CS 466/666: Algorithm Design and Analysis	**University of Waterloo: Fall 2023**

Homework 4

Due: Monday, November 27, 2023

Problem 1. Recall the notion of a *spanner* from Lecture 15, namely, subgraphs that preserve the distances between pairs of vertices up to a *multiplicative* approximation. In this question, we consider the same problem, but this time, with an *additive* approximation guarantee.

We say that a subgraph $H = (V, E_H)$ of a graph $G = (V, E)$ is a **+2-additive spanner** iff for every vertices $u, v \in V$,

$$dist_G(u, v) \leqslant dist_H(u, v) \leqslant dist_G(u, v) + 2.$$

In this question, we will show that every undirected (unweighted) graph G admits a +2-additive spanner with $O(n\sqrt{n} \cdot \log n)$ edges.

(a) Suppose we sample each vertex in G independently and with probability $(10 \log n)/\sqrt{n}$ in a set S. Prove that with high probability every vertex with degree at least $\sqrt{n}$ has at least one neighbor in S. **(10 points)**

(b) Let S be a set as chosen in part (a). Let H be a subgraph of G that contains a BFS tree from every vertex in S, plus the set of all edges on vertices with degree $< \sqrt{n}$. Prove that H is a +2-spanner of G with $O(n\sqrt{n} \log n)$ edges with high probability. **(15 points)**

Problem 2. Suppose we have two players Alice and Bob, who have strings $x \in \{0, 1\}^n$ and $y \in \{0, 1\}^n$, respectively. The goal for us is to decide if Alice and Bob have the same string or not, i.e., is $x = y$ or not. The problem is that Alice and Bob cannot talk to each other and they can only toss their own random coins (as in there is no shared source of randomness between them). Design a solution where both Alice and Bob only send a single message of size

$$O(\sqrt{n}) \cdot \log^{O(1)}(n)$$

bits each to us *directly* and based on their messages, we can decide, with high probability, if $x = y$ or not.

Problem 3. Consider the following linear program for the set cover problem with sets $S_1, \ldots, S_m$ from the universe $[n]$, which we studied in Lecture 10:

$$\begin{aligned} \min_{x \in \mathbb{R}^m} \quad & \sum_{i=1}^{m} x_i \\ \text{subject to} \quad & \sum_{S_i \ni e} x_i \geqslant 1 \quad \forall e \in [n] \\ & x_i \geqslant 0 \quad \forall i \in [m]. \end{aligned}$$

We use the MWU technique to design a $(1 + \varepsilon)$-approximation algorithm for this LP for a given $\varepsilon > 0$.

(a) For every element $e \in [n]$, maintain a weight w_e and let $W := \sum_{e \in [n]} w_e$. Consider this oracle LP:

$$\begin{aligned} \min_{x \in \mathbb{R}^m} \quad & \sum_{i=1}^{m} x_i \\ \text{subject to} \quad & \sum_{e \in [n]} w_e \cdot \sum_{S_i \ni e} x_i \geqslant W \\ & x_i \geqslant 0 \quad \forall i \in [m]. \end{aligned}$$

Design an algorithm for finding the optimum solution to this LP and prove that the value of this solution is always upper bounded by that of the original LP for set cover. **(10 points)**

(b) Consider the following update rule in the MWU for a given solution $x^{(t)}$ to the LP of part (a) for weights $w^{(t)}$ at iteration $t \geqslant 1$. For any element $e \in [n]$, define $x_e^{(t)} := \sum_{S_i \ni e} x_i^{(t)}$ and update:

$$w_e^{(t+1)} \leftarrow (1 - \eta \cdot x_e^{(t)}) \cdot w_e^{(t)},$$

for some $\eta > 0$ that you will need to choose later. Prove the following two equations after running the MWU algorithm with the above update rules for T iterations:

$$W^{(T)} \leqslant \exp\left(-\eta \cdot T + \ln n\right)$$
$$w_e^{(T)} \geqslant \exp\left(-\eta \cdot \sum_{t=1}^{T} x_e^{(t)} - \eta^2 \cdot \sum_{t=1}^{T} {x_e^{(t)}}^2\right).$$

Note that you need to pick η properly to be able to prove the above bounds. **(20 points)**

(c) Use the previous two steps to design a polynomial time algorithm based on MWU that outputs a $(1+\varepsilon)$-approximation to the set cover LP. Remember to both bound the number of iterations of your MWU algorithm as well as the time that it takes to solve the oracle LP in each iteration. **(20 points)**

Problem 4 (**Extra Credit**). Design a solution to the oracle LP for the Set Cover in Problem 3 so that the number of iterations of the resulting MWU algorithm becomes $O(\frac{\log n}{\varepsilon^2})$ iterations only with each iteration taking $O(m \cdot \text{poly}\log(m))$ time. **(+15 points)**

Problem 5 (**Extra Credit**). Design a +4-additive spanner H of any undirected graph $G = (V, E)$ with

$$O(n^{4/3}) \cdot \log^{O(1)}(n)$$

edges. A +4-spanner is defined as in Problem 1 except that we allow the distance in H to at most +4 larger than the distance in G, i.e., for every vertices $u, v \in V$,

$$dist_G(u, v) \leqslant dist_H(u, v) \leqslant dist_G(u, v) + 4.$$

(+25 points)

Hint: This is a *really hard* question – good luck!