Assignment 5

Due February 14 at the beginning of lecture

Question 1 (8pt) When the edge costs are not distinct, there may be more than one minimum spanning tree. The execution, and hence the output, of Kruskal's algorithm depend on the ordering of edges that have the same cost—recall that Kruskal's algorithm works by first sorting the edges in nondecreasing order of cost.

The question here is, for any graph G and cost function c, and any minimum spanning tree T of G, is there an execution of Kruskal's algorithm that outputs T? Either prove this, or give a counterexample (i.e., exhibit a graph G with a cost function c and a minimum spanning tree T such that no matter how edges of G are ordered, Kruskal's algorithm does not produce T as the output).

Solution We will prove the statement. That is, given a minimum spanning tree T of a graph G, we will show that there is an ordering of the edges of G in non-decreasing order of cost so that T is the output of running the Kruskal's algorithm on G. Following a technique given in class for proving the correctness of Kruskal's algorithm (when the costs might not be distinct), we will increase the costs by a tiny amount such that T becomes the unique minimum spanning tree, and such that ordering the edges in increasing order of the new cost also orders the edges in non-decreasing order of the original cost. Because T is the unique minimum spanning tree, it must be the output of Kruskal's algorithm (running with the new cost function). Then we will argue that with the same ordering of the edges, the Kruskal's algorithm under the original cost function must also output T.

Assume without loss of generality that all costs are positive integers. Now list all edges in G as $e_1, e_2, \ldots, e_m$, where the first $n-1$ edges are the edges of T. Increase the cost of the edge e_i by an additional amount of $i\epsilon$, for $\epsilon = \frac{1}{m+1}$. Denote the new cost function by c'. Then note that the total increase in cost for tree T is

$$c'(T) - c(T) = \epsilon + 2\epsilon + \ldots + (n-1)\epsilon = \frac{(n-1)n}{2}\epsilon$$

and the increase for any other spanning tree is more than $(n-1)n/2\epsilon$, because the sum of any sequence of $n-1$ distinct positive integers is at least $(n-1)n/2$. Therefore T is the unique minimum spanning tree of G under the new cost function.

Now order the edges in increasing order of the new costs:

$$e'_1, e'_2, \ldots, e'_m$$

Because T is the unique minimum spanning tree (under the new cost function c'), Kruskal's algorithm under c' outputs T. Now since the cost for each edge increases by at most $\frac{m}{m+1}$ which is less than 1, this ordering is also nondecreasing under the original cost function. In other words,

$$c(e'_1) \leq c(e'_2) \leq \ldots \leq c(e'_m)$$

We will argue that with this ordering of edges, the Kruskal's algorithm on the original cost also outputs T.

We prove by induction on the step of the Kruskal's algorithms that the two executions (under two cost functions c and c') make the same decision. For the first step this is true because both selects e'_1. Now suppose that they have agreed upto (including) the first i steps. In step $i+1$ for the i-th step, edge e'_{i+1} is selected if and only if it doesn't create cycle with the edges selected so far. But the edges selected so far are the same in the two executions, therefore the two executions make the same decision about whether e'_{i+1} is selected or not.

Question 2 (10pt) Consider the following problem, called Weighted Job Scheduling (WJS) here. In this problem there is a single processor that can only process one job at a time. There is a set of n jobs to be processed, the i-th job requires a contiguous period of duration t_i and it has weight w_i. All t_i and w_i are positive integers. Timing starts at 0. For a schedule of the jobs, let d_i denotes the finish time of job i. Then the "weighted duration" of the schedule is defined to be

$$\sum_{i=1}^{n} w_i d_i$$

The WJS problem is to obtain a schedule of all jobs with smallest possible weighted duration.

Give a greedy algorithm to solve this problem. Clearly state the criterion that you optimize for each step, give the pseudo-code for the algorithm, and prove its optimality.

Solution The criterion for our algorithm is to schedule job with smallest ratio weight/duration first. The algorithm is as follows.

1. Sort the jobs so that
$$\frac{w_1}{t_1} \geq \frac{w_2}{t_2} \geq \ldots \geq \frac{w_n}{t_n}$$
2. Schedule the jobs in this order

Proof of optimality. Let $\mathcal{O}$ be any optimal schedule, and $\mathcal{S}$ be the output of our algorithm. We show that they have the same weighted duration.

First, we show that in $\mathcal{O}$ there cannot be any "inversion", i.e., a pair of two jobs i and j such that $\frac{w_i}{t_i} > \frac{w_j}{t_j}$ but i is scheduled before j. We prove this by contraction. So suppose that such an inversion exists in $\mathcal{O}$. Then there must be an inversion u, v where $\frac{w_u}{t_u} > \frac{w_v}{t_v}$ and u is scheduled *immediately* before v. (This is similar to the argument given in lecture for the Job Scheduling problem. To repeat the argument: let (i, j) be an inversion in $\mathcal{O}$. Look at the schedule $\mathcal{O}$, start from job i going to the right there must be the first time where the weight/duration ratio strictly decreases, i.e., we meet two jobs u and v as claimed.) Now consider the schedule $\mathcal{O}'$ which is obtained from $\mathcal{O}$ by swapping the jobs u and v. Let W denote the weighted duration of $\mathcal{O}$ and W' denote the weighted duration of $\mathcal{O}'$. Then

$$\begin{aligned} W' &= W - \big(w_u t_u + w_v(t_u + t_v)\big) + \big(w_v t_v + w_u(t_u + t_v)\big) \\ &= W - w_v t_u + w_u t_v \\ &> W \end{aligned}$$

(The last inequality is because $\frac{w_u}{t_u} > \frac{w_v}{t_v}$.) This violates the fact that $\mathcal{O}$ is optimal, a contradiction.

So now we have established that in $\mathcal{O}$ there are no inversions. Thus $\mathcal{O}$ and $\mathcal{S}$ differ only in the ordering of jobs that have the same weight/duration ratio. Note that if we swap any two jobs u, v

in $\mathcal{O}$ that have the same ratio, then the same derivation as above shows that the weighted duration of the new schedule is the same as that of $\mathcal{O}$ (because $w_v t_u = w_u t_v$). Therefore we can permute the jobs in $\mathcal{O}$ with the same weight/duration ratio to obtain $\mathcal{S}$ without affecting the weighted duration. This shows that $\mathcal{S}$ and $\mathcal{O}$ have the same weighted duration. So $\mathcal{S}$ is an optimal solution.