Lecture 12

Turing model of computation.

Definition 14 (Classical and quantum algorithms in the Turing model (informal))**.** *Classical deterministic, classicla randomized, and quantum algorithms* for solving a problem in the Turing model are specified by uniform families of circuits composed of elementary gates.

1. *Classical deterministic*: classical circuits made of AND, OR, NOT, and FANOUT gates with input $x_1 \dots x_N \in \{0,1\}^N$.
2. *Classical randomized*: same as deterministic but with another possible gate: the COIN gate that outputs a bit 0 or 1 with probability 1/2 each.
3. *Quantum*: quantum circuits with input $|x_1 \dots x_N\rangle \in \mathbb{C}^{2^N}$ together with ancilla/workspace qubits[10] initialized in state $|0^k\rangle$ for some non-negative integer k made of CNOT, H, and T gates followed by computational basis measurement at the end.[11]

The output of these algorithms can be a single bit or multiple bits depending on the problem. In the quantum case, if the problem has M-bit output, then can wlog take the output to be the last M bits (out of $k+N$ bits) of the measurement outcome.

Solving a problem. We say that deterministic algorithm solves the problem if the output of the circuit is always correct for every input. We say (randomized/quantum) algorithms solves the problem with error ϵ if the output of the circuit is correct with probability at least $1-\epsilon$ for every input: common to just say "solves the problem with bounded error" without qualification in which case ϵ is conventionally taken to be 1/3.

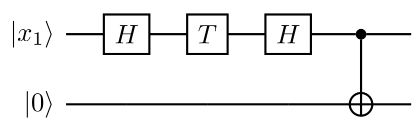
Uniform family means that there exists a classical Turing machine (this has a complicated formal definition but can think of as a program/computer code/circuit) that for every N generates the circuit that solves the problem on all inputs $x \in \{0,1\}^N$ of size N.

The *time complexity* of the (deterministic/randomized/quantum) algorithm is the *time of generation* (which is at least the number of gates).

Remark 5. The set of gates does not matter too much as long as they can be implemented physically and are "universal" meaning

1. In the deterministic case: any function of the input can be implemented by some circuit.
2. In the randomized case: any stochastic map can be approximated by some circuit.
3. In the quantum case: any unitary can be approximated by some circuit. (For the gate set we considered, this follows from the Solovay-Kitaev theorem.)

Example 3. *We analyzed the following example in class.*



Definition 15 (Complexity classes)**.** Given a decision problem $\mathcal{P}$ (decision means $\mathcal{P}$ has a single-bit so can be modelled as $\mathcal{P}\colon \{0,1\}^* \to \{0,1\}$), we say $\mathcal{P}$ is in $\{P, BPP, BQP\}$ if it can be solved by a {deterministic, randomized, quantum} algorithm with time complexity scaling polynomially in the input size N.

Transforming a classical circuit to a quantum one.

Proposition 6. *Suppose we have a classical circuit (implementing the function) $C\colon \{0,1\}^N \to \{0,1\}$. We can efficiently transform C to a quantum circuit Q implementing the unitary $Q\colon |x\rangle|b\rangle \mapsto |x\rangle|b \oplus C(x)\rangle$ for all $x \in \{0,1\}^N$ and $b \in \{0,1\}$ (note that this acts on $\mathbb{C}^{2^{N+1}}$).*

Proof. First assume quantum circuits also have Toffoli gates. (See HW2.)

AND can be simulated by Toffoli, OR can be simulated by Toffoli and X (think de Morgan's law), NOT is simulated by X, FANOUT is simulated by CNOT (with target set to $|0\rangle$).

Comment: Then draw U-copy-$U^\dagger$ and explain. □

Remark 6. This implies P is contained in BQP. In fact BPP is also contained in BQP: can simulate the COIN gate by the Hadamard gate (and use principle of deferred measurement).

[10] You might wonder why there were no ancilla/workspace bits in the deterministic/randomized case. The reason is that you could generate them yourself using FANOUT and NOT: $0 = x_1 \wedge \neg x_1$ but in the quantum case simulating FANOUT using the CNOT gate requires ancilla (see below).

[11] You may have read a version of this lecture with X (Pauli-X) included in the gate set. But that's actually redundant since X can be made using H and T gates (how?).