Algorithm Design - Homework 1

Academic year 2018/2019

Instructors: Prof. Stefano Leonardi, Prof. Chris Schwiegelshohn

November 13, 2018

Due date: December 7th 2018, 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 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. Cristina is interested in a metric space (X, d), where all distances $d(x, y)$ are either 0 (in which case $x = y$), 1 or 3. Further, all distances are symmetric and obey the triangle inequality, that is $d(a, b) + d(b, c) \geq d(a, c)$. Cristina wants to cluster the points of X. Unfortunately, she does not yet know how many clusters she has to use. She therefore will find a permutation $\pi(X)$ and use, for every $k \in \{1, \ldots, |X|\}$, the first k elements $C = \{\pi(X)_1, \ldots, \pi(X)_k\}$ of the permutation as centers.

Goal: For any such k, the output should be optimal with respect to following objective function:

$$\max_{x \in X} \min_{c \in C} d(x, c).$$

Hint: Use a greedy algorithm. The running time should be no larger than $O(|X|^2)$

Exercise 2. Consider a city with m parallel horizontal streets and n parallel vertical avenues. These lines cross in $m \times n$ intersections. On $k \in \{1, \ldots, m \times n\}$ of these intersections, special checkpoints are placed. We want to place video cameras on a subset of the streets and of the avenues such that each checkpoint is in the visibility range of a camera. A camera allows to monitor all the checkpoints of an avenue or of a street. The subset of may contain both horizontal streets and vertical avenues. Clearly, you can always select all $m + n$ of these streets and avenues. The challenge therefore is to select a smallest subset of these streets and avenues such that each checkpoint is in the visibility range of a camera.

Problem: Give an algorithm that finds such a smallest subset of streets and avenues where to place cameras in time polynomial in m and n. Prove that the algorithm is correct and provide an analysis of its running time.

Hint: Use the maximum-flow, minimum-cut theorem and the Ford-Fulkerson algorithm.

Exercise 3. Michele's birthday is coming up and he is thinking about who to invite for the party. He has male friends, denoted by the set M, and female friends, denoted by the set F. Between any two friends $x, y \in M \cup F$, there exists a score $w(x, y) \in \{0, 1\}$. The score tells us whether the two friends like each other or not (0 meaning that they do not like each other, 1 meaning that they like each other). For the party to be successful we have to consider the following two constraints.

- Michele would like to maximize the number of liked guests over the total number of invited guests. Formally, let $I \subseteq M \cup F$ be the invited guests. Then we want to maximize $\frac{1}{|I|} \sum_{x,y \in I} w(x, y)$.
- Michele *insists* upon having an equal number of female and male guests. That is $|I \cap M| = |I \cap F|$.

Goal: Show that this problem is NP-complete.

Exercise 4. A consulting company can execute tasks requested from its customers either with hired personnel or with freelance workers. The set of tasks is presented on a subset S of time instants $\{1, \ldots, T\}$. If task j_t is assigned to a hired employee of the consulting company, the cost is given by his daily salary s. If task j_t is outsourced to a freelance worker, the paid cost is c_t and it depends on the specific task and time instant. A worker can be hired at any time by paying him a hiring cost C and fired at any later time by paying a severance cost S.

Goal: Design an optimal strategy that runs in polynomial time that minimizes the total cost of executing the tasks. The total cost should include the costs paid to the freelances workers and the costs paid for hiring, firing and the salaries of the the hired personnel. Prove that the algorithm is correct and provide an analysis of its running time. Implement the algorithm with a programming language of your choice.

Assume now that task j_t requires a set $W_t \subseteq W$, $|W| = k$, of a constant number k of different types of workers. The company should therefore decide which types of workers to hire and which types of workers to outsource. The salary cost for any time instant is the same for all types of workers as well as the hiring and the firing cost. The cost of worker $j \in W$ varies with time: the cost of worker $j \in W$ is equal to c_t^j.

Goal: Design an optimal strategy that runs in polynomial time that minimizes the total cost of executing the tasks.

Hint: Use dynamic programming for both excercises. The polynomial running time of the final algorithm in the second excercise should depend on T and on 2^k. Start with the case $k = 2$ an generalize the approach from there.

Exercise 5. Federico is a mathematician who doesn't like stories and wants the exercise to get to the point.

Goal: We are given a weighted graph $G(V, E)$. Let $e \in E$.

1. Design an algorithm that decides whether or not there exists a minimum spanning tree containing e. For full marks, the algorithm must run in time at most $O(|V| + |E|)$.
2. Design an algorithm that computes a minimum spanning tree containing e, if one exists. For full marks, the algorithm must run in time at most $O(|E| \log |E|)$. Implement the algorithm with a programming language of your choice.