Advanced Algorithms 2021 Deadline: November 10, 11:59 pm

Graded Homework 1

Lecturer: Mohsen Ghaffari *Student Name:*

Note 1: *Solutions must be typeset in LaTeX and the related pdf file should be uploaded via moodle by 11:59 pm on November 10, 2021 (e.g., the pdf of your solution to problem i should be uploaded in the assignment titled "GHW1: Problem i" on the moodle homepage). Late submissions will not be graded. If you would like to add a drawing in your solution, you can simply include a picture of a hand-drawn figure in your latex. You can submit only pdf one file for each problem and the drawing should be incorporated in the rest of your explanation.*

Note 2: *This is a theory course and any claim should be substantiated with a proof. You can discuss the problems with the other students but you should write your own solutions independently, and you should be able to orally explain your submitted solution to the instructors, if asked. It is strictly prohibited to share any (hand)written or electronic (partial) solutions with fellow students. Moreover, if you discussed a problem with another student, you must list their names on your submitted solution.*

Note 3: *All your algorithms should have polynomial time complexity. Unless specified otherwise, your algorithms should be deterministic, although you may get partial points also for a randomized algorithm.*

1 Routing Messages in a Cycle (15 points)

Consider n nodes numbered from 0 to $n-1$ that are arranged clockwise around a circle and there is an edge between node i and node $i+1 \pmod n$ for each $i \in \{0, \ldots, n-1\}$. There are m messages, and m pairs of vertices $\{u_i, v_i\}$ such that the i-th message should be routed from u_i to v_i for $i = 1, .., m$. We have to decide for each message whether to route it clockwise or counterclockwise. The load of an edge is the number of messages that are sent over it. Provide a polynomial-time 2-approximation algorithm that minimizes the maximum load.

2 Minimum Dominating Set in Special Graphs (15 points)

We say that a graph is an *a-forest* if the edges of the graph can be split into a edge-disjoint forests. In each one of the forests we fix a root for each one of the trees, and orient the edges towards their parents in the trees. We say that u is a *parent* of v if u is a parent of v in one of the a forests.

Recall that a set of vertices D is a *dominating set (DS)* of a graph G, if each vertex is either in D or has a neighbor in D. We say that a set of vertices P is a *parent-DS* of an a-forest graph G if each vertex is either in P or has a parent in P.

1. Design an $O(a)$-approximation algorithm for finding a minimum size parent-DS in an a-forest graph G. The input to the graph is the a-forest, together with the orientation of the edges.

2. Design an $O(a^2)$-approximation algorithm for finding a minimum size DS in an a-forest graph G. The input to the graph is the a-forest, together with the orientation of the edges. You can assume you have an algorithm for part 1 of the problem, even if you did not solve that part.

3 Avid Traveler (15 points)

There are n cities $\{1, \cdots, n\}$ in Europe, and a set of m local train tickets $\{T_1, \cdots, T_m\}$, where T_i is a list of cities you can travel to using the ith ticket, and let $p_i \in \{1, \cdots, n^{100}\}$ to be the price of the ith ticket. We are guaranteed that each city c_i is in at least 1 local ticket, but is in at most 10 local tickets. There are also 10 super tickets $\{S_1, \cdots S_{10}\}$, each costs s_i. You want to buy a set of tickets such that you can travel to any city in Europe while minimizing the cost. Moreover, as a condition for buying super ticket, we are not allowed to have a city simultaneously in a local ticket and a super ticket we bought (i.e. $T_i \cap S_j = \emptyset$ for any T_i and S_j we buy). Devise a polynomial-time approximation algorithm for this problem with a constant approximation guarantee, and for the smallest approximation factor that you can achieve.

4 Vertex Cover (15 points)

Consider the linear program that is used to 2-approximate the weighted vertex cover problem, that is we minimize $\sum_{u \in V(G)} w_u x_u$ given $x_u + x_v \geq 1$ for every edge uv and $x_u \geq 0$ for each node u. Prove that there is a polynomial time algorithm that finds an optimum solution to this linear program such that it has the additional property that for every node u we have $x_u \in \{0, 1/2, 1\}$.

Note: You can assume that the LP solver provides an optimal solution with polynomial precision. That is, each x_u has polynomial many bits.

5 Forbidden Words (20 points)

We are given n bits and m bit strings (forbidden words) $S_1, S_2, \ldots, S_m$. We say an n bit string is legal if it does not contain any S_i as a consecutive substring. For example, let $n = 5$, and $S_1 = 010$, then 11101 is a legal string, but 10101 is not because $x_2x_3x_4 = 010$. Moreover, suppose that $x_1 \cdots x_l$ is a forbidden word, then $\overline{x_1} \cdots x_k$ for $1 \leq k \leq l$ are not forbidden words where $\overline{x_1} = 1$ if $x_1 = 0$, and $\overline{x_1} = 0$ if $x_1 = 1$. In your algorithm, you can call a procedure LegalGenerator that generates a uniformly random legal string given its length and a set of forbidden words. Devise a fully polynomial randomized approximation scheme (FPRAS) for estimating the total number of legal n bit strings.

Hint. Can you say anything in the case $k > l$?

6 Fast Streaming Algorithm for MST (20 points)

In the exercises of week 5, we discussed a streaming algorithm for computing a minimum spanning tree (MST) in $O(\log^2 n)$ passes. The goal of this question is to show a faster algorithm. Recall that in a k-pass streaming algorithm the algorithm is allowed to have k passes over the input graph.

Design a streaming algorithm for computing the MST of the graph, using $\tilde{O}(n)$ total memory and $O(\log n \log\log n)$ passes. The algorithm can use randomization and should work with high probability. You can assume that the weights of the edges in the input graph are non-negative integers in $\{1, 2, \ldots, n^{10}\}$.

Hint. Note that if you want to compute outgoing edges from a small number of connected components, you can sample a large number of edges adjacent to each connected component without violating the memory constraints of the algorithm. Show how to exploit this.