Algorithm Design - Homework 1

Academic year 2019/2020

Instructor: Prof. Stefano Leonardi
Teaching Assistants: Philip Lazos and Rebecca Reiffenhäuser

Deadline: January 10, 2020

Make sure that the solutions are typewritten and 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. In the final exam, you will be asked to explain your solutions and/or to go over your mistakes.

Collaboration policy. You can work on the exercises in teams of two people, and hand them in accordingly. However, make sure that each one of you has fully understood every solution you present. Do not copy any work from other students, the internet or other sources, and do not share your work with others outside your team. If at any point, any part of the exercises you hand in is apparent to be a copy of other work, this will result in the following consequences: All of your exercises, previous as well as upcoming ones, will be treated as if you did not hand them in at all, and you will have to participate in the written exam to make up for this. Please note that there will be no exception made, even if you are the original author of work someone else copied, or if your exercise partner is the one responsible. Therefore, please make sure to only choose a partner that you trust, and do not hand out your exercise solutions to others.

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.

Formal Requirements: Please typeset your solutions (11 pt at least), and state the names of both team members at the beginning of each sheet you hand in. Make sure to name the exact exercise each part of the solution refers to, otherwise it will not be graded. Please start a new page for each main exercise in the assignment (i.e. exercise 1, 2, etc., but not for each subquestion in them). Make sure your solution takes no more than one page for each main exercise, because everything after the first page will not be taken into account. So, for example, when the assignment has four exercises, please hand in four pages, one for each exercise. All solutions have to be sent via email to

lazos@diag.uniroma1.it **or** rebeccar@diag.uniroma1.it

with subject line 'Algorithm Design - Homework 1'.

Office Hours: Philip and/or Rebecca are available for questions during fixed office hours which will be announced on Piazza.

Please refer to course's Web page for updates about the above aspects.

Exercise 1. Michele is very interested in fashion and owns a whole bunch of nice outfits, collectively forming the set O. A set F of his friends all want to borrow one of Michele's outfits to go to a wedding. This should be doable, since $|F| \leq |O|$. However, due to different restrictions like size, body form, or adjustements that have to be made to an outfit to really look wedding-appropriate, there will be some extra effort $w(f, o) \geq 0$ required when any friend f wants to wear outfit o.

a) Model the problem as an integer linear program, and relax this to a corresponding LP.

b) Usually, the optimal solution to a problem's LP-relaxation is better than that of the original ILP. The factor by which both can differ is called the *integrality gap*. For Michele's problem, there is no such gap and for every fractional LP-solution, there exists an integral feasible solution with the same cost. Give a polynomial-time algorithm that, from any given optimal LP-solution, computes such an optimal integer assignment.

Exercise 2. Chris' crazy working hours and his descent from southern Germany have given him a business idea: Since in Bavaria, the shops close very early, he will open a grocery service for working people (located in a set B of office buildings) who never get off work early enough to do their own shopping. His employees will get the ordered groceries for everyone and stash them in a special room in some of the buildings for the customers to then pick up. To save costs, Chris will not rent a room (and equip it with a refrigerator...) for each of the buildings, but wants to do so for as few buildings as possible. For every pair (b_1, b_2) from B, he knows if these two buildings allow occupants to go from one to the other in reasonably short time or not. If yes, this is represented by an edge (b_1, b_2) in the graph G on vertex set B, meaning that people from building b_1 can pick up their groceries also at b_2, and vice versa. Chris has the following idea for a randomized algorithm computing a good subset $B' \subseteq B$ of buildings to rent a room in, such that all customers can get their groceries:

> Fix some order $e_1, e_2, \ldots, e_m$ of all edges in the edge set E of G, and set $B' = \emptyset$.
> Add to B' all isolated vertices, i.e. the ones without any incident edges.
> For every edge $e_1, e_2, \ldots$, check if one of its endpoints is already contained in B'. If not, flip a fair coin deciding which of the endpoints to choose, and add this endpoint to B'.

a) Show that in expectation, this algorithm will output a feasible solution where Chris has to rent at most twice as many rooms as in the optimal one.
(**Hint:** Look at the situation for one vertex, together with all of its direct neighbors.)

b) Chris would like to derandomize the algorithm to prevent himself from possibly ending up with a ridiculously bad solution. Prove that indeed, for every constant $c \geq 1$, the algorithm might produce a B' with $|B'| \geq c|OPT|$.

c) Give a short argument why there is not much hope of deriving an efficient, deterministic version using the method of conditional expectation as was presented in the lecture.
(**Hint:** You can assume the problem is not efficiently approximable up to any factor better than two.)

Exercise 3. Philip decided to play a rock, paper, scissors with his friends for money. The game works as usual, with the loser paying the victor one euro. However, because Philip likes to win, every time he chooses rock he also gets α euros out of thin air, at no cost to the other player.

Goal: Given $\alpha \geq 0$ calculate Philip's payoff at some mixed Nash equilibrium, for the following cases:

1. At first, Philip doesn't want to cheat too much, so he sets $\alpha < 1$.
2. Then he decides this was not enough so he changes it to $\alpha \geq 1$.

Exercise 4. Every time the students submit their homework, the department tries to give out awards to as many students as possible, by finding students that are provably better than the rest. Each homework has two exercises which are graded with extremely high accuracy: the scores are $a_i, b_i \in [0, 1]$ for exercises one and two respectively. A student is 'among the best' if there is no other student with a *better* score in *both* exercises. Of course the students do not cheat, so you can assume that the students scores on both exercises are independent for each student. The exercises themselves have little relation to each other, so the scores a student gets to either exercise are completely independent as well.

Goal: Given n, the number of students, and assuming that each student a_i, and b_i is uniformly distributed in $[0, 1]$, show that there are $O(\log n)$ students worthy of an award, with high probability.

Exercise 5. (Bonus Exercise) Forming teams for the AD homework is a difficult task. Consider an undirected graph $G = (V, E)$ representing the social network of friendship/trust between students. We would like to form teams of three students that know each other. The question is to decide whether the network allows for enough such teams, without checking all the triples of graph G. For this reason, we use random sampling to design an efficient estimator of the number of connected triples.

We partition the set of node triples into four sets T_0, T_1, T_2, and T_3. A node triple v_1, v_2, v_3 belongs to T_0 iff no edge exists between the nodes v_1, v_2, and v_3, - T_1 iff exactly one of the edges (v_1, v_2), (v_2, v_3), and $(v_3; v_1)$ exists,- T_2 iff exactly two of the edges (v_1, v_2), (v_2, v_3), and (v_3, v_1) exist,- T_3 iff all of the edges (v_1, v_2), (v_2, v_3), and $(v_3; v_1)$ exist.

$|T3|$ denotes the number of connected triples in the graph that is the quantity we need to estimate.

Consider the following algorithm:

- Sample an edge $e = (a, b)$ uniformly chosen from E
- Choose a node v uniformly from $V \setminus \{a, b\}$
- if $(a, v) \in E$ and $(b, v) \in E$ then $x = 1$, else $x = 0$

1. Show that $|T_1| + 2|T_2| + 3|T_3| = |E|(|V| - 2)$
2. Let $x_1, x_2, \ldots x_s$ be the outcomes of s independent executions of the algorithm. Show that $\frac{1}{s}\sum_{i=1}^{s} x_i \frac{|E|(|V|-2))}{3}$ is an estimator of T_3.
3. Find a nontrivial number s of executions of the algorithm which are sufficient in order obtain an $(1 + \epsilon)$ and an $(1 - \epsilon)$ approximation of T_3 with probability at least $1 - \delta$.