CSE 291 / Math 277A - Quantum Complexity Theory	October 6, 2022
Lecture 5	
Lecturer: Daniel Grier	*Scribe: Ishaan Kavoori*

1 Classical complexity classes

Polynomial Time (P):
Languages L such that there exists a deterministic poly-time Turing machine M such that M accepts x iff $x \in L$.

Non-deterministic Polynomial Time (NP):
Language L such that there exists deterministic poly-time Turing machine M and polynomial q such that for all $x \in \{0,1\}^n$

- If $x \in L$, $\exists y \in \{0,1\}^{q(n)}$ such that $M(x,y)$ accepts
- If $x \notin L$, $\forall y \in \{0,1\}^{q(n)}$, $M(x,y)$ rejects.

NP is a generalization of P (just forget about the witness string y), so $\mathsf{P} \subseteq \mathsf{NP}$. It is widely conjectured that $\mathsf{P} \neq \mathsf{NP}$, but we do not have a proof!

Bounded-error Probabalistic Polynomial Time (BPP):
Languages L such that there exists a deterministic poly-time Turing machine M and polynomial q such that for all $x \in \{0,1\}^n$

- If $x \in L$, then $M(x,y)$ accepts with at least probability $\frac{2}{3}$ over uniform choice of $y \in \{0,1\}^{q(n)}$.
- If $x \notin L$, then $M(x,y)$ accepts with probability at most $\frac{1}{3}$ over uniform choice of $y \in \{0,1\}^{q(n)}$.

Once again, it is clear that $\mathsf{P} \subseteq \mathsf{BPP}$ since we can just forget about the extra random bits. However, we do not know if $\mathsf{BPP} \subseteq \mathsf{NP}$ or if $\mathsf{NP} \subseteq \mathsf{BPP}$, though it is widely conjectured that $\mathsf{P} = \mathsf{BPP}$.

Probabilistic Polynomial Time (PP):
The same setup as BPP, but with a smaller gap between the acceptance and rejection probabilities:

- If $x \in L$, then $M(x,y)$ accepts more than $\frac{1}{2}$ of strings $y \in \{0,1\}^{q(n)}$.
- If $x \notin L$, then $M(x,y)$ accepts at most $\frac{1}{2}$ of strings $y \in \{0,1\}^{q(n)}$.

We have that $\mathsf{BPP} \subseteq \mathsf{PP}$ simply by definition—PP allows for a narrower gap. In fact, PP is powerful enough even to contain NP. To see this, take any NP machine M, and alter it in the following way. If $M(x,y)$ accepts, then accept. If $M(x,y)$ rejects, then flip an unbiased coin (to be completely rigorous, one would need to extend the length of the random string y)—if heads, accept, and if tails, reject. Notice that if there are no accepting y for the original machine, then the new machine accepts with exactly 50% probability. However, if there is any accepting y, then the new machine accepts with greater than 50%, and so the inclusion $\mathsf{NP} \subseteq \mathsf{PP}$ follows.

Polynomial Space (PSPACE):
Languages L such that there exists a deterministic Turing machine M that uses at most polynomial space and M accepts x iff $x \in L$.

We have that $\mathsf{PP} \subseteq \mathsf{PSPACE}$ since a PSPACE machine can simply count all the $y \in \{0,1\}^{q(n)}$ that make a poly-time Turing machine accept. There are exponentially many such y, but this is not an issue since we can erase the previous computation as we are enumerating over all the y.

Exponential Time (EXP):
Languages L such that there exists a deterministic Turing machine M and a polynomial q such M halts in $2^{q(n)}$ time and M accepts x iff $x \in L$.

We have that PSPACE $\subseteq$ EXP because a Turing machine that uses polynomial space can only have exponentially many configurations. And, if you were to reach the same configuration twice, then you will be in an infinite loop, so you might as well halt.

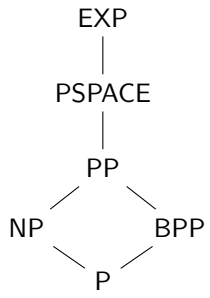

Figure 1: Diagram of class inclusions. A is below B if $A \subseteq B$.

2 Circuit encodings and BQP

To properly define the quantum complexity class BQP, we need to first discuss how a quantum circuit is encoded. Let us suppose that the circuit is constructed from some reasonable universal gate set (i.e., all the amplitudes used in the gates are efficiently computable). We will use the notation $\langle Q \rangle$ to denote the encoding of a circuit Q as a bit string. We now discuss the requirement for a proper encoding:

1. The mapping from a circuit to its encoding must be *injective*.
2. The encoding is of polynomial length. If Q has m gates, then $\langle Q \rangle$ has at most poly(m) bits.
3. The encoding is of length at least m (avoid "super clever" encodings that compress things too much).

Definition 1 (Poly-time Uniformity)**.** *A circuit family $\{Q_n\}_{n=1}^{\infty}$ is poly-time uniform if there exists a poly-time Turing machine such that on input 1^n outputs $\langle Q_n \rangle$.*

Bounded-error Quantum Polynomial Time (BQP):
Languages L such that there exists poly-time uniform class of quantum circuits $\{Q_n\}$ such that $\forall x \in \{0,1\}^n$:

- If $x \in L$, probability of measuring 1 on the first qubit of $Q_n(|x\rangle \otimes |0 \ldots 0\rangle)$ is at least $\frac{2}{3}$.
- If $x \notin L$, probability of measuring 1 on the first qubit of $Q_n(|x\rangle \otimes |0 \ldots 0\rangle)$ is at most $\frac{1}{3}$.

The reason that we include the ancillary register $|0 \ldots 0\rangle$ in the definition is to allow for some robustness in the definition of the class. For example, recall the principle of deferred measurement from the last lecture.

Theorem 2. BQP $\subseteq$ EXP.

Proof. An n-qubit state can be represented as a vector $|x\rangle \in \mathbb{C}^{2^n}$. Each unitary operation is equivalent as multiplying the state vector with a unitary matrix of size $2^n \times 2^n$. Since matrix-vector products can be computed in time $(2^n)^2 = 4^n$, and simulating the circuit requires at most polynomially many matrix-vector product operations, we then have BQP $\subseteq$ EXP. □

Theorem 3. BQP $\subseteq$ PP.

Proof in the next lecture...