Probabilistic Algorithms

Non-Deterministic vs. Probabilistic

All algorithms we have seen so far are either deterministic or impractical (non-deterministic)

To make non-deterministic algorithms more practical we introduce probabilistic algorithms

A probabilistic algorithm (Turing Machine) is a non-deterministic algorithm that makes non-deterministic choices randomly, e.g. by flipping a coin

This is still not practical, because sometimes the algorithm should be extremely lucky to solve problems

Interactive Proofs

Prover
- Has unlimited computational power
- Wants to convince Verifier in something

Verifier
- Can perform polynomial time computations
- Accepts or rejects after performing some computation

They can exchange messages

Proofs for Problems in NP

SAT
- Prover and Verifier get an instance of SAT
- Prover solves the instance using his unlimited computational power and send a satisfying assignment to Verifier
- Verifier checks (in polynomial time) if what obtained is a satisfying assignment, and accepts if it is or rejects otherwise

Problems from coNP

Graph Non-Isomorphism
- Instance: Graphs G and H
- Question: Are G and H isomorphic?

This problem belongs to coNP, but is not believed to be coNP-complete

Apparently, there is no way to prove interactively that two graphs are not isomorphic

Randomized Verifier

Now suppose that Verifier has a fair coin

Given graphs G and H
- Verifier chooses one of G and H by flipping a coin
- Verifier then renames somehow the vertices of the chosen graph and send it to Prover
- Prover decides which graph it received
- Prover send the answer to Verifier
- Repeat the procedure
If the graphs are not isomorphic then Prover always gives the right answer.
If the graphs are isomorphic then Prover gives a correct answer with probability 1/2.

Therefore if Prover is wrong we conclude that the graphs are isomorphic.
If after $n$ repetitions of the protocol, Prover gives only right answers, then Verifier can conclude with probability $\frac{1}{2}$ that graphs are not isomorphic.

Define the probability that $PT^{'}$ accepts $w$ to be $\sum_{\text{path}} \Pr[PT^{'} \text{ accepts } w]$, where $k$ is the number of coin flips made along this path.

Let $X_1, X_2, \ldots, X_n$ be a series of independent experiments (for example, coin flips) such that the probability of success in each of them is $p$.

The amplification lemma states that for any polynomial $m(x)$ and a probabilistic TM $PT^{'}$, there is a probabilistic TM $PT_{m(x)}^{'}$ that operates with error probability $\epsilon$.

The main idea is to run $PT^{'}$ many times and then output the majority of votes.

Thus, with each computational path, we can associate the probability of taking this path. This probability is equal to $\frac{1}{2^k}$ where $k$ is the number of coin flips made along this path.

Denote this probability by $Pr[p]$. 

Clearly, $Pr[wPT^{'} accepts w] = 1 - Pr[wPT^{'} rejects w]$.

Class BPP

A Probabilistic Turing Machine $PT^{'}$ recognizes language $L$ with error probability $\epsilon$ if
- $w \in L$ implies $Pr[PT^{'} accepts w] \geq 1 - \epsilon$
- $w \notin L$ implies $Pr[PT^{'} rejects w] \geq 1 - \epsilon$

We say that $PT^{'}$ operates with error probability $2^{-\Omega(n)}$ if the above inequalities hold for $\epsilon = 2^{-\Omega(n)}$, where $n$ is the length of $w$.

BPP is the class of languages that are recognizable by probabilistic Turing Machines with error probability of $1/3$.

Math Prerequisites

Let $X_1, X_2, \ldots, X_n$ be a series of independent experiments (for example, coin flips) such that the probability of success in each of them is $p$.

Theorem (Chernoff Bound)

If $\mu = \frac{1}{2} + \epsilon$ for some $\epsilon > 0$, then the probability that the number of successes in a series of $n$ experiments is less than $\mu$ is at most $e^{-\frac{n\epsilon^2}{4}}$. 
Proof of Amplification Lemma

Machine $PT$, works as follows

On input $w$
  - for $i = 1$ to $n(|w|)$ do
    - simulate $PT_i$ on $w$
  - if most runs of $PT_i$ accept, then accept; otherwise reject

Analysis

The number $n(|w|)$ must be such that

$$\lambda \frac{\ln n}{\varepsilon} < 2^{-\varepsilon n}$$
$$\varepsilon n \ln n < -\ln(\varepsilon) n \ln 2$$
$$\varepsilon n \ln n > \ln(\varepsilon) n \ln 2$$
$$n \ln \ln \frac{1}{\varepsilon} \geq 6\ln 2 / \varepsilon^2 n$$