Lecture 5

Important note: the definition of quantum circuit is considerably different from what I gave in class (see the footnote on this page for an explanation). I'll explain the updated definition a bit more at the start of the next lecture.

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:
$$\mathcal{P} \leftrightarrow L \subseteq \{0,1\}^*;\ L := \mathcal{P}^{-1}(1). \tag{61}$$

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 $1, \ldots, a$ that have no incoming edges and exactly one outgoing edge ("a input bits") and $b \in \mathbb{N}$ vertices labelled uniquely by $1', \ldots, b'$ that have no outgoing edges and exactly one incoming edge ("b output bits"), and all other vertices are labelled by symbols from set
$$\text{cGATES} := \{\text{FANOUT}, \text{AND}, \text{OR}, \text{NOT}\}, \tag{62}$$
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 has "a input and output bits".)
   (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
$$\text{qGATES} := \{T, H, \text{Toffoli}\}. \tag{63}$$

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:
$$(H, i) \to \mathbb{1}_2^{\otimes j-1} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \mathbb{1}_2^{\otimes n-j}, \quad (T, j) \to \mathbb{1}_2^{\otimes j-1} \otimes \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{pmatrix} \otimes \mathbb{1}_2^{n-j},$$
$(\text{Toffoli}, (k_1, k_2, k_3)) \to$ unitary $\text{Toffoli}_{k_1,k_2,k_3}$ defined by:
$$|\alpha\rangle_{k_1} |\beta\rangle_{k_2} |\gamma\rangle_{k_3} |w\rangle_{\text{rest}} \mapsto |\alpha\rangle_{k_1} |\beta\rangle_{k_2} |\gamma \oplus \alpha \cdot \beta\rangle_{k_3} |w\rangle_{\text{rest}} \text{ for all } \alpha, \beta, \gamma \in \{0,1\}, w \in \{0,1\}^{a-3}, \tag{64}$$
where the subscripts denotes the position of the ket (e.g. when $a = 5$, $|0\rangle_2 |0\rangle_3 |0\rangle_5 |11\rangle_{\text{rest}}$ means $|10010\rangle$). Finally, measure using $\{\Pi_z := |z\rangle\langle z| \mid z \in \{0,1\}^a\}$ at the end.

[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 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 $= \{T, H, \mathrm{CNOT}\}$. 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 $y \in \widetilde{\mathrm{cGATES}}^*$, where $\widetilde{\mathrm{cGATES}} = \mathrm{cGATES} \cup \{0, 1, \mathrm{Blank}\}$. (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 $y \in \widetilde{\mathrm{qGATES}}^*$, where $\widetilde{\mathrm{qGATES}} = \mathrm{qGATES} \cup \{0, 1, \mathrm{Blank}\}$. 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 this).

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 the description of a classical circuit C_y taking a-bit input and 1-bit output such that $C_y(0^a) = \mathcal{P}(y)$. Comment: Alternatively, we could define it such that $\mathcal{A}$ just outputs $\mathcal{P}(y)$ but I did it this way so that the three sub-definitions here follow the same structure.

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 taking a-bit input and output 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

$$T(N) = O\Big(\tau(N) \cdot Q(n(N)) + \sum_{i=0}^{d} \tau_i(N)\Big). \tag{65}$$

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, and I will say more about this at the start of 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. (65). □

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