we use the alternative framework of Quantum Markov chains.

Quantum Markov Chains

A discrete-time classical random walk on an N-vertex graph can be represented by an $N \times N$ matrix P. the entry P_{kj} represents the probability of making a transition to k from j, so the initial probability $P \in \mathbb{R}^n$ becomes Pp. To preserve normalization we require $\sum_{k=1}^{N} P_{jk} = 1$ and call these matrices are *stochastic*.

For any $N \times N$ stochastic P, we can define a corresponding discrete-time quantum walk, a unitary operation on $\mathbb{C}^N \otimes \mathbb{C}^N$. To this end we introduce:

$$|\psi_j\rangle := |j\rangle \otimes \sum_{k=1}^{N} \sqrt{P_{kj}}|k\rangle = \sum_{k=1}^{N} \sqrt{P_{kj}}|j,k\rangle \tag{17.6}$$

Each state is normalized due to P being stochastic. Let:

$$\Pi := \sum_{j=1}^{N} |\psi_j\rangle\langle\psi_j| \tag{17.7}$$

denote the projection onto the span of the $|\psi_j\rangle$s. Let:

$$S := \sum_{j,k=1}^{N} |j,k\rangle\langle k,j| \tag{17.8}$$

be the operator that swaps the two registers. Then, one step is $U := S(2\Pi - 1)$. Notice that if $P_{jk} = A_{jk}/\deg(k)$ then this is the coined quantum walk using the Grover diffusion operator as the coin flip. Two steps gives:

$$[S(2\Pi - 1)]^2 = (2S\Pi S - 1)(2\Pi - 1) \tag{17.9}$$

which can be inrepreted as the reflection about $\text{span}\left\{|\psi_j\rangle\right\}$ followed by reflection about $\text{span}\left\{S|\psi_j\rangle\right\}$ (the states where we condition on the second register to do a coin operation on the first). To understand the behavior of the walk, we will now compute the spectrum of U; but note that it is also possible to compute the spectrum of a product of reflections more generally.

18 Quantum Walk Continued

Missed this lecture - notes based on provided references. To understand the behavior of a discrete-time quantum walk, it will be helpful to compute its spectral decomposition. Let us show the following:

Theorem: Spectrum of Quantum Walk

Fix an $N \times N$ stochastic matrix P and let $\{|\lambda\rangle\}$ denote a complete set of orthonormal eigenvectors of the $N \times N$ matrix D with entries $D_{jk} = \sqrt{P_{jk}P_{kj}}$ with eigenvalues $\{\lambda\}$. Then, the eigenvalues of the discrete-time quantum walk $U = S(2\Pi - 1)$ corresponding to P are ± 1 and $\lambda \pm i\sqrt{1-\lambda^2} = \exp(\pm i \arccos \lambda)$.

Proof. Define an isometry:

$$T := \sum_{j=1}^{N} |\psi_j\rangle\langle j| = \sum_{j,k=1}^{N} \sqrt{P_{kj}}|j,k\rangle\langle j| \tag{18.1}$$

which maps states in $\mathbb{C}^N$ to states in $\mathbb{C}^N \otimes \mathbb{C}^N$, and let $|\tilde{\lambda}\rangle := T|\lambda\rangle$. Notice then that:

$$TT^\dagger = \sum_{j,k=1}^{N} |\psi_j\rangle\langle j|k\rangle\langle\psi_k| = \sum_{j=1}^{N} |\psi_j\rangle\langle\psi_j| = \Pi \tag{18.2}$$

whereas:

$$T^\dagger T = \sum_{j,k=1}^{N} |j\rangle\langle\psi_j|\psi_k\rangle\langle k| = \sum_{j,k,l,m=1}^{N} \sqrt{P_{lj}P_{mk}}|j\rangle\langle j,l|k,m\rangle\langle k| = \sum_{j,l} P_{l,j}|j\rangle\langle j| = I \tag{18.3}$$

and:

$$T^\dagger S T = \sum_{j,k=1}^{N} |j\rangle\langle\psi_j|S|\psi_k\rangle\langle k| = \sum_{j,k,l,m=1}^{N} \sqrt{P_{lk}P_{mk}}|j\rangle\langle j,l|S|k,m\rangle\langle k| = \sum_{j,k=1}^{N} \sqrt{P_{jk}P_{kj}}|j\rangle\langle k| = D. \tag{18.4}$$

Applying the walk operator U to $|\tilde{\lambda}\rangle$ gives:

$$U|\tilde{\lambda}\rangle = S(2\Pi - 1)|\tilde{\lambda}\rangle = S(2TT^\dagger - 1)T|\lambda\rangle = 2ST|\lambda\rangle - ST|\lambda\rangle = S|\tilde{\lambda}\rangle \tag{18.5}$$

We see tht the subspace span $\left\{|\tilde{\lambda}\rangle, S|\tilde{\lambda}\rangle\right\}$ is invariant under U, so we can find eigenvectors of U within this subspace. Now, let $|\mu\rangle := |\tilde{\lambda}\rangle - \mu S|\tilde{\lambda}\rangle$, and let us choose $\mu \in \mathbb{C}$ so that $|\mu\rangle$ is an eigenvector of U. We then have:

$$U|\mu\rangle = S|\tilde{\lambda}\rangle - \mu(2\lambda S|\tilde{\lambda}\rangle - |\tilde{\lambda}\rangle) = \mu|\tilde{\lambda}\rangle + (1 - 2\lambda\mu)S|\tilde{\lambda}\rangle \tag{18.6}$$

Thus μ will be an eigenvalue of U corresponding to $|\mu\rangle$ provided that $\mu^2 - 2\lambda\mu + 1 = 0$, so:

$$\mu = \lambda \pm i\sqrt{1-\lambda^2}. \tag{18.7}$$

Finally, note that for any vector in the orthogonal complement of span$\left\{|\tilde{\lambda}\rangle, S|\tilde{\lambda}\rangle\right\}$, U simply acts as $-S$ (since $\Pi = TT^\dagger = \sum_\lambda T|\lambda\rangle\langle\lambda|T^\dagger = \sum_\lambda |\lambda\rangle\langle\lambda|$ projects onto span$\left\{|\tilde{\lambda}\rangle\right\}$). In this subspace, the eigenvalues are ± 1. □

19 Quantum Walk Conclusion, Quantum Phase Estimation

Missed this lecture - notes based on provided references.

Hitting Time for Random Walk

We can use random walks to formulate a generic search algorithm, and quantizing this algorithm gives a generic square root speedup. Consider a graph $G = (V, E)$ with some subset $M \subset V$ of the vertices denoted as *marked*. We will compare classical and quantum walk algorithms for deciding whether any vertex in G is marked.

Classically, a straightforward aproach to this problem is to take a random walk defined by some stochastic matrix P, stopping if we encounter a marked vertex. In other words, we modify the original walk P to give a walk P' defined as:

$$P'_{jk} = \begin{cases} 1 & k \in M \text{ and } j = k \\ 0 & k \in M \text{ and } j \neq k \\ P_{jk} & k \neq M \end{cases}. \tag{19.1}$$

Let us assume from now on that the original walk P is symmetric, thought he modified walk P' clearly is not provided M is non-empty. If we order the vertices so that the marked ones come last, the matrix P' has the block form:

$$P' = \begin{pmatrix} P_M & 0 \\ Q & I \end{pmatrix} \tag{19.2}$$

where P_M is obtained by deleting the rows and columns of P corresponding to vertices in M.

Suppose we take t steps of the walk. A simple calculation shows:

$$(P')^t = \begin{pmatrix} P_M^t & 0 \\ Q(I + P_M + \cdots + P_M^{t-1}) & I \end{pmatrix} = \begin{pmatrix} P_M^t & 0 \\ Q\frac{P_M^t - I}{P_M - I} & I \end{pmatrix}. \tag{19.3}$$

Now if we start from the uniform distribution over unmarked items (if we start from a marked item we are done, so we might as well condition on this not happening), then the probability of not reaching a marked item after t steps is:

$$\frac{1}{N - |M|} \sum_{j,k \notin M} [P_M^t]_{jk} \leq \left\| P_M^t \right\| = \|P_M\|^t \tag{19.4}$$

where the inequality follows because the left hand side is the expectation of P_M^t in the normalized state $|V \setminus M\rangle = \frac{1}{\sqrt{N-|M|}} \sum_{j \notin M} |j\rangle$. Now if $\|P_M\| = 1 - \Delta$, then the probability of reaching a marked item after t steps is at least $1 - \|P_M\|^t = 1 - (1-\Delta)^t$, which is $\Omega(1)$ provided $t = O(1/\Delta) = O(\frac{1}{1-\|P_M\|})$.

It turns out we can bound $\|P_M\|$ away from 1 knowing only the fraction of marked vertices and the spectrum of the original walk. Thus we can upper bound the *hitting time*, the time required to reach some marked vertex with constant probability.

Lemma: Bound on stochastic matrix norm

If the second largest eigenvalue of P (in absolute value) is at most $1 - \delta$ and $|M| \geq \epsilon N$, then $\|P_M\| \leq 1 - \delta\epsilon$.

Proof. Let $|v\rangle \in \mathbb{R}^{N-|M|}$ be the principal eigenvector of P_M, and let $|w\rangle \in \mathbb{R}^N$ be the vector obtained by padding $|v\rangle$ with 0s for the marked vertices.

We will decompose $|w\rangle$ in the eigenbasis of P. Since P is symmetric, it is actually doubly stochastic, and the uniform vector $|V\rangle = \frac{1}{\sqrt{N}} \sum_j |j\rangle$ corresponds to the eigenvalue 1. All other eigenvectors $|\lambda\rangle$ have eigenvalues $1-\delta$ by assumption. Now:

$$\begin{aligned}\|P_M\| = \langle v|P_M|v\rangle = \langle w|P|w\rangle &= |\langle V|w\rangle|^2 + \sum_{\lambda\neq 1} \lambda |\langle \lambda|w\rangle|^2 \\ &\leq |\langle V|w\rangle|^2 + (1-\delta)\sum_{\lambda\neq 1}|\langle\lambda|w\rangle|^2 = 1-\delta\sum_{\lambda\neq 1}|\langle\lambda|w\rangle|^2 = 1-\delta(1-|\langle V|w\rangle|^2)\end{aligned}$$

So applying Cauchy-Shwarz:

$$|\langle V|w\rangle|^2 = |\langle V|\Pi_{V\backslash M}|w\rangle|^2 \leq \left\|\Pi_{V\backslash M}|V\rangle\right\|^2 \||w\rangle\|^2 = \frac{N-|M|}{N} \leq 1-\epsilon \tag{19.5}$$

where $\Pi_{V\backslash M} = \sum_{j\notin M}|j\rangle\langle j|$. Thus, $\|P_M\| \leq 1-\delta\epsilon$ as claimed.

Thus, we see the classical hitting time is $O(1/\delta\epsilon)$.

Now we turn to the quantum case. Our strategy will be to perform phase estimation with sufficiently high precision on the operator U, the quantum walk corresponding to P', with the state:

$$|\psi\rangle := \frac{1}{\sqrt{N-|M|}}\sum_{j\notin M}|\psi_j\rangle. \tag{19.6}$$

This state can be easily prepared by first starting from the state:

$$T|V\rangle = \frac{1}{\sqrt{N}}\sum_j |\psi_j\rangle \tag{19.7}$$

nd measuring whether the first register corresponds to a marked vertex. If it does, we're done, and if does not, we've prepared $|\psi\rangle$.

The matrix D for the walk P' is:

$$\begin{pmatrix} P_M & 0 \\ 0 & I\end{pmatrix} \tag{19.8}$$

so according to the spectral theorem, the eigenvalues of the resulting walk operator U are ± 1 and $\exp(\pm i \arccos\lambda)$, where λ runs over eigenvalues of P_M. If the marked set M is empty, then $P' = P$, and $|\psi\rangle$ is an eigenvector of U with eigenvalue 1, so phase estimation with U is guaranteed to return 0. But if M is non-empty, then $|\psi\rangle$ lives entirely within the subspace with eigenvlues $\exp(\pm i\arccos\lambda)$. Thus if we perform phase estimation on U with precision $O(\min_\lambda \arccos\lambda)$, we will see a phase different from 0. Since $\arccos\lambda \geq \sqrt{2(1-\lambda)}$, we see that precision $O(\sqrt{1-\|P_M\|})$ suffices. So the quantum algorithm can decide whether there is a marked vertex in time $O(1/\sqrt{1-\|P_M\|}) = O(1/\delta\epsilon)$. □

Quantum Phase Estimation

In the last portion of the proof of the quantum walk hitting time, we used the quantum phase estimation algorithm - let's discuss that now! It is an algorithm that unifies various aspects of quantum algorithms we have seen thus far. It is based on the QFT over $\mathbb{Z}_N$, where $N = 2^n$.

Imagien we are given unitary U, a black-box that allows us to apply a controlled U^j, and a eigenstate $|\psi\rangle$ with eigenvalue $\exp(2\pi i\phi)$ for $0 \leq \phi < 1$. We wish to determine ϕ to n bits of precision.

To this end, prepare an n-qubit register to $|\psi\rangle$ and apply H to each qubit in the first register to get:

$$\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle|\psi\rangle \tag{19.9}$$

We then apply the unitary operator:

$$U' = \sum_{x=0}^{N-1}|x\rangle\langle x|\otimes U^x \tag{19.10}$$

This operator can be thought of as performing the map where if the first register contains x, we can apply U x times to the second. By expressing x in binary we can implement U^{2^j} gates for different integers j, controlled on different qubits in the first register. After applying U', we are left with the state:

$$\frac{1}{\sqrt{N}}\sum_{x=0}^{N-1} e^{2\pi i\phi x}|x\rangle|\psi\rangle \tag{19.11}$$

note that teh second register is left unchanged. Apply Q^{-1} to first register and measure, getting outcome y. Output the binary fraction:

$$0.y_1y_2\dots y_n = \frac{y_1}{2}+\frac{y_2}{4}+\dots+\frac{y_n}{2^n} \tag{19.12}$$

as our guess for ϕ.

Why does this work? When we perform the last measurement, we measure x with probability:

$$\frac{1}{N^2}|\sum_{y=0}^{N-1} e^{2\pi i\phi y-2\pi ixy/N}|^2 = \frac{1}{N^2}|\sum_{y=0}^{N-1} e^{2\pi i(\phi-x/N)}|^2 \tag{19.13}$$

First imagine that the binary expansion of ϕ is at most n bits long, or in other words $\phi = z/N$ for some $0\leq z\leq N-1$. In this case we have:

$$\frac{1}{N^2}|\sum_{y=0}^{N-1} e^{2\pi iy(\phi-x/N)}|^2 = \frac{1}{N^2}|\sum_{y=0}^{N-1} e^{2\pi iy(z-x)/N}|^2 = \delta_{xz} \tag{19.14}$$

by the unitarity of the QFT, so the measurement outcome is guaranteed to be z, implying that the algorithm outputs ϕ with certainty. If the binary expansion of ϕ is longer than n bits, we now show that we still get the best possible answer with probability $\Omega(1)$, and indeed are very likely to get an answer close to ϕ. The proof turns out to be very similar to that of the correctness of the periodicity determination algorithm in the approximate case.

Theorem: Correctness of QPE

The probability that the above algorithm outputs the real niumber with n binary digits which is closest to ϕ is at least $4/\pi^2$. Further, the probability that the algorithm outputs θ such that $|\theta-\phi|\geq\epsilon$ is at most $O(1/(N\epsilon))$.

Proof. If the binary expansion of ϕ has n binary digits are fewer, we are done by the argument above (it gives the exact answer). So assuming it does not, let $\tilde{\phi}$ be the closest approximation to ϕ that has n binary digits, and write $\tilde{\phi} = a/N$ for some integer $0\leq a\leq N-1$. For any z, define $\delta(z) := \phi - z/N$ and note that $0<|\delta(a)|\leq 1/(2N)$. For any ϕ, the probability of getting z from the final measurement is:

$$Pr[z] = \frac{1}{N^2}|\sum_{y=0}^{N-1} e^{2\pi iy(\phi-x/N)}|^2 = \frac{1}{N^2}|\sum_{y=0}^{N-1} e^{2\pi iy\delta(z)}|^2 = \frac{1}{N^2}|\frac{1-e^{2\pi iN\delta(z)}}{1-e^{2\pi i\delta(z)}}|^2 = \frac{\sin^2(\pi N\delta(z))}{N^2\sin^2(\pi\delta(z))} \tag{19.15}$$

where we evaluate the sum using the geometric series formula. This quantity should be familiar from the proof of correctness of the periodicity determination algorithm.

We first lower bound this expression for $z=a$ to prove the first part of the lemma. As $|\delta(a)|\leq 1/(2N)$, we have $N\pi\delta(a)\leq\pi/2$. Then:

$$Pr[a] = \frac{\sin^2(\pi N\delta(a))}{N^2\sin^2(\pi\delta(a))} \geq \frac{(2N\delta(a))^2}{N^2\pi(\delta(a))^2} = \frac{4}{\pi^2} \tag{19.16}$$

using trigonometric inequalities for sine.

In order to prove the secone part, we now find an upper bound on $Pr[z]$. First, it is clear that $\sin^2(\pi N\delta(z)) \leq 1$ always. For the denominator, by the same arguemtn above we have $\sin(\pi\delta(z)) \geq 2\delta(z)$ and hence for all z:

$$Pr[z] \leq \frac{1}{N^2}\left(\frac{1}{2\delta(z)}\right)^2 = \frac{1}{4N^2\delta(z)^2} \tag{19.17}$$

we now sum this expression over all z such that $|\delta(z)| \geq \epsilon$. The sum is symmetric about $\delta(z) = 0$, and as z is an integer, the terms in this sum corresponding to $\delta(z) > 0$ are $\delta_0, \delta_0 + 1/N, \ldots$ for some $\delta_0 \geq \epsilon$. The sum will be maximized when $\delta_0 = \epsilon$, when we obtain:

$$Pr[z \text{ with } |\delta(z)| \geq \epsilon] \leq \frac{1}{4N^2}\sum_{k=0}^{\infty}\frac{1}{(\epsilon + k/N)^2} \leq \frac{1}{4}\int_0^{\infty}\frac{1}{(N\epsilon + k)^2}dk = \frac{1}{4}\int_{N\epsilon}^{\infty}\frac{1}{k^2}dk = O(\frac{1}{N\epsilon}) \tag{19.18}$$

□

We observe the following properties regarding the behaviour of this algorithm:

1. What happens if we do not know an eigenvector of U? If we input an arbitrary state $|\varphi\rangle$ to to the QPE algorithm, we can write it as $\sum_j \alpha_j|\psi_j\rangle$ over the eigenvectors $\left\{|\psi_j\rangle\right\}$. Therefore, the algorithm will output an estimate of each corresponding eigenvalue ϕ_j with probability $|\alpha_j|^2$. This may or may not allow us to infer anything useful, depending on what we know about U in advance.

2. In order to approximate ϕ to n bits of precision, we needed to apply U^{2m} for all $0 \leq m \leq n-1$. If we are given U as a black box, this may be prohibitively expensive as we need to use the black box exponentially times in n. However, if we have an explicit circuit for U, then we may be able to find a more efficient way of computing U^{2m}, e.g. as is the case for modular exponentiation.

Applications of QPE to Phase Estimation

An elegant application of QPE is to generalize unstructured (Grover) search. Imagine we have an oracle $f : \{0,1\}^n \to \{0,1\}$ which takes the value 1 on k inputs, for some unknown k, and again set $N = 2^n$. We would like to estimate k by querying f.

Classically, a natural way to do this is via sampling. Imagine that we query f on q random inputs and get that f is 1 on l of those inputs. Then as our estimate of k we output $\tilde{k} = lN/q$. One can show using properties of the binomial distribution that this achieves:

$$|\tilde{k} - k| = O(\sqrt{\frac{k(N-k)}{q}}) \tag{19.19}$$

with high probability. We can achieve improved accuracy by using the QPE algorithm. Consider the "Grover iteration" $G = -H^{\otimes n}U_0H^{\otimes n}U_f$. As G is a rotation through angle 2θ in a 2D plane, where $\sin\theta = \sqrt{k/N}$, its eigenvalues are $e^{2i\theta}$ and $e^{-2i\theta}$. In order to estimate k, we can apply the QPE to G to estimate either one of these eigenvalues. As it doesn't matter which one we estimate, we can input any state within this 2D plane to the QPE as a claimed eigenvector of G. In particular, $\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n}|x\rangle$ will work.

By the previous theorem, if we apply QPE to G, we can find the closest m-difit number to θ, for any m, with constant probability of sucess using $O(2^m)$ queries. For small θ, $\theta \approx \sqrt{k/N}$, so we learn $\sqrt{k/N}$ up to additive error $O(1/2^m)$ with $O(2^m)$ queries. Setting $2^m = \sqrt{N}/\delta$ for some real $\delta > 0$, we have learned $\sqrt{k}$ up to additive error $O(\delta)$ using $O(\sqrt{N}/\delta)$ queries; or in other words have learnt k up to additive error $O(\delta\sqrt{k})$ using $O(\sqrt{N}/\delta)$ queries. In order to achieve a similar level of accuracy classically, we would need $\Omega(N/\delta^2)$ queries for small k. Note that QPE can also be applied to the order finding problem.

20 Quantum Walk for OR and ED

Missed this lecture - notes based on provided references.

Unstructured Search

Now we begin to discuss applications of quantum walks to search algorithms. We start with the most basic of all search problems, the unstructured search problem (which is solved optimally by Grover's algorithm). We discuss how this problem fits into the framework of quantum walk search, and also describe amplitude amplification and quantum counting in this setting. We also discuss quantum walk algorithms for the search problem under locality constraints.

In the unstructured search problem, we are given a black box $f : S \to \{0,1\}$ where S is a finite set with $|S| = N$. The inputs $x \in M$ where $M := \{x \in S : f(x) = 1\}$ are called *marked items*. In the decision version of the problem, our goal was to determine whether M was empty or not. Now we are interested in finding a marked item when it exists.

The decision problem requires $\Omega(N)$ classical queries, and N queries suffice, so unstructured search is classically $\Theta(N)$.

We are already familiar with Grover which does this in $O(\sqrt{N})$. We start with $|S\rangle = \sum_{x\in S}|x\rangle/\sqrt{N}$ and alternatively apply the reflection about the set of marked items $\sum_{x\in M} 2|x\rangle\langle x| - 1$, and the reflection about S, $2|S\rangle\langle S| - 1$. The former is done with two queries to f and the latter none. It is straightforward to show that there is some $t = O(\sqrt{N/|M|})$ for which t steps of this gives a state with constant overlap on $|M\rangle$ so a measurement reveals a marked item with constant probability.

It can be shown that unstructured search requires $\Omega(\sqrt{N/|M|})$ queries. We discuss this when we discuss adversary lower bounds.

Consider the discrete-time random walk on the complete graph, which has stochastic matrix:

$$P = \frac{N}{N-1}|S\rangle\langle S| - \frac{1}{N-1}I \tag{20.1}$$

It has eigenvalues 1 (non-degenerate) and $-1/(N-1)$ (degeneracy $N-1$). Since the graph is highly connected, its spectral gap is very large, with $\delta = 1 + \frac{1}{N-1} = \frac{N}{N-1}$.

This random walk gives rise to a very simple classical algorithm for unstructured search. In this algorithm, we start from a uniformly random item and repeatedly choose a new item uniformly at random from the other $N-1$ possibilities, stopping when we reach a marked item. The fraction of marked items is $\epsilon = |M|/N$, so the hitting time of this walk is

$$O(\frac{1}{\delta\epsilon}) = \frac{(N-1)N}{N|M|} = O(N/|M|) \tag{20.2}$$

(this is only an upper bound on the hitting time, but in this case we know it is optimal). Of course, if we have no a priori lower bound on $|M|$ if it is non-empty, we can only say that $\epsilon \geq 1/N$ so the running time is $O(N)$.

The corresponding quanrum walk has hitting time:

$$O(\frac{1}{\sqrt{\delta\epsilon}}) = O(\sqrt{N/|M|}) \tag{20.3}$$

corresponding to Grover's running time. To see that we get a total algorithm which is $O(\sqrt{N/|M|})$, we need to show the quantum walk takes $O(1)$ quantum queries.

We can modify the classical walk matrix to where the first item is marked, and then $|\psi_1\rangle = |1,1\rangle$ and $|\psi_j\rangle = |j, S\setminus\{j\}\rangle = \sqrt{\frac{N}{N-1}}|j,S\rangle - \frac{1}{\sqrt{N-1}}|j,j\rangle$ for $j = 2,\dots,N$. With a general M, the projector onto the span of these states is:

$$\Pi = \sum_{j\in M}|j,j\rangle\langle j,j| + \sum_{j\notin M}|j,S\setminus\{j\}\rangle\langle j,S\setminus\{j\}| \tag{20.4}$$

so $2\Pi - 1$ acts as Grover diffusion over the neighbors when when the vertex is unmarked, and as a phase flip when the vertex is marked. (Note that since we start from the state $|\psi\rangle = \sum_{j \notin M} |\psi_j\rangle$, we stay in the subspace of $|j,k\rangle$ with (j,k) edges, and have zero support for $|j,j\rangle$ for $j \in V$ so $2\Pi - 1$ acts as -1 when the first register holds a marked vertex. ach such step can be implemented using two queries of the black box, one to compute whether we are at a marked vertex and one to uncompute that information; the subsequent swap operation requires no queries. Thus the query complexity is indeed $O(\sqrt{N/|M|})$.

This is not exactly the same as Grover - it works in $\mathbb{C}^N \otimes \mathbb{C}^N$ instead of $\mathbb{C}^N$. But it is very closely related. In Grover, we can view $2|S\rangle\langle s| - 1$ as a discrete time quantum walk.

The algorithm we have described so far only solves the decision version of unstructured search. To find marked item, we could use bisection, but this would introduce a logarithmic overhead. In fact, it can be shown that the final state of the quantum walk algorithm actually encodes a marked item when one exists.

Quantum Walk Algorithm

Ambainis's algorithm is to quantize a walk on the Johnson graph $J(n,m)$ where m is chosen appropriately. The graph has $\binom{n}{m}$ vertices corresponding to subsets of $\{1,2,\dots,n\}$ of size m, and two vertices are connected by an edge if the subsets differ in exactly one element.

To simplify, we use the Hamming graph $H(n,m)$ - where the vertices are m-tuples of values from $\{1,2,\dots,n\}$ (so there are n^m vertices). Two vertices are connected vy an edge if they differ in exactly one coordinate. There are two main differences betwen the two graphs; the Hamming graph allows for repeated elements, and the order matters. This doesn't impact the performance of the algorithm significantly.

At each vertex, we store the values of the function at the corresponding inputs, i.e. $(x_1, x_2, \dots, x_m) \in \{1,2,\dots,n\}^m$ is represented by the state

$$|x_1, x_2, \dots, x_m, f(x_1), f(x_2), \dots, f(x_m)\rangle \tag{20.5}$$

To prepare such states, we must query the black-box function. In particular, to prepare an initial superposition over vertices of this graph takes m queries. However, we can move from one vertex to an adjacent one using two queries; to replace x by y in a particular coordinate, we use one query to erase $f(x)$ and another to compute $f(y)$.

In the search problem, the marked vertices are those containing some $x \neq y$ with $f(x) = f(y)$. Notice that, given the stores function values, we can check whether we are at a marked vertex with no additional queries. The total number of marked vertices (in this case, where the elements are not all distinct) is at least $\binom{m}{2}(n-2)^{m-2}$, so the fraction of marked vertices is:

$$\epsilon \geq \frac{m(m-1)(n-2)^{m-2}}{2n^m} \tag{20.6}$$

To analyze, we need the eigenvalues of the relevant Markov chain. The adjacency matrix of $H(n,m)$ is $A = \sum_{i=1}^{m} (J - I)^{(i)}$, where J denotes the $n \times n$ matrix of all 1s, and the superscript indicates that this matrix acts on the ith coordinate. The eigenvalues of J are n and 0, so the eigenvalues of $J - I$ are $n-1$ and -1. Hence the largest eigenvalue of A is $m(n-1)$ (the degree of any vertex of $H(n,m)$) and the second largest is $(m-1)(n-1) - 1 = m(n-1) - n$. Normalizing by the degree, we find the spectral gap to be:

$$\delta = \frac{n}{m(n-1)} \tag{20.7}$$

Finally, how many queries does the algorithm use? Taking into account the initial m queries used to prepare the starting state and 2 per step of the walk, we have the total number:

$$m + 2O\left(\frac{1}{\sqrt{\delta\epsilon}}\right) = m + O\left(\frac{n}{\sqrt{m}}\right) \tag{20.8}$$

We can again set the terms to be equal to optimize the performance - since $m^{3/2} = O(n)$, we should take $m = \Theta(n^{2/3})$ and so the total number of queries is $O(n^{2/3})$ which matches the lower bound (and is therefore optimal).

Note that for the classical random walk search algorithm that we have quantized, the corresponding query complexity is $m + O(n^2/m)$, optimized by $m = n$. This gives no improvement over querying every input (as we knew).

Quantum Walk with Auxiliary Data

Algorithms based on similar ideas turn out to be useful for a wide variety of problems, including deciding whether a graph contains a triangle (or various other related graph properties), checking matrix multiplication, and testing whether a group is abelian. In general, as in the element distinctness case, we may need to store some data at each vertex, and we need to take into account the operations on this data when analyzing the walk.

Suppose we have setup cost S, cost U to update the state after a step of the walk, and C to check whether a vertex is marked. In the ED problem, we had $S = m$ to query m positions, $U = 2$ to remove one item and add another, and $C = 0$ since the function values for the subset are stored. In general, there is an algorithm to solve such a problem with total cost:

$$S + \frac{1}{\sqrt{\delta\epsilon}}(U + C). \tag{20.9}$$

It turns out that for some problems, when $C \gg U$, it is advantageous to take many steps on the unmarked graph before performing a phase flip on the marked sites. Thsi is how Ambainis' algorithm originally worked, though for ED its not necessary. Using this idea, one can give a general quantum walk search algorithm with total cost:

$$S + \frac{1}{\sqrt{\epsilon}}\left(\frac{1}{\sqrt{\delta}}U + C\right) \tag{20.10}$$

In fact, it is also possible to modify the general algorithm so it finds a marked item when one exists.

21 Adversary Method

Missed this lecture - notes based on provided references.

The quantum adversary method provides a way to prove quantum query lower bounds. In fact, we will find that the generalized version is an upper bound on quantum query complexity, up to constant factors.

Quantum adversaries

Motivation for the quantum adversary method comes from the following construction. Suppose the oracle is operated by an adversarial party who holds a quantum state determining the oracle string, which is in some superposition $\sum_{x\in S}|a_x\rangle|x\rangle$ over the possible oracles. To implement eac query, the adversary performs the "super-oracle":

$$O := \sum_{x\in S}|x\rangle\langle x| \otimes O_x. \tag{21.1}$$

An algorithm does not have direct access to the oracle string, and hence can only perform unitary operations that act as the identity on the adversary's superposition. After t steps, an algorithm maps the overall state to:

$$|\psi^t\rangle := (I\otimes U_t)O\ldots(I\otimes U_2)O(I\otimes U_1)O\left(\sum_{x\in S}a_x|x\rangle\otimes|\psi\rangle\right) = \sum_{x\in S}a_x|x\rangle\otimes|\psi_x^t\rangle. \tag{21.2}$$

The main idea of the approach is that for the algorithm to learn x, this state must become very entangled. To measure the entanglement of the pure state $|\psi^t\rangle$, we can consider the reduced density matrix of the oracle,

$$\rho^t := \sum_{x,y\in S} a_x^* a_y \langle\psi_x^t|\psi_y^t\rangle |x\rangle\langle y| \tag{21.3}$$

Initially, the state ρ^0 is pure. Our goal is to quantify how mixed it must become (i.e. how entangled the overall state must be) before we can compute f with error at most ϵ. To do this we could consider, for example, the entropy of ρ^t. However, it turns out that other measures are easier to deal with.

In particular, we have the following fact about the distinguishability of quantum states

Fact

Given one of pure states $|\psi\rangle, |\phi\rangle$, we can make a measurement that determines which state we have with error probability at most $\epsilon \in [0, 1/2]$ if and only if $|\langle\psi|\phi\rangle| \leq 2\sqrt{\epsilon(1-\epsilon)}$.

Thus, it is convenient to consider measures that are linear in the inner products $\langle\psi_x^t|\psi_y^t\rangle$.

The adversary method

To obtain an adversary lower bound, we choose a matrix $\Gamma \in \mathbb{R}^{|S|\times|S|}$ with rows and columns indexed by the possible black-box inputs. The entry $\Gamma_{x,y}$ is meant to characterize how hard it is to distinguish between x and y. We say that Γ is an *adversary matrix* if:

1. $\Gamma_{xy} = \Gamma_{yx}$
2. If $f(x) = f(y)$ then $\Gamma_{xy} = 0$.

The second condition reflects that we do not need to distinguish between x and y if $f(x) = f(y)$.

The original adversary method made the additional assumption that $\Gamma_{xy} \geq 0$ but it turns out this condition is not actually unecessary. Sometimes we refer to the *negative* or *generalized* adversary method to distinguish it from the original, positive-weighted method. While it may not be intuitively obvious what it would mean to give a negative weight to the entry characterizing distinguishability of two inputs, it turns out that this flexibility can lead to significantly improved lower bounds for some functions..

Given an adversary matrix Γ, we can define a weight function:

$$W^j := \sum_{x,y\in S} \Gamma_{xy} a_x^* a_y \langle\psi_x^j|\psi_y^j\rangle. \tag{21.4}$$

Note that this is a simple function of the entries of ρ^j. The idea of the lower bound is to show that W^j starts out large, must become small to compute f, and cannot change by much if we make a query.

the initial value of the weight function is:

$$W^0 = \sum_{x,y\in S} \Gamma_{xy} a_x^* a_y \langle\psi_x^0|\psi_y^0\rangle = \sum_{x,y\in S} a_x^* \Gamma_{xy} a_y \tag{21.5}$$

since $|\psi_x^0\rangle$ cannot depend on x. To make this as large as possible, we take a to be a principal eignevector of Γ, an eigenvector with eigenvalue $\pm\|\Gamma\|$. Then $|W^0| = \|\Gamma\|$.

The final value of the weight function is easier to bound if we assume a nonnegative adversary matrix. The final value is constrained by the fact that we must distinguish x from y with error probability at most ϵ whenever $f(x) \neq f(y)$. For this to hold after t queries, we need $|\langle\psi_x^t|\psi_y^t\rangle| \leq 2\sqrt{\epsilon(1-\epsilon)}$ for all pairs $x, y \in S$ with $f(x) \neq f(y)$ (by the above fact). Thus, if Γ has nonnegative entries, we have:

$$|W^t| \leq \sum x, y \in S\Gamma_{xy} a_x^* a_y 2\sqrt{\epsilon(1-\epsilon)} = 2\sqrt{\epsilon(1-\epsilon)}\|\Gamma\|. \tag{21.6}$$

Here we can include the terms where $f(x) = f(y)$ in the sum since $\Gamma_{xy} = 0$ for such pairs. We also used the fact that the principal eigenvector of a nonnegative matrix can be taken to have nonnegative entries (by the Perron-Frobenius theorem).

A similar bound holds if Γ has negative entries, but we need a different argument. In general, one can only show that $|W^t| \leq (2\sqrt{\epsilon(1-\epsilon)} + 2\epsilon)\|\Gamma\|$. But if we assume $f : S \to \{0,1\}$ has Boolean output, then we can prove the same bound as in the non-negative cae, and the proof is simpler than for a general output space. We can use the following simple result, stated in terms of the Frobenius norm $\|X\|_F^2 := \sum_{a,b} |X_{ab}|^2$:

Proposition

For any $X \in \mathbb{C}^{m\times n}$, $Y \in \mathbb{C}^{n\times n}$, $Z \in \mathbb{C}^{n\times m}$, we have $|\mathrm{Tr}(XYZ)| \leq \|X\|_F \|Y\| \|Z_F\|$.

Proof. We have:

$$\mathrm{Tr}(XYZ) = \sum_{a,b,c} X_{ab} Y_{bc} Z_{ca} = \sum_a (x^a)^\dagger Y z^a \tag{21.7}$$

where $(x^a)_b = X^*_{ab}$ and $(z^a)_c = Z_{ca}$. Thus:

$$\begin{aligned} |\mathrm{Tr}(XYZ)| &\leq \sum_a \|x^a\| \|Yz^a\| \\ &\leq \|Y\| \sum_a \|x^a\| \|z^a\| \\ &\leq \|Y\| \sqrt{\sum_a \|x^a\|^2 \sum_{a'} \left\|z^{a'}\right\|^2} \\ &= \|Y\| \|X\|_F \|Z\|_F \end{aligned}$$

as claimed, where we use Cauchy-Shwarz in the first and third steps. □

The upper bound $|W^t|$ for the negative adversary with Boolean output, write $W^t = \mathrm{Tr}(\Gamma V)$ where $V_{xy} := a_x^* a_y \langle \psi_x^t | \psi_y^t \rangle \delta[f(x) \neq f(y)]$. Now define:

$$C := \sum_{x\in S} a_x \Pi_{f(x)} |\psi_x^t\rangle\langle x| \tag{21.8}$$

$$\overline{C} := \sum_{x\in S} a_x \Pi_{1-f(x)} |\psi_x^t\rangle\langle x| \tag{21.9}$$

with Π_0, Π_1 projectors onto the $f(x) = 0, 1$ respectively. Then:

$$(C^\dagger \overline{C})_{xy} = a_x^* a_y \langle \psi_x^t | \Pi_{f(x)} \Pi_{1-f(y)} | \psi_y^t \rangle, \tag{21.10}$$

so:

$$(C^\dagger \overline{C} + \overline{C}^\dagger C)_{xy} = a_x^* a_y \langle \psi_x^t | (\Pi_{f(x)} \Pi_{1-f(y)} + \Pi_{1-f(x)} \Pi_{f(y)}) | \psi_y^t \rangle = a_x^* a_y \langle \psi_x^t | \psi_y^t \rangle \delta[f(x) \neq f(y)], \tag{21.11}$$

i.e. $V = C^\dagger \overline{C} + \overline{C}^\dagger C$. Thus we have:

$$W^t = \mathrm{Tr}(\Gamma(C^\dagger \overline{C} + \overline{C}^\dagger C)) = \mathrm{Tr}(\overline{C} \Gamma C^\dagger) + \mathrm{Tr}(C \Gamma \overline{C}^\dagger). \tag{21.12}$$

By the proposition, $|W^t| \leq 2\|\Gamma\| \|C\|_F \left\|\overline{C}\right\|_F$. Finally, we upper bound $\|C\|_F, \left\|\overline{C}\right\|_F$:

$$\|C\|_F^2 + \left\|\overline{C}\right\|_F^2 = \sum_{x,y\in S} |a_x|^2 (|\langle y | \Pi_{f(x)} | \psi_x^t \rangle|^2 + |\langle y | \Pi_{1-f(x)} | \psi_x^t \rangle|^2) = 1 \tag{21.13}$$

$$\left\|\overline{C}\right\|_F^2 = \sum_{x\in S} |a_x|^2 \left\|\Pi_{1-f(x)}|\psi_x^t\rangle\right\|^2 \leq \epsilon \tag{21.14}$$

Therefore $\|C\|_F \left\|\overline{C}\right\|_F \leq \max_{x\in[0,\epsilon]} \sqrt{x(1-x)} = \sqrt{\epsilon(1-\epsilon)}$ assuming $\epsilon \in [0,1/2]$ and thus $|W^t| \leq 2\sqrt{\epsilon(1-\epsilon)}\|\Gamma\|$ as claimed.

It remains to understand how much the weight function can decrease at each step of the algorithm. We have:

$$W^{j+1} - W^j = \sum_{x,y\in S} \Gamma_{xy} a_x^* a_y (\langle \psi_x^{j+1}|\psi_y^{j+1}\rangle - \langle \psi_x^j|\psi_y^j\rangle) \tag{21.15}$$

Consider how the state changes when we make a query. We have $|\psi_x^{j+1}\rangle = U_{j+1} O_x |\psi_x^j\rangle$. Thus the elements of the Gram matrix of the states $\left\{|\psi_x^{j+1}\rangle : x \in S\right\}$ are:

$$\langle \psi_x^{j+1}|\psi_y^{j+1}\rangle = \langle \psi_x^j|O_x^\dagger (U_{j+1})^\dagger U_{j+1} O_y|\psi_y^j\rangle = \langle \psi_x^j|O_x O_y|\psi_y^j\rangle \tag{21.16}$$

since U_{j+1} is unitary and O_x is Hermitian. Therefore:

$$W^{j+1} - W^j = \sum_{x,y\in S} \Gamma_{xy} a_x^* a_Y \langle \psi_x^j|(O_x O_y - I)|\psi_y^j\rangle. \tag{21.17}$$

Observe that $O_x O_y|i,b\rangle = (-1)^{b(x_i\oplus y_i)}|i,b\rangle$. Let $P_0 = I \otimes |0\rangle\langle 0|$ denote the projection onto the $b=0$ states, and let P_i denote the projection $|i,1\rangle\langle i,1|$. (As with O_x, the projections P_i implicitly act as the identity on any ancilla registers, so $\sum_{i=0}^n P_i = I$.) Then $O_x O_y = P_0 + \sum_{i=1}^n (-1)^{x_i\oplus y_i} P_i$, so $O_x O_y - I = -2\sum_{i:x_i\neq y_i} P_i$. Thus we have:

$$W^{j+1} - W^j = -2 \sum_{x,y\in S} \sum_{i:x_i\neq y_i} \Gamma_{xy} a_x^* a_y \langle \psi_x^j|P_i|\psi_y^j\rangle \tag{21.18}$$

Now for each $i \in \{1,\ldots,n\}$, let Γ_i be a matrix with:

$$(\Gamma_i)_{xy} := \begin{cases} \Gamma_{xy} & \text{if } x_i \neq y_i \\ 0 & \text{if } x_i = y_i \end{cases} \tag{21.19}$$

Then we have (doing more algebra and the proposition):

$$|W^{j+1} - W^j| \leq 2 \max_{i\in\{1,\ldots,n\}} \|\Gamma_i\|. \tag{21.20}$$

Since $|\Omega^0| = \|\Gamma\|$, we have:

$$|W^t| \geq \|\Gamma\| - 2t \max_{i\in\{1,\ldots,n\}} \|\Gamma_i\| \tag{21.21}$$

Thus, we have $|W^t| \leq 2\sqrt{\epsilon(1-\epsilon)}|\Gamma|$, we require:

$$t \geq \frac{1 - 2\sqrt{\epsilon(1-e)}}{2} \mathrm{Adv}(f) \tag{21.22}$$

where:

$$\mathrm{Adv}(f) := \max_{\Gamma} \frac{\|\Gamma\|}{\max_{i\in\{1,\ldots,n\}} \|\Gamma_i\|} \tag{21.23}$$

with the maximum taken over all adversary matrices Γ for the function f. Sometimes the notation $\mathrm{Adv}(f)$ is reserved for the maximization over nonnegative adversary matrices, with $\mathrm{Adv}^{\pm}(f)$ for the generalized method.

Unstructured search

As a simple application of this method, we prove the optimality of Grover's algorithm. It suffices to consider the problem of distinguishing between the case of no marked items and the case of a unique marked item (in an unknown location). Thus, consider the partial function where S consists of strings of Hamming weight 0 or 1, and f is the logical OR of the input bits.

For this problem, adversary matrices take the form:

$$\Gamma = \begin{pmatrix} 0 & \gamma_1 & \cdots & \gamma_n \\ \gamma_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_n & 0 & \cdots & 0 \end{pmatrix} \tag{21.24}$$

for some nonnegative $\gamma_1, \ldots, \gamma_n$. By symmetry we set them all equal - this can be formalized, but for now let's just consider it as an ansatz.

Setting $\gamma_1 = \ldots = \gamma_n = 1$ (since the overall scale is irrelevant for the bound), we have:

$$\Gamma^2 = \begin{pmatrix} n & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \cdots & 1 \end{pmatrix} \tag{21.25}$$

which has norm $\left\|\Gamma^2\right\| = n$, and hence $\|\Gamma\| = \sqrt{n}$. We also have:

$$\Gamma_1 = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & \\ 1 & 0 & 0 & \cdots & 0 & \\ 0 & 0 & 0 & \cdots & 0 & \\ & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \end{pmatrix} \tag{21.26}$$

and similarly for the other Γ_i, so $\|\Gamma_i\| = 1$. Thus, we find that $\mathrm{Adv}(\mathrm{OR}) \geq \sqrt{n}$, and so $Q_\epsilon(\mathrm{OR}) \geq \frac{1-2\sqrt{\epsilon(1-\epsilon)}}{2}\sqrt{n}$. This shows that Grover's algorithm is optimal up to a constant factor (recall that Grover finds a unique marked iterm with constant probability in $\frac{\pi}{4}\sqrt{n} + o(1)$ queries).

22 Adversary methods continued, Divide and Conquer

There is some repetition here with the material in the last section, which was a missed lecture.

Adversary bound

Proposition

Let $f : D \subseteq \{0,1\}^n \to \{0,1\}$. Then $Q_\epsilon(f) \geq \frac{1-2\sqrt{\epsilon(1-\epsilon)}}{2}\mathrm{Adv}(f)$.

Proof. We have:

$$W^j := \sum_{x,y \in D} a_x^* a_y \Gamma_{xy} \langle \psi_x^j | \psi_y^j \rangle. \tag{22.1}$$

Γ is an adversary matrix, i.e. it is symmetric, real, and $f(x) = f(y) \implies \Gamma_{xy} = 0$.

By choosing $a \in \mathbb{C}^{|D|}$ such that $\|\Gamma\| = \langle a|\Gamma|a\rangle$, we have $|W^0| = \|\Gamma\|$ and $|W^T| \leq 2\sqrt{\epsilon(1-\epsilon)}\|\Gamma\|$ (shown previously). Let us calculate the difference $W^{j+1} - W^j$:

$$W^{j+1} - W^j = \sum_{xy} a_x^* a_y \Gamma_{xy} \left(\langle \psi_x^{j+1} | \psi_y^{j+1} \rangle - \langle \psi_x^j | \psi_y^j \rangle \right) \tag{22.2}$$

note that $|\psi_x^{j+1}\rangle = U_{j+1} O_x |\psi_x^j\rangle$. Since the U cancels when we take the inner product, we note that:

$$\langle \psi_x^{j+1} | \psi_x^{j+1} \rangle = \langle \psi_x^j | O_x O_y | \psi_y^j \rangle \tag{22.3}$$

and in fact this motivates the definition of the “step”.

We use the phase oracle definition of O_x, i.e. rather than $O_x|i,b\rangle = |i, x_{i+1} \oplus b\rangle$ we take $O_x|i,b\rangle = (-1)^{b \cdot x_{i+1}}|i,b\rangle$. Then:

$$O_x O_y = \sum_{i,b} (-1)^{b(x_{i+1}+y_{i+1})} |i,b\rangle\langle i,b| \tag{22.4}$$

but since $\mathbb{I} = \sum_{i,b} |i,b\rangle\langle i,b|$. So then, $O_x O_y - \mathbb{I} = -2\sum_{i, x_i \neq y_i} P_i$ with $P_i = |i,1\rangle\langle i,1|$.

Now looking back at the difference of the Ws, we can write:

$$W^{j+1} - W^j = \sum_{x,y \in D} \sum_i a_x^* a_y (\Gamma_i)_{xy} \langle \psi_x^j | P_i | \psi_y^j \rangle = \sum_{i=1}^n \text{Tr}(Q \Gamma_i Q^\dagger) \tag{22.5}$$

where $Q = \sum a_x P_i |\psi_x^j\rangle\langle x|$. Then:

$$|W^{j+1} - W^j| \leq \sum_{i=1}^n \|Q_i\|_2^2 \|\Gamma_i\| \tag{22.6}$$

Then:

$$\|Q_i\|_2^2 = \text{Tr}(Q_i^\dagger Q_i) = \text{Tr}\left(\sum_x a_x^* |x\rangle \langle \psi_x^i | P_i^\dagger \sum_y a_y P_i |\psi_y^i\rangle\langle y| \right) = \sum_x |a_x|^2 \langle \psi_x^j | P_i | \psi_x^j \rangle \tag{22.7}$$

so then:

$$|W^{j+1} - W^j| \leq \max_i \|\Gamma_i\| \sum_i^n \|Q_i\|_2^2 = \max_i \|\Gamma_i\| \sum_i^n \sum_x \left\| P_i |\psi_x^j\rangle \right\|^2 \tag{22.8}$$

the norm square appearing in the above is less than 1.

Combining the three results $|W^0| = \|\Gamma\|$, bound on $|W^T|$, bound on $|W^{j+1} - W^j|$, we obtain:

$$T \geq \frac{1 - 2\sqrt{\epsilon(1-\epsilon)}}{2} \frac{\|\Gamma\|}{\max_i \|\Gamma\|} \implies T \geq \frac{1 - 2\sqrt{\epsilon(1-\epsilon)}}{2} \text{Adv}(f). \tag{22.9}$$

□

A simple example of divide-and-conquer

Proposition

$\exists c_1, c_2 > 0$ such that for any $f : \Sigma^n \to \{0,1\}$, $c_1 \text{Adv}(f) \leq Q(f) \leq c_2 \text{Adv}(f)$.

We won’t prove the upper bound part, but we’ll use it. The proof is similar. As a comment, you can take the Adv quantity, then consider a semidefinite program (then dualize) to get a minimizing program, then construct a quantum algorithm.

Why should we consider this adversary quantity when trying to study divide and conquer quantum algorithms? We consider:

Fact

For $f : \Sigma^a \to \{0,1\}$ and $g : \Sigma^b \to \{0,1\}$, if we consider $F : \Sigma^a \times \Sigma^b \to \{0,1\}$ iwth $F(x,y) = f(x) \wedge g(y)$ (or $\vee$) then $\text{Adv}(F) = \sqrt{\text{Adv}(f)^2 + \text{Adv}(g)^2}$.

Now if we consider $OR_n(x) = OR(OR_{n/2}(x_L), OR_{n/2}(x_R))$ but then this implies $Q_n \leq \sqrt{2}Q_{n/2}$ if we solve the recurrence so $Q_n = O(\sqrt{n})$. But this bound of $Q_n \leq \sqrt{2}Q_{n/2}$ is NOT TRUE for the quantum query complexity.

But if we consider the adversary quantity, $\text{Adv}(OR_n) \leq \sqrt{\text{Adv}(OR_{n/2})^2 + \text{Adv}(OR_{n/2})^2} = \sqrt{2}\text{Adv}(OR_{n/2})$. So what we hoped for for the quantum complexity holds exactly for the adversary object. Therefore $\text{Adv}(OR_n) \leq \sqrt{n}$.

As an example, suppose $\Sigma = \{0,1,2\}$. Then, $f : \Sigma^n \to \{0,1\}$. Then $f(x) = 1$ iff x contains a substring of the form $20 * 2$ ($*$ denoting any number of repeating zeros). So 20021 would be $f = 1$, and 20102 would be $f = 0$.

So how would we find this using divide and conquer? There are three possibilites if we split the input string; either the substring can appear in x_L or x_R or between the two. So, we consider:

$$f_n(x) = f_{n/2}(x_L) \vee f_{n/2}(x_R) \vee g(x) \tag{22.10}$$

with g accounting for the center part. So then:

$$\text{Adv}(f_n)^2 \leq 2\text{Adv}(f_{n/2})^2 + \text{Adv}(g)^2 \leq 2\text{Adv}(f_{n/2})^2 + O(Q(g)^2) \tag{22.11}$$

We do this by grover binary search, because we check to see if the bit string on the right is all zero $O(\sqrt{n})$, then halve that which is $O(\sqrt{n/2})$, and so on. Therefore the $O(Q(g)^2) = O((\sqrt{n})^2) = O(n)$. Therefore, the recurrency relation reads:

$$\text{Adv}(f(n))^2 = 2\text{Adv}(f_{n/2})^2 + O(n) \tag{22.12}$$

The $O(n)$ does not enter the recurrence part, and we end up solving it to find:

$$\text{Adv}_n(f_n) \leq O(\sqrt{n \log n}) \tag{22.13}$$

note that a direct Grover search method before this result gave $O(\sqrt{n}\log n)$.

Another note - bonus HW 2 problem. Doable with just Grover, but it's basically the same as this problem. The hard instance of this problem are when all the edges and the top and bottom are entirely present, and this technique gives a nice way to do it.

k-common subsequence

We take k a constant. Consider $\Sigma^n \to x =$ einstein and $\Sigma^n \to y =$ entwined. Here the e-n-t-i-n would be a common 4-subsequence. The k-common subsequence problem is whether the two strings share a k-common subsequence or not. We denote this as $-kCS$

The randomized complexity of this problem is $\Omega(n)$. If we assume we know what x is, then all it is is a search problem on y. $Q_(1 - CS) \leq O(n^{2/3})$ by its commonality to element distinctness. If we generalize, we find $O(k - CS) \leq O(n^{(k+1)/(k+2)})$ using quantum walk.

But in fact we can do better.

Proposition

For all constant k, $Q(kCS) \leq \tilde{O}(n^{2/3})$ ($\tilde{}$ denoting polylogarithmic factors).

We split up x, y into m parts. k-CS could be SIMPLE if the commonality occurs within the same part, or COMPOSITE otherwise.

For COMPOSITE, we have $O(\sum_{j=1}^{k-1} a_j(n)\log(n))$ where $a_j(n)$ is the Adv. quantity of $j - CS$.

For the SIMPLE case, A priori there are m^2 possible paths. We need to do a search of m^2 parts. But this is not actually what we need - we can get away with $O(m)$. $2m-1$ suffices. Do the following precomputation. Compute the m^2 lines between the blobs, where the line indicates whether the two blobs have a common symbol. But importantly, this is not whether the blobs share a k-CS, but rather a common symbol/1-CS. I can then afford to do m^2 checks because we only do a 1-CS. This is just ED so is $O(m^2 n^{2/3})$ (really $(n/m)^{2/3}$, but this wont matter). The worst case is lines going from all top ones to the first blob on the bottom and all bottom ones to the first blob on the top

23 Hamiltonian Simulation

What is a useful task we can do with a quantum computer? Shor is a strict negative for humanity, and even if it works we just move to quantum-safe encryption techniques.

One useful goal would be to simulate physical systems. An n-qubit system is specified by 2^n complex numbers/amplitudes, so any classical algorithm would generically have $O(2^n)$ space complexity. But on an n-qubit quantum system, we would expect we only need an n-qubit universal quantum computer - and this turns out to be indeed the case, with quantum systems able to be simulated in poly(n) time.

Hamiltonian H (hermitian) encodes the dynamics of the quantum system, with $\langle\psi|H|\psi\rangle$ denoting the average energy. The time evolution is given by the SE:

$$\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = -iH|\psi(t)\rangle \tag{23.1}$$

with solution $|\psi(t)\rangle = \exp(-iHt)|\psi(0)\rangle$. $\exp(-iHt)$ is the matrix exponential defined by:

$$\exp(A) = \sum_{j=0}^{\infty}\frac{A^j}{j!} \tag{23.2}$$

so, given H, t we want to simulate $U(t) = \exp(-iHt)$ to a suitable degree of approximation. To this end we review/introduce some useful definitions:

Definition: Operator norm

The *operator norm* of operator A is:

$$\|A\| = \max_{|\psi\rangle}\|A|\psi\rangle\| \tag{23.3}$$

with the max taken over normalized $|\psi\rangle$. If A is diagonalizable, then $\|A\|$ is the maximum eigenvalue of A.

Proposition: Operator norm inequalities

$$\|A+B\| \le \|A\| + \|B\| \tag{23.4}$$

$$\|AB\| \le \|A\|\|B\| \tag{23.5}$$

We will work with k-local Hamiltonians - for this fixed k, we expect poly(n) scaling (but not as we increase it).

Definition: k-local Hamiltonians

A Hamiltonian H is k-local on n qubits if:

$$H = \sum_{j=1}^{m} H_j, \tag{23.6}$$

where H_j acts on at most k qubits, i.e $H_j = \tilde{H}_j \otimes I$ where $\tilde{H}_j$ acts on some k qubits and I the others as identity.

The number of m terms we need in the above definition is bounded by:

$$m \leq \binom{n}{k} = O(n^k). \tag{23.7}$$

Some examples:

1. $H = X \otimes I \otimes I - Z \otimes I \otimes Y$ is 2-local on 3 qubits.
2. $H = J \sum_{i,j=1}^{n-1} Z_{(i,j)} Z_{(i,j+1)} + Z_{(i,j)} Z_{(i+1,j)}$ is the Ising model on an $n \times n$ lattice of qubits. It is 2-local on n^2 qubits.
3. $H = \sum_{i=1}^{n-1} J_x X_i X_{i+1} + J_y Y_i Y_{i+1} + J_z Z_i Z_{i+1}$ is the Heisenberg model on a n qubit line. It is 2-local on n bits.

Why is the idea of k-locality useful? The idea is we can simulate each $\exp(iH_j t)$ separately and combine. However, unless the $\left\{H_j\right\}$ are mutually commuting, then in general:

$$\exp(-i \sum_j H_j t) \neq \prod_j \exp(-iH_j t) \tag{23.8}$$

so we need to somehow solve this problem. Putting it aside for now, we can start with the quantum simulation problem. We make use of the following, technical, theorem (which proof is given, e.g., in Nielsen and Chuang):

Theorem: Solovay-Kitaev

Let U be a unitary be a unitary operator on k qubits and S any universal set of quantum gates. Then U can be approximated to within ϵ using $O(\log^c \frac{1}{\epsilon})$ from S, with $c < 4$.

Thus, we can simulate each $\exp(-iH_j t)$ with modest overhead in circuit side for improved error, assuming we fix k.

We also need to keep track of the accumulation of errors. To this end, the following Lemma is useful:

Lemma: Error Accumulation

Let $\{U_i\}, \{V_i\}$ be sets of unitary opeartors with:

$$\|U_i - V_i\| \leq \epsilon \tag{23.9}$$

then:

$$\|U_m \dots U_1 - V_m \dots V_1\| \leq m\epsilon. \tag{23.10}$$

Proof. (Sketch) Unitary gates preserve the size of vectors, and hence do not blow up errors - they simply accumulate linearly. □

Warm-up; easy case with mutually commuting terms.

Proposition: Commuting Hamiltonian Case

Let:

$$H = \sum_{j=1}^{m} H_j \tag{23.11}$$

be any k-local Hamiltonian with commuting terms.
Then for any $t, \exp(-iHt)$ can be approximated to within ϵ by a circuit of:

$$O\left(m\text{poly}\left(\log(\frac{m}{\epsilon})\right)\right) \tag{23.12}$$

gates from any universal gate set.

Proof. Pick $\epsilon' = \epsilon/m$ and approximate $\exp(-iH_j t)$ to within ϵ'. Then the total error is bounded by $m\epsilon' = \epsilon$, and this uses:

$$O\left(m\text{poly}\left(\log(\frac{m}{\epsilon})\right)\right) \tag{23.13}$$

gates. □

Full-commutative case; for this, note the piece of notation that $X + O(\epsilon)$ means $X + E$ with $\|E\| = O(\epsilon)$.

Lemma: Lie-Trotter Product Formula

Let A, B matrices with $\|A\|, \|B\| \leq K < 1$. Then:

$$\exp(-iA)\exp(-iB) = \exp(-i(A+B)) + O(K^2). \tag{23.14}$$

Proof.

$$\exp(-iA) = I - iA + \sum_{k=2}^{\infty} \frac{(iA)^k}{k!} = I - iA + (iA)^2 \sum_{k=0} \frac{(-iA)^k}{(k+2)!} \tag{23.15}$$

noticing that $\left\|(iA)^2\right\| \leq K^2$, and the final sum has norm bounded by $e^K < e$, so:

$$\exp(-iA) = I - iA + O(K^2). \tag{23.16}$$

Then we have:

$$\exp(-iA)\exp(-iB) = \exp(-i(A+B)) + O(K^2) \tag{23.17}$$

where we use Eq. (23.16) twice and then need that $\|A+B\| \leq 2K = O(K)$ and $\|AB\| \leq K^2 = O(K^2)$. □

We now apply this repeatedly to accumulate sums $H_1, H_2, \ldots, H_m$ in the exponent. First of all, we note that if each $\|H_i\| < K$, then $\|H_i + \ldots + H_l\| < lK$. We want this to be < 1 for all $l \leq m$. So for now, we assume $K < \frac{1}{m}$. Also, take $t = 1$ for now. Then:

$$\prod_j \exp(-iH_j) = \exp(-i\sum_j H_j) + O(m^3K^2) \tag{23.18}$$

where we repeatedly multiply and then use that $\sum_{n=1}^{m} n^2 = O(m^3)$. We write teh error as Cm^3K^2.

This is ok for small K, but it won't be in general. For general K and t values, we introduce a large N such that:

$$\left\| \frac{H_j t}{N} \right\| < \frac{Kt}{N} \le \|K\| < 1. \tag{23.19}$$

Imn other words, we divide time up into small t/N intervals.

$$U = \exp(-i \sum_j H_j t) = (\exp(-i \sum_j \frac{H_j t}{N}))^N \tag{23.20}$$

which holds because $\sum_j \frac{H_j t}{N}$ commutes with itself.

We now want to make sure the final error for U is $< \epsilon$. So, we know each term $\exp(-i \sum_j \frac{H_j t}{N})$ needs to be approximated to ϵ/N. So, using our previous formula, we want:

$$Cm^3\tilde{K}^2 < \frac{\epsilon}{N}, \tag{23.21}$$

Doing some algebraic manipulation, we find that we need:

$$N > \frac{Cm^3K^2t^2}{\epsilon} \tag{23.22}$$

We now have Nm gates of the form $\exp(iH_jt/N)$, so the circuit size is at most:

$$O(\frac{m^4(Kt)^2}{\epsilon}). \tag{23.23}$$

Recall for n qubits, a general k-local Hamiltonian has $m = O(n^k)$. So the circuit size is:

$$\|C\| = O(\frac{n^{4k}(Kt)^2}{\epsilon}). \tag{23.24}$$

Now this is in terms of the number of $\exp(iH_jt/N)$ gates. If we want to express this in terms of universal gates, then each needs to be approximated to $O(\epsilon/|C|)$. We then need $O(\log^c(\frac{|C|}{\epsilon}))$ gates for each, for some $c < e$. So we only get an extra modest multiplicative factor in $|C|$. Note tht for fixed n but variable t, the quantum process runs in t but our simulation needs $O(t^2)$, which can be improved to $O(t^{1+\delta})$ for any $\delta > 0$ by using "better" Lie-Trotter expansions.

Local Hamiltonian Problem

There are many other things we might want to do with a k-local Hamiltonian. One question we might be interested in is the eigenvalues of H. Suppose we are given a 5-local Hamiltonian:

$$H = \sum_{j=1}^{m} H_j \tag{23.25}$$

on n qubits. We suppose $\|H_i\| < 1$, and we are given two numbers $a < b$, such as $a = \frac{1}{3}$ and $b = \frac{2}{3}$. We are promised that the smallest eigenvalue E_0 of H is $< a$ or $> b$. The problem is to decide to whether $E_0 < a$.

The reason why we have these funny a, b is so that we don't have to worry about precision. If we only had a single a and want to determine if $E_0 > a$ or $E_0 < a$, it would be difficult if $E_0 \approx a$.

Kitaev's theorem says that the above problem is complete for a complexity class known as QMA, i.e. it is the "hardest" QMA problem. In other words, any problem in QMA (the quantum version of NP) can be translated into a local Hamiltonian problem.

24 Quantum Signal Processing

QSP is a powerful, unifying framework. Given an operator represented via a block encoding, we can encode spectral information via qubitization and then transform via QSP to implement a function of it. Applying this to the Hamiltonian Simulation problem, and using amplitude amplification, we can simulate sparse Hs with optimal complexity tradeoff. These techniques can also be applied in a wide variety of other algorithms.

Block Encoding

We say a unitary U is a block encoding of A if:

$$U = \begin{pmatrix} A & \cdot \\ \cdot & \cdot \end{pmatrix} = |0\rangle\langle 0| \otimes A + \ldots = \left(\langle 0| \otimes I\right) U \left(|0\rangle \otimes I\right) \tag{24.1}$$

Note that for A to have a blcok encoding, $\|A\| \leq 1$, but we can consider them under rescaling - i.e. A/α with $\alpha \geq \|A\|$ can be block encoded. α measures the quality of the encoding (with α small corresponding to a better encoding).

Obviously an efficient quantum circuit block-encodes itself, but we can give efficient block encodings of many other kinds of matrices - of particular interest is sparse matrices. Suppose $A \in \mathbb{C}^{N\times N}$ is d-sparse and efficiently row and column computable. Furthermore suppose $\max_{i,j}|A_{i,j}| \leq 1$. Then we can efficiently implement unitaries R, C acting on $\mathbb{C}^{3\times N\times N}$ as:

$$R : |0\rangle|0\rangle|i\rangle \mapsto |0\rangle \frac{1}{\sqrt{d}} \sum_{k=1}^{N} \sqrt{A^*_{ik}}|i\rangle|k\rangle + |1\rangle|i\rangle|\mu_j\rangle \tag{24.2}$$

$$R : |0\rangle|0\rangle|j\rangle \mapsto |0\rangle \frac{1}{\sqrt{d}} \sum_{k=1}^{N} \sqrt{A_{lj}}|l\rangle|j\rangle + |1\rangle|j\rangle|\nu_j\rangle \tag{24.3}$$

For some states $|\mu_i\rangle, |\nu_j\rangle$. This can be done via quantum walk implementations. Then, $R^\dagger C$ is a block encoding of A/d.

Block encodings have nice closure properties, e.g. A, B block encodings can be used to efficiently construct a block encoding of AB.

QSP

A key problem is tranforming a block encoding of one matrix into a related one. In particular, given a block encoding of A, when we can produce a block encoding of $f(A)$, and at what cost? This is addressed by quantum signal processing.

We describe Low/Yoder/Chuang's result for 2×2 matrices, which generalizes to higher dimensions.

Suppose we are given:

$$W(x) := \begin{pmatrix} x & i\sqrt{1-x^2} \\ i\sqrt{1-x^2} & x \end{pmatrix} = \exp(i \arccos(x)\sigma_x) \tag{24.4}$$

Our goal is to generate a matrix whose entries are polynomial in x. We do this by interspersing $W(x)$ with z-rotations, giving circuit:

$$W_\Phi(x) := \exp(i\phi_0\sigma_z)W(x)\exp(i\phi_1\sigma_z)W(x)\ldots W(x)\exp(i\phi_k\sigma_z) \tag{24.5}$$

where $\Phi := (\phi_0, \phi_1, \ldots, \phi_k)$. the functions $W(x)$ that can be realized this way are captured by the following Lemma:

Lemma: QSP

There exists $\Phi \in \mathbb{R}^{k+1}$ such that:

$$W_\Phi(x) = \begin{pmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix} \tag{24.6}$$

iff $P, Q \in \mathbb{C}[x]$ satify:

1. $\deg(P) \leq k$ and $\deg(Q) \leq k-1$.
2. P has parity $k \mod 2$ and Q has parity $k-1 \mod 2$
3. $\forall x \in [-1,1]$, $|P(x)|^2 + (1-x^2)|Q(x)|^2 = 1$.

Proof. We first show by induction on k that Eq. (24.6) implies three conditions:

For the induction step, we find that:

$$W_{(\phi_0,\ldots\phi_{k+1})} = \begin{pmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix} W(x)e^{i\phi_k\sigma_z} = \begin{pmatrix} \tilde{P}(x) & i\tilde{Q}(x)\sqrt{1-x^2} \\ i\tilde{Q}^*(x)\sqrt{1-x^2} & \tilde{P}^*(x) \end{pmatrix} \tag{24.7}$$

With:

$$\tilde{P}(x) = e^{i\phi_k}\left(xP(x) - (1-x^2)Q(x)\right) \tag{24.8}$$

$$\tilde{Q}(x) = e^{-i\phi_k}\left(P(x) + xQ(x)\right) \tag{24.9}$$

clearly satify the three conditions.

For the converse, we show by induction that the three conditions suffice to construct the decomposition.

For $k = 0$, $\deg(P) = 0$, so the third condition implies that $P(x) = \exp(i\phi_0)$ for some $\phi_0 \in \mathbb{R}$ and $Q(x) = 0$. Proof details can be filled in via the AMC notes. □

When lifting this decomposition to higher-dimensional cases via qubitizations, we will use a variant of QSP with reflections.

Qubitization

Qubitization lets us map a high-dimensional block encoding to a single qubit, to which we can apply QSP. This mapping relies on a decomposition of the block encoding into two-dimensional subspaces. This allows us to perform QSP in superposition on the high-dimensional target space.

Applications to Hamiltonian Simulation

QSP yields an optimal algorithm for simulation of sparse Hamiltonians.

As described in the block encoding section, we can construct an efficient block encoding of sparse H (scaled down by sparsity times largest magnitude of a matrix element). Our goal is to turn this into a block encoding of the evolution operator $\exp(-iHt)$.

To do this, we use the Jacobi-Anger expansion:

$$\exp(it\cos\theta) = \sum_{k=-\infty}^{\infty} i^k J_k(t)\exp(ik\theta) = J_0(t) + 2\sum_{k=1}^{\infty} i^k J_k(t) T_k(\cos\theta) \tag{24.10}$$

Where J_k is a Bessel function and $T_k(\theta) = \cos(k\theta)$ is a Chebyshev polynomial. By truncating this expression to the first K terms, we get a degree-K polynomial in x:

$$J_0(-t) + 2\sum_{k=1}^{K} i^k J_k(t) T_k(\cos\theta) \approx \exp(-itx) \tag{24.11}$$

Using this polynomial as f in the QSP Lemma, we get a good approximation of $\exp(-iHt)/2$.

To understand the quality of the approximation, we must bound the error incurred by truncating of the infinite series. We omit the details here as it is technial, but it turns out to be $O((t/2)^K/K!)$. To make this $O(\epsilon)$, we take:

$$K = \left(t + \frac{\ln(1/\epsilon)}{\ln(e + \ln(1/\epsilon)/t)}\right) \tag{24.12}$$

and this expression is tight.

Since this is scaled by a factor of 2, we need to scale it back up to achieve the desired deterministic simulation. We can use this via robust oblivious amplitude amplification with only constant-factor overhead. Overall, this gives a quantum algorithm for sparse Hamiltonian simulation with optimal dependence on both t and ϵ.