Algorithm Design - Homework 2

Academic year 2017/2018

Instructor: Prof. Stefano Leonardi

December 23, 2017

Due date: January 14, 11.59pm

Make sure that the solutions are typewritten or clear to read. A complete answer consists of a clear description of an algorithm, followed by an analysis of its running time and a proof that it works correctly. Exercise 1 requires an implementation of the algorithm. Each exercise can be awarded a maximum of 6 points. Bonus exercises add the total if points were deducted from other exercises, including those from the first homework.

Send your solutions to *schwiegelshohn"at"dis.uniroma1.it* (replace "at" with @) and keep a copy for yourself. Office hours are not held for the duration of the assignment, but you can send emails to Chris (email address above) who will reply as soon as possible. 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 your solutions in detail 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. Let A be a set of n 2-dimensional points. The diameter of A is the maximum Euclidean distance between any two points in A. It is straightforward to compute a diameter in $O(n^2)$ time by computing the distance between all pairs of points. We want to design a faster algorithm that returns an approximation. Specifically, design an algorithm that returns a $\sqrt{2}$ approximate diameter, where an α-approximate diameter $\tilde{\Delta}$ is defined as $\frac{\max_{a,b\in A} \mathrm{dist}(a,b)}{\tilde{\Delta}} \leq \alpha$. For full marks the algorithm must run in linear time ($O(n)$).

Exercise 2. Let $G(V, E)$ be an unweighted and undirected graph. Let S be a set of $\frac{24n}{\varepsilon^2}$ edges sampled uniformly at random without replacement. Assume that we only have access to the sampled edges of graph. Our goal is to estimate the weight of the max-cut $w(A, B) = |(A\times B)\cap E|$. Specifically, show that we can obtain a $(1\pm\varepsilon)$ approximate estimate by computing

$$|(A \times B) \cap S| \cdot \frac{|E|}{|S|},$$

while also showing that for any cut (A', B') that is not the max-cut, the probability that the estimated cut value $|(A' \times B') \cap S| \cdot \frac{|E|}{|S|}$ is greater than

$$|(A' \times B') \cap E| + \varepsilon \cdot |(A \times B) \cap E|$$

is small. Your algorithm has to succeed with probability at least 1/2, but can take an arbitrary time.

Base your estimation on the following two probabilistic inequalities:

- Given a set of $|S|$ i.d.d. Bernoulli random variables $X_1, \ldots X_{|S|}$ with expected value μ, the *Chernoff bounds* state

$$\mathbb{P}\left[\sum_{i=1}^{|S|} X_i > (1+\varepsilon)|S| \cdot \mu\right] < \exp\left(-\frac{\varepsilon^2 \cdot |S| \cdot \mu}{3}\right) \text{ and}$$

$$\mathbb{P}\left[\sum_{i=1}^{|S|} X_i < (1-\varepsilon)|S| \cdot \mu\right] < \exp\left(-\frac{\varepsilon^2 \cdot |S| \cdot \mu}{2}\right).$$

- The *union bound* states that given multiple not necessarily disjoint Bernoulli random variables E_i,

$$\mathbb{P}[\cup E_i] \leq \sum \mathbb{P}[E_i].$$

Exercise 3. Consider the Metric Facility Location problem defined on a set F of facilities and a set C of cities. Let G be a bipartite graph connecting facilities to cities. The costs c_{ij} of connecting city $j \in C$ to facility $i \in F$ form a metric and therefore satisfy the triangle inequality. The problem is to find a subset $I \subseteq F$ of facilities that should be opened, and a function $\phi : C \to I$ assigning the cities to opened facilities in a way that the total cost of opening facilities and connecting cities to open facilities is minimised.

1. Now, let us assume that the cost of connecting city j to facility i is also a function of the amount of demand d_j which is expected from city j. The cost of connecting city j to facility i is equal to $d_j c_{ij}$. You are asked to extend the primal-dual 3-approximation algorithm presented in the course in order to take into account the demand d_j from city j in the connection cost.

2. **Bonus problem.** In a second variation of the basic version of problem, we assume that a facility can be opened multiple times, but each time the facility is opened, it can serve at most u_i cities. Extend the basic primal dual scheme in order to obtain a constant factor approximation for this second problem.

Exercise 4. Assume we have n agents with each agent having a job with weight w_i. There is a set of m identical machines. Each agent chooses a machine. The machine chosen by agent i is denoted by $m(i)$. Let $L_j = \sum_{i:m(i)=j} w_i$ the load of machine j, i.e., the sum of the weights of the agents who choose it. The individual goal of every agent is to minimize the load of her machine. The global objective of the problem is to minimise the makespan, i.e., maximum load $\max_j L_j$. A pure Nash Equilibrium is an assignment of agents to machines so that no agent can unilaterally deviate to a different machine and decrease her load.

1. Prove that starting from any solution it is possible to converge to a pure Nash equilibrium. **Hint:** Use the the potential function $\Phi = \sum_j \frac{1}{2} L_j^2$.

2. Prove that the makespan of the pure Nash equilibrium is at most twice the minimum makespan.

Exercise 5. (Bonus Problem) On Christmas eve, one of the elves gets to accompany Santa. This year, the elves cannot agree on who should get picked. One of the elves produces a coin and suggests using a coin toss to determine the lucky party. Since the elves are notoriously mischievous, none of the other elves trust the coin to be fair. Indeed, after a few trials, the elves see that one side appears more frequently than the other, but they have not idea as to how big the bias is. Help the elves by giving an algorithm that simulates a fair coin toss while having access only to an unfair coin. Also, be sure that algorithm does not take too long. The sleigh is already ready and Santa is impatient.

Happy holidays and a happy new year!