Advanced Algorithms Deadline: November 23, 2018, 11:59 pm

Graded Homework 2

Lecturer: Mohsen Ghaffari

Note 1: *For all problems, besides providing the algorithm, you should present a complete analysis that proves the claimed performance, e.g., approximation guarantee or running time.*

Note 2: *Your solutions must by typeset in LaTeX, and you should email the pdf version to Sebastian Brandt (brandts@ethz.ch) by the indicated deadline. Please include the phrase "Advanced Algorithms 2018: Graded Homework 2" in the title of your email, to help us in finding the submissions.*

1 FPRAS for satisfaction probability of DNFs (20 points)

The input is a DNF formula f with n Boolean variables $x_1, \ldots, x_n$, and probabilities $p_1, \ldots, p_n$ for setting the respective Boolean variable to be true. Devise an FPRAS for approximating the probability that f is satisfied, in a random assignment.

2 Max Directed Cut Approximation (5 point)

Given a edge-weighted directed graph $G = (V, E, w)$, devise a polynomial-time $\frac{1}{4}$-approximation (randomized) algorithm for the objective of finding a subset $S \subseteq V$ that maximizes the cut size, i.e., the total weight of the edges going out of S. The algorithm's expected cut size should be at least a $1/4$ factor of the maximum cut.

3 Alternative Randomized Rounding for MAX-SAT (10 point)

Consider the setting of the MAX-SAT problem (as discussed in exercise 1 of problem set 4), and suppose that y_i^* is the assignment for variable y_i in the optium solution to the Linear Program. Consider the following non-linear randomized rounding where we set the Boolean variable x_i to be true with probability $f(y_i^*)$, in which the function f is defined in three cases, as follows: (A) If $y \in [0, 1/3]$, then $f(y) = \frac{3}{4}y + \frac{1}{4}$. (B) If $y \in [1/3, 2/3]$, then $f(y) = \frac{1}{2}$. (C) If $y \in [2/3, 1]$, then $f(y) = \frac{3}{4}y$. Prove that with this non-linear randomized rounding, we can obtain a $3/4$ approximation of MAX-SAT.

4 Approximation for Modified MAX-SAT (Williamson-Shmoys's Problem 5.8, 20 points)

Consider the setting of the MAX-SAT problem (as discussed in exercise 1 of problem set 4), but now with an additional constraint that all variables appear positively in all clauses. Given this change, we could easily satisfy all clauses. But we now want to maximize a different measure instead: suppose the i^{th} clause has a weight w_i and the j^{th} variable has weight w'_j. We want to maximize the total weight of satisfied clauses plus the total weight of variables set to false. Give an Integer Programming formulation of this problem, and a randomized rounding for it that leads to a $2(\sqrt{2} - 1)$ approximation.

5 k-median in Metric Spaces (Williamson-Shmoys's Problem 8.10, 25 points)

The input is a set of locations N in a metric space, and a parameter k. Let c_{ij} be the distance between $i, j \in N$. The goal is to select a subset $S \subseteq N$ of locations for the centers, where $|S| = k$, such that we minimize the sum of distances of each location in N to its nearest center in S, that is, we want to choose S such that it minimizes $\sum_{j \in N} (min_{i \in S}\ c_{ij})$. We develop an algorithm for this in two steps.

(a) This question has 15 points: Suppose that the metric comes from a tree metric, i.e., c_{ij} are distances between the nodes of a tree T that is known to you. Give a polynomial-time exact algorithm for this case.

(b) This question has 10 points: Give a polynomial-time randomized $O(\log N)$-approximation algorithm for the general case.

6 An Alternative Approach for Number of Distinct Elements (20 points)

Here, we devise an alternative algorithm for the problem approximating the number of distinct elements in a stream $a_1, a_2, \ldots, a_m \in \{1, 2, \ldots, n\}$, using a small memory.

(a) This question has 5 points: First, suppose that we just want a tester for whether the number d of distinct elements is larger than some value k or not. Define a random subset $S \subseteq \{1, \ldots, n\}$ by including each $i \in \{1, \ldots, n\}$ in set S with probability $1/k$. For this problem, you can assume that different elements are included in S independently and you do not need to worry about the space for maintaining S; although there are ways of reducing this space. During the stream, just remember $z = \sum_{i \in S} f_i$, which is the summation of the frequencies of the elements in set S. Let d be the number of distinct elements. Prove that if $d > (1 + \varepsilon)k$, then $Pr[z = 0] \leq 1/e - \varepsilon/3$, and if $d < (1 - \varepsilon)k$, then $Pr[z = 0] \geq 1/e + \varepsilon/3$.

(b) This question has 5 points: Obtain a randomized tester for whether $d \geq k$ or not. In particular, if $d > (1 + \varepsilon)k$, your tester should say 'yes' with probability at least 0.99 and if $d < (1 - \varepsilon)k$, your tester should say 'no' with probability at least 0.99. In other cases, the tester is allowed to output arbitrarily. Hint: think about repeat the above scheme for $O(1/\varepsilon^2)$ iterations (all done in parallel, through the stream).

(c) This question has 10 points: Extend the tester provided above to obtain a $1 + \varepsilon$ approximation of the number of distinct elements. Your algorithm should have success probability $1 - \delta$, for a given $\delta > 0$. How much memory do you need, overall?