Chapter 34

Complexity classes

By Sariel Har-Peled, April 26, 2022[1]

"I'm a simple man, a guileless man," Panin answered. "There is a norm. The norm is five gravities. My simple, uncomplicated organism cannot bear anything exceeding the norm. My organism tried six once, and got carried out at six minutes some seconds. With me along."

Almost the same, Arkady and Boris Strugatsky

34.1. Las Vegas and Monte Carlo algorithms

Definition 34.1.1. 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 34.1.2. A ***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 matrix multiplication algorithm is an example of a Monte Carlo algorithm.

34.2. 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[2].

Definition 34.2.1. The class **P** consists of all languages L that have a polynomial time algorithm **Alg**, such that for any input Σ^*, we have

(A) $x \in L \Rightarrow$ **Alg**(x) accepts,
(B) $x \notin L \Rightarrow$ **Alg**(x) rejects.

Definition 34.2.2. The class **NP** consists of all languages L that have a polynomial time algorithm **Alg**, such that for any input Σ^*, we have:

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

[2]There is also the internet.

Definition 34.2.3. 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

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

4 where $\overline{L} = \Sigma^* \setminus L$.

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

(i) If $x \in L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] \geq 1/2$.
(ii) $x \notin L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] = 0$.

An **RP** algorithm is a Monte Carlo algorithm, but this algorithm can make a mistake only if $x \in L$. As such, 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 34.2.5. The class **ZPP** (for Zero-error Probabilistic Polynomial time) is the class of languages that have a Las Vegas algorithm that runs in expected polynomial time.

Definition 34.2.6. The class **PP** (for Probabilistic Polynomial time) is the class of languages that have a randomized algorithm **Alg**, with worst case polynomial running time, such that for any input $x \in \Sigma^*$, we have

(i) If $x \in L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] > 1/2$.
(ii) If $x \notin L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] < 1/2$.

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

Exercise 34.2.7. Provide a **PP** algorithm for 3SAT.

Consider the mind-boggling stupid randomized algorithm that returns either yes or no with probability half. This algorithm is almost in **PP**, as it return the correct answer with probability half. An algorithm is in **PP** needs to be *slightly* better, and be correct with probability better than half. However, how much better can be made to be arbitrarily close to 1/2. In particular, there is no way to do effective amplification with such an algorithm.

Definition 34.2.8. The class **BPP** (for Bounded-error Probabilistic Polynomial time) is the class of languages that have a randomized algorithm **Alg** with worst case polynomial running time such that for any input $x \in \Sigma^*$, we have

(i) If $x \in L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] \geq 3/4$.
(ii) If $x \notin L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] \leq 1/4$.

34.3. Bibliographical notes

Section 34.1 follows [MR95, Section 1.5].

References

[MR95] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge, UK: Cambridge University Press, 1995.