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

Homework 3

Due: Thursday, November 14, 2024

Problem 1. In the **Knapsack problem**, we have a set of n items where item i has a weight w_i and a profit p_i. We are additionally given a knapsack of total capacity W. The goal is to pick a set of items with maximum total profit, subject to their total weight being less than W, i.e., find

$$\arg\max_{S\subseteq[n]} \sum_{i\in S} p_i \quad \text{subject to} \quad \sum_{i\in S} w_i \leqslant W.$$

This is a well-known NP-hard problem and admits a textbook dynamic programming solution. In this question, we study approximation algorithms for this problem.

1. Write an LP relaxation for Knapsack and justify its correctness. **(5 points)**
2. Design a simple greedy algorithm for solving the LP relaxation of the Knapsack in $O(n \log n)$ time. **(10 points)**

   Note that in this part, you are <u>not</u> solving the original Knapsack problem, but rather the *fractional* knapsack problem wherein you can pick a fraction of an item (and its weight and profit will also become the same fraction of their original value).
3. Design an algorithm for the original Knapsack problem that given any fractional solution x to the LP relaxation, returns a feasible S to Knapsack with total profit at least

$$\left(\sum_{i=1}^{n} x_i \cdot p_i\right) - \max_{i=1}^{n} p_i.$$

   (10 points)
4. Use the previous parts to design a 2-approximation algorithm for Knapsack in $O(n \log n)$ time. **(5 points)**

Problem 2. 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 3. Consider the following online problem. Unfortunately, you have lost a bet to your (most unreasonable) friend and now have to accommodate them: every day, you either have to buy lunch for your groups of friends at a cost of L dollars, or on that day, you just give up and pay a total of G dollars to your friend to end your misery (at least in this bet). However, your friend can also decide at any day that they are done with this bet and no longer want to continue this and you have no idea on if or when they do this[1].

1. Design a *deterministic* strategy such that the total money you spend is at most $\left(2 - \frac{L}{G}\right)$ times the minimum amount you had to spend if you knew which day your friend decides to stop this bet.

   (10 points)

2. Prove that the above bound is optimal, i.e., *any* deterministic strategy that you choose cannot results in a ratio less than $\left(2 - \frac{L}{G}\right)$ in the above bound. **(10 points)**

You may assume in this question that L divides G to simplify the math.

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 $\text{polylog}(n)$ 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.

[1] As I told you, they are particularly unreasonable ...

Problem 5 (**Extra Credit**)**.** Recall the online bipartite matching problem from Lecture 12. In this question, we prove that the $(1 - 1/e)$-competitive ratio obtained for this problem in the class is optimal.

First, prove that any randomized algorithm here implies a deterministic algorithm for *fractional* online matching with the same competitive ratio. In the fractional online matching, we only need to maintain a fractional matching instead of an integral one; when a vertex v arrives, we can increase the fractional values of edges incident on v (subject to not violating fractional matching constraint), but after v passes, we can no longer change these values. **(+10 points)**

Second, prove that any deterministic algorithm for fractional online matching cannot achieve a better than $1 - 1/e$-competitive ratio on the following family of graphs: the first arriving vertex is connected to every vertex on the other side, the second vertex is connected to all but one randomly chosen vertex, the third vertex is connected to all but the previously excluded vertex and another randomly chosen one, and so on and so forth. See Figure 1 for an example of this graph. **(+10 points)**

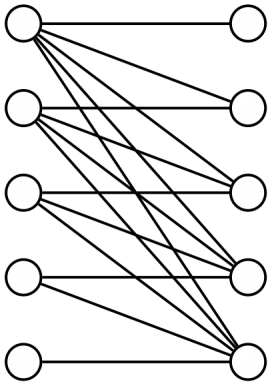

Figure 1: An example of the graph constructed against algorithms for online fractional matching. Here, there are only 5 vertices on each side but in this problem, we are interested when the number of vertices n (on each side) goes to infinity.

Problem 6 (**Extra Credit**)**.** Design a randomized algorithm for Problem 3 such that the expected money you spend is at most $\frac{e}{e-1}$ times the minimum amount you had to spend if you knew which day your friend decides to stop this bet.

To be able to design a *randomized* algorithm and analyze it properly, you can assume that your friend has decided ahead of time which day to stop the bet and does not change their decision based on what your strategy is in a given day. **(+15 points)**