1 Probabilistic Complexity Classes

1.1 BPP

BPP (Bounded Probabilistic Polynomial Time) is a class of languages that can be defined as follows:

$L \in$ **BPP** if $\exists$ PPT (Probabilistic Polynomial Time) algorithm A with the following properties:

- **Completeness:** $\forall x \in L, \underset{\text{A's coinflips}}{Pr}[A(x) = 1] > \frac{2}{3}$. In other words, if x is in the language, algorithm A will output 1 with probability greater than 2/3
- **Soundness:** $\forall x \in L, \underset{\text{A's coinflips}}{Pr}[A(x) = 1] < \frac{1}{3}$. Or the probability of the algorithm outputting 1 on an input not in the language is less that 1/3.

It is a large open question in computer science whether or not $P = BPP$.

1.1.1 Equality of Definition

The bound of 2/3 and 1/3 are selected arbitrary. It is the case that any algorithm that demonstrates an advantage of $(\frac{1}{2} + \varepsilon)$ can be converted into an algorithm A' that has an negligible chance of failing. The conversion is as follows.

A' : For an algorithm A with advantage $(\frac{1}{2} + \varepsilon)$, run algorithm $A(x)$ k times with fresh randomness each time and output 1 if the majority of the k runs outputs 1, else output 0.

To demonstrate that this is correct, and to find the value needed for k, we need the following bound from probability.

Chernoff Bound: Let $X_1, X_2, ..., X_n$ be independent $\{0, 1\}$ random variables w/ probability that $X_i = 1$ is $0 \leq p \leq 1$.

- Let $S = \sum_{i=1}^{n} X_i$. Then the expected value $E[S] = p \cdot n$

The Chernoff Bound states that

$$\forall \varepsilon \ \ 0 \leq \varepsilon \leq 1, \ \Pr[S > (1 + \varepsilon)E[S]] \leq e^{\frac{-\varepsilon^2 \cdot E[S]}{3}}$$

Plugging this into algorithm A', we will seek the probability that $s > \frac{n}{2}$. Let $X_i = 1$ if on run i, it makes a mistake, else 0. This will give us the total number of mistakes that the algorithm A' makes, which is expected to be $n/3$, or $E[S] = n/3$. We set $\varepsilon = 1/2$ to get the following:

$$\Pr[S > (1 + \frac{1}{2})\frac{n}{3}] \leq e^{\frac{-\frac{1}{4}\cdot\frac{n}{3}}{3}}$$
$$\Pr[S > \frac{n}{2}] \leq e^{-\frac{n}{27}}$$

To get a bound of $1/e^{80}$, we would simply need to run the procedure $27 * 80$ times, a constant amount.

1.1.2 Other Probabilistic Complexity Classes

Clearly, since the properties of IP problems and BPP problems are the same, this strategy can be applied to IP problems.

EP (Expected Polynomial): The class of problems that are expected to run in polynomial time, but can potentially run in exponential time with negligible probability. (This can be achieved by running multiple algorithms in paralell)
(Did not write down the others he talked about)

2 co-NP $\in$ IP

co-NP is the class of all languages L such that for every string $x \notin L$ there exists a polynomial-size witness w with which a polynomial-time verifier can confirm that $x \notin L$.

For example, 3-SAT is an NP-Complete problem that aims to determine if x, a conjunction of 3-literal clauses can be satisfied given some truth assignment for all $x_0, ..., x_n$.

For example, an instance $\Phi = (x_1 \vee \overline{x_3} \vee x_5) \wedge (x_1 \vee \overline{x_7} \vee x_2)$...

This problem is in NP, langauge membership of Φ can be conducted in polynomial time. However, testing if $\Phi \notin L$ is a much more difficult task, as a proof would essentially involve enumerating through all truth assignments to demonstrate none satisfy Φ. Because of this, the problem 3-UNSAT is co-NP (since testing if $x \in L$ is the same as 3-SAT). 3-UNSAT is a co-NP complete problem, meaning any problem in co-NP can be converted to an instance of 3-UNSAT in polynomial time. With this, we will show that 3-UNSAT $\in IP$ which show that co-NP $\in IP$.

2.1 Arithmeticizing 3-Sat

Arithmeticization is the process of converting a valid instance of 3-UNSAT to an algebraic formula that will allow us to perform an interactive proof.

- If $x = 0$ x is False
- If $x \neq 0$ x is True
- $\vee$ gate (or) becomes $+$
- $\wedge$ gate (and) becomes $\cdot$
- $\neg x = 1 - x$

For notation purposes, we will refer to the arithmeticized boolean formula as $\Phi(x_1...x_n)$. To show that all truth assignments of $X = x_1...x_n$ do not satisfy Φ, we demonstrate the following:

$$\sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} ... \sum_{x_n \in \{0,1\}} \Phi(x_1...x_n) = 0$$

Or in other words, the sum of all truth assignments of the arithmeticized formula is 0. The proof idea involves the prover P demonstrating that they are able to perform calculations that are exponential in time, and providing the verifier with the ability to check that they are using the original equation as well as verifying that they are not cheating.

The proof uses the following:

- Arithmetisized clause for any assignment must be ≤ 3 ($(1 + 1 + 1)$ is at most 3)
- There are m clauses, so the total value must be $\leq 2^n \cdot 3^m$ (The max value is obtained where all m clauses yield a value of 3 for all 2^n possible truth value assignments)
- We pick a prime $q > 2^n 3^m$ and perform all calculations over a finite field $\mathbb{F}_q$