Lecture notes on quantum algorithms

Daochen Wang

December 2, 2024

Preface

This is a set of lecture notes on quantum algorithms. A feature of these notes is that it starts from (and focuses on) query complexity. Benefits of this approach are discussed in Lecture 1. The connection to the more standard measure of complexity, time complexity, is introduced only later, in Lecture 5. Much of the second half of this course follows a set of excellent lecture notes by Andrew M. Childs, available at [AMC]. In that case, I have referred the reader to the relevant parts of [AMC] that I covered.

Lecture 1

Brief introduction. Quantum computing is computation using the laws of quantum mechanics that were discovered in the early 20th century. Sometimes the power of quantum computing is described as: n quantum bits or qubits can be in the 2^n states of n bits at the same time. Therefore, it can solve a problem by simply trying 2^n, i.e., a large number for n large, possible solutions at the same time. But this explanation is not really correct. In fact, employing a similar logic, we may also say that n classical bits can be in 2^n states at the same time in a classical randomized computation where the n bits are determined by n random coin flips. In fact, quantum computation is similar to randomized computation in many ways. However, we will see that it subsumes randomized computation and can be provably more powerful than randomized computation in the so-called "query model" of computation. Much of this course will focus on the query model, which is *not* the same as the standard model of computation, known as the Turing model. We do this for three reasons:

1. Quantum-over-classical speedups (or simply quantum speedups) can be proven rigorously in the query model. In contrast, there is no rigorous proof of quantum speedup in the Turing model. This is because it is notoriously difficult to prove classical lower bounds in the Turing model.

2. Quantum speedups in the query model often translates into "apparent speedups" in the Turing model, where "apparent speedups" means quantum algorithms that are faster than any *known* classical algorithm. Moreover, quantum algorithms in the query model often contain the key ideas of their translations in the Turing model. For example, the apparent quantum speedup for factoring discovered by Peter Shor was motivated by a quantum speedup in the query model for the so-called Simon's problem.

3. We can describe quantum algorithms in the query model with much less setup than in the Turing model.

Let's get started! An alphabet is a finite non-empty set. $\mathbb{N}$ denotes the positive integers (no zero).

Let $n, m \in \mathbb{N}$, $\Sigma \coloneqq \{0, 1, \ldots, m-1\}$, D and Γ be alphabets, and

$$f \colon D \subseteq \Sigma^n \to \Gamma. \tag{1}$$

Mainly deal with case where $\Gamma = \{0, 1\}$.

Remark 1. (Remarks are informal.)

1. Main question of query complexity: given $x \in D$, how many bits of x need to be read (queried) to compute $f(x)$? (At most n.)

2. Three models of query computation: deterministic $D(f)$, randomized $R(f)$, quantum $Q(f)$. Quantum speedup means $Q(f) < R(f)$. Note that this means to establish a quantum speedup, we not only need to give a quantum algorithm but a classical lower bound. Therefore, a portion of this class will be devoted to *classical* analysis.

3. The access to the input y is different in the Turing model: y would be input to a classical Turing machine as a string. Quantum time complexity, for example, is then defined to be the time taken by that Turing machine to output the description of a quantum circuit C_y that computes the solution of the problem. The $x \in \{0, 1\}^n$ in the query model more appropriately translates to a function $x \colon [n] \to \{0, 1\}$ whose circuit can be described efficiently using the input y in the Turing model. Hopefully this will make more sense when we discuss how the query speedup for OR translates to an apparent speedup for kSAT – see Lectures 5 and 6.

Example 1.

1. $f \colon \{0, 1\}^3 \to \{0, 1\}$, $f(x) = (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)$. How many bits of the input $x \in \{0, 1\}$ do we need to read to determine f? Could always read all 3 bits but here there's a cleverer way. Read the bit x_1, then read x_2 if $x_1 = 1$ and x_3 if $x_1 = 0$.

2. $f = \mathrm{OR}_n \colon \{0, 1\}^n \to \{0, 1\}$

$$\mathrm{OR}_n(x) = x_1 \vee x_2 \vee \cdots \vee x_n. \tag{2}$$

   Computing OR_n models search among n elements since computing $\mathrm{OR}_n(x)$ is equivalent to finding a 1 among the n bits of x. Facts:

   (a) $D(\mathrm{OR}_n) = R(\mathrm{OR}_n) = \Theta(n)$ [1] Intuition: for a "worst-case" input x, you might query the first $n-1$ bits and they all show up as 0 so still need to make one more query to decide whether $\mathrm{OR}(x) = 0$ or $\mathrm{OR}(x) = 1$.

   (b) $Q(\mathrm{OR}_n) = \Theta(\sqrt{n})$ – due to Grover search, will see in Lecture 3.

[1] Asymptotic notation: for $f \colon \mathbb{N} \to \mathbb{C}$, $f(n) = \Theta(n)$ means there exist constants $0 < c_1 < c_2$ and $n_0 \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ s.t. $n > n_0$: $c_1 n \leq |f(n)| \leq c_2 n$.

(Note that computing OR_{2^n} relates to solving SAT on n Boolean variables since the latter is about searching for a satisfying assignment among 2^n possible assignments.)

The definitions of query complexity are dependent on the domain and codomain of the f under consideration, i.e., they are dependent on D, m, n, Γ. For convenience of notation, these dependencies are left implicit.

Definition 1 (Deterministic decision tree (or query algorithm))**.** A deterministic decision tree is an m-ary tree T with a unique vertex labelled as "root", together with the following additional data:

1. Each leaf[2] of T is labelled by an element in Γ.
2. Each non-leaf vertex of T is labelled by an element in $[n] \coloneqq \{1, 2, \ldots, n\}$.
3. For all non-leaf vertices v, the m edges between v and its m children are each labelled by a unique element in $\{0, 1, \ldots, m-1\}$. (For each non-leaf vertex of T, its neighbors going away from the root are known as its children. Each non-leaf vertex of T has exactly m children.)

Definition 2 (Deterministic query computation)**.** Let T be a deterministic decision tree and $x \in D$. We write $T(x) \in \{0, 1\}$ for the bit output by the following procedure

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

We say that a deterministic decision tree T computes f if

$$\forall x \in D,\ T(x) = f(x). \tag{3}$$

[The "for all" quantifier means this is sometimes referred to as "T computes f in the worst case".]

Definition 3 (Deterministic query complexity)**.** Given a deterministic decision tree (DDT) T, its depth $\mathrm{depth}(T)$ is the maximum length of a root-to-leaf path in T. Then

$$D(f) \coloneqq \min_{T \text{ DDT},\ T \text{ computes } f} \mathrm{depth}(T) \tag{4}$$

Definition 4 (Randomized decision tree (or query algorithm))**.** A randomized decision tree is a probability distribution $\mathcal{T}$ over deterministic decision trees.

Definition 5 (Randomized query computation)**.** Given $x \in D$ and a randomized decision tree $\mathcal{T}$, we write $\mathcal{T}(x)$ for the random variable on Γ defined by: for all $i \in \Gamma$,

$$\Pr[\mathcal{T}(x) = i] \coloneqq \Pr[T(x) = i \mid T \leftarrow \mathcal{T}]. \tag{5}$$

Let $\epsilon \in (0, 1/2)$. We say that a randomized decision tree $\mathcal{T}$ computes f with (two-sided) bounded-error ϵ if

$$\forall x \in D,\ \Pr[\mathcal{T}(x) = f(x)] \geq 1 - \epsilon. \tag{6}$$

Note that

$$\Pr[\mathcal{T}(x) = f(x)] = \Pr[T(x) = f(x) \mid T \leftarrow \mathcal{T}] = \sum_T \Pr[T \mid T \leftarrow \mathcal{T}] \cdot \mathbb{1}[T(x) = f(x)]. \tag{7}$$

Definition 6 (Randomized query complexity)**.** Given a randomized decision tree (RDT) T, its depth is defined by

$$\mathrm{depth}(\mathcal{T}) \coloneqq \max\{\mathrm{depth}(T) \mid \Pr[T \mid T \leftarrow \mathcal{T}] > 0\}. \tag{8}$$

Then for $\epsilon \in (0, 1/2)$,

$$R_\epsilon(f) \coloneqq \min\{\mathrm{depth}(\mathcal{T}) \mid \mathcal{T} \text{ RDT},\ \mathcal{T} \text{ computes } f \text{ with bounded-error } \epsilon\}. \tag{9}$$

Also standard to write

$$R(f) \coloneqq R_{1/3}(f). \tag{10}$$

Proposition 1. *$D(\mathrm{OR}_n) = n$.*

Proof. $D(\mathrm{OR}_n) \leq n$ is obvious (what's the DDT?).

For $D(\mathrm{OR}_n) \geq n$. Suppose for contradiction that there is a DDT T with $\mathrm{depth}(T) < n$ that computes OR_n. Consider the root-to-leaf path defined by following edges labelled by 0. We may assume wlog (without loss of generality) that the leaf vertex on this path is labelled by 0, else $T(0^n) = 1 \neq \mathrm{OR}_n(0^n)$, contradiction. Suppose the vertices on this path are labelled by $i_1, \ldots, i_d$, where $d < n$. Let $j \in [n] - \{i_1, \ldots, i_d\}$ (exists since $d < n$). Let $x \in \{0, 1\}^n$ be the all-zeros bitstring except for a 1 at position j. Then $T(x) = 0 \neq \mathrm{OR}_n(x)$ contradiction. □

[2] A leaf of T is a vertex of degree 1.

Lecture 2

Proposition 2. *For $\epsilon \in (0, 1/2)$, we have $R_\epsilon(\mathrm{OR}_n) \geq (1-2\epsilon)n$.*

Proof. Let $\mathcal{T}$ be a randomized decision tree that computes f with bounded-error ϵ. Then, for all $x \in D$,

$$\begin{aligned} 1-\epsilon &\leq \Pr[T(x) = f(x) \mid T \leftarrow \mathcal{T}] \\ &\leq \sum_T \Pr[T \mid T \leftarrow \mathcal{T}] \cdot \mathbb{1}[T(x) = f(x)]. \end{aligned}$$

Let μ be a distribution over $\{0,1\}^n$. Take the expectation of above equation over μ

$$1-\epsilon \leq \sum_T \Pr[T \mid T \leftarrow \mathcal{T}] \,\mathrm{E}[\mathbb{1}[T(x) = f(x) \mid x \leftarrow \mu]] = \sum_T \Pr[T \mid T \leftarrow \mathcal{T}] \Pr[T(x) = f(x) \mid x \leftarrow \mu]. \tag{11}$$

Since $\Pr[T \mid T \leftarrow \mathcal{T}]$ is a probability distribution over DDTs, there must exist a T^* in the support of $\mathcal{T}$ such that

$$\Pr[T^*(x) = f(x) \mid x \leftarrow \mu] \geq 1-\epsilon. \tag{12}$$

(The above steps form the "easy direction of Yao's principle" – see the remark after the proof.)

The above holds for an arbitrary distribution μ. To continue the proof, we set μ as follows

$$\mu(x) = \begin{cases} \frac{1}{2} & \text{if } x = 0^n, \\ \frac{1}{2n} & \text{if } |x| = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

where $|x|$ denotes Hamming weight.

For $x \in D$, let $T^*[x]$ denote the path followed by T^* on input x. Then define

$$Bad \coloneqq \{x \in D \mid \text{All edges of } T^*[x] \text{ are labelled by } 0\}. \tag{14}$$

Now

$$\Pr[x \in Bad \mid x \leftarrow \mu] = \frac{1}{2} + \frac{n-k}{2n} = 1 - \frac{k}{2n}, \tag{15}$$

where $k \leq \mathrm{depth}(T)$ is the length of the root-to-leaf path where all edges are labelled by 0. (Note that this is a *fixed* path.)

Comment: The point is that the decision tree T^*, sees the input x as the all-zeros string whp (with high probability) over $x \leftarrow \mu$, so the function it computes is a constant whp.

Two cases:

1. T^* outputs 0 if it only sees all-zeros. Then,

$$\begin{aligned} &\Pr[T^*(x) = 0] \\ =&\Pr[T^*(x) = 0 \mid x \in Bad] \Pr[x \in Bad] + \Pr[T^*(x) = 0 \mid x \notin Bad] \Pr[x \notin Bad] \geq 1 - \frac{k}{2n}, \end{aligned}$$

where all probabilities are over $x \leftarrow \mu$. Then

$$\begin{aligned} \Pr[T^*(x) = f(x)] &= \Pr[T^*(x) = f(x) = 0] + \Pr[T^*(x) = f(x) = 1] \\ &\leq \Pr[f(x) = 0] + \Pr[T^*(x) = 1] \\ &\leq \frac{1}{2} + \frac{k}{2n}. \end{aligned}$$

(Note that $\Pr[f(x) = 0] = 1/2$.) Therefore, for the left-hand side to be at least $1-\epsilon$, we require

$$k \geq (1-2\epsilon)n, \tag{16}$$

and therefore

$$\mathrm{depth}(T) \geq (1-2\epsilon)n. \tag{17}$$

2. T^* outputs 1 if it only sees all-zeros. Then,

$$\begin{aligned} &\Pr[T^*(x) = 1] \\ =&\Pr[T^*(x) = 1 \mid x \in Bad] \Pr[x \in Bad] + \Pr[T^*(x) = 1 \mid x \notin Bad] \Pr[x \notin Bad] \geq 1 - \frac{k}{2n}, \end{aligned}$$

where all probabilities are over $x \leftarrow \mu$.

[The analysis is symmetrical to the first case.] For completeness:

$$\begin{aligned}\Pr[T^*(x) = f(x)] &= \Pr[T^*(x) = f(x) = 0] + \Pr[T^*(x) = f(x) = 1] \\ &= \Pr[f(x) = 1] + \Pr[T^*(x) = 0] \\ &\leq \frac{1}{2} + \frac{k}{2n}.\end{aligned}$$

Therefore, for the left-hand side to be at least $1 - \epsilon$, we require

$$k \geq (1 - 2\epsilon)n, \tag{18}$$

and therefore

$$\text{depth}(T) \geq (1 - 2\epsilon)n. \tag{19}$$

□

Remark 2.

1. Question: can we prove the best possible lower bound on $R_\epsilon(f)$ by lower bounding the depth of a DDT (*deterministic* decision tree) that succeeds in computing $f(x)$ with probability $\geq 1-\epsilon$ where x is sampled according to some distribution μ? Answer: Yes! This is the content of the "hard direction of Yao's principle"; it is a corollary of von Neumann's minimax theorem that was first observed by Andrew Yao.

2. The above proof actually shows the following stronger result. Let $\text{OR}_n^{\leq 1} \colon D_{\leq 1} \to \{0,1\}$, where $D_{\leq 1} \coloneqq \{x \in \{0,1\}^n, |x| \leq 1\}$ ($|\cdot|$ denotes Hamming weight.) Then

$$R_\epsilon(\text{OR}_n^{\leq 1}) \geq (1 - 2\epsilon)n. \tag{20}$$

This is because the distribution μ defined in the proof is supported on $D_{\leq 1}$.

Quantum query computation. To define quantum computation, we will use Dirac notation. This notation is an alternative notation for linear algebra. It is possible to do quantum information without using this notation (e.g., famous quantum information theorist John Watrous has a book called "Theory of Quantum Information" that avoids Dirac notation entirely) but it has become standard and I find it convenient.

Definition 7 (Dirac notation)**.** For $n \in \mathbb{N}$, an n-dimension quantum state (or simply, state) is a unit column vector in $\mathbb{C}^n$, i.e., $v = (v_1, \ldots, v_n)^\top$ with $\|v\|_2 = 1$. (Euclidean norm: $\|v\| \coloneqq \sqrt{\sum_i |v_i|^2}$.)

The vector v is written in Dirac notation as $|v\rangle$ and called a "ket". The complex conjugate transpose of $|v\rangle$ is written $\langle v|$ and called a "bra", i.e., $\langle v| \coloneqq v^\dagger$. The naming is based on the observation that a bra $\langle v|$ (a row vector) can be right-multiplied by a $|w\rangle$ to give the inner product "braket" between v and w, i.e., $\langle v|w\rangle \coloneqq \langle v|\,|w\rangle = v^\dagger w = \langle v, w\rangle \in \mathbb{C}$. Note that $\langle v|$ can also be left-multiplied by $|w\rangle$ to give the outer product between w and v, i.e., $|w\rangle\langle v| = wv^\dagger \in \mathbb{C}^{n\times n}$. The tensor product of quantum states is their Kronecker product: if $|v\rangle \in \mathbb{C}^{n_1}$ and $|w\rangle \in \mathbb{C}^{n_2}$, then $|v\rangle\,|w\rangle \coloneqq |v\rangle \otimes |w\rangle \in \mathbb{C}^{n_1} \otimes \mathbb{C}^{n_2} = \mathbb{C}^{n_1 n_2}$.

Example of Kronecker product:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \otimes \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} u_1 v_1 \\ u_1 v_2 \\ u_1 v_3 \\ u_2 v_1 \\ u_2 v_2 \\ u_3 v_3 \end{pmatrix}. \tag{21}$$

The computational basis of $\mathbb{C}^N$ is the set of vectors $\{e^{(1)}, \ldots e^{(N)}\}$, where $e^{(i)} = (0, \ldots, 1 \ldots, 0)^\top$ is the all-zeros vector except with a 1 in the ith coordinate. It is conventional to reserve the symbol $|i\rangle$ for $e^{(i+1)}$ and $\langle i|$ for $e^{(i+1)\dagger}$. Then, for example, $|v\rangle = \sum_{i=0}^{N-1} v_i\,|i\rangle$ and $\langle v| = \sum_{i=0}^{N-1} v_i^*\,\langle i|$. An n-qubit quantum state is a 2^n-dimensional quantum state. Then, an n-qubit state can be written as $\sum_{x\in\{0,1\}^n} \alpha_x\,|x\rangle$, where $\alpha_x \in \mathbb{C}$ for all x and $|x\rangle \coloneqq |x_1\rangle\,|x_2\rangle \ldots |x_n\rangle \in (\mathbb{C}^2)^{\otimes n} = \mathbb{C}^{2^n}$.

Can also take Kronecker products of matrices, e.g., suppose V is an $n \times n$ matrix, then

$$\begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{pmatrix} \otimes V = \begin{pmatrix} u_{11}V & u_{12}V \\ u_{21}V & u_{22}V \end{pmatrix} \quad \text{RHS is a } 2n \times 2n \text{ matrix.} \tag{22}$$

Definition 8 (Projective measurement)**.** Let Γ be an alphabet and $d \in \mathbb{N}$. A Γ-outcome projective measurement $\mathcal{M}$ on $\mathbb{C}^d$ is a set of orthogonal projectors $\{\Pi_i \mid i \in \Gamma\}$, i.e., $\forall i, j \in \Gamma$, $\Pi_i\Pi_j = \delta_{i,j}\Pi_i$, $\forall i \in \Gamma, \Pi_i^\dagger = \Pi_i$, and $\sum_{i\in\Gamma} \Pi_i = \mathbb{1}_d$.

Given a quantum state $|\psi\rangle \in \mathbb{C}^d$, measuring $|\psi\rangle$ using $\mathcal{M}$ refers to a process that

1. Outputs $i \in \Gamma$ with probability $\|\Pi_i |\psi\rangle\|^2$. This i is referred to as the measurement outcome.

2. Changes the quantum state to

$$\frac{\Pi_i |\psi\rangle}{\|\Pi_i |\psi\rangle\|}. \tag{23}$$

Measurement in the computational basis on $\mathbb{C}^d$ refers to the d-outcome projective measurement defined by $\{|i\rangle\langle i| \mid i \in \{0, 1, \ldots, d-1\}\}$.

Lecture 3

Definition 9 (Quantum query algorithm)**.** A quantum query algorithm of depth $d \in \mathbb{N}$ is defined by the following data:

1. $w \in \mathbb{N}$. (Dimension of the workspace, i.e., non-query, part of the algorithm.)
2. $d+1$ unitary matrices $U_0, U_1, \ldots, U_d \in \mathbb{C}^n \otimes \mathbb{C}^m \otimes \mathbb{C}^w = \mathbb{C}^{nmw}$.
3. A Γ-outcome projective measurement $\mathcal{M} \coloneqq \{\Pi_s \mid s \in \Gamma\}$ on $\mathbb{C}^{nmw}$.

Definition 10 (Quantum oracle)**.** For $x \in \{0, \ldots, m-1\}^n$, the quantum oracle of x is the unitary matrix $O_x \in \mathbb{C}^{nm \times nm}$ defined by

$$O_x |i\rangle |j\rangle = |i\rangle |j + x_{i+1} \mod m\rangle , \tag{24}$$

for all $i \in \{0, 1, \ldots, n-1\}$ and $j \in \{0, \ldots, m-1\}$. (And linearly extended. $\mod m$ maps integers to the range $\{0, \ldots, m-1\}$)

In the special case where $m = 2$, this is the same as

$$O_x |i\rangle |b\rangle = |i\rangle |b \oplus x_{i+1}\rangle , \tag{25}$$

for all $i \in \{0, 1, \ldots, n-1\}$ and $b \in \{0, 1\}$, where $\oplus$ denotes XOR and $|b\rangle$ represents a 1-qubit quantum state.

Definition 11 (Quantum query computation)**.** Given $x \in D$ and a quantum query algorithm $\mathcal{A}$, we write $\mathcal{A}(x)$ for the random variable on $\{0, 1\}$ defined by, for all $i \in \Gamma$

$$\Pr[\mathcal{A}(x) = i] \coloneqq \| \Pi_i \cdot U_d (O_x \otimes \mathbb{1}_w) \ldots U_1 (O_x \otimes \mathbb{1}_w) U_0 |0\rangle \|^2, \tag{26}$$

where $\mathbb{1}_w \in \mathbb{C}^{w \times w}$ is the identity matrix and we recall $|0\rangle \in \mathbb{C}^{nmw}$ is the first computational basis vector. (Note there are d occurrences of O_x on the RHS.)

For $\epsilon \in (0, 1/2)$, we say that a quantum query algorithm $\mathcal{A}$ computes f with (two-sided) bounded-error ϵ if

$$\forall x \in D, \Pr[\mathcal{A}(x) = f(x)] \geq 1 - \epsilon, \tag{27}$$

where the probability is over the random variable $\mathcal{A}(x)$.

Definition 12 (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 (two-sided) bounded-error ϵ. Also standard to write $Q(f) = Q_{1/3}(f)$.

Grover's algorithm. Recall

$$\mathrm{OR}_n \colon \{0, 1\}^n \to \{0, 1\}. \tag{28}$$

It will be convenient to work with an alternative form of the quantum oracle.

Definition 13 (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

$$U_x |i\rangle |b\rangle = (-1)^{x_{i+1} \cdot b} |i\rangle |b\rangle . \tag{29}$$

Let

$$H \coloneqq \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{30}$$

denote the Hadamard matrix.

By the next lemma, it is clear that quantum query complexity does not change whether we use the phase oracle or the normal oracle.

Lemma 1 (Phase kickback trick)**.** *For all $x \in \{0, 1\}^n$, $U_x = (\mathbb{1}_n \otimes H) O_x (\mathbb{1}_n \otimes H)$. Moreover, since $H^2 = \mathbb{1}_2$, we also have $O_x = (\mathbb{1}_n \otimes H) U_x (\mathbb{1}_n \otimes H)$.*

Note that the quantum phase oracle of x can be implemented using one call to the quantum oracle of x together with unitaries *independent* of x, and vice versa. Therefore, if we defined quantum query complexity using the quantum phase oracle instead of the quantum oracle, the value of quantum query complexity would not change.

Proof. Note that for $b \in \{0, 1\}$, we have

$$H |b\rangle = \frac{1}{\sqrt{2}} (|0\rangle + (-1)^b |1\rangle). \tag{31}$$

Then

$$\begin{aligned}
&|i\rangle|b\rangle \overset{\mathbb{1}_n\otimes H}{\mapsto} |i\rangle \frac{1}{\sqrt{2}}(|0\rangle + (-1)^b|1\rangle) \overset{O_x}{\mapsto} \frac{1}{\sqrt{2}}|i\rangle(|x_{i+1}\rangle + (-1)^b|x_{i+1}\oplus 1\rangle)\\
\overset{\mathbb{1}_n\otimes H}{\mapsto} &\frac{1}{\sqrt{2}}|i\rangle(\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_{i+1}}|1\rangle) + (-1)^b\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_{i+1}\oplus 1}|1\rangle))\\
=&\frac{1}{2}|i\rangle((1+(-1)^b)|0\rangle + (-1)^{x_{i+1}}(1-(-1)^b)|1\rangle)\\
=&(-1)^{x_{i+1}\cdot b}|i\rangle|b\rangle,
\end{aligned}$$

as required. □

Remark 3. There is a generalization of the quantum phase oracle definition for $m > 2$ ($x \in \{0,1,\ldots,m-1\}^n$) — see [AMC] Section 20.2.

For $t \in \mathbb{N}$, define $\mathrm{OR}_n^{0,t}$ to be OR_n with the restricted domain $D_{0,t} \coloneqq \{x \in \{0,1\}^n \mid |x| \in \{0,t\}\}$.

Proposition 3 (Grover's algorithm)**.** *For all $n, t \in \mathbb{N}$ with $t \le n/3$,*

$$Q(\mathrm{OR}_n^{0,t}) \le \frac{\pi}{4}\sqrt{\frac{n}{t}} + \frac{1}{2}. \tag{32}$$

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

$$|\psi\rangle \coloneqq \frac{1}{\sqrt{n}}\sum_{i=0}^{n-1}|i\rangle, \tag{33}$$

and let $G \in \mathbb{C}^{n\times n}$ denote the following unitary matrix

$$G \coloneqq \mathbb{1}_n - 2|\psi\rangle\langle\psi|. \tag{34}$$

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

$$V_x \coloneqq \sum_{i=0}^{n-1}(-1)^{x_{i+1}}|i\rangle\langle i| = \mathbb{1}_n - 2\sum_{i|x_{i+1}=1}|i\rangle\langle i|. \tag{35}$$

(V_x can be instantiated using the quantum phase oracle with b set to 1, and still uses 1 call to O_x.)

Let

$$\Pi_0 \coloneqq |\psi\rangle\langle\psi| \quad \text{and} \quad \Pi_1 \coloneqq \mathbb{1}_n - \Pi_0. \tag{36}$$

Clearly, $\{\Pi_0, \Pi_1\}$ defines a $\{0,1\}$-outcome measurement on $\mathbb{C}^n$.

For $k \in \mathbb{N}$, we now consider the following quantity, which can be seen as the probability that a k-query quantum algorithm outputs 0:

$$p_x \coloneqq \|\Pi_0(GV_x)^k|\psi\rangle\|^2. \tag{37}$$

Two cases:

1. $x = 0^n$. In this case $V_x = \mathbb{1}_n$ and $G^k|\psi\rangle = (-1)^k|\psi\rangle$ so $p_x = 1$.
2. $|x| = t$. Define the following orthogonal quantum states:

$$|\psi_0\rangle \coloneqq \frac{1}{\sqrt{n-t}}\sum_{i|x_{i+1}=0}|i\rangle, \tag{38}$$

$$|\psi_1\rangle \coloneqq \frac{1}{\sqrt{t}}\sum_{i|x_{i+1}=1}|i\rangle. \tag{39}$$

Then

$$|\psi\rangle = \sqrt{1-\frac{t}{n}}|\psi_0\rangle + \sqrt{\frac{t}{n}}|\psi_1\rangle = \cos(\theta)|\psi_0\rangle + \sin(\theta)|\psi_1\rangle, \tag{40}$$

where $\theta \coloneqq \arcsin(\sqrt{t/n}) \in (0, \pi/2]$.

We have

$$\begin{aligned}
GV_x|\psi_0\rangle &= G|\psi_0\rangle = |\psi_0\rangle - 2\cos(\theta)|\psi\rangle = (1-2\cos^2(\theta))|\psi_0\rangle - 2\cos(\theta)\sin(\theta)|\psi_1\rangle = -\cos(2\theta)|\psi_0\rangle - \sin(2\theta)|\psi_1\rangle.\\
GV_x|\psi_1\rangle &= -G|\psi_1\rangle = -|\psi_1\rangle + 2\sin(\theta)|\psi\rangle = 2\sin(\theta)\cos(\theta)|\psi_0\rangle + (2\sin^2(\theta)-1)|\psi_1\rangle = \sin(2\theta)|\psi_0\rangle - \cos(2\theta)|\psi_1\rangle.
\end{aligned}$$

Therefore, GV_x applied to the state $|\psi\rangle$ always stays in the 2-dimensional subspace $\mathrm{span}(|\psi_0\rangle, |\psi_1\rangle) \leq \mathbb{C}^n$. Therefore, we can reduce the analysis to linear algebra in $\mathbb{C}^2$ by working in the basis $|\psi_0\rangle, |\psi_1\rangle$. In this basis, $|\psi\rangle$ is represented as

$$\begin{pmatrix}\cos(\theta)\\ \sin(\theta)\end{pmatrix}, \tag{41}$$

and $-GV_x$ is represented as

$$A \coloneqq \begin{pmatrix}\cos(2\theta) & -\sin(2\theta)\\ \sin(2\theta) & \cos(2\theta)\end{pmatrix}. \tag{42}$$

This is the rotation matrix by angle 2θ anticlockwise. Therefore,

$$A^k = \begin{pmatrix}\cos(2k\theta) & -\sin(2k\theta)\\ \sin(2k\theta) & \cos(2k\theta)\end{pmatrix}. \tag{43}$$

(This is geometrically intuitive, but can also prove this rigorously by diagonalizing A and then taking the kth power, as in Homework 1.)

Therefore,

$$A^k \begin{pmatrix}\cos(\theta)\\ \sin(\theta)\end{pmatrix} = \begin{pmatrix}\cos(2k\theta)\cos(\theta) - \sin(2k\theta)\sin(\theta)\\ \sin(2k\theta)\cos(\theta) + \cos(2k\theta)\sin(\theta)\end{pmatrix} = \begin{pmatrix}\cos((2k+1)\theta)\\ \sin((2k+1)\theta)\end{pmatrix}. \tag{44}$$

Therefore,

$$(GV_x)^k |\psi\rangle = (-1)^k(\cos((2k+1)\theta)|\psi_0\rangle + \sin((2k+1)\theta)|\psi_1\rangle). \tag{45}$$

Therefore,

$$p_x = [\cos(\theta)\cos((2k+1)\theta) + \sin(\theta)\sin((2k+1)\theta)]^2 = \cos^2(2k\theta).$$

Let $r \coloneqq \frac{\pi}{4\theta}$ and $k \coloneqq \lfloor r \rceil \in [r - 1/2, r + 1/2]$ (where $\lfloor\cdot\rceil$ denotes rounding to the nearest integer). Then

$$p_x = \cos^2(2k\theta) \leq \cos^2(2(r-1/2)\theta) \qquad \text{(to see the } \leq\text{, draw } \cos^2(A) \text{ around } A = \pi/2) \tag{46}$$

$$= \cos^2(\pi/2 - \theta) = \sin^2(\theta) = \frac{t}{n} \leq 1/3. \qquad \text{(last } \leq \text{ by proposition conditions)} \tag{47}$$

Therefore,

$$Q(\mathrm{OR}_n^{0,t}) \leq k \coloneqq \lfloor \frac{\pi}{4\theta} \rceil \leq \frac{\pi}{4\theta} + \frac{1}{2} = \frac{\pi}{4\arcsin(\sqrt{t/n})} + \frac{1}{2} \leq \frac{\pi}{4}\sqrt{\frac{n}{t}} + \frac{1}{2}, \tag{48}$$

where the last inequality uses $\arcsin(a) \geq a$ for all $a \in [0, 1]$.

□

Lecture 4

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

In this lecture, we'll see two very useful principles of quantum algorithm design given as the two items in Fact 1 below. We will apply these two principles to show that the quantum query complexity of OR_n (without any restriction on domain) is also $O(\sqrt{n})$. In later lectures, we will take these principles for granted and not explicitly mention them.

Fact 1.

1. Quantum (query) algorithms can efficiently simulate randomized (query) algorithms. In particular $Q(f) \leq R(f)$ for any f. Reference: Section 2.3.3 of [de Wolf thesis].

   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.

   We will use the following

   Fact (*). Suppose $g\colon \{0, 1 \ldots, a-1\} \to \{0, 1, \ldots, b-1\}$, then there exists a unitary U_g (in fact 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

   $$U_g |i\rangle |j\rangle = |i\rangle |j + g(i) \mod b\rangle \tag{49}$$

   for all $i \in \{0, 1, \ldots, a-1\}$ and $j \in \{0, 1, \ldots, b-1\}$.

   (Proof: Eq. (49) completely defines U_g, check that the definition implies U_g is unitary – in fact, a permutation matrix.)

   Let $I\colon \{0,1\} \to \{2,3\}$ 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 $h\colon \{0,1\} \times \{0,1,2\} \times \{0,1\} \to \{0,1\}$ be defined by

   $$h(0, 2-1, 0) = 0,\ h(0, 2-1, 1) = 1,\ h(1, 3-1, 0) = 1,\ h(1, 3-1, 1) = 0. \tag{50}$$

   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 $x_I = b$.

   Register dimensions $\mathbb{C}^3 \otimes \mathbb{C}^2 \otimes \mathbb{C}^3 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$:

   $$\begin{aligned}
   &\underbrace{|0\rangle |0\rangle}_{\text{query registers}} \quad \underbrace{|0\rangle |0\rangle |0\rangle}_{\text{workspace registers}} \\
   &\overset{O_x}{\mapsto} |0\rangle |x_1\rangle |0\rangle |0\rangle |0\rangle \\
   &\overset{U_{I-1}}{\mapsto} |0\rangle |x_1\rangle |I(x_1) - 1\rangle |0\rangle |0\rangle && \text{notation follows fact (*)} \\
   &\overset{O_x}{\mapsto} |0\rangle |x_1\rangle |I(x_1) - 1\rangle |x_{I(x_1)}\rangle |0\rangle \\
   &\overset{U_h}{\mapsto} |0\rangle |x_1\rangle |I(x_1) - 1\rangle |x_{I(x_1)}\rangle |h(x_1, I(x_1) - 1, x_{I(x_1)})\rangle && \text{notation follows fact (*)} \\
   &= |0\rangle |x_1\rangle |I(x_1) - 1\rangle |x_{I(x_1)}\rangle |T(x)\rangle && \text{definition of } h
   \end{aligned}$$

   where the $\overset{A}{\mapsto}$ notation means application of matrix A (suitably tensored with identity matrices), and the last line uses the definition of h. Then measuring using $\{\Pi_0 \coloneqq \mathbb{1}_{36} \otimes |0\rangle\langle 0|, \Pi_1 \coloneqq \mathbb{1}_{36} \otimes |1\rangle\langle 1|\}$ gives outcome $T(x)$ (with probability 1).

   What about RDTs? Recall an RDT is a distribution $(p_i, T_i)_{i=0}^{K-1}$ over DDTs. 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 $\{U_j^i\}_{j=0,\ldots,d}$. Then the RDT can be simulated by a quantum query algorithm $\mathcal{A}$ that starts with the state

   $$|\psi_0\rangle \coloneqq \sum_{i=0}^{K-1} U_0^i |0\rangle \otimes \sqrt{p_i}\, |i\rangle\,. \tag{51}$$

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

   $$U_j \coloneqq \sum_{i=0}^{K-1} U_j^i \otimes |i\rangle\langle i|. \tag{52}$$

   □

The measurement of $\mathcal{A}$ is still $\{\Pi_0 \coloneqq |0\rangle\langle 0|, \Pi_1 \coloneqq |0\rangle\langle 0|\}$ (tensored with identities so that the Π_bs only act non-trivially on the single register that contains $\{T_i(x) \mid i \in \{0, \dots, K-1\}\}$.

2. Principle of deferred measurement.

   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.

   Proof of principle of deferred measurement. Suppose we make a measurement $\mathcal{M} \coloneqq \{\Pi_1, \dots, \Pi_k\}$ 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 (later), need $|\psi'\rangle$ to be the postmeasurement state of $|\psi\rangle$—but the proof is the same. Then the effect of this procedure is that with probability $\|\Pi_i |\psi\rangle\|^2$, we end up with final state $U_i |\psi'\rangle$.

   Now consider the following simulation: we apply the unitary

$$U \coloneqq \sum_{i=1}^{n} \Pi_i \otimes U_i \tag{53}$$

   to the state $|\psi\rangle |\psi'\rangle$ and *then* measure the first register using $\mathcal{M}$. (Note U is unitary: $UU^\dagger = \sum_{i=1}^n \Pi_i \otimes U_i \cdot \sum_{j=1}^n \Pi_j \otimes U_j^\dagger = \sum_{i=1}^n \Pi_i \otimes I = I$; Likewise $U^\dagger U = I$.)
   Then the probability of observing outcome $i \in [k]$ is

$$\|(\Pi_i \otimes \mathbb{1})U |\psi\rangle |\psi'\rangle\|^2 = \|\Pi_i |\psi\rangle \otimes U_i |\psi'\rangle\|^2 = \|\Pi_i |\psi\rangle\|^2, \tag{54}$$

   where the last equality uses the fact that $\|u \otimes v\| = \|u\|\|v\|$ and $\|Vu\| = \|V\|$ 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 facts (implicitly), can show the following.

Proposition 4. *There exists $c > 0$ such that for all $n \in \mathbb{N}$, we have*

$$Q(\mathrm{OR}_n) \le c\sqrt{n}. \tag{55}$$

Proof sketch. 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

$$\left(1 - \frac{0.01n}{n}\right)^{10000} \le e^{-100} = \text{negligible}^3 \tag{56}$$

where the inequality uses $1 - x \le e^{-x}$ for all $x \ge 0$.

From the analysis before, we see that, on input $x \in \{0,1\}^n$ using k queries we can get the probability of outputting 0 to be

$$p_x(k) = \cos^2(2\theta_x k) = \frac{1 + \cos(4\theta_x k)}{2}, \tag{57}$$

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

$$15 \le \frac{\pi}{2 \arcsin\sqrt{0.01}} \le T_x \coloneqq \frac{\pi}{2\theta_x} \le \frac{\pi}{2}\sqrt{n}, \tag{58}$$

where the second 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 $a \in [0,1]$).

Observation: as k runs over the interval $[1, \lceil \frac{\pi}{2}\sqrt{n} \rceil]$, $p_x(k)$ runs over at least one period (by the second inequality of Eq. (58)) and each period must span over at least 15 positive integers (by the first inequality of Eq. (58)).

[3] Note that this is "negligible" since we only care about computing OR_n with bounded error 1/3 and *compared to 1/3*, e^{-100} is negligible. To argue this formally, we need to consider the probabilities of failure from all sources (there's another source later on), add them together (cf. the "union bound" or "Boole's inequality" on Wikipedia) and show that the sum is $\le 1/3$.

The last step of the algorithm is:

> Repeat the following 10000 times: choose $k \in \mathbb{N}$ uniformly at random from $[1, \lceil \frac{\pi}{2}\sqrt{n} \rceil]$, run Grover's quantum query algorithm which has $p_k(x)$ probability of outputting 0 (i.e., the measurement outcome is 0). If the output is 1, return 1.
>
> If all repeats give output 0, return 0.

Clearly the quantum query algorithm uses $O(\sqrt{n})$ queries.

The intuition for correctness is that if we choose an integer k uniformly at random from $[1, \lceil \frac{\pi}{2}\sqrt{n} \rceil]$, the previous "Observation" means that $p_x(k)$ will be 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.[4] □

Remark 4.

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 [Aaronson notes] – 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 $O(\sqrt{n})$, is analyzed in detail in Section 4 of [BBHT'96].

4. In fact, there's yet another algorithm for computing OR_n 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 [Yoder, Low, Chuang'14]. Maybe we'll have time to discuss this when we talk about quantum signal processing.

Proposition 5 (Error suppression/Chernoff bound)**.** *Let* $\epsilon \in (0, 1/3)$. *Let* $f\colon D \subseteq \{0, 1, \ldots, m-1\}^n \to \Gamma$. *Then* $R_\epsilon(f) \leq R(f)\lceil 18\ln(1/\epsilon)\rceil$ *and* $Q_\epsilon(f) \leq Q(f)\lceil 18\ln(1/\epsilon)\rceil$.

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

Suppose $\mathcal{T}$ is an 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 output the correct answer on x, the probability that each copy outputs the correct answer is $p = \frac{1}{2} + \delta$, where $\delta \geq 1/6$ and the probability that each copy outputs the incorrect answer is $q = 1 - p = \frac{1}{2} - \delta \leq 1/3$. Correct $\iff X > k/2$. So probability of incorrect is

$$\begin{aligned}
\Pr[X \leq k/2] &= \sum_{i=0}^{k/2} \Pr[X = i] = \sum_{i=0}^{k/2} \binom{k}{i} p^i q^{k-i} \\
&\leq \sum_{i=0}^{k/2} \binom{k}{i} p^{k/2} q^{k/2} \leq 2^k (pq)^{k/2} \\
&= 2^k \Big(\frac{1}{2} + \delta\Big)^{k/2} \Big(\frac{1}{2} - \delta\Big)^{k/2} = 2^k \Big(\frac{1}{4} - \delta^2\Big)^{k/2} \\
&= (1 - 4\delta^2)^{k/2} \leq e^{-2k\delta^2} \qquad \forall x \geq 0,\ 1 - x \leq e^{-x}.
\end{aligned}$$

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 5. We have shown that given k i.i.d. random variables $X_1, \ldots, X_k$ taking values 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.

[4] The proof of this is similar to how we got the first "negligible" and the footnote about the first "negligible" also applies here.

Lecture 5

The Turing model and its relation to the query model. In the Turing model, we typically consider decision problems and the time complexity of solving them. Comment: everything in this class can be generalized to non-decision, aka "search", problems, but for convenience of notation, we'll stick with decision problems. Time complexity in the Turing model matches our normal intuition of how long it takes a problem to be solved. In this lecture, we will define time complexity formally and see how it relates to query complexity in the quantum setting.

In the following, $\{0,1\}^*$ is the set of all finite-length bitstrings, including the empty bitstring, denoted ϵ.

Definition 14. A decision problem is a function $\mathcal{P}\colon \{0,1\}^* \to \{0,1\}$.

Remark 6.

1. This corresponds to the following "problem" in the colloquial sense: input $y \in \{0,1\}^*$, problem is to output $\mathcal{P}(y)$.
2. For those who have taken complexity theory, a decision problem is equivalent to a language by the correspondence:

$$\mathcal{P} \leftrightarrow L \subseteq \{0,1\}^*;\ L \coloneqq \mathcal{P}^{-1}(1). \tag{59}$$

We will define deterministic, randomized, and quantum time-complexity of solving $\mathcal{P}$ via classical and quantum *circuits*.

Definition 15 (Classical and quantum circuits)**.**

1. A classical (Boolean/bit) circuit is a directed acyclic graph with $a \in \mathbb{N}$ vertices labelled uniquely by $1,\dots,a$ that have no incoming edges and at most one outgoing edge ("a input bits") and $b \in \mathbb{N}$ vertices labelled uniquely by $1',\dots,b'$ that have no outgoing edges and exactly one incoming edge ("b output bits"), and all other vertices are labelled by symbols from set

$$\text{cGATES} \coloneqq \{\text{FANOUT}, \text{AND}, \text{OR}, \text{NOT}\}, \tag{60}$$

such that vertices labelled AND and OR have two incoming edges and one outgoing edge, vertices labelled by FANOUT have one incoming edge and two outgoing edges, and vertices labelled by NOT have one incoming edge and one outgoing edge.

2. A quantum (Boolean/qubit) circuit is defined by the following data[5]:
   (a) $a \in \mathbb{N}$. (In this case, we say the quantum circuit "is on a qubits" or has "a input and output qubits".)
   (b) A finite sequence of elements each of the form (H,i), (T,j), or $(\text{Toffoli},(k_1,k_2,k_3))$, where $i,j,k_1,k_2,k_3 \in [a]$ and k_1,k_2,k_3 are distinct.

   We also define the set of symbols

$$\text{qGATES} \coloneqq \{T, H, \text{Toffoli}\}. \tag{61}$$

Definition 16 (Computation using classical and quantum circuits.)**.** A classical circuit with a input bits and b output bits computes in the natural way with the natural interpretations of the symbols AND, OR, NOT as (Boolean logic) gates; FANOUT is interpreted as the gate that takes an input bit and fans it out into two copies . Given $y \in \{0,1\}^a$, $C(y) \in \{0,1\}^b$ is defined in the natural way.

A quantum circuit C with a input and output qubits computes as follows. Given $y \in \{0,1\}^a$, $C(y)$ is the random variable on $\{0,1\}^a$ defined by applying the finite sequence defining C on $|y\rangle \in \mathbb{C}^{2^a}$ in order, under the interpretation:

$$(H,i) \to \mathbb{1}_2^{\otimes i-1} \otimes \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \otimes \mathbb{1}_2^{\otimes a-i}, \quad (T,j) \to \mathbb{1}_2^{\otimes j-1} \otimes \begin{pmatrix}1 & 0\\ 0 & \exp(i\pi/4)\end{pmatrix} \otimes \mathbb{1}_2^{a-j},$$

$(\text{Toffoli},(k_1,k_2,k_3)) \to$ unitary $\text{Toffoli}_{k_1,k_2,k_3}$ defined by:

$$|\alpha\rangle_{k_1}|\beta\rangle_{k_2}|\gamma\rangle_{k_3}|w\rangle_{\text{rest}} \mapsto |\alpha\rangle_{k_1}|\beta\rangle_{k_2}|\gamma \oplus \alpha\cdot\beta\rangle_{k_3}|w\rangle_{\text{rest}} \text{ for all } \alpha,\beta,\gamma \in \{0,1\}, w \in \{0,1\}^{a-3}, \tag{62}$$

where the subscripts denote the position of the ket (e.g., when $a=5$, $|0\rangle_2|0\rangle_3|0\rangle_5|11\rangle_{\text{rest}}$ means $|10010\rangle$). Then, measuring using $\{\Pi_z \coloneqq |z\rangle\langle z| \mid z \in \{0,1\}^a\}$.

[5] In class, I defined a quantum circuit analogously to a classical circuit, i.e., as a directed acyclic graph with non-input/output vertices labelled by elements in qGATES. However, upon reflection, this definition is problematic when it comes to defining computation. In particular, this definition makes the subsequent definition of computation not well-defined since the latter definition interprets Toffoli as a *non-symmetric* gate, i.e., we don't have $\text{Toffoli}|x_1x_2x_3\rangle = |x_{\sigma(1)}x_{\sigma(2)}x_{\sigma(3)}\rangle$ for all $x_1,x_2,x_3 \in \{0,1\}, \sigma \in S_3$ (S_3 = set of permutations on 3 objects). The definition given here follows Definition 6.1 of [Kitaev, Shen, Vyalyi].

Remark 7. There are alternative definitions of classical and quantum circuits that use other gate sets. But the resulting time-complexities don't differ significantly – just like how the definition of Turing machine is rather robust to number of tapes, size of alphabets, etc.

For example, often see qGATES $= \{T, H, \text{CNOT}\}$. But this only leads to "constant" complexity differences since 1 CNOT can be simulated by 1 Toffoli gate and, conversely, 1 Toffoli can be simulated by 6 CNOT gates, 2 H gates and 25 T gates.

A classical circuit can be described in a canonical way by a string $y \in \widetilde{\text{cGATES}}^*$, where $\widetilde{\text{cGATES}} = \text{cGATES} \cup \{0, 1, \text{Blank}\}$. (For example, the $0, 1$ can specify the location of the gates in binary, and Blank can indicate that we've finished specifying a single gate.) Similarly, a quantum circuit can be described in a canonical way by a string $y \in \widetilde{\text{qGATES}}^*$, where $\widetilde{\text{qGATES}} = \text{qGATES} \cup \{0, 1, \text{Blank}\}$. In the following definition, the word "description" refers to a fixed cannonical description.

Comment: the quantum part of the following definition is taken from Definition 9.2 of [Kitaev, Shen, Vyalyi]. The randomized and deterministic parts of this definition are written in a form to be similar to the quantum definition. These definitions involve the notion of Turing machines, which have a formal definition, but if that's unfamiliar, just think of it as an algorithm/systematic procedure/computer program. I'm implicitly using the arbitrary but constant number of tapes definition of [Arora-Barak] (Section 1.2), not the single-tape definition of [Sipser] (since these can give rise to quadratically different time complexities, e.g., for PALINDROME, see [Kabanets notes]).

Definition 17 (Deterministic, randomized, and quantum time complexity for decision problems)**.** Let $T\colon \mathbb{N} \to \mathbb{N}$ and $\mathcal{P}$ be a decision problem.

1. We say $\mathcal{P}$ can be solved in deterministic time T ("by a deterministic algorithm in time T") if there exists a Turing machine $\mathcal{A}$ that, for all $N \in \mathbb{N}$, satisfies the following:

   For all inputs $y \in \{0,1\}^N$, $\mathcal{A}$ runs in $\leq T(N)$ steps and outputs $\mathcal{P}(y)$.

2. We say $\mathcal{P}$ can be solved in randomized time T ("by a randomized algorithm in time T") if there exists a (same-notion-as-before) Turing machine $\mathcal{A}$ that, for all $N \in \mathbb{N}$, satisfies the following:

   For all inputs $y \in \{0,1\}^N$, $\mathcal{A}$ runs in $\leq T(N)$ steps and outputs the description of a classical circuit C_y taking a-bit input and 1-bit output such that $\Pr[C_y(r) = \mathcal{P}(y) \mid r \leftarrow \{0,1\}^a] \geq 2/3$.

3. We say $\mathcal{P}$ can be solved in quantum time T ("by a quantum algorithm in time T") if there exists a (same-notion-as-before) Turing machine that, for all $N \in \mathbb{N}$, satisfies the following:

   For all inputs $y \in \{0,1\}^N$, $\mathcal{A}$ runs in $\leq T(N)$ steps and outputs the description of a quantum circuit C_y on a qubits such that $\Pr[C_y(0^a)_1 = \mathcal{P}(y)] \geq 2/3$, where $C_y(0^a)_1$ denotes the first bit of the random variable $C_y(0^a)$.

Definition 18 (P, BQP, BPP)**.** A decision problem $\mathcal{P}$ is said to be in

1. P if there exists $c > 0$ and $T\colon \mathbb{N} \to \mathbb{N}$ with $T(n) = O(n^c)$ such that $\mathcal{P}$ can be solved in deterministic time T.
2. BPP if there exists $c > 0$ and $T\colon \mathbb{N} \to \mathbb{N}$ with $T(n) = O(n^c)$ such that $\mathcal{P}$ can be solved in randomized time T.
3. BQP if there exists $c > 0$ and $T\colon \mathbb{N} \to \mathbb{N}$ with $T(n) = O(n^c)$ such that $\mathcal{P}$ can be solved in quantum time T.

P stands for polynomial time, BPP stands for bounded-error probabilistic polynomial time, BQP stands for bounded-error quantum polynomial time.

Relation between the query model and Turing model. The quantum query complexities of functions $f_n\colon \{0,1\}^n \to \{0,1\}$, where $n \in \mathbb{N}$, can be used to upper bound the quantum time complexity of the decision problem $\mathcal{P}$ if there exists a Turing machine $\mathcal{A}$ that for all $N \in \mathbb{N}$ and for all inputs $y \in \{0,1\}^N$ to $\mathcal{P}$, outputs

1. a natural number $n = n(N) \in \mathbb{N}$ (as a bitstring) and the description of a *classical* circuit for some function $x\colon [n] \to \{0,1\}$ such that $f_n(x) = \mathcal{P}(y)$ in $\leq \tau(N)$ steps. (Note that x is a function but can be identified with a bistring in the natural way, and so can serve as input to f_n.)
2. the description of quantum circuits that (approximately) equal U_i in $\leq \tau_i(N)$ steps for all $i \in \{0, 1, \ldots, Q(n)\}$, where $Q(n)$ is the quantum query complexity of f_n and the U_is are the $Q(n) + 1$ unitaries that specify a quantum query algorithm for f_n of minimum depth.

In this case, $\mathcal{P}$ can be solved in quantum time $T\colon \mathbb{N} \to \mathbb{N}$ such that

$$T(N) = O\Big(\tau(N) \cdot Q(n(N)) + \sum_{i=0}^{d} \tau_i(N)\Big). \tag{63}$$

Proof sketch. The quantum query algorithm can be thought of as an abstract quantum circuit (recall the diagram we drew in class). But using $\mathcal{A}$, we can convert that abstract quantum circuit into a bonafide quantum circuit with gates in qGATES:

1. For each U_i, the conversion takes $\leq \tau_i(N)$ steps.
2. For the quantum oracle of x, O_x, $\mathcal{A}$ outputs the description D_x of a *classical* circuit for $x\colon [n] \to \{0,1\}$ in $\leq \tau(N)$ steps but it is a fact that there's another Turing machine that can convert D_x to a description of a *quantum* circuit for O_x in a number of steps that's linear in the length of D_x. Comment: this is a non-trivial fact, see the next lecture.

   But the length of D_x is $O(\tau(N))$. So converting each O_x to its quantum circuit takes $O(\tau(N))$ steps. Since the quantum query algorithm uses $Q(n(N))$ O_xs, the total number of steps to convert all of them to their quantum circuits is $O(\tau(N) \cdot Q(n(N)))$.

Adding the number of steps in the two cases together gives Eq. (63). □

Remark 8. One way to think of the proof is that a quantum query algorithm serves as a *template* for a quantum circuit. To get the associated time complexity requires us to *fill in* that template.

Comment: this will hopefully make more sense when we discuss kSAT.

Lecture 6

Fact 2 (Quantum oracle instantiation.)**.** Let $\widetilde{\text{cGATES}}$ and $\widetilde{\text{qGATES}}$ be as above. There is a Turing machine that does the following:

1. Input: a description $y \in \widetilde{\text{cGATES}}^*$ of a classical circuit computing the function $x\colon [n] \to \{0, 1, \ldots, m-1\}$.
2. Output: a description $y' \in \widetilde{\text{qGATES}}^*$ of a quantum circuit computing the quantum oracle, O_x of x. (Recall $O_x \in \mathbb{C}^{nm \times nm}$ maps $|i\rangle\,|j\rangle \mapsto |i\rangle\,|j + x_{i+1} \mod m\rangle$.)

Moreover, the Turing machine runs in time $O(\text{length}(y))$.

Proof. See Section 1.4.1 of [Nielsen and Chuang] or [Watrous video starting at 45:37]. □

It follows immediately from this fact that BPP $\subseteq$ BQP. (P $\subseteq$ BPP is obvious.)

kSAT, the Strong Exponential-Time Hypothesis (SETH), and quantum time complexity. Let $k \in \mathbb{N}$, the kSAT decision problem is defined as follows.

Input: A Boolean formula y in Conjunctive Normal Form of width k in l Boolean variables having m clauses. (y can be represented as a string in $\{0,1\}^N$ under a suitable encoding.) This means the input describes an AND of m clauses, where each clause is an OR of $\leq k$ variables or their negations.

Example: Input in the case $k = 3$, $l = 5$, $m = 4$:

$$y = (u_1 \vee \neg u_4 \vee u_3) \wedge (u_5 \vee \neg u_2 \vee \neg u_3) \wedge (\neg u_5 \vee u_4 \vee u_3) \wedge (u_1 \vee \neg u_2), \tag{64}$$

where $u_1, \ldots, u_5 \in \{0,1\}$ are the variables.

Problem: decide if y is satisfiable, i.e., output 1 if there exists $u \in \{0,1\}^l$ such that y evaluated on u is equal to 1

Example: yes since $F = 1$ if we set $u_1, u_4, u_5 = 1$ and u_2, u_3 to be arbitrary, say 1.

Conjecture 1 (Strong Exponential Time Hypothesis)**.** SETH *conjectures that for every $\epsilon > 0$, there exists $k \in \mathbb{N}$ such that no $O(2^{(1-\epsilon)l}\,\text{poly}(m,l))$-time randomized algorithm solves kSAT.*

Obviously, for each $k \in \mathbb{N}$, there is a deterministic algorithm that runs in time $O(2^l\,\text{poly}(m,l))$. Less trivially:

1. For each $k \in \mathbb{N}$, there is a randomized algorithm for that runs in time $O(2^{l(1-1/k)}\,\text{poly}(m,l))$.
2. For $k = 2$, there is a deterministic algorithm that runs in time $O(\text{poly}(m,l))$.

SETH is generally believed to be true. If SETH is indeed true, then there is a definitive (quadratic) quantum advantage for kSAT for large k due to the next proposition.

Proposition 6. *For any $k \in \mathbb{N}$, kSAT can be solved by a quantum algorithm in time $O(\sqrt{2^l}\,\text{poly}(m,l))$.*

Proof sketch. From the definitions, we need to give a classical algorithm that takes the input Boolean formula y and outputs the description of a quantum circuit $C_{N,y}$ that can decide whether y is satisfiable.

Observe that y already describes a classical circuit computing the function

$$x\colon \{0,1\}^l \to \{0,1\}, \tag{65}$$

such that $x(u_1, \ldots, u_l)$ is equal to y evaluated on $u_1, \ldots, u_l$.

Therefore, by the fact on quantum oracle instantiation, there is an algorithm that outputs the description of a quantum circuit computing the quantum oracle, O_x of x in time $O(\text{length}(y)) = O(\text{poly}(m,l))$.

Viewing x as a 2^l-bit string, it suffices for $C_{N,y}$ to compute $\text{OR}_{2^l}(x)$ since this equals whether or not y is satisfied.

But we can use Grover's algorithm for this which uses $O(\sqrt{2^l})$ calls to O_x. Each O_x now has an explicit description as a quantum circuit that takes time $O(\text{poly}(m,l))$ to generate. Moreover, the other non-O_x unitaries used in Grover's query algorithm can be described in time $O(\text{poly}(m,l))$ (see Homework 2). So the total time is $O(\sqrt{2^l}\,\text{poly}(m,l))$.

□

Back to query complexity! Interesting applications of Grover search, or how to combine Grover with classical algorithmic ideas.

The collision problem. Let $n \in \mathbb{N}$ be even. The function in this case is

$$\text{Collision}_n \colon D_0 \dot\cup D_1 \subseteq \{0,1,\dots,n-1\}^n \to \{0,1\}, \tag{66}$$

where

$$D_0 \coloneqq \{x \in \{0,1,\dots,n-1\}^n \mid \forall i,j \in [n], i \neq j \implies x_i \neq x_j\}, \tag{67}$$

$$D_1 \coloneqq \{x \in \{0,1,\dots,n-1\}^n \mid \forall i \in [n], \exists! j \in [n] \text{ s.t. } x_j = x_i \text{ and } j \neq i\}, \tag{68}$$

and $\text{Collision}_n(x) = 1$ if and only if $x \in D_1$.

Example $n = 6$, then

$$301245 \in D_0 \quad \text{and} \quad 313155 \in D_1. \tag{69}$$

Note $x \in \{0,1,\dots,n-1\}^n$ can be naturally identified with a function $\tilde{x} \colon [n] \to \{0,1,\dots,n-1\}$ such that $\tilde{x}(i) = x_i$. Abuse notation and write $\tilde{x}$ also as x. Then $x \in D_1$ means x is a two-to-one function.

Proposition 7. *$R(\text{Collision}_n) = O(\sqrt{n})$ and $Q(\text{Collision}_n) = O(n^{1/3})$.*

Remark 9. In fact, these bounds are tight. We will prove an even stronger randomized lower bound when we discuss Simon's problem. Proving the quantum lower bound is non-trivial (may do later in the course).

Proof.

1. Randomized query algorithm for computing Collision_n.

   Note that the following description can be formally phrased in terms of a distribution over decision trees (how?). Given input $x \in D_0 \dot\cup D_1$:

   > Sample a uniformly random subset $\{i_1,\dots,i_k\} \subset [n]$ of size $k \in \mathbb{N}$. Query $x_{i_1},\dots,x_{i_k}$, if there is a collision, i.e., $i_a \neq i_b$ with $a,b \in [k]$, such that $x_{i_a} = x_{i_b}$, then output 1, else output 0.

   How large of a $k \leq n/2$ do we need to pick? (Note if $k > n/2$, guaranteed to find a collision.) If $x \in D_0$, then will never observe a collision, so always correct in this case. So the probability of error is the probability that no collision is observed if $x \in D_1$.

   $$\begin{aligned}\frac{n(n-2)(n-4)\dots(n-2(k-1))/k!}{\binom{n}{k}} &= 1 \cdot \Big(1 - \frac{1}{n-1}\Big) \cdot \Big(1 - \frac{2}{n-2}\Big) \dots \Big(1 - \frac{k}{n-k+1}\Big) \\ &\leq \exp\Big(-\sum_{i=1}^{k-1} \frac{i}{n-i}\Big) \leq \exp\Big(-\sum_{i=1}^{k-1} \frac{i}{n}\Big) = \exp\Big(-\frac{k(k-1)}{2n}\Big) \leq \exp\Big(-\frac{(k-1)^2}{2n}\Big).\end{aligned}$$

   Therefore the probability of error is $\leq \epsilon$ if

   $$\exp\Big(-\frac{(k-1)^2}{2n}\Big) \leq \epsilon \iff N \geq \sqrt{2n\ln(1/\epsilon)} + 1. \tag{70}$$

   Therefore, $R_\epsilon(\text{Collision}_n) \leq \min(\sqrt{2n\ln(1/\epsilon)} + 1, n/2)$.

2. Quantum query algorithm for computing Collision_n.

   Let $k \in \mathbb{N}$ be such that $k \leq n/2$. On input $x \in D_0 \dot\cup D_1$:

   (a) Classically query $x_1, \dots x_k$.

   (b) If there is a collision, then done. If there is not a collision, *quantumly* search for the k distinct symbols among the remaining $n-k$ symbols.

   This is essentially the same as computing $\text{OR}^{0,k}_{n-k}$ because can use O_x twice, where x is an input to Collision_n, to instantiate a query to an input to $\text{OR}^{0,k}_{n-k}$. Let $F \colon \{0,1,\dots,n-1\} \to \{0,1\}$ be defined by $F(s) = \mathbb{1}[s \in \{x_1,\dots,x_k\}]$. Then for $i \in \{k,\dots,n-1\}$ and $b \in \{0,1\}$,

   $$\begin{aligned}|i\rangle\,|0\rangle\,|b\rangle &\overset{O_x[1,2]}{\mapsto} |i\rangle\,|x_{i+1} \mod n\rangle\,|b\rangle \\ &\overset{U_F[2,3]}{\mapsto} |i\rangle\,|x_{i+1} \mod n\rangle\,|b \oplus \mathbb{1}[x_{i+1} \in \{x_1,\dots,x_k\}]\rangle \\ &\overset{O_x^\dagger[1,2]}{\mapsto} |i\rangle\,|0\rangle\,|b \oplus \mathbb{1}[x_{i+1} \in \{x_1,\dots,x_k\}]\rangle\,.\end{aligned} \tag{71}$$

   Instantiating $O_x^\dagger$ using $O_x \in \mathbb{C}^{n^2 \times n^2}$: let $\text{neg}_n \in \mathbb{C}^{n \times n}$ be defined by $|j\rangle \mapsto |-j \mod n\rangle$.

   $$|i\rangle\,|j\rangle \overset{\text{neg}_n[2]}{\mapsto} |i\rangle\,|-j \mod n\rangle \overset{O_x}{\mapsto} |i\rangle\,|-j + x_{i+1} \mod n\rangle \overset{\text{neg}_n[2]}{\mapsto} |i\rangle\,|j - x_{i+1} \mod n\rangle\,. \tag{72}$$

   The number of queries in the second step is $O(\sqrt{(n-k)/k}) = O(\sqrt{n/k})$.

Overall query complexity is $O(k + \sqrt{n/k})$ which is minimized at $k = n^{1/3}$.

□

Comment: optional material on time complexity, QRAM, and RAM. The relevance of the quantum query algorithm for collision to the real world (i.e., in the Turing model) is less clear than Grover's query algorithm. The reason is due to concerns about the time-efficiency of describing the circuit for F. Often researchers side-step the issue by working in the "Quantum Random Access Memory" (QRAM) model, which implies that the circuit for F can be described in $O(\log(k))$ time. Then, also assuming the circuit for $x\colon [n] \to \{0, 1, \ldots, n-1\}$ can be described in $O(\log(n))$ time, the quantum query algorithm can be described as a bonafide quantum circuit in $O(k \log(n) + \sqrt{n/k}(\log(n) + \log(k))) \approx O(n^{1/3})$ time, again taking $k = n^{1/3}$.

Working in the QRAM model means that we effectively assume:

QRAM assumption. For all $N \in \mathbb{N}$, a quantum circuit for the unitary $\mathrm{QRAM}_N \in \mathbb{C}^N \otimes \mathbb{C}^{2^N}$ defined by

$$\mathrm{QRAM}_N |i\rangle |b\rangle |x\rangle = |i\rangle |b \oplus x_{i+1}\rangle |x\rangle \tag{73}$$

for all $i \in \{0, 1, \ldots, N-1\}$, $b \in \{0, 1\}$, and $x \in \{0, 1\}^N$ can be described in $O(\log(N))$ steps.

What if we don't assume QRAM? Still assume that the time needed to describe a quantum circuit for O_x is $O(\log(n))$. The query algorithm is making $\sqrt{n/k}$ calls to U_F. The circuit for F can be described in time roughly $O(k)$: the circuit simply compares the input s with each of $x_1, \ldots, x_k$. So the quantum circuit for U_F can be described in time $O(k)$. Then the time taken to descibe the Grover search circuit would be $O(k \log(n) \sqrt{n/k})$ so the overall time complexity is $O((k + \sqrt{nk}) \log(n))$ which is minimized at $k = 1$ and gives $\approx O(\sqrt{n})$ – same as classical.

The QRAM model is supposed to be a quantum analogue of the RAM model, and frequently authors cite [Giovannetti, Lloyd, and Maccone'08] to justify the assumption. However, whether the assumption is justified is highly controversial, see, e.g., [Jaques and Rattew'23]. The assumption is provably false in the standard Turing model: QRAM_N provably requires $\Omega(N)$ gates to implement in terms of gates in qGATES (and therefore requires $\Omega(N)$ time to describe in terms of gates in qGATES).

Working in the RAM model means that we effectively assume:

RAM assumption. For all $N \in \mathbb{N}$, a classical circuit for the gate RAM_N that takes input $x \in \{0, 1\}^N$ and $i \in [N]$, and outputs x_i can be described in $O(\log(N))$ steps.

The RAM assumption is also provably false in the standard Turing model since the number of gates in cGATES needed to describe RAM_N is $\Omega(N)$. The RAM_N gate models a "RAM device" (outside the Turing model) that can process an input N-bit string x in $O(\log(N))$ steps to produce a random access memory for x that allows for reading x_i for *arbitrary* ("random") i in a unit time step – the word "random" is really a misnomer since there's no probability distribution involved here. Proponents of QRAM argue that "if RAM is reasonable, then so is QRAM, and RAM is generally regarded as reasonable". However, it is not clear how to engineer a QRAM device out of a RAM device – see [Jaques and Rattew'23] for a discussion.

Lecture 7

Directed st-connectivity in the hypercube. The quantum query algorithm we'll discuss this lecture translates to a quantum algorithm for the travelling salesman problem. Based on [ABIKPV'18].

A hypercube is a graph with 2^n vertices labelled by n-bit strings $\{0,1\}^n$. There is an edge between two vertices $u, v \in \{0,1\}^n$ iff u and v differ in a single bit (alternatively, $|u \oplus v| = 1$, where $|\cdot|$ denotes Hamming weight). This is written $u \sim v$.

The number of edges is $m \coloneqq n2^n/2$. Therefore, $\{0,1\}^m$ bijects with the set of subgraphs of the hypercube graph. We will identity these two sets. The query problem is defined by

$$\text{Hypercube}_n \colon \{0,1\}^m \to \{0,1\}, f(x) = 1 \text{ iff } 0^n \text{ and } 1^n \text{ is connected by a directed path in } x. \tag{74}$$

Here, a directed path means: a sequence $0^n = u_0, u_1, \ldots, u_n = 1^n$ such that $|u_i| = i$ and $u_i \sim u_{i+1}$.

Some obvious ideas that don't work:

1. Grover search over all possible directed paths. $n!$ paths. Each path takes $O(\sqrt{n})$ queries to verify presence. So overall $O(\sqrt{n!n}) = O^*(\sqrt{n!})$, where O^* suppresses factors polynomial in n. But $n! \approx \sqrt{2\pi n}(n/e)^n$ (Stirling's approximation) so $\sqrt{n!} \geq (n/3)^{n/2}$ which is much larger than the trivial bound $n2^n/2$ for large n.

2. Given a vertex v with Hamming weight $n/2$, can decide if it is connected to 0^n using $O^*(2^{n/2})$ queries (there are $O^*(2^{n/2})$ edges that could possibly be part of a path connecting 0^n to v, classically query them all). Similarly can decide if it is connected to 1^n using $O^*(2^{n/2})$ queries. So can decide if there's a directed path from 0^n to 1^n via v using $O^*(2^{n/2})$ queries. Now, Grover search over all v costs $O^*(\sqrt{2^{n/2}}\sqrt{2^{n/2}}) = O^*(2^n)$, better than the first strategy but essentially same as the trivial bound.

Lemma 2. *Let $n \in \mathbb{N}$. Then*

$$\forall k \in \mathbb{N}, 1 \leq k \leq n/2, \binom{n}{\leq k} \coloneqq \sum_{i=0}^{k} \binom{n}{i} \leq 2^{h(k/n)n}, \tag{75}$$

$$\forall l \in \mathbb{N}, 1 \leq l \leq n, \binom{n}{l} \leq 2^{h(l/n)n}, \tag{76}$$

where $h \colon [0,1] \to [0,1]$ denotes the binary entropy function $h(p) \coloneqq -p\log_2(p) - (1-p)\log_2(1-p)$.

Proposition 8. $Q(\text{Hypercube}_n) = O^*(2^{0.9693n})$.

Let $\alpha \in [0, 1/2]$, will set α later.

1. Classically query all edges between Hamming weight $[0, \alpha n]$ and $[(1-\alpha)n, n]$. Cost is $O^*(2^{h(\alpha)n})$.

2. Given a vertex v with Hamming weight $n/2$ and u with Hamming weight αn, can decide if v is connected to u layer using $O^*(2^{(1/2-\alpha)n})$ queries. So can decide if v is connected to 0^n by searching over those $u \leq v$ such that u is connected to 0^n, which costs $O^*(2^{(1/2-\alpha)n}\sqrt{\binom{n/2}{\alpha n}})$ queries. (Note that we know which us are connected to 0^n from the first part.) Similarly, this is also the cost of deciding if v is connected to 1^n. Then Grover search over all v costs $O^*(2^{(1/2-\alpha)n}\sqrt{\binom{n/2}{\alpha n}}\sqrt{\binom{n}{n/2}})$.

The overall query complexity is of O^*:

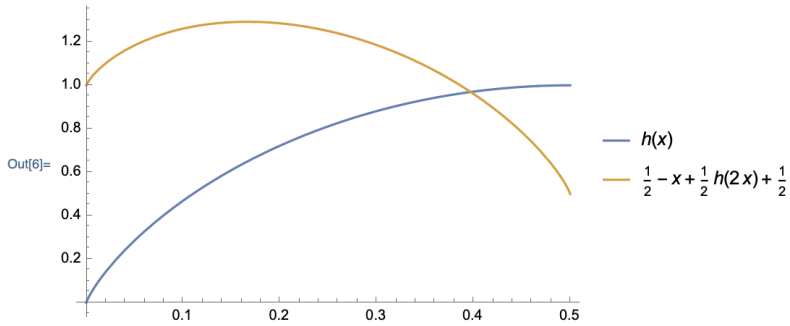
$$\begin{aligned}
&2^{h(\alpha)n} + 2^{(1/2-\alpha)n}\sqrt{\binom{n/2}{\alpha n}}\sqrt{\binom{n}{n/2}} \\
\leq &2^{h(\alpha)n} + 2^{(1/2-\alpha)n}\sqrt{\binom{n/2}{\alpha n}}\sqrt{\binom{n}{n/2}} \\
\leq &2^{h(\alpha)n} + 2^{(1/2-\alpha+h(2\alpha)/2+1/2)n} \\
\leq &2 \cdot 2^{\max(h(\alpha),(1/2-\alpha+h(2\alpha)/2+1/2))n}
\end{aligned} \tag{77}$$

The exponent is minimized by setting $\alpha = 0.397201...$ which gives an upper bound of $O^*(2^{0.9693n})$. (See Fig. 1.)

Remark 10. Can analyze more layers and get $O^*(2^{0.8615n})$. The best known lower bound is $\Omega^*(2^{n/2})$ it is a 5-year-old open question to improve either the lower bound or the upper bound. Another similar open question concerns the directed $2D$ $n \times n$ grid: best known upper bound is $O(n^2)$ (trivial), best known lower bound is $\Omega(n^{1.5})$ – see [ABIKKPSSV'20].

```
In[1]:= h[x_] := -x Log2[x] - (1 - x) Log2[1 - x];

In[6]:= Plot[{h[x], 1/2 - x + h[2 x]/2 + 1/2}, {x, 0, 1/2}, PlotLegends → "Expressions"]
```



Out[6]=

```
In[7]:= NSolve[h[x] == 1/2 - x + h[2 x]/2 + 1/2 && 0 ≤ x && x ≤ 1/2, {x}]

Out[7]= {{x → 0.397201}}

In[8]:= h[0.3972013305447702`]

Out[8]= 0.96929
```

Figure 1: Minimizing the exponent in the quantum query complexity of directed st-connectivity in the hypercube. The minimum is achieved where the two lines cross.

Lecture 8

Simon's problem. Let $n, k \in \mathbb{N}$ be such that $n = 2^k$. So that $\{0,1,\dots,n-1\}^n$ bijects with $\{0,1,\dots,n-1\}^{\{0,1\}^k}$ and be identified under a fixed bijection. In the following, we will switch between these two notations.

$$\mathrm{Simon}_n \colon D \coloneqq D_0 \dot\cup D_1 \subseteq \{0,1,\dots,n-1\}^n \to \{0,1\}, \tag{78}$$

where

$$D_0 \coloneqq \{x \in \{0,1,\dots,n-1\}^{\{0,1\}^k} \mid \forall s,t \in \{0,1\}^k, s \neq t \implies x(s) \neq x(t)\}, \tag{79}$$

$$D_1 \coloneqq \{x \in \{0,1,\dots,n-1\}^{\{0,1\}^k} \mid \exists a \in \{0,1\}^k - \{0^k\}, \forall s,t \in \{0,1\}^k, x(s) = x(t) \iff s \in \{t, t \oplus a\}\}, \tag{80}$$

and $\mathrm{Simon}_n(x) = 0 \iff x \in D_0$.

Proposition 9. $Q(\mathrm{Simon}_n) = O(\log(n))$.

We need some lemmas.

Lemma 3. *Let $x \in \{0,1\}^k$ and $|x\rangle = |x_1\rangle \dots |x_k\rangle$ be a k-qubit state. Let $H^{\otimes k} \coloneqq H \otimes \cdots \otimes H$ (k times). Then*

$$H^{\otimes k} |x\rangle = \frac{1}{\sqrt{2^k}} \sum_{y \in \{0,1\}^k} (-1)^{x \cdot y} |y\rangle . \tag{81}$$

Proof. We have

$$\begin{aligned}
H^{\otimes k} |x\rangle =& H |x_1\rangle \otimes \cdots \otimes H |x_n\rangle \\
=& \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_1} |1\rangle) \otimes \cdots \otimes \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_1} |1\rangle) && \text{Eq. (31) (Phase kickback)} \\
=& \frac{1}{\sqrt{2^k}} \sum_{y_1,\dots,y_k \in \{0,1\}} (-1)^{x_1 y_1 + \cdots + x_k y_k} |y_1\rangle |y_2\rangle \dots |y_k\rangle && \text{think about phase for fixed } y \\
=& \frac{1}{\sqrt{2^k}} \sum_{y \in \{0,1\}^k} (-1)^{x \cdot y} |y\rangle ,
\end{aligned}$$

as required. □

Lemma 4. *Let $K \in \mathbb{N}$. Suppose $z_1, \dots, z_K \leftarrow \mathbb{F}_2^k$. Then the probability that the dimension of the span of the z_is, i.e., the dimension of the subspace*

$$V \coloneqq \{a_1 z_1 + \cdots + a_K z_K \mid a_1, \dots, a_K \in \mathbb{F}_2\} \leq \mathbb{F}_2^k \tag{82}$$

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

Based on [StackExchange post].

Proof. Let $A \in \mathbb{F}_2^{K \times k}$ denote the matrix whose rows are the z_is. The dimension of V is the same as the row-rank (dimension of the span of the rows) of A, which is equal to the column-rank of A by a standard fact in linear algebra. Now, the column-rank of A is k if and only if the kernel of A is $\{0\}$ by the rank-nullity theorem, where the kernel of A is defined by

$$\ker(A) \coloneqq \{x \in \mathbb{F}_2^k \mid Ax = 0\}. \tag{83}$$

Since the z_is are chosen uniformly from $\mathbb{F}_2^k$, A is a uniformly random matrix in $\mathbb{F}_2^{K \times k}$. In the following, the probability is over $A \leftarrow \mathbb{F}_2^{K \times k}$.

$$\begin{aligned}
\Pr[\ker(A) \neq \{0\}] =& \Pr[\exists x \in \mathbb{F}_2^k, x \neq 0, Ax = 0] && \text{definition} \\
\leq& \sum_{x \in \mathbb{F}_2^k, x \neq 0} \Pr[Ax = 0] && \text{union bound} \\
=& (2^k - 1)\frac{1}{2^K} && Ax \text{ is unif. random in } \mathbb{F}_2^K, \text{ e.g., suppose } x_K = 1 \\
\leq& \frac{2^k}{2^K}.
\end{aligned}$$

Therefore $\Pr[\dim(V) = k] = \Pr[\ker(A) = \{0\}] \geq 1 - 2^{k-K}$. □

Lemma 5. *Let $K \in \mathbb{N}$ and $0 \neq a \in \mathbb{F}_2^k$. Let $z_1, \ldots, z_K \in \mathbb{F}_2^k$ (arbitrary) be such that $\forall i \in [K]$, $a \cdot z_i = 0$. Then the dimension of the span of the z_is is at most $k-1$.*

Proof. It suffices to prove that the dimension of the following subspace is $k-1$:

$$U \coloneqq \{z \in \mathbb{F}_2^k \mid a \cdot z = 0\}. \tag{84}$$

Note that U is the kernel of the $1 \times k$ matrix $A \coloneqq (a_1, \ldots, a_k)$. Now, the column-rank of A is 1 since $a \neq 0$. Therefore, by the rank-nullity theorem, $\dim(U) = k-1$. □

With the lemmas in place, we can now prove Proposition 9.

Proof of Proposition 9. Create the state using 1 query to $x \in D$:

$$\frac{1}{\sqrt{2^k}} \sum_{s \in \{0,1\}^k} |s\rangle |x(s)\rangle . \tag{85}$$

Measure the second register in the computational basis. There are two cases depending on whether $x \in D_0$ or $x \in D_1$.

1. $x \in D_0$. Obtain a value $y_0 \in \{0, 1, \ldots, n-1\}$ (with probability $1/2^n$ but the precise value doesn't matter for the later analysis) and the state becomes
$$|s_0\rangle |y_0\rangle , \tag{86}$$
where $x(s_0) = y_0$.

2. $x \in D_1$. Obtain a value $y_0 \in x(\{0,1\}^k)$ (with probability $2/n$ – note $|x(\{0,1\}^k)| = n/2$) and the state becomes
$$\frac{1}{\sqrt{2}}(|s_0\rangle + |s_0 \oplus a\rangle) |y_0\rangle , \tag{87}$$
where $x(s_0) = y_0$.

Now apply $H^{\otimes k}$ to the first register. Then measure the first register in the computational basis. (Will ignore the second register for notational convenience since it just stays $|y_0\rangle$.) Analyze two cases $x \in D_0$ and $x \in D_1$ separately:

1. $x \in D_0$. After applying $H^{\otimes k}$:
$$\frac{1}{\sqrt{2^k}} \sum_{z \in \{0,1\}^k} (-1)^{s_0 \cdot y} |z\rangle . \tag{88}$$
After measurement in the computational basis: obtain $z \in \{0,1\}^k$ uniformly at random.

2. $x \in D_1$. After applying $H^{\otimes k}$:
$$\frac{1}{\sqrt{2^k}} \sum_{y \in \{0,1\}^k} ((-1)^{s_0 \cdot z} + (-1)^{(s_0 \oplus a) \cdot z}) |z\rangle = \frac{1}{\sqrt{2^k}} \sum_{z \in \{0,1\}^k} (-1)^{s_0 \cdot y} (1 + (-1)^{a \cdot z}) |z\rangle . \tag{89}$$
After measurement in the computational basis: obtain $z \in \{0,1\}^k$ such that $a \cdot z = 0 \mod 2$ with probability $2/2^k$. (Note that there are 2^{k-1} zs satisfying $a \cdot z = 0$.)

Repeat the entirety of the above K times and output 0 if and only if

$$d \coloneqq \text{dimension of the span of the } K \text{ } z\text{s obtained (viewed as vectors in } \mathbb{F}_2^k) = k. \tag{90}$$

Analyze two cases $x \in D_0$ and $x \in D_1$ separately:

1. $x \in D_0$. By Lemma 4: with probability at least $1 - 2^{k-K}$, $d = k$. Therefore the probability of the output being correct, i.e., 0, is at least $1 - 2^{k-K}$.

2. $x \in D_1$. By Lemma 5: $d \leq k-1$. Therefore, the output is always correct, i.e., equal to 1.

So if we take $K \geq k+2$, then, for all $x \in D$, the probability of being correct is at least $2/3$.
Since each repeat costs only 1 query. The overall query complexity is $K = k + 2 = O(\log(n))$, as required. □

Remark 11. In the case $x \in D_1$, a slight modification of the algorithm above can also recover a: choose K large enough (how large?) such that in the case $x \in D_1$, we have $d = k-1$ whp; collect the $k-1$ linearly independent vectors $z^{(1)}, \ldots z^{(k-1)} \in \mathbb{F}_2^k$ into the rows of a matrix $A \in \mathbb{F}_2^{(k-1)\times k}$ and compute the kernel of A, which will have size 2. a is the non-zero element. Moreover, note that, since $n = 2^k$, we can identify $\{0, 1, \ldots, n-1\}^n$ with $\{0, 1, \ldots, n-1\}^{\mathbb{F}_2^k}$.

Therefore, we also have an $O(\log(n))$ quantum algorithm for the following query problem:

$$\text{Simon}'_n \colon D' \subseteq \{0, 1, \ldots, n-1\}^{\mathbb{F}_2^k} \to \mathbb{F}_2^k \tag{91}$$

where $x \in D'$ if and only if there exists an $a \in \mathbb{F}_2^k - \{0^k\}$ such that $\forall s, t \in \mathbb{F}_2^k$, $x(s) = x(t) \iff s \in \{t, t+a\}$ (addition as defined in the group $\mathbb{Z}_2^k$, i.e., component-wise addition), and $\text{Simon}'_n(x)$ outputs the a (period) associated with x. (Writing it this way is to allow for direct comparison with the order finding problem at the heart of Shor's algorithm later.)

Lecture 9

Proposition 10. $R(\mathrm{Simon}_n) = \Omega(\sqrt{n})$.

We will need the following lemma.

Lemma 6. *Let $f, T\colon D \coloneqq D_0 \dot\cup D_1 \subseteq \Sigma^n \to \{0,1\}$. Let $f(D_0) = \{0\}$ and $f(D_1) = \{1\}$. Suppose μ_0 is a distribution on D_0 and μ_1 is a distribution on D_1. Let μ denote the distribution on D such that $x \leftarrow \mu$ is defined by $b \leftarrow \{0,1\}$ and $x \leftarrow \mu_b$. Let $P_1 \subseteq D_1$. Suppose that for all $b \in \{0,1\}$,*

$$\Pr[T(x) = b \mid x \leftarrow \mu_0] = \Pr[T(x) = b \mid x \in P_1, x \leftarrow \mu_1]. \tag{92}$$

Then

$$\Pr[T(x) = f(x) \mid x \leftarrow \mu] \le \frac{1}{2} + \frac{1}{2}\Pr[x \notin P_1 \mid x \leftarrow \mu_1]. \tag{93}$$

Proof.

$$\begin{aligned}
&\Pr[T(x) = f(x) \mid x \leftarrow \mu] \\
=&\frac{1}{2}\Pr[T(x) = 0 \mid x \leftarrow \mu_0] + \frac{1}{2}\Pr[T(x) = 1 \mid x \leftarrow \mu_1] && \text{definition of } \mu \\
=&\frac{1}{2}\Pr[T(x) = 0 \mid x \leftarrow \mu_0] + \frac{1}{2}(\Pr[T(x) = 1 \mid x \in P_1, x \leftarrow \mu_1]\Pr[x \in P_1 \mid x \leftarrow \mu_1] \\
&\quad + \frac{1}{2}\Pr[T(x) = 1 \mid x \notin P_1, x \leftarrow \mu_1]\Pr[x \notin P_1 \mid x \leftarrow \mu_1]) && \text{law of total probability} \\
\le&\frac{1}{2}\Pr[T(x) = 0 \mid x \leftarrow \mu_0] + \frac{1}{2}\Pr[T(x) = 1 \mid x \leftarrow \mu_0] + \frac{1}{2}\Pr[x \notin P_1 \mid x \leftarrow \mu_1] && \text{by lemma condition} \\
=&\frac{1}{2} + \frac{1}{2}\Pr[x \notin P_1 \mid x \leftarrow \mu_1],
\end{aligned}$$

as required. □

Comment: apply this lemma to $f = \mathrm{Simon}_n$ and T the (function induced by the) decision tree.

Proof of Proposition 10. (A more rigorous version of [de Wolf notes].) By the averaging argument/easy direction of Yao's principle (i.e., the arguments we used at the beginning of the randomized lower bound proof for OR_n), it suffices to show the following. There exists a distribution μ over D such that if a DDT T satisfies

$$\Pr[T(x) = \mathrm{Simon}_n(x) \mid x \leftarrow \mu] \ge 2/3, \tag{94}$$

then the depth d of T is at least $\Omega(\sqrt{n})$.

We assume without loss of generality (wlog) that

1. T never queries x at the same index twice, i.e., in all paths from root to leaf, the labels of the nodes are distinct.
2. T is balanced, i.e., every root-to-leaf path is length d.

This is wlog since any T without these properties can be simulated by another DDT with these two properties of no greater depth.

To define μ, we first define two distributions μ_0 and μ_1 on D_0 and D_1 respectively by the following sampling procedures. Then we define $x \leftarrow \mu$ by $b \leftarrow \{0,1\}$ and $x \leftarrow \mu_b$.

1. Definition of $x \leftarrow \mu_0$. For each $s \in \{0,1\}^k$, pick a distinct value in $\{0, 1, \ldots, n-1\}$ for $x(s)$ uniformly at random. (So x is a uniformly random permutation of $\{0, 1, \ldots, n-1\}$.)
2. Definition of $x \leftarrow \mu_1$. Pick $a \leftarrow \{0,1\}^k - \{0^k\}$, then for each set $\{s, s \oplus a\}$, where $s \in \{0,1\}^k$, pick a distinct value in $\{0, 1, \ldots, n-1\}$ for $x(s) = x(s \oplus a)$ uniformly at random. Comment: the distribution defined is independent of how the "for each" loop is ordered.

Case $x \leftarrow \mu_0$. The sequence of d responses to the d queries T makes is a uniformly random sequence of d distinct elements in $\{0, 1, \ldots, n-1\}$.

Case $x \leftarrow \mu_1$. Let $t \in \{1, \ldots, d\}$. Let $v_1, \ldots, v_{t-1} \in \{0, 1, \ldots, n-1\}$ be distinct. Let $s_1, \ldots s_t$ denote the sequence of indices that T queries on x given $x(s_1) = v_1, \ldots, x(s_{t-1}) = v_{t-1}$. (Note $s_1, \ldots, s_t$ are uniquely defined, in particular, s_1 is the

label of the root of T.) Say the sequence $x(s_1), \ldots, x(s_t)$ is good if all its values are all distinct. Writing Pr for probability over $x \leftarrow \mu_1$, we have

$$\begin{aligned}
&\Pr[x(s_1), \ldots, x(s_t) \text{ is good} \mid x(s_1) = v_1, \ldots, x(s_{t-1}) = v_{t-1}] \\
=&\Pr[x(s_t) \notin \{x(s_1) = v_1, \ldots, x(s_{t-1}) = v_{t-1}\} \mid x(s_1) = v_1, \ldots, x(s_{t-1}) = v_{t-1}] \\
=&\Pr[a(x) \notin \{s_1 \oplus s_t, \ldots, s_{t-1} \oplus s_t\} \mid x(s_1) = v_1, \ldots, x(s_{t-1}) = v_{t-1}] \qquad a(x) = \text{the } a \text{ corresp. to } x
\end{aligned}$$

Comment: the point of conditioning like this is to explicitly see that s_t is *fixed* and not a function of x; without such conditioning, the queried indices are generally functions of x and we would need to argue why, e.g., we can't have $s_1 = 0^k$ and $s_t = a(x)$, so that $a(x)$ is always in $\{s_1 \oplus s_t\}$. This is why I have chosen to be more rigorous here than [de Wolf notes].

Since the v_is are distinct, conditioning on $x(s_1) = v_1, \ldots, x(s_{t-1}) = v_{t-1}$ implies that $a(x)$ cannot belong to $\{s_i \oplus s_j \mid i, j \in [t-1], i \neq j\} \cup \{0^k\}$ but can take any other value. Since a is initially chosen uniformly from $\{0,1\}^k - \{0^k\}$, $a(x)$ is uniformly distributed over the set of other values, i.e.,

$$\{0,1\}^k - \{0^k\} - \{s_i \oplus s_j \mid i, j \in [t-1], i \neq j\}, \tag{95}$$

which has at least $2^k - 1 - \binom{t-1}{2}$ elements. Therefore, by the union bound,

$$\Pr[a(x) \notin \{s_1 \oplus s_t, \ldots, s_{t-1} \oplus s_t\} \mid x(s_1) = v_1, \ldots, x(s_{k-1}) = v_{k-1}] \geq 1 - \frac{t-1}{2^k - 1 - \binom{t-1}{2}}. \tag{96}$$

Write x is t-good if the responses to the first t queries T makes on x are distinct. Then, since the above analysis holds for all distinct $v_1, \ldots, v_{t-1}$, we have

$$\Pr[x \text{ is } t\text{-good} \mid x \text{ is } (t-1)\text{-good}] \geq 1 - \frac{t-1}{2^k - 1 - \binom{t-1}{2}}, \tag{97}$$

using the fact that $\Pr[A \mid \dot\cup_i B_i] \geq \min_i \Pr[A \mid B_i]$.

Therefore, since the last inequality holds for all $t \in \{1, \ldots, d\}$,

$$\begin{aligned}
\Pr[x \text{ is } d\text{-good}] \geq& \prod_{t=1}^{d} \left(1 - \frac{t-1}{2^k - 1 - \binom{t-1}{2}}\right) \\
\geq& 1 - \sum_{t=1}^{d} \frac{t-1}{2^k - 1 - \binom{t-1}{2}} \qquad \forall a, b \in [0,1], (1-a)(1-b) \geq 1 - a - b.
\end{aligned}$$

Assume wlog that d is such that $1 + \binom{d-1}{2} \leq 2^k/2$ (else we're done) so

$$\Pr[x \text{ is } d\text{-good}] \geq 1 - \frac{2}{2^k}\frac{1}{2}d(d-1) \geq 1 - \frac{d^2}{2^k}. \tag{98}$$

Conditioned on the event that x is d-good, the sequence of d responses to the d queries T makes is a uniformly random sequence of d distinct elements in $\{0, 1, \ldots, n-1\}$, as in the case $x \leftarrow \mu_0$. Comment: this is intuitively clear from the definition of μ_1 but can also verify this by computing a product of conditional probabilities.

Therefore, if we let $P_1 \coloneqq \{x \in D_1 \mid x \text{ is } d\text{-good}\}$, then for all $b \in \{0,1\}$,

$$\Pr[T(x) = b \mid x \leftarrow \mu_0] = \Pr[T(x) = b \mid x \in P_1, x \leftarrow \mu_1]. \tag{99}$$

Finally, we apply Lemma 6 to find that

$$\Pr[T(x) = \text{Simon}_n(x) \mid x \leftarrow \mu] \leq \frac{1}{2} + \frac{1}{2}\frac{d^2}{2^k}. \tag{100}$$

Therefore, we must have $d \geq \sqrt{2^k/3} = \Omega(\sqrt{n})$, as required. □

Remark 12. The D_0 of Simon_n is the same as the D_0 of Collision_n (when n is a power of 2). On the other hand, the D_1 of Simon_n is a subset of D_1 of Collision_n. Therefore, any randomized decision tree that computes Collision_n (with bounded-error 1/3) can also be used to compute Simon_n (with bounded-error 1/3). Therefore $R(\text{Collision}_n) \geq R(\text{Simon}_n)$. Therefore $O(\sqrt{n}) \geq R(\text{Collision}_n) \geq R(\text{Simon}_n) \geq \Omega(\sqrt{n})$, where the first inequality is from a few lectures ago and the last inequality is what we just proved. So $R(\text{Simon}_n), R(\text{Collision}_n) = \Theta(\sqrt{n})$.

Lecture 10-11

Period finding in $\mathbb{Z}$. Also known as the Hidden Subgroup Problem (HSP) over $\mathbb{Z}$.

Comment: the following description of Shor's algorithm follows the exposition in Section 3.3 of [Jozsa notes] which explains [Shor'95]. This is so that the similarity of the analysis of the query problem underlying Shor's algorithm, Period_N below, to the analysis of Simon's problem is self-evident. Indeed, Shor was directly inspired by Simon's algorithm when he came up with his algorithm: see [video]. The exposition here differs from that of, e.g., [Nielsen and Chuang], which explains Kitaev's variant of Shor's algorithm.

We study the following query problem.

$$\text{Period}_N \colon D \subset (\mathbb{Z}_N)^{\mathbb{Z}} \to \{1, \dots, N\}, \tag{101}$$

where $x \in D$ if and only if there exists $r \in \mathbb{N}$ such that

$$x(s) = x(t) \iff s - t \in r\mathbb{Z} \coloneqq \{rz \mid z \in \mathbb{Z}\}. \tag{102}$$

Comment: in the language of the HSP, the ambient group of this problem is $\mathbb{Z}$ and the hidden subgroup is $r\mathbb{Z}$.

Observe that for a given $x \in D$, the "r" associated with it is unique and we denote it by $\text{per}(x)$ (called "period of x"). (Proof: suppose both $r, r' \in \mathbb{N}$ are associated with x, then $x(0) = x(r) = x(r') \implies r - r' \in r\mathbb{Z} \cap r'\mathbb{Z} \implies r|(r - r')$ and $r'|(r - r') \implies r|r', r'|r \implies r = \pm r' \implies r = r'$ as $r, r' \in \mathbb{N}$.)

This does not technically fall into the query problem setup since the input x *can* be queried at an infinite number of points in $\mathbb{Z}$ but the algorithm we describe will only query x at points in $\{0, \dots, 2^n - 1\}$ for $n \coloneqq \lceil 2 \log_2(N) \rceil + 1$.

The quantum algorithm uses the Quantum Fourier Transform.

Definition 19 (Quantum Fourier Transform)**.** For $M \in \mathbb{N}$, the quantum Fourier transform on $\mathbb{C}^M$ is the unitary $\text{QFT}_M \in \mathbb{C}^{M \times M}$ defined by

$$\text{QFT}_M |j\rangle = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} \omega_M^{jk} |k\rangle , \tag{103}$$

for all $j \in \{0, 1 \dots, M-1\}$, where $w_M \coloneqq \exp(2\pi i/M)$. Comment: exercise: check that it is unitary.

Lemma 7 (Coprimality lemma)**.** *Let $r \in \mathbb{N}$ such that $r \geq 100$. The number of elements in $\{0, 1, \dots, r-1\}$ that are coprime to r (i.e., $\forall d \in \mathbb{N}, d|j, d|r \implies d = 1$), denoted $\phi(r)$ (Euler's totient function), satisfies*

$$\phi(r) \geq \frac{r}{5 \ln \ln(r)}, \tag{104}$$

where ln *is the natural logarithm.*

Remark 13. There's a more refined estimate: for $r \in \mathbb{N}, r \geq 3$, we have

$$\phi(r) \geq \frac{r}{e^\gamma \ln \ln(r) + \frac{3}{\ln \ln(r)}}, \tag{105}$$

where $e^\gamma \in [1.7810, 1.7812]$.

Proposition 11. $Q(\text{Period}_N) = O(\log \log(N))$.

Proof. Given input $x \in D$, let $r \coloneqq \text{per}(x)$. Assume wlog $N \geq 100$ as the claimed result is asymptotic. Assume wlog $r \geq 100$, else r will be found by classically querying $x(0), \dots, x(99)$. Observe that we must have $r \leq N$.

Let $n \coloneqq \lceil 2 \log_2(N) \rceil + 1$ so that $2^n > N^2$ and write

$$2^n - 1 = Br + b, \tag{106}$$

in quotient remainder form, so that $B \in \{0, 1, \dots\}$ and $0 \leq b < r$.

Comment: a lot of the technical complications of this proof can be avoided if we assume $r|2^n$, as explained in Lecture 11. (The assumption is invalid in general.)

Create the state

$$\frac{1}{\sqrt{2^n}} \sum_{s=0}^{2^n-1} |s\rangle |x(s)\rangle . \tag{107}$$

Measure the second register. Then the state of the first register becomes

$$\frac{1}{\sqrt{A+1}} \sum_{k=0}^{A} |s_0 + kr\rangle , \tag{108}$$

for some $s_0 \in \{0, 1, \ldots, r-1\}$, where $A = B$ if $s_0 \leq b$ and $A = B - 1$ if $s_0 > b$.

Now we apply QFT_{2^n} to Eq. (108) to obtain

$$\frac{1}{\sqrt{2^n(A+1)}} \sum_{y=0}^{2^n-1} \sum_{k=0}^{A} \omega^{(s_0+kr)y} |y\rangle = \frac{1}{\sqrt{2^n(A+1)}} \sum_{y=0}^{2^n-1} \omega^{s_0 y} \Big(\sum_{k=0}^{A} \omega^{kry} \Big) |y\rangle \,, \tag{109}$$

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

We analyze the sum in brackets:

$$1 + \omega^{ry} + \cdots + \omega^{Ary}. \tag{110}$$

For $j \in \{0, 1, \ldots, r-1\}$, let $y_j \in \{0, 1, \ldots, 2^n - 1\}$ be the closest integer to $j2^n/r$ (if there's a tie, let y_j be the smaller), so that

$$\Big| y_j - j\frac{2^n}{r} \Big| \leq \frac{1}{2}. \tag{111}$$

Note that the y_js defined this way must be distinct since $2^n/r > N^2/N = N \geq 2$. Comment: if $r|2^n$, then y_j would just exactly equal $j2^n/r$.

Then, we have

$$\Big| \frac{ry_j}{2^n} - j \Big| \leq \frac{1}{2}\frac{r}{2^n}. \tag{112}$$

and so we can write

$$\frac{ry_j}{2^n} = j + \eta_j, \tag{113}$$

where $|\eta_j| \leq r/2^{n+1}$. Then

$$S_j \coloneqq \sum_{k=0}^{A} \omega^{kry_j} = \sum_{k=0}^{A} \exp(2\pi i \cdot kry_j/2^n) = \sum_{k=0}^{A} \exp(2\pi i \cdot k\eta_j). \tag{114}$$

Two cases:

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

$$\begin{aligned}
|S_j|^2 =& \Big| \frac{1 - \exp(2\pi i \cdot (A+1)\eta_j)}{1 - \exp(2\pi i \cdot \eta_j)} \Big|^2 && \text{sum geometric series} \\
=& \Big| \frac{\exp(-\pi i \cdot (A+1)\eta_j) - \exp(\pi i \cdot (A+1)\eta_j)}{\exp(-\pi i \cdot \eta_j) - \exp(\pi i \cdot \eta_j)} \Big|^2 \\
=& \frac{\sin^2(\pi(A+1)\eta_j)}{\sin^2(\pi\eta_j)} \\
\geq& \frac{\sin^2(\pi(A+1)\eta_j)}{\pi^2\eta_j^2} && \forall \theta \in \mathbb{R}, \sin^2(\theta) \leq \theta^2
\end{aligned}$$

Now,

$$|\pi(A+1)\eta_j| = \pi(A+1)|\eta_j| \leq \pi(B+1)r/2^{n+1} \leq \pi/2 + \pi r/2^{n+1} \leq \pi/2 + \pi N/(2N^2) \leq 0.505\pi, \tag{115}$$

where the last inequality uses $N \geq 100$. But $\sin^2(\theta) \geq \theta^2/3$ for all $\theta \in [-0.505\pi, 0.505\pi]$ Comment: for safety, I've used a rather loose bound here, so

$$|S_j|^2 \geq \frac{(A+1)^2}{3}. \tag{116}$$

Therefore, if we measure the state in Eq. (109) in the computational basis, the probability of the measurement outcome

being y_j for some $j \in \{0, 1, \ldots, r-1\}$ that is coprime to r is at least:

$$\begin{aligned}
&\frac{r}{5\ln\ln(r)} \cdot \frac{1}{2^n(A+1)} \cdot \frac{(A+1)^2}{3} \\
=&\frac{1}{\ln\ln(r)} \frac{r(A+1)}{15 \cdot 2^n} \\
\geq&\frac{1}{\ln\ln(r)} \frac{rB}{15 \cdot 2^n} \\
=&\frac{1}{\ln\ln(r)} \frac{2^n - 1 - b}{15 \cdot 2^n} \\
\geq&\frac{1}{\ln\ln(r)} \frac{2^n - r}{15 \cdot 2^n} \\
\geq&\frac{1}{\ln\ln(r)} \frac{1}{15}\left(1 - \frac{1}{N}\right) && r/2^n \leq N/2^n < N/N^2 = 1/N \\
\geq&\frac{1}{\ln\ln(r)} 0.05 && N \geq 100
\end{aligned}$$

Comment: it took many lines above to nail down the details but the point is just that at the second line we have $r(A+1) \approx 2^n$.

The overall algorithm is described as follows:

Set $r^* = N + 1$.

Repeat the following $10000 \ln\ln(N)$ times.

Run the procedure described above and let z be the outcome of the measurement. Compute some $r' \in \mathbb{N}$, $r' \leq N$ and $j' \in \{0, 1, \ldots, r'-1\}$ coprime to r' such that

$$\left|\frac{z}{2^n} - \frac{j'}{r'}\right| \leq \frac{1}{2}\frac{1}{2^n}. \tag{117}$$

There are two cases:

(a) If no such pairs r', j' exist, then skip to the next repeat.

(b) If r', j' exist, verify if $x(0) = x(r')$ using 2 queries. If $r' \leq r^*$, set $r^* = r'$.

After all repeats are finished, output r^*.

Comment: (i) $x(0) = x(r')$ does not imply $r' = \mathrm{per}(x)$ as r' could be a multiple of the period; (ii) the computation can be done by trying all possible pairs r', j' – we don't care about the cost of this for query complexity. However, this takes $\Omega(N)$ steps and is *emphatically not* what we would do if we want a $O(\mathrm{poly}(\log(N)))$ *time* quantum algorithm, see example below and next lecture.

We now argue that with very high probability, this procedure will yield the period r. We consider the following two cases at each repeat. Let z denote the measurement outcome.

1. z is indeed y_j for some j coprime to r. Then

$$\left|\frac{rz}{2^n} - j\right| \leq \frac{1}{2}\frac{r}{2^n} \tag{118}$$

and so

$$\left|\frac{z}{2^n} - \frac{j}{r}\right| \leq \frac{1}{2}\frac{1}{2^n}. \tag{119}$$

So certainly r', j' exist. But $r' \in \mathbb{N}$, $r' \leq N$ and $j' \in \{0, 1, \ldots, r'-1\}$ is coprime to r' satisfies

$$\left|\frac{z}{2^n} - \frac{j'}{r'}\right| \leq \frac{1}{2}\frac{1}{2^n}. \tag{120}$$

Then we must have $j/r = j'/r'$, since

$$jr' \neq j'r \implies \left|\frac{j}{r} - \frac{j'}{r'}\right| = \left|\frac{jr' - j'r}{rr'}\right| \geq \frac{1}{N^2}, \tag{121}$$

but

$$\left|\frac{j}{r} - \frac{j'}{r'}\right| \leq \frac{1}{2^n} < \frac{1}{N^2}, \tag{122}$$

which is a contradiction. Comment: we've not used co-primeness before this, only $r, r' \leq N$: the above analysis shows that for any $a \in \mathbb{R}$ there can be at most one fraction with denominator less than or equal to N that approximates a to precision $< 1/N^2$; this fact is what motivated the choice of n to be $\lceil 2\log_2(N) \rceil + 1$ But j is coprime to r and j' is coprime to r' so $j/r = j'/r' \implies j = j'$ and $r = r'$. Therefore, $r' = r$ and r^* is set to r.

2. z is not y_j for some j coprime to r. Assume wlog that r', j' still exist and satisfies $x(0) = x(r')$ (else, nothing happens in this repeat). Then $r' - 0 \in r\mathbb{Z}$ so $r | r'$ so $r' \geq r$ and so r^* is set to an integer that is at least r.

The above analysis means if the first case occurs in at least one of the $10000 \ln\ln(N)$ repeats, then the output of the algorithm will be correct.

At each repeat, the probability that the first case occurs is at least

$$0.05 \frac{1}{\ln\ln(r)} \geq 0.05 \frac{1}{\ln\ln(N)}. \tag{123}$$

Therefore, the probability of the first case not occuring across all repeats (this is the probability of failure) is at most

$$\left(1 - 0.05 \frac{1}{\ln\ln(N)}\right)^{10000 \ln\ln(N)} \leq e^{-500} \leq 1/3, \tag{124}$$

as required.

□

Example 2 (Continued-fractions algorithm)**.** The continued-fractions algorithm can be used to compute r', j' in the above algorithm. We will formally state the properties of this algorithm in the next lecture. For now, let's consider the following example.

Suppose $N = 6$ and $r = 5$. In this case, $n = \lceil 2\log_2(N) \rceil + 1 = 7$. Then

$$y_0 = 0, \quad y_1 = 26, \quad y_2 = 51, \quad y_3 = 77, \quad \text{and} \quad y_4 = 102. \tag{125}$$

Suppose z is equal to *one of the* y_i*s*, then on input $z/2^n$, *exactly one* of the convergents p/q output by the continued-fractions algorithm satisfies:

1. $q \in \mathbb{N}$, $q \leq N$, $p \in \{0, 1, \ldots, q-1\}$ is coprime to p,
2. p/q is $(1/2^{n+1})$-close to $z/2^n$, i.e.,
$$\left| \frac{z}{2^n} - \frac{p}{q} \right| \leq \frac{1}{2}\frac{1}{2^n} = \frac{1}{256}. \tag{126}$$

Example: suppose $z = y_2 = 51$ so $z/2^n = 51/128$.

$$\begin{aligned} \frac{51}{128} &= \frac{1}{\frac{128}{51}} = \frac{1}{2 + \frac{26}{51}} \\ &= \frac{1}{2 + \frac{1}{\frac{51}{26}}} = \frac{1}{2 + \frac{1}{1 + \frac{25}{26}}} \\ &= \frac{1}{2 + \frac{1}{1 + \frac{1}{\frac{26}{25}}}} = \frac{1}{2 + \frac{1}{1 + \frac{1}{1 + \frac{1}{25}}}}. \end{aligned}$$

So the convergents[6], are

$$\frac{0}{1}, \quad \frac{1}{2}, \quad \frac{1}{2+1} = \frac{1}{3}, \quad \frac{1}{2 + \frac{1}{1+1}} = \frac{2}{5}, \quad \frac{51}{128}. \tag{127}$$

Out of these, *only* 2/5 has denominator $\leq N = 6$ and satisfy Eq. (126). Note the denominator of 2/5 recovers $r = 5$ as expected since, in this example, $z = y_2$ and the subscript "2" is coprime to $r = 5$.

We can prove a "poor man's version" of the coprimality lemma (Lemma 7). Reference: Appendix 4, Problem 4.1 of [Nielsen and Chuang]. Since we can prove this lemma from first principles, it means that we can prove $Q(\text{Period}_N) = O(\log N)$ from first principles (which would also lead to a poly$(\log(N))$ time quantum algorithm for factoring $N \in \mathbb{N}$).

Lemma 8. *For $r \in \mathbb{N}$, let $\pi(r)$ denote the number of elements in $\{1, \ldots, r\}$ that are prime. Then*

$$\pi(2r) \geq \frac{r}{\log_2(2r)}. \tag{128}$$

In particular, $\phi(2r) \geq r/\log_2(2r)$.

[6]The first convergent, 0/1, is always a convergent of any input number $a \in [0, 1)$. It corresponds to the integer part of a, which is 0.

Proof. First observe that

$$\binom{2r}{r} = \frac{(2r)}{r}\frac{(2r-1)}{r-1}\cdots\frac{r+1}{1} \geq 2^r. \tag{129}$$

Second, observe that for any $m \in \mathbb{N}$ and prime $p \in \mathbb{N}$, the number of times that p appears in $m!$ is

$$\left\lfloor \frac{m}{p} \right\rfloor + \cdots + \left\lfloor \frac{m}{p^k} \right\rfloor, \tag{130}$$

where k is such that $p^k \leq m < p^{k+1}$. Comment: proof by example, $m = 5, p = 2$.

Therefore, the number of times a prime p appears in $\binom{2r}{r} = \frac{(2r)!}{(r!)^2}$ is given by

$$\left\lfloor \frac{2r}{p} \right\rfloor + \cdots + \left\lfloor \frac{2r}{p^{k_p}} \right\rfloor - 2\left(\left\lfloor \frac{r}{p} \right\rfloor + \cdots + \left\lfloor \frac{r}{p^{k_p}} \right\rfloor\right) \leq k_p, \tag{131}$$

where k_p is such that $p^{k_p} \leq 2r < p^{k_p+1}$ and we used the inequality $\forall x > 0, \lfloor 2x \rfloor - 2\lfloor x \rfloor \leq 1$.

Clearly, only primes p with $1 \leq p \leq 2r$ can appear in the factorization of $\binom{2r}{r}$. Therefore

$$2^r \leq \binom{2r}{r} \leq \prod_{\text{p prime}, 1 \leq p \leq 2r} p^{k_p} \leq (2r)^{\pi(2r)}. \tag{132}$$

Taking base-2 logarithms and rearranging yields the lemma. □

Lecture 12

Time complexity of period finding and Shor's algorithm

Order finding problem. Given $a, N \in \mathbb{N}$ with $a < N$ such that a and N are coprime (only shared factor is 1). The order of a modulo N is the least $r \in \mathbb{N}$ such that $a^r = 1 \mod N$, and is denoted $\mathrm{ord}_N(a)$.

Note that the well-definedness of $\mathrm{ord}_N(a)$ uses the coprimality of a and N: must have $a^i = a^j$ (*) for some $j > i$ since $a, a^2, \ldots a^{N+1}$ can't be all distinct $\mod N$; but a coprime to N implies a^{-1} exists $\mod N$; multiply (*) by $(a^{-1})^i$.

Proposition 12. *There is a quantum algorithm that solves the order finding problem in time $O(\log^3(N)\log\log(N))$.*

Proof sketch. Consider the function $x\colon \mathbb{Z} \to \mathbb{Z}_N$ defined by

$$x(s) = a^s \mod N. \tag{133}$$

Then this x satisfies

$$x(s) = x(t) \iff s - t \in r\mathbb{Z}, \tag{134}$$

where $r \coloneqq \mathrm{ord}_N(a)$. Comment: prove this.

We saw that the period finding quantum query algorithm can use only $O(\log\log(N))$ calls to O_x to extract $\mathrm{ord}_N(a)$ with probability at least 2/3. The quantum query algorithm can be seen as giving us an abstract quantum circuit. To get to time complexity we need to analyze the number of steps it takes to describe that abstract quantum circuit as a bonafide quantum circuit. This analysis can be split into two parts: O_x part and the non-O_x part.

1. O_x part. Recall Fact 2 that the number of steps it takes to describe a quantum circuit for O_x is of the same order as the number of steps it takes to describe a classical circuit for x.

   Recall the quantum algorithm only needs to evaluate x at points $\{0, 1 \ldots, 2^n - 1\}$, where $n = \lceil 2\log_2(N)\rceil + 1$. So we need to be able to compute a to a power of up to $4N^2$ modulo N.

   We can compute a^k modulo N using $O(\log(k)\log^2(N))$ steps by repeated squaring:

   $$a \to a^2 \to (a^2)^2 \to (a^4)^2 \to \cdots \to a^k, \tag{135}$$

   each arrow is a multiplication of two $\log(N)$ bit numbers (since we're working modulo N) so each arrow takes $\log^2(N)$ steps (or gates; this accounting is by naive school-boy multiplication in fact can be $O(\log(N)\log\log(N))$ by Fast-Fourier-Transform (FFT) based methods). The number of arrows is $\log(k)$, so the overall cost is $O(\log(k)\log^2(N))$.

   So, in the worst case of $k = 2^n - 1 = \Theta(N^2)$, the number of steps is $O(\log^3(N))$.

2. Non-O_x part. This itself has three parts. In the following, we write $n \coloneqq \lceil 2\log_2(N)\rceil + 1$.

   (a) Creating the uniform superposition state

   $$\frac{1}{\sqrt{2^n}} \sum_{s=0}^{2^n-1} |s\rangle \tag{136}$$

   can be done by n Hadamard gates.

   (b) Quantum Fourier Transform.

   Fact 3 (Quantum Fourier Transform)**.** There exists a Turing machine Comment: algorithm that on input $n \in \mathbb{N}$ and $\epsilon \in (0, 1)$, outputs the description of a quantum circuit that implements a unitary approximating QFT_{2^n} to error $\leq \epsilon$ in spectral norm distance, in $O(n^2\,\mathrm{poly}(\log(n/\epsilon)))$ steps. Comment: combination of: decomposition of QFT into Hadamard and controlled rotation gates (see [Wikipedia]) + Solovay-Kitaev theorem.

   Suffices to take $\epsilon = O(1/\log n)$ since QFT_{2^n} will be applied $O(\log\log(N)) = O(\log n)$ times. Then the number of steps to describe one QFT is $O(n^2\,\mathrm{poly}(\log n))$.

   (c) Continued-fractions algorithm.

   Fact 4. The continued-fractions algorithm on input $0^n \neq b_1, \ldots, b_n \in \{0,1\}^n$, outputs all the convergents

   $$(p_0 \coloneqq 0, q_0 \coloneqq 1), (p_1, q_1), \ldots, (p_k, q_k) \tag{137}$$

   of $\varphi \coloneqq 0.b_1 \ldots b_n$ in $O(n^3)$ steps, where $p_i, q_i \in \mathbb{N}$, $p_i/q_i < 1$ and p_i is coprime to q_i for all $i > 0$.
   Moreover, suppose there exists $q \in \mathbb{N}$ and $p \in \{0, 1, \ldots, q-1\}$ coprime to q such that

   $$\left|\varphi - \frac{p}{q}\right| \leq \frac{1}{2q^2}, \tag{138}$$

   then there exists $i \in \{0, 1, \ldots, k\}$ such that $p = p_i$ and $q = q_i$.

Using this fact, for each convergent (p_i, q_i) of $z/2^n$, we check whether

$$\left|\frac{z}{2^n} - \frac{p_i}{q_i}\right| \leq \frac{1}{2}\frac{1}{2^n} \quad \text{and} \quad q_i \leq N \tag{139}$$

and output q_i as the solution for r when this is satisfied. Comment: assume check takes unit time.

Correctness when z is equal to y_j for some $j \in \{0, 1, \ldots, r-1\}$ coprime to r:

i. (j, r) must appear as a convergent of $z/2^n$ since j is coprime to r and

$$\left|\frac{z}{2^n} - \frac{j}{r}\right| \leq \frac{1}{2} \cdot \frac{1}{2^n} \leq \frac{1}{2N^2} \leq \frac{1}{2r^2}. \tag{140}$$

Moreover, $r \leq N$.

ii. if a convergent (p_l, q_l) satisfies

$$\left|\frac{z}{2^n} - \frac{p_l}{q_l}\right| \leq \frac{1}{2}\frac{1}{2^n}, \tag{141}$$

then noting that $q_l \in \mathbb{N}$, $q_l \leq N$ and $p_l \in \{0, 1, \ldots, q_l - 1\}$ is coprime to q_l, by the argument in the proof of Proposition 11, we must have $p_l = j$ and $q_l = r$.

Therefore, the time complexity (number of steps needed to describe the circuit) of *each* repeat is of order

$$n + \log^3(N) + n^2 \operatorname{poly}(\log n) + n^3, \tag{142}$$

where the first term is from creating the uniform superposition, the second term is from the oracle call, the third term is from the QFT, and the last term is from the continued-fractions algorithm.

Therefore, taking into account the $O(\log\log(N))$ repeats, the overall cost is

$$O(\log^3(N) \log\log(N)), \tag{143}$$

since $n = \lceil 2\log_2(N) \rceil + 1$.

□

Quantum factoring algorithm (Shor)
Input: $N \in \mathbb{N}$, N is composite (i.e., not prime).

1. Check N for parity. If N is even, then output "2", else go to Step 2.

   Cost: $O(1)$ by checking if the last binary digit is 0.

2. Check if N is the kth power of some natural number for $k = 2, \ldots, \lceil \log_2(N) \rceil$. If $N = y^k$, where $y \in \mathbb{N}$, then output "y", else go to Step 3.

   Cost of taking square root is $O(\log(N) \cdot \log^2(N)) = O(\log^3(N))$ by a binary-search-type method: $O(\log(N))$ iterations, each iteration multiplies two $\log(N)$-length numbers at cost $O(\log^2(N))$; by repeated square-rooting, cost of taking kth root is $O(\log(k)\log^3(N))$. So doing all of this costs $O(\log^3(N)\log\log(N))$.

3. Choose $a \leftarrow \{1, \ldots, N-1\}$. Compute $b = \gcd(a, N)$ by the Euclidean algorithm.

   Cost: $O(\log^3(N))$. One $\log(N)$ from number of iterations, each iteration takes $O(\log^2(N))$ by school-boy long division.

   If $b > 1$, then output "b", else proceed to Step 4.

4. Compute $r := \operatorname{ord}_N(a)$ using the quantum period finding algorithm with success probability $\geq 2/3$. (Recall this means r is the least natural number such that $a^r = 1 \mod N$.)

   Cost: $O(\log^3(N)\log\log(N))$ *quantumly.*[7]

   If r is odd, output "don't know", else go to Step 5.

5. Compute $d = \gcd(a^{r/2} - 1 \mod N, N)$. If $d > 1$, then output "d"; otherwise output "don't know".

   Cost: $O(\log^3(N))$. Note the $\mod N$ on the $a^{r/2} - 1$ is important and has to be done "online" while $a^{r/2}$ is being computed: $a^{r/2}$ could be of length $\Omega(N)$ so even writing it down takes $\Omega(N)$ time.

[7] The $\log\log(N)$ factor came from the number of repeats in the period-finding algorithm. This factor can be reduced to $O(1)$ by modifying the algorithm slightly. Then the cost here becomes $O(\log^3(N))$. In fact, the cost of the *quantum component* of period finding (i.e., creating the uniform superposition, oracle call, and QFT_{2^n}) can be further reduced to $O(\log^2(N)\log\log(N))$ if we used FFT-multiplication to instantiate the oracle instead of school-boy multiplication.

Runtime. $\tilde{O}(\log^3(N))$ by the red text.

Correctness. (Of a repeated version of the algorithm, see proof.)

Proof. Since the input N is composite, there are only the following two cases:

1. N is composite with $m = 1$ distinct prime factor. That is $N = p^k$ for some prime p and integer $k \geq 2$. In this case, step 2 will output the prime.
2. N is composite with $m \geq 2$ distinct prime factors. In this case, we claim that with probability $\geq 1/3$, the algorithm doesn't output "don't know", in which case it must output a non-trivial factor of N.

Therefore, we can repeat the algorithm 10000 times and if it returns "don't know" across all repeats, we declare that N is prime. If at any repeat it returns an integer, then that integer must be a non-trivial factor of N by definition and we output it. The probability of being incorrect is $\leq (2/3)^{10000}$.

Proof of the claim in the second case. If the algorithm outputs an integer in steps 1, 2, 3, then it is definitely a non-trivial factor of N by definition. Therefore, we analyze from step 4. At the start of step 4:

1. N is odd.
2. N has $m \geq 2$ distinct prime factors.
3. a is a natural number chosen uniformly at random from the set

$$\{\alpha \in \{1, \ldots, N-1\} \mid \alpha \text{ is coprime to } N\}. \tag{144}$$

We use the following non-trivial fact from number theory:

Fact 5 (See, e.g., Theorem 5.3 of [Nielsen and Chuang]– the "m" in the theorem should be $m-1$ or Section 13.3 of [Kitaev, Shen, Vyalyi].)**.** Suppose $N \in \mathbb{N}$ is odd and has $m \geq 2$ distinct prime factors. Then

$$\Pr[r \coloneqq \mathrm{ord}_N(a) \text{ is even and } a^{r/2} \neq -1 \mod N] \geq 1 - \frac{1}{2^{m-1}} \geq \frac{1}{2}, \tag{145}$$

where the probability is over the uniformly random choice of a from the set in Eq. (144).

Conditioned on the event in Eq. (145),

$$(a^{r/2} - 1)(a^{r/2} + 1) = 0 \mod N, \tag{146}$$

so

$$N | (a^{r/2} - 1)(a^{r/2} + 1), \tag{147}$$

yet

$$N \nmid (a^{r/2} - 1) \quad \text{and} \quad N \nmid (a^{r/2} + 1), \tag{148}$$

where the first not-divide is because r is the period. So N must share a non-trivial factor (i.e., not 1 or N) with $a^{r/2} - 1$, which will be extracted at step 5.

The computation of the r is correct with probability $\geq 2/3$, so the probability of the output being correct is at least $1/2 \times 2/3 = 1/3$. □

Remark 14. What if we don't assume the input N to be composite so that we want the algorithm to also be able to tell if N is prime? This can be done by adding an *efficient classical* preprocessing step that tests whether N is prime: see, e.g.,

1. Miller-Rabin primality test, which is randomized classical.
2. AKS primality test, which is deterministic (!) classical.

Lecture 13

Definition 20 (Group)**.** A group $G = (S, \alpha)$ is defined by a set S and a function $\alpha\colon S \times S \to S$ with the following conditions. (For $g, h \in S$, we write $g \cdot h$ or simply gh as shorthand for $\alpha(g, h)$.)

1. Identity. There exists an $e \in S$ such that $\forall g \in S$, $ge = eg = g$ — this definition implies the e is unique.
2. Inverse. For all $g \in S$, there exists $h \in S$ such that $gh = hg = e$.
3. Associativity. For all $g, h, k \in S$, $(gh)k = g(hk)$. (That is, $\alpha(\alpha(g, h), k) = \alpha(g, \alpha(h, k))$.)

If we additionally have: for all $g, h \in S$, $gh = hg$, then the group is called abelian.

Observation: for a given $g \in S$, there is a unique element h such that $gh = hg = e$ and we can denote it without ambiguity by g^{-1}. (Proof: suppose $h_1, h_2 \in S$ both satisfies this, then

$$g = gh_1 = gh_2 \implies h_1(gh_1) = h_1(gh_2) \implies (h_1g)h_1 = (h_1g)h_2 \implies eh_1 = eh_2 \implies h_1 = h_2, \tag{149}$$

where the second implication uses associativity.)

The size of G or the "order of G" is the size of the set defining G and is denoted $|G|$. G is a finite group if its size is finite.

Example 3.

1. The set $S = \mathbb{F}_2^n$ and $\alpha =$ component-wise addition mod 2. This is abelian.
2. The set $S = \mathbb{Z}$ and $\alpha =$ addition. This is abelian.
3. The set S of invertible $n \times n$ complex matrices and $\alpha =$ multiplication. This is not abelian – matrix multiplication does not commute.
4. The set S of "symmetries of the regular n-gon" and $\alpha =$ "composition of symmetries". S has size $2n$ — n reflections and n rotations. Formally, S can be expressed as

$$S = \{\sigma^s \tau^a \mid x \in \mathbb{Z}_n, a \in \mathbb{Z}_2\}. \tag{150}$$

We will write $(s, a) \in \mathbb{Z}_n \times \mathbb{Z}_2$ for $\sigma^s \tau^a$ for notational convenience. Then the α is defined by

$$\alpha((s, a), (t, b)) = (s + (-1)^a t, a + b). \tag{151}$$

This implies that $(s, a)^{-1} = (-(-1)^a s, a)$.

The group is often denoted D_{2n}.

Definition 21. Let $G = (S, \alpha)$ be a group. We say $T \subseteq S$ forms a subgroup of G if:

1. T contains the identity element of G.
2. T is closed under α, i.e., $g, h \in T \implies gh \in T$.
3. T contains inverses, i.e., $g \in T \implies g^{-1} \in T$.

This definition means that $(T, \alpha|_T)$ is a group, where $\alpha|_T$ is the natural restriction of α to T. We say $(T, \alpha|_T)$ is a subgroup of G. Often the function α (aka group operation) is implicit in which case it's common to abuse language and identify the set S with the group G and the set T with the subgroup $(T, \alpha|_T)$.

Example 4.

1. For any G, $\{e\}$ and G itself are subgroups of G.
2. For $r \in \mathbb{Z}$, $r\mathbb{Z} \coloneqq \{rz \mid z \in \mathbb{Z}\}$ is a subgroup of $\mathbb{Z}$.
3. For $y \in \mathbb{Z}_n$,

$$T \coloneqq \{(0, 0), (s, 1)\} = \langle (y, 1) \rangle \tag{152}$$

is a subgroup of D_{2n} (Follows from $(s, 1)(s, 1) = (s - s, 1 + 1) = (0, 0)$; order-2 subgroup of reflections.)

Definition 22 (Hidden subgroup problem (HSP))**.** Let G be a finite group and let $\mathcal{H}$ denote a set of subgroups of G. Let Σ be an alphabet with $|\Sigma| \geq |G|$. The hidden subgroup problem $\mathrm{HSP}(G, \mathcal{H})$ is the problem of computing the function

$$f\colon D \subseteq \Sigma^G \to \mathcal{H}, \tag{153}$$

where $x \in D$ if and only if there exists $H \in \mathcal{H}$ such that

$$x(g) = x(h) \iff gH = hH, \tag{154}$$

where $gH \coloneqq \{gh \mid h \in H\}$, and $f(x)$ is defined to be the H associated with x. Comment: homework: check that this is well-defined.

Remark 15. The definition could be generalized to infinite groups G in the case where

$$\max_{H \in \mathcal{H}} \left|G : H\right| < \infty, \tag{155}$$

where $\left|G : H\right|$ is the index of H in G – see [Wikipedia]. Then require $|\Sigma| \geq \max_{H\in\mathcal{H}} |G : H|$. This would capture Shor's case: $\mathcal{H} \coloneqq \{r\mathbb{Z} \mid r \in \{1, 2, \ldots, N\}\}$.

Example 5. So far we have studied the HSP for two abelian groups.

1. Simon's problem. $G = \mathbb{F}_2^n$ (with component-wise addition), $\mathcal{H} \coloneqq \{\{0, a\} \mid 0 \neq a \in \mathbb{F}_2^n\}$, and $\Sigma = \{0, 1, \ldots, n-1\}$. Note that
$$gH = hH \iff \{g \cdot 0, g \cdot a\} = \{h \cdot 0, h \cdot a\} \iff \{g, g + a\} = \{h, h + a\} \iff g \in \{h, h + a\}. \tag{156}$$
2. Period finding problem. $G = \mathbb{Z}$, $\mathcal{H} = \{r\mathbb{Z} \mid r \in \{1, 2, \ldots, N\}\}$, and $\Sigma = \mathbb{Z}_N$.

Interlude on mixed quantum states. We know that quantum states are complex vectors $|\psi\rangle \in \mathbb{C}^d$. We also know that a Γ-outcome measurement is a process that when performed on $|\psi\rangle$ results in a measurement outcome $i \in \Gamma$ and a post-measurement state $|\psi_i\rangle$ with probability p_i. In the analysis of Simon's problem and period finding, we either:

1. did not care about what the distribution on $(i, |\psi_i\rangle)$ was (the first measurement: any i works as the next step is a QFT that washes it out) or
2. did care about the distribution on i but not what the post-measurement state was (the second measurement: uniformly random i used to lower bound probability of linear independence/coprimality, respectively).

In either case, we did not care about the distribution on the post-measurement state.

However, in the algorithm for the HSP, we will post-process the post-measurement state $|\psi_i\rangle$ in a way that requires us to care about the distribution on the post-measurement state when performing its analysis. Now, there are two ways of doing the analysis.

1. Naive but hard way. For each $i \in \Gamma$, we analyze how the post-processing behaves on $|\psi_i\rangle$ and then average the behaviour for each i over the distribution $\{p_i\}$. Hard because the analysis loops over Γ; especially bad when Γ is massive.
2. Sophisticated but easier way. We define a single object that describes the post-measurement state in a way that takes the probability distribution into account:
$$\rho \coloneqq \sum_{i\in\Gamma} p_i |\psi_i\rangle\langle\psi_i| \tag{157}$$
and then analyze how the post-processing behaves on ρ. Easier because there's no more loop over Γ.

Quantum information theory has developed tools for analyzing how processing behaves on ρ in a way that is completely consistent with the naive analysis. One of them is the next lemma. First some definitions.

Lecture 14

Definition 23 (Mixed quantum state)**.** Let $d \in \mathbb{N}$. A d-dimensional mixed (quantum) state is a matrix $\rho \in \mathbb{C}^{d\times d}$ that is positive semi-definite (PSD) and has trace 1.

Quantum states $|\psi\rangle \in \mathbb{C}^d$ are often called "pure quantum states" and can be viewed as mixed quantum states of rank 1 via $|\psi\rangle \leftrightarrow |\psi\rangle\langle\psi|$.

For example, the matrix in Eq. (157) is a mixed quantum state – follows from the definition of a measurement. In general, a mixed quantum state can be used to describe any probability distribution over quantum states $(p_i, |\psi_i\rangle) \mapsto \sum_i p_i |\psi_i\rangle\langle\psi_i|$. (I.e., not just restricted to the distribution arising from a measurement.) Comment: the mapping is not invertible but it does not destroy information, since the information content of $(p_i, |\psi_i\rangle)$ *is* $\sum_i p_i |\psi_i\rangle\langle\psi_i|$. Put another way: you cannot distinguish between $(p_i, |\psi_i\rangle)$ and $(p'_i, |\psi'_i\rangle)$ using any procedure if their corresponding mixed states are the same. E.g., $(\frac{1}{2}, |0\rangle), (\frac{1}{2}, |1\rangle)$ and $(\frac{1}{2}, |+\rangle), (\frac{1}{2}, |-\rangle)$.

Definition 24 (Measurement on mixed quantum states)**.** Let Γ be an alphabet and $d \in \mathbb{N}$. Let $\mathcal{M} \coloneqq \{\Pi_i \mid i \in \Gamma\}$ be a measurement. Given a mixed quantum state $\rho \in \mathbb{C}^d$, to measure ρ using $\mathcal{M}$ refers to a process that

1. Outputs $i \in \Gamma$ with probability $\mathrm{tr}[\Pi_i \rho]$. This i is referred to as the measurement outcome.
2. The mixed quantum state changes to
$$\frac{\Pi_i \rho \Pi_i}{\mathrm{tr}[\Pi_i \rho]}. \tag{158}$$

The effect of a quantum measurement on a mixed quantum state is consistent with the effect of a quantum measurement on (pure) quantum states.

Definition 25 (Schatten p-norms)**.** Let $p \in [1, \infty)$ and $d \in \mathbb{N}$. The Schatten p-norm of $A \in \mathbb{C}^{d\times d}$ is defined to be
$$\|A\|_p \coloneqq [\mathrm{Tr}[(A^\dagger A)^{p/2}]]^{1/p} \tag{159}$$
The Schatten ∞-norm of A is defined to be the the spectral norm of A, which coincides with $\lim_{p\to\infty} \|A\|_p$.

Definition 26 (Fidelity between mixed quantum states)**.** For $\rho, \sigma \in \mathbb{C}^{d\times d}$ that are mixed quantum states, the fidelity between ρ and σ is defined to be
$$F(\rho, \sigma) \coloneqq \|\sqrt{\rho}\sqrt{\sigma}\|_1, \tag{160}$$
Comment: have $F(\rho, \sigma) = \mathrm{tr}[|\sqrt{\rho}\sqrt{\sigma}|)$, where $|X| \coloneqq \sqrt{X^\dagger X}$. (This definition is symmetric.) Note: for C Hermitian of the form $\sum_i \lambda_i |v_i\rangle\langle v_i|$ with $\lambda_i > 0$, $\sqrt{C}$ is defined to be $\sum_i \sqrt{\lambda_i} |v_i\rangle\langle v_i|$. Also $F(A, B) = F(B, A)$ follows from $\mathrm{tr}[|X|] = \mathrm{tr}[|X^\dagger|]$ for any $X \in \mathbb{C}^d$ – think SVD.

Lemma 9 (Pretty Good Measurement)**.** *Let $N, d \in \mathbb{N}$. Let $\rho_1, \ldots, \rho_N \in \mathbb{C}^{d\times d}$ be mixed quantum states. Then there exists an $[N]$-outcome measurement $\mathcal{M} \coloneqq \{\Pi_1, \ldots, \Pi_N\}$ on $\mathbb{C}^N \otimes \mathbb{C}^d$ such that*
$$\forall i \in [N]\colon\ \mathrm{tr}[\Pi_i |0\rangle\langle 0| \otimes \rho_i] \geq 1 - N\sqrt{\max_{a\neq b} F(\rho_a, \rho_b)}, \tag{161}$$
where $|0\rangle$ is the first standard basis vector of $\mathbb{C}^d$.

Proof. Omitted. See [Harrow and Winter'06] (which applies von Neumann's minimax theorem to [Barnum and Knill'00]). □

Lemma 10 (Hölder's inequality for Schatten p-norms)**.** *Let $p \in [1, \infty]$ and $p^* \in [1, \infty]$ satisfy $1/p + 1/p^* = 1$. Let $d \in \mathbb{N}$. Then, for $A, B \in \mathbb{C}^{d\times d}$, we have*
$$\|AB\|_1 \leq \|A\|_p \|B\|_{p^*}. \tag{162}$$

Proof. Omitted, non-trivial. Variant more often seen is $|\langle A, B\rangle| \leq \|A\|_p \|B\|_{p^*}$, where $\langle A, B\rangle \coloneqq A^\dagger B$. Here's a reduction to that variant. Let $r \coloneqq \mathrm{rk}(A)$. Write
$$AB = \sum_{i=1}^r \sigma_i |u_i\rangle\langle v_i|, \tag{163}$$
so $\|AB\|_1 = \sum_{i=1}^r \sigma_i$.

Let $U \in \mathbb{C}^{d\times d}$ unitary be such that $U^\dagger |v_i\rangle = |u_i\rangle$. Then
$$AB = \sum_{i=1}^r \sigma_i U^\dagger |v_i\rangle\langle v_i|. \tag{164}$$

So

$$\operatorname{tr}[UAB] = \sum_{i=1}^{r} \sigma_i. \tag{165}$$

But

$$\operatorname{tr}[UAB] = \langle A^\dagger U^\dagger, B\rangle \leq \|A^\dagger U^\dagger\|_p \|B\|_{p^*} = \|A\|_p \|B\|_{p^*}, \tag{166}$$

where the last equality uses the unitary invariance of the Schatten p-norm and its invariance under conjugate transpose. □

Remark 16. Further ingredients: von Neumann's trace inequality: $|\operatorname{tr}[AB]| \leq \sum_{i=1}^{d} \sigma_i(A)\sigma_i(B)$ (see [Stackexchange post]) and the normal Hölder's inequality.

Lemma 11. *Let ρ, σ be mixed quantum states. Then $F(\rho,\sigma) \leq \sqrt{\operatorname{tr}[\Pi_\rho \cdot \sigma]}$, where Π_ρ is the orthogonal projector onto the support of ρ.*

Proof. Follows from Hölder's inequality with $p = 2$. □

Lemma 12. *Let G be a finite group, $H, H' \leq G$ (subgroups of G), and $g, g' \in G$. Then*

$$|gH \cap g'H'| = \begin{cases} |H \cap H'| & \text{if } g^{-1}g' \in HH', \\ 0 & \text{otherwise.} \end{cases} \tag{167}$$

Proof. See Homework 3. □

Lemma 13. *Let N denote the number of subgroups of G. Then $N \leq (|G|+1)^{\log_2 |G|}$.*

Proof. Each subgroup K can be specified by a set of independent generators $g_1, \ldots, g_k$. Moreover $k \leq \log_2 |K| \leq \log_2 |G|$, where the first inequality follows since each new independent generator increases the size of the subgroup by at least 2.

Therefore, the number of subgroups is at most

$$\binom{|G|}{\leq \log_2 |G|} \leq (|G|+1)^{\log_2 |G|}. \tag{168}$$

□

All of the above serves as preliminaries for the following result.

Proposition 13. *Let G be a finite group. Let $\mathcal{H}$ be the set of all subgroups of G. Then $Q(\mathrm{HSP}(G,\mathcal{H})) = O(\log^2(|G|))$.*

Proof. We covered this following Chapter 10.3 of [AMC]. □

Lecture 15

For $n \in \mathbb{N}$, let Dihedral_{2n} denote the HSP with $G = D_{2n}$, $\mathcal{H} = \{\langle (s,1)\rangle \mid s \in \mathbb{Z}_n\}$ and $\Sigma = \{0,1,\dots,2n-1\}$.

Definition 27 (Quantum non-oracular time complexity)**.** Let $f\colon D \subseteq \Sigma^n \to \Gamma$. The quantum non-oracular time complexity of a quantum query algorithm $\mathcal{A}$ computing f with bounded error $1/3$, denoted $T(\mathcal{A})$, is the depth of $\mathcal{A}$ plus the total number of steps needed for a Turing machine to describe the quantum circuits implementing the non-oracular unitaries in $\mathcal{A}$. The quantum non-oracular time complexity of f, denoted $T(f)$, is defined to be

$$T(f) = \min_{\mathcal{A}\text{ computes } f \text{ with bounded error } 1/3} T(\mathcal{A}). \tag{169}$$

By definition $Q(f) \le T(f)$.

Proposition 14 (Kuperberg's algorithm)**.** $T(\text{Dihedral}_{2n}) = 2^{O(\sqrt{n})}$.

Proof. Comment: first steps are the same as the generic steps in the HSP. Let $x \in D \subseteq \Sigma^G$ be the input. Let $\langle s,1\rangle$ be the hidden subgroup associated with x, where $s \in \mathbb{Z}_n$.

Create the state

$$\frac{1}{\sqrt{2n}} \sum_{t\in\mathbb{Z}_n, a\in\mathbb{Z}_2} |t,a\rangle\,|x(t,a)\rangle\,, \tag{170}$$

where $x(t,a)$ means $x((t,a))$.

Measure the second register in the computational basis, which yields an element $x_0 \in \{0,1,\dots,2n-1\}$. The state of the first register becomes

$$\frac{1}{\sqrt{2}}(|t_0', a_0\rangle + |t_0' + (-1)^{a_0} s, a_0 + 1\rangle), \tag{171}$$

where $x(t_0', a_0) = x_0$. Comment: can verify by:

$$\begin{aligned} x(t',a') = x(t_0',a_0) &\iff (t',a')\langle (s,1)\rangle = (t_0',a_0)\langle (s,1)\rangle \\ &\iff \{(t',a'), (t' + (-1)^{a'} s, a' + 1)\} = \{(t_0',a_0), (t_0' + (-1)^{a_0} s, a_0 + 1)\} \\ &\iff (t',a') \in \{(t_0',a_0), (t_0' + (-1)^{a_0} s, a_0 + 1)\}. \end{aligned}$$

Eq. (171) can be written

$$\frac{1}{\sqrt{2}}(|t_0, 0\rangle + |t_0 + s, 1\rangle), \tag{172}$$

for some $t_0 \in \mathbb{Z}_n$. Comment: two cases: $a_0 = 0$ and $a_0 = 1$.

We now do QFT_n on the first register Comment: i.e, do $\text{QFT}_n \otimes \mathbb{1}_2$. Write $\omega \coloneqq \exp(2\pi i/n)$. This gives

$$\begin{aligned} &\frac{1}{\sqrt{2}}\Big(\frac{1}{\sqrt{n}}\sum_{k=0}^{n-1} \omega^{t_0 k}\,|k,0\rangle + \frac{1}{\sqrt{n}}\sum_{k=0}^{n-1} \omega^{(t_0+s)\cdot k}\,|k,1\rangle\Big) \\ =&\frac{1}{\sqrt{n}}\sum_{k=0}^{n-1} \omega^{t_0 k}\,|k\rangle\,\frac{1}{\sqrt{2}}(|0\rangle + \omega^{ks}\,|1\rangle) \end{aligned}$$

Measure the first register in the computational basis. Obtain uniformly random $k \in \{0,\dots,n-1\}$ and the state of the second register becomes:

$$|\psi_k\rangle \coloneqq \frac{1}{\sqrt{2}}(|0\rangle + \omega^{ks}\,|1\rangle). \tag{173}$$

Then started covering Kuperberg's algorithm following Chapter 13 of [AMC]. □

Lecture 16

Finished covering Kuperberg's algorithm following Chapter 13 of [AMC].

Another approach to DHSP. Comment: elaboration on Chapter 13.6 of [AMC]

HSP on D_{2n} with hidden subgroup $\langle s, 1\rangle$, where $s \in \mathbb{Z}_n$.

We know how to generate the following

$$k \leftarrow \mathbb{Z}_n \quad \text{and} \quad |\psi_k\rangle := \frac{1}{\sqrt{2}}(|0\rangle + \omega^{ks}|1\rangle), \tag{174}$$

where $\omega = \exp(2\pi i/n)$.

Measure in the Hadamard basis

$$\Pr[+ \mid k] = |\langle +|\psi_k\rangle|^2 = \cos^2\Big(\frac{\pi k s}{n}\Big). \tag{175}$$

Comment: two random correlated variables here: $K \in \mathbb{Z}_n$ and $B \in \{+,-\}$; formally, should write $\Pr[K = k \mid B = +]$.

So by Bayes' rule

$$\Pr[k \mid +] = \frac{\Pr[+ \mid k]\cdot\Pr[k]}{\Pr[+]} = \frac{\cos^2\big(\frac{\pi k s}{n}\big)\cdot\frac{1}{n}}{\frac{1}{n}\sum_{j=0}^{n-1}\cos^2\big(\frac{\pi j s}{n}\big)} = \begin{cases} \frac{1}{n} & \text{if } s = 0, \\ \frac{2}{n}\cos^2\big(\frac{\pi k s}{n}\big) & \text{if } s \neq 0. \end{cases} \tag{176}$$

Trick for analyzing the sum in the denominator:

$$\frac{1}{n}\sum_{j=0}^{n-1}\cos^2\Big(\frac{\pi j s}{n}\Big) = \frac{1}{n}\sum_{j=0}^{n-1}\frac{1+\cos(\frac{2\pi j s}{n})}{2} = \frac{1}{2} + \frac{1}{2n}\mathrm{Re}\left[\sum_{j=0}^{n-1}\exp(i\cdot 2\pi s/n)^j\right] = \frac{1}{2}, \tag{177}$$

where the last equality follows from summing a geometric series.

It is known that distinguishing between the distributions $\{D_s \mid s \in \mathbb{Z}_n\}$ takes $m = O(\log(n))$ samples, but it is not known how to do this in $O(\mathrm{poly}(\log(n)))$ time.

In fact, a time-inefficient way of doing this is by

$$\hat{s} := \arg\max_{s' \in \mathbb{Z}_n}\sum_{i=1}^{m}\Pr[k_i \mid +;\, s'], \tag{178}$$

where the k_is are samples from D_s. (Like maximum likelihood estimation.)

Quantum walk. Motivating problem: Element distinctness $\mathrm{ED}_n\colon [n]^n \to \{0,1\}^n$, $\mathrm{ED}_n(x) = 1$ if and only if x contains a repeated symbol. Example $n = 5$:

$$\begin{aligned} \mathrm{ED}_n(1\ 2\ 3\ 4\ 5) &= 0. \\ \mathrm{ED}_n(1\ 2\ 2\ 3\ 4) &= 1. \end{aligned}$$

Before discussing quantum walk. Let's discuss another useful approach that can be applied here to give a non-optimal bound – but is very useful in general.

Proposition 15 (Amplitude amplification)**.** *Let $d \in \mathbb{N}$ and $\theta \in [0, \pi/2]$. Let $|\psi\rangle$, $|\psi_0\rangle$, $|\psi_1\rangle$ be d-dimensional quantum states and*

$$|\psi\rangle := \cos(\theta)|0\rangle|\psi_0\rangle + \sin(\theta)|1\rangle|\psi_1\rangle. \tag{179}$$

Let

$$G := 1_{2d} - 2|\psi\rangle\langle\psi| \quad \text{and} \quad U := 1_{2d} - 2|1\rangle\langle 1| \otimes \mathbb{1}_d. \tag{180}$$

Then, for all $k \in \mathbb{N}$,

$$(GU)^k|\psi\rangle = (-1)^k(\cos((2k+1)\theta)|0\rangle|\psi_0\rangle + \sin((2k+1)\theta)|1\rangle|\psi_1\rangle) \tag{181}$$

Proof. Essentially the same as the steps leading to Eq. (45) in Lecture 3. □

In particular, writing $\sin(\theta) = \sqrt{p}$ (so that the probability of measuring 1 in the computational basis on the first register of the initial state is p), then we can take

$$k = \lfloor \frac{\pi}{4\theta} - \frac{1}{2} \rceil \in \Big[\frac{\pi}{4\theta} - 1, \frac{\pi}{4\theta}\Big]; \quad k \leq \frac{\pi}{4}\frac{1}{\sqrt{p}} \tag{182}$$

to have

$$\sin((2k+1)\theta) \in \left[\sin(\frac{\pi}{2} - \theta), 1\right] = [\sqrt{1-p}, 1] \tag{183}$$

Typical application. Suppose we have a unitary $\mathcal{A}$ such that $\mathcal{A}|0\rangle = |\psi\rangle$ so $G = \mathcal{A}(1 - 2|0\rangle\langle 0|)\,\mathcal{A}^{-1}$. The success probability is the probability of measuring $|1\rangle$ in the first register. Then $\mathcal{A}$ has success probability p. But to amplify the success probability to close to 1 takes $O(1/\sqrt{p})$ applications of $\mathcal{A}$ and $\mathcal{A}^{-1}$ (these usually have the same or similar costs). Quadratically better than the classical case.

Lecture 17

Covered:

1. Application of amplitude amplification to element distinctness – resulting bound is suboptimal.
2. Introduction to the quantum walk framework.

References: Chapter 19.1, Chapter 17.1-2 of [AMC].

Lecture 18

Covered: Quantum walk spectral theorem: relating the spectrum of the Szegedy quantum walk operator of a stochastic $n \times n$ matrix P to that of the discriminant matrix of P.

References: Chapter 17.3 of [AMC].

Lecture 19

Covered: quantum walk algorithm for detecting existence of marked items.

References: Chapter 17.4 of [AMC].

Additional material: Let $\epsilon, \delta \in (0,1)$. Quantum walk algorithm for deciding between no marked items and $\geq \epsilon n$ marked items based on symmetric stochastic $n \times n$ matrix P and a "marked" subset $M \subseteq [n]$ that is either empty or $|M| \geq \epsilon n$. Assume 2nd largest eigenvalue of P is $\leq 1 - \delta$.

1. turn P to P', the variant of P that stops upon reaching M.
2. create the state
$$\frac{1}{\sqrt{n-|M|}} \sum_{j \notin M} |\psi_j\rangle = T \frac{1}{\sqrt{n-|M|}} \sum_{j \notin M} |j\rangle , \tag{184}$$
where $T \coloneqq \sum_{j=1}^{n} |\psi_j\rangle\langle j|$.
3. phase estimation with unitary U equal to Szegedy quantum walk operator corresponding to P' and state equal to the state above with accuracy $O(\sqrt{\delta\epsilon})$ and confidence 0.99. Comment: stress the dash/prime!

Let P_M be the $(n-|M|) \times (n-|M|)$ submatrix of P corresponding to indices not in M. Note that the discriminant matrix of P' is then
$$D(P') = \begin{pmatrix} P_M & 0 \\ 0 & \mathbb{1}_{|M|} \end{pmatrix} \tag{185}$$

Proposition 16 (Quantum phase estimation)**.** *Let $U \in \mathbb{C}^{n \times n}$ be unitary. Let $|\psi\rangle \in \mathbb{C}^n$ be a quantum state. Let $\epsilon, \delta \in (0,1)$. Let $|\theta_1\rangle, \ldots, |\theta_n\rangle$ be a complete orthonormal basis of eigenvectors of U with eigenvalues $e^{i\theta_1}, \ldots, e^{i\theta_n}$, respectively. Write*
$$|\psi\rangle = \sum_{j=1}^{n} a_j |\theta_j\rangle , \tag{186}$$
for some $a_j \in \mathbb{C}$. Then there exists a quantum circuit using $O(\epsilon^{-1} \log(1/\delta))$ applications of c-$U \coloneqq |0\rangle\langle 0| \otimes \mathbb{1}_n + |1\rangle\langle 1| \otimes U \in \mathbb{C}^{2n \times 2n}$ that outputs a $\theta \in [0, 2\pi)$ such that with probability $|a_j|^2$:
$$\Pr[|\theta - \theta_j|_{circle} \leq \epsilon] \geq 1 - \delta, \tag{187}$$
where for $x \in \mathbb{R}$, $|x|_{circle} \coloneqq \min\{|y| \mid y \in \mathbb{R}, y = x \mod 2\pi\}$.

Remark 17. Note that θ_j is only defined up to multiples of 2π but the proposition is still well-defined due to the circle norm being used. (Similarly, there is some freedom in $|\theta_j\rangle$ and a_j but the proposition is still well-defined.)

Lecture 20

Covered: application of the quantum walk framework to OR_n and ED_n.

References: Chapters 18.1-18.2 and Chapter 19 of [AMC].

Lecture 21

Covered: adversary method, application to quantum lower bound for OR_n, started proof that the adversary quantity lower bounds quantum query complexity.

References: Chapter 22 of [AMC].

Lecture 22

Covered: finished proof that the adversary quantity lower bounds quantum query complexity, stated that it also provides an upper bound and its composition properties with respect to OR and AND, covered quantum divide and conquer using the adversary quantity including two applications: (a) recognizing 20^*2 and (b) k-common subsequence.

References: Chapter 22 of [AMC] and [CKKSW'22].

Lecture 23

Block encoding and quantum signal processing (QSP).

Motivating example: quantum simulation or Hamiltonian simulation.

Definition 28. An n-qubit Hamiltonian is a Hermitian matrix in $\mathbb{C}^{2^n \times 2^n}$.

Quantum simulation problem.
Input: the description of an n-qubit Hamiltonian H, $t \in (0, \infty)$, and $\epsilon \in (0, 1)$.
Output: the description of the quantum circuit that implements a unitary $U \in \mathbb{C}^{(2^{n+m}) \times (2^{n+m})}$ that approximates $\exp(-itH)$ to error ϵ in operator norm distance on the subspace $\mathbb{C}^{2^n} \oplus 0$. That is

$$\|(\langle 0| \otimes \mathbb{1}_{2^n})U |0\rangle |\psi\rangle - \exp(-itH) |\psi\rangle \|_2 \leq \epsilon \tag{188}$$

for all $|\psi\rangle \in \mathbb{C}^{2^n}$, where $|0\rangle$ denotes the first standard basis vector in $\mathbb{C}^{2^m}$.

For the quantum simulation problem, we care both about its time complexity and the size of the quantum circuit that is generated. In fact, we often care more about the latter since that circuit will be run on real quantum hardware which (given current experimental progress) is very difficult if the circuit size is not small.

There are multiple ways that H can be described. Two of the most important and practically relevant are the *sparse oracle* description and the *Pauli-decomposition* description. For concreteness, let's focus on the latter.

Definition 29. An n-qubit Pauli matrix is a $2^n \times 2^n$ matrix of the form $A_1 \otimes \cdots \otimes A_n$, where $A_i \in \{I, X, Y, Z\}$ and

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{189}$$

X, Y, Z are known as the "Pauli-X,Y,Z" matrices.

In many quantum physics/chemistry setups, a system of "size" n (e.g., number of particles) can be described by an n-qubit Hamiltonian that is a weighted sum of a small number of Pauli matrices

$$H = \sum_{j=1}^{N} a_j P_j, \tag{190}$$

where $a_j \in \mathbb{R}$ and P_j is an n-qubit Pauli matrix, and $N = O(\text{poly}(n))$. Example: for electronic systems under certain approximations, $N = O(n^4)$ where n is the number of electrons. Comment: assuming number of spin orbitals used $= O(n)$; reasonable to use $2n$ spin orbitals as each electron can have two spins. Observe that even though H is a $2^n \times 2^n$ matrix, its description length is $O(\text{poly}(n))$.

We'll assume $\sum_{j=1}^{n} |a_j| = 1$ as otherwise can scale the simulation time t to compensate. This implies $\|H\| \leq 1$.

Method 1: Trotterization. Historically, the main approach to Hamiltonian simulation when the input is described as a Pauli decomposition has been Trotterization. This method is based on the observations that

1. the quantum circuit for $\exp(-ia_j P_j)$ is easily described with $O(n)$ gates (see Section 4.7.3 of [Nielsen and Chuang]),
2. $\exp(\sum_j A_j) = \lim_{m \to \infty} (\prod_j \exp(A_j/m))^{1/m}$.

However, this approach outputs a quantum circuit with $O(nN^4t^2/\epsilon)$ gates. The time complexity of generating the circuit, i.e., actually solving the quantum simulation problem, is also $O(nN^4t^2/\epsilon)$. For more details, see Section 6 of [Jozsa notes]. Comment: I'm assuming here that single-qubit rotation gates don't need compilation, otherwise there'll be some more logs from Solovay-Kitaev. I'm also assuming that each a_j costs constant time to compute.

Method 2: QSP. A more recent approach outputs a quantum circuit with

$$O(nN(t + \log(1/\epsilon))) \tag{191}$$

gates which can be shown to be optimal in certain query formulations of the Hamiltonian simulation problem.

The high-level approach is:

1. Generate a circuit implementing a unitary U such that

$$U = \begin{pmatrix} H & \cdot \\ \cdot & \cdot \end{pmatrix} \tag{192}$$

Here, $\cdot$ stands for arbitrary matrices (subject to U being unitary). U is called a block encoding of H. As long as $\|H\| \leq 1$, such a U always exists – see Homework 4. (In fact, the optimality mentioned above can be proven in terms of calls (queries) to U.)

Can construct U explicitly using the "Prepare-Select-Prepare^{-1} circuit" (similar to the LCU proposition in the next lecture) which takes time $O(nN)$. The resulting circuit also has size $O(nN)$ and implements $U \in \mathbb{C}^{N2^n \times N2^n}$ that acts on $k \coloneqq O(\log(N) + n)$ qubits.

2. Compute a polynomial $p \in \mathbb{C}[x]$ of degree $d \coloneqq O(t + \log(1/\epsilon))$ that approximates the function $\exp(-itx)$ to error ϵ in infinity norm over the interval $[-1, 1]$. Compute $\alpha > 0$ such that[8] $|p(x)/\alpha| \leq 1/4$ for all $x \in [-1, 1]$. For the Hamiltonian simulation example, $\alpha = 8$ suffices because if $p(x)$ is $\epsilon \in (0, 1)$ close to $\exp(-itx)$ for all $x \in [-1, 1]$, then $|p(x)| \leq 2$ for all $x \in [-1, 1]$.

Use the *quantum signal processing framework* to generate a circuit involving a single use of single-qubit controlled-U, $O(d)$ uses of (U, $U^\dagger$, a simple gate acting on $k + 1$ qubits, and a simple gate acting on 3 qubits) that implements a unitary[9]

$$\tilde{V} \coloneqq \begin{pmatrix} p(H)/\alpha & \cdot \\ \cdot & \cdot \end{pmatrix} \tag{193}$$

acting on $k + 3$ qubits. Comment: +1 from the original signal processing of P and Q (see below), +2 from LCU of 4 parts: real even, real odd, imaginary even, imaginary odd (see paragraph after Theorem 56 in [GSLW'18]).

Theorems in the framework says this takes time $O(\text{poly}(d, \log(1/\epsilon)) + d^2 + kd + nNd)$ Comment: the first term is to produce the P, Q pair that satisfies all the conditions of Proposition 17 in the next lecture, the second term is to construct the phase sequence for P and Q. The resulting circuit has size $O(kd + nNd) = O(nN(t + \log(1/\epsilon)))$.

3. Use a variant of amplitude amplification (called oblivious amplitude amplification) to generate a circuit implementing unitary

$$V \coloneqq \begin{pmatrix} p(H) & \cdot \\ \cdot & \cdot \end{pmatrix} \tag{194}$$

This uses $O(\alpha)$ calls of $\tilde{V}$ and all costs above in item 2 are scaled by α. V satisfies the output conditions of the quantum simulation problem.

The key point is the degree bound on the polynomial in the second step since the complexity of QSP scales with the degree. For Hamiltonian simulation, we are approximating the function $f\colon [-1, 1] \to \mathbb{C}$, $f(x) = \exp(-itx)$, which can be done using the Jacobi-Anger expansion:

$$\max_{x \in [-1,1]} \left| e^{-itx} - \left(J_0(-t) + 2 \sum_{k=1}^{K} i^k J_k(-t) T_k(x) \right) \right| \leq \epsilon \tag{195}$$

for some $K = O(t + \log(1/\epsilon))$, where J_k is a Bessel function and $T_k(x)$ is the kth Chebyshev polynomial. For more details, see Section 27.4 of [AMC].

The QSP framework is very powerful because it reduces many problems (not just Hamiltonian simulation) to finding polynomial approximations, e.g.,

1. solving linear systems quantumly: approximating $f(x) = 1/x$,
2. solving (time-independent) differential equations quantumly: approximating $f(x) = e^{xt}$,
3. ground state preparation: approximating $f_\mu(x) = \mathbb{1}[x \leq \mu]$ for judiciously chosen values of μ.

For more, see [MRTC'21] and Chapters 6-8 of [Lin notes].

It is important to stress that "solving linear systems quantumly" (and analogously for "solving differential equations quantumly") refers to producing a quantum state $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$ whose amplitudes are proportional to the solution to an $N \times N$ linear system $Ax = b$, which is *very different* from producing x stored as an array (the typical classical output condition). $|x\rangle$ could be used to recover x in time scaling as $O(N)$ which is not what we typically want to do as it removes any exponential (in N) quantum advantage. The hope is that we could perform measurements on $|x\rangle$ to obtain average-statistics about x in time scaling as $O(\log(N))$ which might preserve an exponential advantage. How to do this concretely is a crucially important open problem. For more discussions on this, [read the fine print].

[8] The 1/4 comes from Theorem 56 [GSLW'18] and the remark afterwards.

[9] Technically, should be "whose top-left block is a $\tilde{p}(H)$ where $\tilde{p}$ is a polynomial that is ϵ-close to p/α in infinity norm on $[-1, 1]$", this is due to root-finding algorithm used to produce (P, Q) from p which has complexity scaling as $\text{poly}(d, \log(1/\epsilon))$ – see the next lecture.

Lecture 24

For $x \in [-1,1]$, let

$$U(x) \coloneqq \begin{pmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{pmatrix}. \tag{196}$$

$U(x)$ can be seen as a block encoding of the 1×1 Hermitian matrix x with $\|x\| \leq 1$.

Proposition 17. *Let $P, Q \in \mathbb{C}[x]$. There exists $\Phi \coloneqq (\phi_0, \ldots, \phi_d) \in \mathbb{R}^{d+1}$ such that*

$$U_\Phi \coloneqq e^{i\phi_0 Z} \prod_{j=1}^{d} (U(x) e^{i\phi_j Z}) = \begin{pmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix} \tag{197}$$

if and only if P, Q satisfy

1. $\deg(P) \leq d$, $\deg(Q) \leq d-1$.
2. *P has parity $d \mod 2$ and Q has parity $d-1 \mod 2$.*
3. $\forall x \in [-1,1]$, $|P(x)|^2 + (1-x^2)|Q(x)|^2 = 1$.

Here, $\deg(Q) = -1$, means $Q = 0$.

Proof. The "only if" direction is obvious. The "if" direction can be proven by induction, see, e.g., Lemma 27.1 in [AMC] and the paragraph after the proof. (The proof also gives a way of computing Φ in time $O(d^2)$). □

However, the third condition in Proposition 17 is hard to work with. Fortunately, we have the following.

Proposition 18. *There exists an algorithm with the following properties.*

1. ***Input.*** *$p \in \mathbb{R}[x]$, $d \in \mathbb{N}$, and $\delta \in (0,1)$ such that*
   (a) $\deg(p) \leq d$.
   (b) *p has parity $d \mod 2$.*
   (c) $\forall x \in [-1,1]$, $|p(x)| \leq 1$.
2. ***Output.*** *$P, Q \in \mathbb{C}[x]$ and $\Phi \coloneqq (\phi_0, \ldots, \phi_d) \in \mathbb{R}^{d+1}$ such that*
   (a) *$|\mathrm{Re}(P)(x) - p(x)| \leq \delta$ and $|\mathrm{Re}(Q)(x)| \leq \delta$ for all[10] $x \in [-1,1]$.*
   (b) *the following equation holds:*

$$U_\Phi \coloneqq e^{i\phi_0 Z} \prod_{j=1}^{d} (U(x) e^{i\phi_j Z}) = \begin{pmatrix} P(x) & iQ(x)\sqrt{1-x^2} \\ iQ^*(x)\sqrt{1-x^2} & P^*(x) \end{pmatrix}. \tag{198}$$

3. ***Runtime.*** *$O(\mathrm{poly}(d, \log(1/\delta)))$.*

Proof. The idea is to reduce to Proposition 17 by computing the roots of p and using them to construct the P and Q satisfying both condition (a) and the conditions of Proposition 17. Computing the $\leq d$ roots of p to precision δ requires $O(\mathrm{poly}(d, \log(1/\delta)))$ time. See Corollary 10 of [GSLW'18]. □

Remark 18. There are software packages implementing Proposition 18, e.g., [QSPPACK].

The parity constraint may also seem restrictive but it can be handled by the "Linear Combination of Unitaries" (LCU) technique, first introduced in [Childs and Wiebe'12]. Suppose $p \in \mathbb{R}[x]$ satisfies the conditions in Proposition 18, except for the parity condition. Then we can write $p = p_{\mathrm{even}} + p_{\mathrm{odd}}$, where p_{even} and p_{odd} are the even and odd parts of p respectively. Then we can implement a block encoding of $p(x)/2$ by applying the LCU proposition below to the block encoding of $p_{\mathrm{even}}(x)$ and $p_{\mathrm{odd}}(x)$ with $\alpha = \beta = 1/2$. LCU has many other uses besides this.

[10] Could also be $\{x \in \mathbb{C} \mid |x| \leq c\}$ where c is a constant. The computation is by a root finding algorithm that might turn x^2 into, say, $(x-\delta)^2$; the difference of the two is bounded by $O(\delta)$ on any compact set like $\{x \in \mathbb{C} \mid |x| \leq c\}$ but not on $\mathbb{C}$.

Proposition 19 (Linear Combination of Unitaries (LCU))**.** *Let $U, V \in \mathbb{C}^{n\times n}$ be unitaries and $\alpha, \beta \geq 0$ such that $\alpha + \beta = 1$. Then there exists a circuit using the following unitaries*

$$c_0\text{-}U \coloneqq |0\rangle\langle 0| \otimes U + |1\rangle\langle 1| \otimes \mathbb{1}_n, \tag{199}$$

$$c_1\text{-}V \coloneqq |0\rangle\langle 0| \otimes \mathbb{1}_n + |1\rangle\langle 1| \otimes V, \tag{200}$$

once each that implements a $2n \times 2n$ unitary of the form

$$W \coloneqq \begin{pmatrix} \alpha U + \beta V & \cdot \\ \cdot & \cdot \end{pmatrix}. \tag{201}$$

Proof. Let unitary $P \in \mathbb{C}^{2\times 2}$ be such that $P|0\rangle = \sqrt{\alpha}|0\rangle + \sqrt{\beta}|1\rangle$. Verify that

$$(P^{-1} \otimes \mathbb{1}_n) \cdot c_1\text{-}V \cdot c_0\text{-}U \cdot (P \otimes \mathbb{1}_n) \tag{202}$$

is of the form W. □

The remainder of the lecture went over the exposition of qubitization in Chapter 27.3 of [AMC]. Qubitization is a technique that allows us to lift the 1-dimensional analysis above to an arbitrary-dimensional analysis.

Lecture 25 (unused)

Dual polynomials. Let $f\colon \{-1,1\}^n \to \{-1,1\}$ and $\epsilon \geq 0$.

The approximate degree of f is the minimal degree of a (multilinear)[11] polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]$ such that $|p(x) - f(x)| \leq \epsilon$ for all $x \in \{-1,1\}^n$.

A dual polynomial witnessing $\widetilde{\deg}_\epsilon(f) \geq d$ is a multilinear polynomial $\phi \in \mathbb{R}[x_1, \ldots, x_n]$ such that

1. (High correlation) $\sum_{x \in \{-1,1\}^n} f(x)\phi(x) > \epsilon$.
2. (High phd) $\mathrm{phd}(\phi) \geq d$, where $\mathrm{phd}(\phi)$ is the pure-high-degree of ϕ, i.e., the minimum degree of a monomial of ϕ.
3. (Normalization) $\sum_{x \in \{-1,1\}^n} |\phi(x)| = 1$.

Claim 1. $\widetilde{\deg}_\epsilon(f) \geq d$ if and only if *there exists a dual polynomial witnessing* $\widetilde{\deg}_\epsilon(f) \geq d$.

[11] The approximate degree does not change whether we include "multilinear" here or not

Additional unused material

Deciding regular languages. A key algorithmic idea in [Aaronson, Grier, Schaeffer'18].

$\mathrm{Regular}_n\colon \{0,1,2\}^n \to \{0,1\}$. $f(x)=1$ iff x contains a substring of the form 10^j1, where 0^j means a continuous string of j zeros and $j \in \{0,1,2,\dots\}$. Example $n=4$: 1102 (1), 1001 (1), 1021 (0), 2121 (0).

Proposition 20. $Q(\mathrm{Regular}_n) = O(\sqrt{n}\log^3(n))$.

Proof. [**Idea: guess the length of the target substring and verify.**] (Will be less careful with failure probabilities, minor details like divisibility, using ceiling/floor, etc.)

Suppose there is a substring of form 10^j1 of length $l \in [2^i, 2^{i+1})$ – call such a substring "good". Then split the input into disjoint blocks of length $2^i/2$ each. Then, the substring must cover some block in its entirety. Given the index $k \in \{1,\dots,K \coloneqq 2n/2^i\}$ of a block B_k can decide if B_k is covered by a good substring using $O(\sqrt{2^i})$ quantum queries as follows:

1. Check if B_k is all zeros. This is the same as computing $\mathrm{OR}_{2^i/2}$ and uses $O(\sqrt{2^i})$ queries.
2. If yes, find the first 1 to the right of B_k within distance 2^i. Use an exponentially increasing sequence: see if the first 2^j positions contains all zeros, stop when not and backtrack and binary search. Uses $O(\sqrt{2^i}\log(2^i)) = O(i\sqrt{2^i})$ queries. If 1 not found, then output 0. Else check if the positions from the right-boundary of B_k to the position of the first 1 all contain zeros. This costs $O(\sqrt{2^i})$
3. Repeat the last step on the left of B_k.

We can use the above algorithm $\mathcal{A}$, repeated $\log(K)$ times and taking majority vote to suppress errors, to instantiate essentially error-free quantum queries to a K-bit string y where bit $k \in [K]$ indicates whether B_k is covered by a good substring. (See paragraph 3 of the introduction of [Høyer, Mosca, de Wolf'03], the rest of this paper explains why we don't actually need to repeat $\log(K)$ times.)

Then we compute $\mathrm{OR}_K(y)$. This takes $O(\sqrt{K})$ queries to y. Therefore the number of queries is $O(i\sqrt{2^i}\log(K)\sqrt{K}) = O(i\sqrt{n}\log(n))$. Doing this for $i = 1, 2\dots, \log(n)$ (to cover all possible lengths of a 10^j1 substring) costs order

$$\sqrt{n}\log(n)\sum_{i=1}^{\log(n)} i = O(\sqrt{n}\log^3(n)). \tag{203}$$

□

Remark 19. The above proof gives an example of a quantum composition theorem. Suppose $f_1,\dots,f_K\colon D \subseteq \Sigma^n \to \{0,1\}$ and $\mathrm{OR}_K\colon \{0,1\}^K \to \{0,1\}$. Then can define $\mathrm{OR}_K \circ (f_1,\dots,f_K)\colon D^K \subseteq \Sigma^{Kn} \to \{0,1\}$ by

$$\begin{aligned} &\mathrm{OR}_K \circ (f_1,\dots,f_K)(x_{11},\dots,x_{1n},x_{21},\dots,x_{2n},\dots,x_{K1},\dots,x_{Kn}) \\ =&\,\mathrm{OR}_K(f_1(x_{11},\dots,x_{1n}),\dots,f_K(x_{K1},\dots,x_{Kn})). \end{aligned} \tag{204}$$

Then $Q(\mathrm{OR}_K \circ (f_1,\dots,f_K)) = O(\sqrt{K}\cdot \max_{i\in[K]} Q(f_i))$. In fact, a stronger statement is possible:

$$Q(\mathrm{OR}_K \circ (f_1,\dots,f_K)) \le O\Big(\Big(\sum_{i=1}^K Q(f_i)^2\Big)^{1/2}\Big). \tag{205}$$

Minimum finding. (Basis for many applications of quantum computing in optimization.)

$$\mathrm{MIN}_n\colon \{0,1,\dots,m-1\}^n \to \{0,1,\dots,m-1\}; \quad \mathrm{MIN}_n(x) = \min_{i\in[n]} x_i. \tag{206}$$

Note when $m=2$, this is the same as the OR_n function.

Naive method: Binary search over $\{0,1,\dots,m-1\}$. Pick $t=m/2$, decide if there is an i such that $x_i \le m/2$– this takes $O(\sqrt{n})$ quantum queries (essentially the same as computing OR_n). If yes then pick $t=m/4$, if not then pick $t=3m/4$ and continue in the same way. Quantum query complexity is $O(\log(m)\sqrt{n})$.

But in fact can remove dependence on m by essentially classical tricks.

Proposition 21. $Q(\mathrm{MIN}_n) \le O(\sqrt{n}\log(n)\log\log(n))$.

Proof. 1. Quantum part:

> **Claim 2.** *There exists a quantum query algorithm $\mathcal{A}$ parametrized by $t \in [m]$ and $R \in \mathbb{N}$ that queries $x \in \{0,1,\dots,m-1\}$ and with probability $\ge 1-1/3^R$:*

(a) outputs t if $t = \min_{i\in[n]} x_i$

(b) else outputs[12] *an i chosen uniformly at random from $\{i \mid x_i < t\}$ except with failure probability $\leq 1/3^R$*

The quantum query complexity of $\mathcal{A}$ is $O(R\sqrt{n})$.

Proof sketch. Part (a) is essentially the same as computing OR_n because can instantiate a query to the input to OR_n using a query to x in the obvious way. Repeat this $O(R)$ times to get failure probability $\leq 1/3^{10R}$. If OR_n is computed to be 0, then output 1. If OR_n is computed to be 1, then continue.

Create the uniform superposition

$$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}|i\rangle = \alpha\sum_{i|x_i\leq t}|i\rangle + \beta\sum_{i|x_i<t}|i\rangle\,, \tag{207}$$

and we can use amplitude amplification to amplify the second part so that $|\beta|^2$ becomes at least $2/3$ using $O(\sqrt{n})$ queries to x – note $|\beta|^2$ is $\geq 1/n$. (This part is sketchy as cannot apply amplitude amplification directly since when $|\beta|^2 \geq 1/n$, we do not know what it is, the idea is to assume different values of $|\beta|$ (an exponentially increasing sequence) and choosing k in Proposition 15 appropriately, then one of these trials will return an i uniformly at random from $\{i \mid x_i < t\}$ probability $\geq 2/3$; note $x_i < t$ is a verifiable condition – see Theorem 3 in quantum amplification paper [BHMT'00]. Repeat this $O(R)$ times to get failure probability to be $< 1/3^{10R}$. □

2. Classical wrapper:

Suppose there is zero failure probability in the first claim (will later set R large enough such that this is essentially true). Input $x \in \{0,1,\ldots,m-1\}^n$. Given $t \in [m]$, let $S_t \coloneqq \{i \in [n] \mid x_i < t\}$. Let $k \in \mathbb{N}$. Consider the following algorithm. The commands in square brackets are only used for the analysis.

Initialize $t = m$. For $i = 0,\ldots,k$, do:

(a) Output t if $t = \min_{i\in[n]} x_i$. [Set $B_j = \emptyset$ for all $j \geq i$.] Else sample $i \leftarrow S_t$. (I use $\leftarrow$ to mean chosen uniformly at random.) [Set $B_i = S_t$.]

(b) Classically query x_i. Set $t = x_i$.

Analysis. If $B_k = 0$, then the algorithm is successful.

For $i \in \{0,1,\ldots,k\}$, let $Y_i \coloneqq |B_i|$ and $y_i \coloneqq \mathrm{E}[Y_i]$. Write $x(B_i) \coloneqq \{x_j \mid j \in B_i\}$ as a multiset. Note Y_i is a random variable with $Y_0 = n$ and $y_0 = n$. Observe that, by definition, if $Y_k = 0$, then the algorithm succeeds in outputting $\min_{i\in[n]} x_i$.

For $z \in \{1,2,\ldots,n\}$, we have

$$\mathrm{E}[Y_{i+1} \mid Y_i = z] = \frac{1}{z}\Big(\sum_{j\in x(B_i)}((\text{rank of } j \text{ in } x(B_i)) - 1)\Big) \leq \frac{1}{z}\sum_{j=1}^{z}(j-1) = \frac{z-1}{2} \leq \frac{z}{2}, \tag{208}$$

where the first inequality is tight if $x(B_i)$ contains all distinct elements. [Example: suppose $B_i = \{1,2,3,4,5\}$ and $x(B_i) = \{1,3,6,7,8\}$, then $\mathrm{E}[Y_{i+1} \mid B_i] = \frac{1}{5}(0+1+2+3+4) = \frac{1}{5} = \frac{1}{5}\frac{1}{2}4(4+1) = 2$; suppose $x(B_i) = \{1,3,3,3,8\}$, then $\mathrm{E}[Y_{i+1} \mid B_i] = \frac{1}{5}(0+1+1+1+4) = 7/5 < 2$.]

But we also have

$$\mathrm{E}[Y_{i+1} \mid Y_i = 0] = 0 \tag{209}$$

by definition.

Therefore, for all $z \in \{0,1,\ldots,n\}$,

$$\mathrm{E}[Y_{i+1} \mid Y_i = z] \leq \frac{z}{2}. \tag{210}$$

Therefore

$$\mathrm{E}[Y_i + 1] = \sum_{z=0}^{n}\Pr[Y_i = z]\,\mathrm{E}[Y_{i+1} \mid Y_i = z] \leq \frac{1}{2}\sum_{z=0}^{n}\Pr[Y_i = z]z = \frac{1}{2}\,\mathrm{E}[Y_i], \tag{211}$$

where the first equality uses the Law of Total Expectation and the last equality uses the definition of expectation.

Therefore, $y_{i+1} \leq y_i/2$. Since $y_0 = n$, we have $y_i \leq n/2^i$.

Therefore, by Markov's inequality[13] $\Pr[Y_k \geq 1/2] \leq 2\,\mathrm{E}[Y_k] = 2n/2^k$. Therefore, since Y_k can only take non-negative integer values, if $k = O(\log(n))$, then this probability can be made to be smaller than $1/100$.

[12] Let $T \coloneqq \{i \mid x_i < t\}$. If $|T| > 0$, then $\forall i \in T$, $\Pr[\mathcal{A}(t) = i] = (1 - 1/3^R)/|T| + 1/3^R q_i$ for some $q_i \in [0,1]$; if $T = 0$, then $\Pr[\mathcal{A}(t) = i] = 0$.

[13] Suppose X is a random variable that only takes non-negative values. Then, for all $a > 0$, $\Pr[X > a] \leq \mathrm{E}[X]/a$.

Now, let's consider the failure probability. Since there are $k = O(\log(n))$ uses of the quantum algorithm in Claim 2.

$$
\begin{aligned}
&\Pr[\forall i \in \{0,1,\ldots,k\}, \text{ no failure at iteration } i] \\
=&1 - \Pr[\exists i \in \{0,1,\ldots,k\}, \text{ failure at iteration } i\ ] && \Pr[A] = 1 - \Pr[A^c] \\
\geq&1 - \sum_{i=0}^{k} \Pr[\text{failure at iteration } i] && \text{union bound, } \Pr[A_1 \cup A_2] \leq \Pr[A_1] + \Pr[A_2] \\
\geq&1 - (k+1)/3^R && \text{Claim 2} \\
\geq&1 - 1/100 && \text{by choosing } R = \log\log(n).
\end{aligned}
$$

The proposition follows: each iteration costs $O(\sqrt{n}\log\log(n))$ and there are $O(\log(n))$ iterations, multiply.

□

Remark 20. In fact, can remove the log factors by a more careful analysis so query complexity is $O(\sqrt{n})$. See [Dürr-Høyer'96].