Complexity Classes

497 - Randomized Algorithms

Sariel Har-Peled

September 4, 2002

1 Las Vegas and Monte Carlo algorithms

Definition 1.1 A *Las Vegas* algorithm is a randomized algorithms that *always* return the correct result. The only variant is that it's running time might change between executions.

An example for a Las Vegas algorithm is the `QuickSort` algorithm.

Definition 1.2 *Monte Carlo* algorithm is a randomized algorithm that might output an incorrect result. However, the probability of error can be diminished by repeated executions of the algorithm.

The `MinCut` algorithm was an example of a Monte Carlo algorithm.

2 Complexity Classes

I assume people know what are Turing machines, **NP**, **NPC**, RAM machines, uniform model, logarithmic model. **PSPACE**, and **EXP**. If you do now know what are those things, you should read about them. Some of that is covered in the randomized algorithms book, and some other stuff is covered in any basic text on complexity theory.

Definition 2.1 The class **P** consists of all languages L that have a polynomial time algorithm A, such that for any input Σ^*,

- $x \in L \Rightarrow A(x)$ accepts.
- $x \notin L \Rightarrow A(x)$ rejects.

Definition 2.2 The class **NP** consists of all languages L that have a polynomial time algorithm A, such that for any input Σ^*,

- $x \in L \Rightarrow \exists y \in \Sigma^*, A(x,y)$ accepts, where $|y|$ (i.e. the length of y) is bounded by a polynomial in $|x|$.
- $x \notin L \Rightarrow \forall y \in \Sigma^* A(x,y)$ rejects.

Definition 2.3 For a complexity class $\mathcal{C}$, we define the complementary class co-$\mathcal{C}$ as the set of languages whose complement is in the class $\mathcal{C}$. That is

$$co-\mathcal{C} = \left\{ L \,\middle|\, \overline{L} \in \mathcal{C} \right\}.$$

It is obvious that $\mathbf{P} = \text{co}-\mathbf{P}$ and $\mathbf{P} \subseteq \mathbf{NP} \cap \text{co}-\mathbf{NP}$. (It is currently unknown if $\mathbf{P} = \mathbf{NP} \cap \text{co}-\mathbf{NP}$ or whether $\mathbf{NP} = \text{co}-\mathbf{NP}$, although both statements are believed to be false.)

Definition 2.4 The class **RP** (for Randomized Polynomial time) consists of all languages L that have a randomized algorithm A with worst case polynomial running time such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \Pr\left[A(x) \text{ accepts}\right] \geq 1/2.$
- $x \notin L \Rightarrow \Pr\left[A(x) \text{ accepts}\right] = 0.$

An **RP** algorithm is Monte Carlo, but the mistake can only be if $x \in L$. co$-$**RP** is all the languages that have a Monte Carlo algorithm that make a mistake only if $x \notin L$. A problem which is in $\mathbf{RP} \cap \text{co}-\mathbf{RP}$ has an algorithm that does not make a mistake, namely a Las Vegas algorithm.

Definition 2.5 The class **ZPP** (for Zero-error Probabilistic Polynomial time) is the class of languages that have Las Vegas algorithms in expected polynomial time.

Definition 2.6 The class **PP** (for Probabilistic Polynomial time) is the class of languages that have a randomized algorithm A with worst case polynomial running time such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \Pr\left[A(x) \text{ accepts}\right] > 1/2.$
- $x \notin L \Rightarrow \Pr\left[A(x) \text{ accepts}\right] < 1/2.$

The class **PP** is not very useful. Why?

Definition 2.7 The class **BPP** (for Bounded-error Probabilistic Polynomial time) is the class of languages that have a randomized algorithm A with worst case polynomial running time such that for any input $x \in \Sigma^*$,

- $x \in L \Rightarrow \Pr\left[A(x) \text{ accepts}\right] \geq 3/4.$
- $x \notin L \Rightarrow \Pr\left[A(x) \text{ accepts}\right] \leq 1/4.$