DIMACS Tutorial on
Fine-Grained Complexity

Prep Sheet

This is a preparatory exercise sheet with some basic exercises on asymptotics and randomness in algorithm design. It is not strictly necessary to solve these exercises before the tutorial, but we recommend that you give it a try.

Asymptotics

Recall that in theoretical computer science we typically state the running time (and space complexity, etc.) of algorithms in terms of the O-notation. That is, instead of stating that an algorithm takes at most $12n + 10$ basic operations, we simply say that the algorithm runs in time $O(n)$. (See `https://en.wikipedia.org/wiki/Big_O_notation` for the formal definition.)

Problem 1 (Asymptotics). Order the following running times according to their asymptotic growth (e.g., $n \log n = O(n^2)$):

$$n^2, \quad n \log n, \quad 2^n, \quad 10n, \quad \log^{10} n, \quad \frac{n}{10}, \quad \frac{n^3}{\log n}, \quad 1.99^n, \quad 100\sqrt{n}, \quad n^3$$

Probability Theory

We recap some basic definitions from probability theory which we will be useful throughout the tutorial. A *random variable* X is a number or object measuring the outcome of some random experiment. Some relevant examples include throwing a coin ($X \in \{\text{heads}, \text{tails}\}$), throwing a die ($X \in \{1, 2, 3, 4, 5, 6\}$), *indicator variables* of events such as “is it going to rain today” ($X \in \{0, 1\}$) or the running time of an algorithm ($X \in \mathbb{Z}$). For numeric random variables, the *expectation* $\mathbf{E}[X]$ is defined

$$\mathbf{E}[X] = \sum_x x \cdot \mathbf{P}(X = x),$$

where we sum over all values x that the random variable can take. For instance, the expected outcome of throwing an unbiased die ($X \in \{1, 2, 3, 4, 5, 6\}$) is

$$\mathbf{E}[X] = \sum_{x=1}^{6} x \cdot \mathbf{P}(X = x) = \sum_{x=1}^{6} x \cdot \frac{1}{6} = 3.5.$$

Recall that the expectation is *linear*, i.e. for random variables $X_1, \dots, X_n$ it holds that $\mathbf{E}[\sum_i X_i] = \sum_i \mathbf{E}[X_i]$.

Let us finally recap two useful bounds that are generally useful in the design of randomized algorithms:

Theorem (Markov’s Inequality). Let $X \geq 0$ be a random variable that only takes nonnegative values. Then for all $t > 0$:

$$\mathbf{P}(X \geq t) \leq \frac{\mathbf{E}[X]}{t}.$$

Theorem (Chernoff' Inequality). Let $X_1, \dots, X_n \in \{0, 1\}$ denote independent random variables, and let $X = X_1 + \cdots + X_n$. Then for all $\epsilon > 0$:

$$\mathbf{P}(|X - \mathbf{E}[X]| > \epsilon\, \mathbf{E}[X]) \leq 2e^{-\epsilon^2\, \mathbf{E}[X]/3}.$$

Problem 2 (Boosting I). Consider a decision problem P (i.e., for each input the answer to P is either YES or NO). We say that an algorithm for P has *one-sided error probability* δ if for a given YES instance the algorithm always returns YES and for a given NO instance the algorithm returns NO with probability at least $1 - \delta$.

Suppose that there is an algorithm for P with one-sided error probability $\frac{1}{2}$ that runs in time $T(n)$. Give an algorithm for P with arbitrarily small one-sided error probability $\delta > 0$ that runs in time $O(T(n) \log(1/\delta))$.

Problem 3 (Boosting II). Consider a decision problem P. We say that an algorithm for P has *two-sided error probability* δ if for any given instance the algorithm returns the correct answer with probability at least $1 - \delta$.

Suppose that there is an algorithm for P with one-sided error probability $\frac{1}{3}$ that runs in time $T(n)$. Give an algorithm for P with arbitrarily small two-sided error probability $\delta > 0$ that runs in time $O(T(n) \log(1/\delta))$.

Hint: Apply Chernoff's bound.