CS 466/666: Algorithm Design and Analysis	**University of Waterloo: Fall 2025**

Homework 3

Due: Thursday, November 13, 2025

Problem 1. Let $G = (V, E)$ be any graph with integer weights $w(e)$ on edges $e \in E$. For this question, we allow the edges to have negative weight. Consider the following approximation algorithm for the maximum weight matching problem, namely, finding a matching $M \subseteq E$ that maximizes $\sum_{e \in M} w(e)$.

1. If all edges in G have a non-positive weight, return $M = \emptyset$ and terminate.
2. Pick an arbitrary edge $e \in E$ with $w(e) > 0$. Create the new graph $G' := G - e$ with weights $w'(f) = w(f)$ for every edge f not incident on e and $w'(f) = w(f) - w(e)$ for every edge f incident on e (basically, we are subtracting $w(e)$ from the weights of all edges incident on e in G').
3. Run the algorithm recursively on G' to obtain a matching M'. If both endpoints of the edge e are unmatched, return $M := M' \cup \{e\}$, otherwise, return $M = M'$ as the final answer.

Prove that this algorithm outputs a 2-approximation to the maximum weight matching problem.

(25 points)

Problem 2. Design a randomized algorithm that given a 3-colorable graph G, with high probability, outputs a proper coloring of G with $\widetilde{O}(n^{1/4}) = O(n^{1/4} \cdot \text{polylog}(n))$ colors. **(25 points)**

Problem 3. 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 4. We re-examine the graph sketching technique of Lecture 14 in this question. Recall the setting: We have a graph $G = (V, E)$ with $V := [n]$ and there is a player for each vertex $v \in V$ who only sees the neighbors of v. The players have access to the same shared source of randomness. Simultaneously with each other, they each send a message of length polylog(n) bits to a referee (who has no input but has access to the same shared randomness), and the referee outputs a solution to the problem. We require the solution to be correct with high probability.

In the class, we designed an algorithm for finding a spanning forest of the input graph. In this question, we examine two other problems in this model.

1. We say a graph $G = (V, E)$ is k-(edge)-connected if one needs to remove at least k edges from G in order to make G disconnected. In other words, the minimum cut of G has at least k edges. We like to design a graph sketching algorithm for this problem wherein each player sends a messages of length $O(k \cdot \text{polylog}(n))$ to the referee.

   Consider the following the process: Pick a spanning forest F_1 of G, then spanning forest F_2 of $G \setminus F_1$, then F_3 of $G \setminus (F_1 \cup F_2)$, and so on and so forth until picking F_k. Prove that G is k-connected if and only if $F_1 \cup F_2 \cup \ldots \cup F_k$ is k-connected.

   Then design an algorithm using ℓ_0-samplers that allows the players to send $O(k \cdot \text{polylog}(n))$-length messages to the referee so that the referee can find the spanning forests $F_1, \ldots, F_k$.

   (12.5 points)

   Hint: Recall that ℓ_0-samples are *linear*: can you obtain a sketch $A \cdot (G \setminus F)$ from the sketch of $A \cdot G$ if you know F already?

2. Let us now switch to finding an *approximate* MST. Suppose every edge $e = (u, v) \in E$ of the graph G also as an integer weight $w(e) \geqslant 1$ which is known only to players on vertices u and v. The players are also all given a parameter $\varepsilon > 0$. They should each send a message of size $\text{poly}(1/\varepsilon, \log(n))$ to the referee and the referee with high probability outputs a spanning T such that weight of T is at most $(1 + \varepsilon)$ times the weight of the MST of G. **(12.5 points)**

   Hint: The approach in Lecture 14 was to implement Boruvka's algorithm by finding *any* edge out of each contracted vertex. Can you generalize the approach to find a $(1 + \varepsilon)$-approximate minimum weight edge instead? You should then be able to run Boruvka's algorithm to find an approximate MST not just a spanning tree.

Problem 5 (**Extra Credit**)**.** Design a randomized algorithm that given a 3-colorable graph G, with high probability, outputs a proper coloring of G with $O(n^{0.2499})$ colors. **(+10 points)**

Problem 6 (**Extra Credit**)**.** Design a polynomial time algorithm for the following problem: given a *directed* graph $G = (V, E)$ and two vertices s, t find any path from s to t that is <u>not</u> a shortest path(!).

(+10 points)

Hint: This is a really hard question, or, is it?