# Algorithm Design - Homework 1

Academic year 2016/2017

Instructors: Prof. Luca Becchetti, Prof. Francesco Pasquale

November 2, 2016

**Due date: November 17th, 11.59pm**

Make sure that the solutions are typewritten or clear to read. A complete answer consists of a clear description of an algorithm (an English description is fine), followed by an analysis of its running time and a proof that it works correctly.

**Hand in your solutions and keep a copy for yourself. Solutions will be posted or presented after due date. In the final exam, you will be asked to explain your solutions and/or to go over your mistakes.**

**Collaboration policy.** You may discuss the homework problems with other members of the class, but you must write up the assignment alone, in isolation. Also, you must understand well your solutions and be able to discuss your choices and their motivations in detail with the instructor. Finally, you should cite any sources you use in working on a homework problem.

**Late policy:** Every homework must be returned by its due date. Homeworks that are late will lose 10% of the grade if they are up to 1 day (24h) late, 20% if they are 2 days late, 30% if they are 3 days late, and they will receive no credit if they are late by more than 3 days.

*Please refer to course's Web page for detailed information about above aspects.*

**Exercise 1.** You have $n$ random coins such that $p_i \in [0,1]$ is the probability the $i$-th coin turns out `Head`. You want to compute the probability $P(n,k)$ that, when you toss all the $n$ coins, exactly $k$ of them turn out `Head`.
For example, if $p_1 = 1/2$, $p_2 = 1/3$, $p_3 = 1/4$, then

$$P(3,0) = (1-p_1)(1-p_2)(1-p_3) = \frac{1}{2}\cdot\frac{2}{3}\cdot\frac{3}{4} = \frac{1}{4},$$

$$\begin{aligned} P(3,1) &= p_1(1-p_2)(1-p_3) + (1-p_1)p_2(1-p_3) + (1-p_1)(1-p_2)p_3 \\ &= \frac{1}{2}\cdot\frac{2}{3}\cdot\frac{3}{4} + \frac{1}{2}\cdot\frac{1}{3}\cdot\frac{3}{4} + \frac{1}{2}\cdot\frac{2}{3}\cdot\frac{1}{4} = \frac{11}{24}. \end{aligned}$$

Notice that, in general, $P(n,k)$ involves a sum of $\binom{n}{k}$ terms, each one being a product of $n$ numbers. Hence, a naïve algorithm for computing $P(n,k)$ would do $n\binom{n}{k}$ products and $\binom{n}{k}$ sums.

Design an algorithm that takes in input two non-negative integers, $n$ and $k$ with $n \geqslant k$, and $n$ values $p_1, \dots, p_n \in [0,1]$ and returns $P(n,k)$ in $\mathcal{O}(nk)$ time (assume that summing and multiplying two numbers takes time $\mathcal{O}(1)$).

**Remark:** note that this is not an exercise about probabilities or probabilistic algorithms. You have to give a *deterministic* algorithm that computes $P(n,k)$, given $n$ and $k$.

**Exercise 2.** Let $G = (V, E)$ be a directed graph and for every edge $e \in E$ let $c(e) \in \mathbb{N}$ be its capacity. Let $s, t \in V$ be a source and a sink node let $f : E \to \mathbb{N}$ be a *max flow* from $s$ to $t$ and assume $f$ is *acyclic*, i.e., there is no cycle in $G$ in which all edges have positive flow.

Suppose the capacity of one specific edge $\hat{e} \in E$ decreases of one unit, i.e., consider the new capacities

$$\hat{c}(e) = \begin{cases} c(e) - 1 & \text{for } e = \hat{e} \\ c(e) & \text{for } e \neq \hat{e} \end{cases} .$$

Design an algorithm that, starting from $f$, finds a max flow from $s$ to $t$ in the graph $G$ with the new capacities $\hat{c}$ in time $\mathcal{O}(|V| + |E|)$.

**Exercise 3.** We have a perfectly rectilinear road, with $n$ homes in positions $0 \leq c_1 < c_2 < \cdots < c_n$. We want to install wireless signal repeaters along the road, so that all homes receive the signal. Each repeater has range $d > 0$, so that a home is covered, whenever it has at least one repeater within distance $d$. Design an algorithm that takes homes' positions and $d$ as input and outputs the list of positions in which repeaters should be installed. **Goal:** minimize the number of repeaters used.

**Exercise 4.** Human blood is grouped into four types: A, B, AB, and 0. Each letter refers to a kind of *antigen*, or protein, on the surface of red blood cells. For example, the surface of red blood cells in Type A blood has antigens known as A-antigens, while 0 is a blood type that has neither A nor B antigens. When blood transfusion is needed, one needs to consider that an individual whose blood does not possess a given antigen, cannot be transfused with blood containing that antigen. So for example, individuals with blood type B cannot receive blood from individual of types A or AB (because of the presence of A-antigens), but they can receive transfusion of B and 0 type bloods (the latter contains no antigens).

Now, assume an hospital has $d_0, d_A, d_B, d_{AB} \in \mathbb{N}$ supplies of blood of each type. Give an efficient algorithm that, taken availabilities $d_0, d_A, d_B, d_{AB}$ and demands $r_0, r_A, r_B, r_{AB}$ of blood types, decides whether it is possible or not to meet all demands.

**Exercise 5.** A 3D printer can produce different items, with different shapes. The times needed to produce different items may differ, but once the desired shape is specified, the time needed to produce the corresponding item is known.

In this scenario, assume we want to produce $n$ different items with processing times $p_1, \ldots, p_n$. Assume once the processing order is specified, the 3D printer produces all objects in the specified order, without interruptions. If we assume without loss of generality that production begins at time 0, the *response time* for item $j$ is $f_j$, the time at which $j$ is output from the printer.

Design an algorithm that takes as input the processing times of the $n$ items and outputs a *schedule*, i.e., the order in which the items will be produced. **Goal:** minimize the average response time $\frac{1}{n} \sum_{j=1}^{n} f_j$.

For example, assume we have three items with production times $p_1 = 1, p_2 = 6, p_3 = 4$ respectively. If we serve them in the order $(1, 2, 3)$ the average response time is $\frac{1}{3}(1 + (1 + 6) + (1 + 6 + 4)) = \frac{19}{3} \approx 6.33$, while if we follow the order $(2, 3, 1)$, average response time increases to $\frac{1}{3}(6 + (6 + 4) + (6 + 4 + 1)) = 9$.

`Hint:` begin by showing that minimizing average response time is the same as minimizing $\sum_{j=1}^{n} f_j$ (easy to show).