Recent Developments in Algorithms, Spring 2025
Homework 3

Q1: Online Vertex Cover

Consider the problem of finding a vertex cover in a node-weighted graph, where the edges of the graph arrive online. Formally, you start with a vertex set V of n vertices, but no edges (yet). Each vertex has a non-negative weight w_v. At each timestep $t = 1, 2, \ldots$, an edge $e_t = \{u_t, v_t\}$ arrives and has to be covered (if not yet covered already). Formally, if $E_t = \{e_1, e_2, \ldots, e_t\}$ are the edges seen by time t, and $G_t = (V, E_t)$, then you want to maintain a set C_t such that C_t is a vertex cover for G_t, and moreover, $C_{t-1} \subseteq C_t$. (Assume $C_0 = \emptyset$.) You want to ensure that at any time t, the weight of your solution $w(C_t) := \sum_{v \in C_t} w_v$ is at most a constant times OPT_t, the cost of the optimal vertex cover for G_t.

When an edge e_t arrives that is not covered by C_{t-1}, here are four different things you could do:

1. add both its endpoints to get $C_t = C_{t-1} \cup e_t$.
2. add its cheaper endpoint, and also its heavier endpoint in case it costs at most two times the cheaper endpoint.
3. add u_t with probability $\frac{w_{v_t}}{w_{u_t}+w_{v_t}}$ and v_t with the remaining probability.

For the first two algorithms, give examples showing on which the competitive ratio of the algorithm is unbounded, as $t \to \infty$. For the last algorithm, show that $\mathbb{E}[w(C_t)] \leq 2OPT_t$.

Q2: A Special Set Cover Problem

Suppose you have a rooted tree $T = (V, E)$. Define the following set system $(E, \mathcal{F})$:

- The elements of the set system are the edges E, and
- Each set $S_{u,v}$ in $\mathcal{F}$ is specified by some pair of non-adjacent vertices $u, v \in E$, and this set contains all the edges on the unique path between u and v in the tree.

Some people like to think of the sets being given by another collection of non-tree edges $F \subseteq \binom{V}{2}$, with the sets being $\mathcal{F} = \{S_f \mid f \in F\}$.

1. Suppose each set in $\mathcal{F}$ is a subset of some root-leaf path. (Call such sets "monotone".) Give an dynamic-programming based algorithm to find the min-cost solution.
2. Now suppose the sets are not all monotone. For each non-monotone set u, v, let x be its least-common ancestor in the rooted tree. Replace the set $S_{u,v}$ with two monotone sets $S_{u,x}$ and $S_{v,x}$. Show that an optimal solution on this new monotone instance is a 2-approximation to the set cover problem on the original non-monotone instance.

Q3: Shortest Paths Problem

This problem is on Aaron's lecture from March 20. Say you have a graph G with integer weights polynomial in n. The "Restricted Graph Theorem" from lecture notes states that we can assume we are working with a *restricted* graph, which has three properties:

- all weights are ≥ -1 (and still integral)
- All weights are $\leq n$ (and still integral)
- Every cycle C has $w(C) \geq |C|$.

In this HW problem we will show why we can assume the first two properties. The third property is a big harder to justify, so we won't go into it.

Note: You can attempt part 2 even if you weren't successful with part 1. In fact part 2 is probably the easier problem.

Part 1

Let $T_{\text{SSSP}}(W, m, n)$ be the worst-case time required to solve negative-weight shortest paths in a graph with m edges, n vertices, and integral weights $\geq -W$.

The Problem: The goal of this problem is to prove that

$$T_{\text{SSSP}}(W, m, n) = O(\log(W) \cdot T_{\text{SSSP}}(1, m, n)) + \tilde{O}(m \log(W)). \tag{1}$$

I will tell you the first step. What you should do is prove the following:

$$T_{\text{SSSP}}(W, m, n) \leq T_{\text{SSSP}}(W/2, m, n) + T_{\text{SSSP}}(2, m, n) + \tilde{O}(m). \tag{2}$$

Combined with the fact shown in class that $T_{\text{SSSP}}(2, m, n) \sim T_{\text{SSSP}}(1, m, n)$, it's easy to see that Equation 2 implies Equation 1. So all you need to do is prove the recurrence in Equation 2.

1. WRITING NOTE: for this problem the analysis is pretty self-evident once you have the right algorithm. So feel free to keep the analysis short, *but make sure to include detailed pseudo-code for the algorithm!*
2. Hint 1: Feel free to assume that W is a power of 2.
3. Hint 2: The simplest proof I can think of uses the fact we proved in class that $T_{\text{SSSP}}(2, m, n) = O(T_{\text{SSSP}}(1, m, n))$.

4. Hint 3: You need to solve an instance where $w(e) \geq -W$ for all edges. First imagine that you were extremely lucky and all your weights were integer multiples of $W/2$. Show that in this lucky case, there is an extremely simple way to solve the problem in time $T_{\text{SSSP}}(2, m, n)$ (you don't even need price functions for this one). Now, if you your weights are not integer multiples of $W/2$, how can you still use the above idea? You will need to use price functions for this non-lucky case.

Part 2

Say you have an algorithm A that solves negative-weight SSSP in $\tilde{O}(m)$ time when all weights are integral and you are also guaranteed that

$$-1 \leq w(e) \leq n \qquad \forall\ e \in E$$

Using A as a blackbox, show that you also solve the problem in $\tilde{O}(m)$ time under the weaker assumption that weights are integral and $w(e) \geq -1$ for all $e \in E$ (i.e. you no longer have the assumption $w(e) \leq n$).

WRITING NOTE: as for part 1, you should include detailed pseudo-code for the algorithm.