**CS 466/666: Algorithm Design and Analysis** **University of Waterloo: Fall 2024**

# Homework 4

Due: Thursday, November 28, 2024

**Problem 1.** 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. Instead, we need to design a solution where each of Alice and Bob send a single message to us, and based on that messages, we output the answer.

(a) Prove that if the messages of both Alice and Bob are deterministic, then each of them needs to send at least $n$ bits. **(5 points)**

(b) Suppose Alice and Bob have access to a shared source of randomness which we can see as well. Prove that $O(1)$ bits of communication is sufficient for us to solve the problem with probability at least $2/3$. **(5 points)**

**Extra credit.** Prove that in this case, even a single bit from each of Alice and Bob is sufficient for us to solve the problem with probability at least $2/3$. **(+5 points)**

(c) Finally, suppose that Alice and Bob do have their own private randomness, but it is not shared between the two of them, nor visible to us. Prove that $O(\sqrt{n}) \cdot \log^{O(1)}(n)$ bits of communication is sufficient for us to solve the problem with probability at least $2/3$. **(15 points)**

**Problem 2.** We consider a model in the spirit of sparse recovery for graphs. Let $G = (V, E)$ be some undirected graph with vertices $V := \{1, 2, \ldots, n\}$ and unknown edges. We can access $G$ by querying it as follows: we specify a cut $(S, \bar{S})$ as our query and receive the *number* of edges in this cut in $G$ as the answer. Our goal is to design an algorithm for finding a (maximal) spanning forest of $G$.

(a) Design an algorithm that given two disjoint sets $A$ and $B$ of vertices in $G$, uses $O(1)$ queries and determine if there is any edge in the graph between $A$ and $B$. **(10 points)**

(b) Design an algorithm that uses the above subroutine to solve the following problem: given a cut $(S, \bar{S})$ in $G$, the algorithm uses $O(\log n)$ queries and finds a single edge in the cut. **(10 points)**

(c) Design an algorithm that uses the above subroutines to find a (maximal) spanning forest of $G$ using $O(n \log n)$ queries. **(5 points)**

**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 oracle 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:

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

Note that you need to pick $\eta$ 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 $O(m \cdot \text{poly}(\frac{1}{\varepsilon}, \log(m)))$ time. **(+15 points)**

**Problem 5** (**Extra Credit**)**.** In this problem, we examine an interesting and natural (in hindsight) property of MWU, wherein the weights on the constraints can form near-optimal dual solutions as well.

Recall the MWU approach used in Lecture 18 for the maximum bipartite matching problem (for any of the oracles used in that lecture). Consider the iteration wherein the objective value of the returned solution for the oracle is *smallest* and let $\{w_v\}_{v \in V}$ denote the MWU weights on the vertices in this iteration. Prove that there exists a fixed choice of $k$ (which you should choose also), such that if we pick the variables $y_v = k \cdot w_v$ for every $v \in V$, then, the variables $\{y_v\}_{v \in V}$ form a $(1 + O(\varepsilon))$ fractional vertex cover of $G$.

**(+15 points)**