Yezheng Li

Question (4-17) *Consider the following variation on the Interval Scheduling Problem. You have a processor that can operate 24 hours a day, every day. People submit request to run daily jobs on the processor. Each such job comes with a start time and an end time; if the job is accepted to run on the processor, it must run continuously, every day, for the period between its start and end times. (Note that certain jobs can begin before midnight and end after midnight; this makes for a type of situation different from what we saw in the Interval Scheduling Problem.)*

Given a list of n such jobs, your goal is to accept as many jobs as possible (regardless of their length), subject to the constraint that the processor can run at most one job at any given point in time. Provide an algorithm to do this with a running time that is polynomial in n. You may assume for simplicity that no two jobs have the same start or end times.

Answer *Here is the algorithm where T_{all} represent the 24–hour time.*

1. *Input $I_1, \ldots, I_N$, N intervals/ requests of time, represented by $I_j \doteq (t_{start}, t_{end})$;*
2. *For $j = 1, \ldots, N$*
   (a) *Denote by S_j, the set of intervals/ requests selected; by A_j, the set of intervals/ requests still available;*
   (initialization) $S_j \leftarrow \{I_j\}$, $A_j \leftarrow \{I_k : k = 1, \ldots, N, I_k \cap I_j = \emptyset\}$;
   (b) *Notice $T_{all} \setminus I_j$ can be regarded as a line when rest of the intervals are to be scheduled selectively. According to classic interval/request scheduling problem, S, A can be modified by first-finish-first schedule routine.*
3. *Compare $\#S_j$ and select $j_0 \in arg\max_{j=1,\ldots,N} \#S_j$, namely the index j_0 for which we can schedule with as many intevals/ requests as possible;*
4. *Output S_{j_0}.*

Running time:

Question (4-20) *Every September, somewhere in a far-away mountainous part of the world, the county highway crews get together and decide which roads to keep clear through the coming winter. There are n towns in this county and the road system can be viewed as a (connected) graph $G = (V, E)$ on this set of towns, each representing a road joining two of them. In the winter, people are high enough up in the mountains that they stop worrying about the length of roads and start worrying about their altitude – this is really what determines how difficult the trip will be.*

So each road – each edge e in the graph – is annotated with a number a_e that gives the altitude of the hightest point on the road. We will assume that no two edges have exactly the same altitude value a_e. The height of a path P in the graph is then the maximum of a_e over all edges e on P. Finally, a path between towns i and j is declared to be winter-optimal if it achieves the minimum possible height over all paths from i to j.

The highway cres are going to select a set $E' \subset E$ of the roads to keep clear through the

winter; the rest will be left unmaintained and kept off limits to travellers. They all agree that whichever subset of roads E' they decide to keep clear, it should have the property that (V, E') should be no greater than it is in the full graph $G = (V, E)$. We will say that (V,E') is a minimum-altitude connected subgraph if it has this property.

Given that they are going to maintain this key property, however, they otherwise want to keep as few roads clear as possible. One year, they hit upon the following conjecture:

- *The minimum spaning tree of G, with respect to the edge weights a_e, is a minimum-altitude connected subgraph.*

(In an earlier problem, we claimed that there is a unique minimum spanning tree when the edge weights are distinct. Thus, thanks to the assumption that all a_e are distinct, it is okay for us to speak of the minimum spanning tree.)

Initially, this conjecture is somewhat counterintuitive, since the minimum spanning tree is trying to minimize the sum of the values a_e, while the goal of minimizing altitude seems to be asking for a fairly different thing. But lacking an argument to the contrary, they begin considering an even bolder second conjecture:

- *A subgraph (V, E') is a minimum–altitude connected subgraph if and only if it contains the edges of the minimum spanning tree.*

Note that this second conjecture would immediately imply the first one, since a minimum spanning tree contains its own edges.

So here is the question.

1. *Is the first conjecture true, for all choices of G and distinct altitudes a_e? Give a proof or a counterexample with explanation.*
2. *Is the second conjecture true, for all choices of G and distinct altitudes a_e? Give a proof or a counterexample with explanation.*

Answer *1. True. By exercise 4-8, the minimal spanning tree is unique and can be obtained by Kruskal's algorithm. The thing to show is that Kruskal's algorithm also produces a minimum-altitude connected subgraph. It suffices to show counterpart of lemma 4.17:*

> **Lemma (4.17')** *Let S be any subset of nodes that is neighter empty nor equal to all of V and let edge $e = (v, w)$ be the minimum–cost edge with one end in S and the other in $V - S$. There exists a minimum spanning tree containing the edge e.*

Proof *Suppose not, that is any spanning tree T containing e cannot be a minimal–altitude subgraph.*

Given ang minimal–altitude subgraph T_1. In order to show there exists a tree T_2 containing e that has at most the same altitude, an exchange argument is applied:

The crux is therefore to find an edge that can be successfully exchanged with e. Recall that the ends of e are $v \in S, w \in V - S$. That T_1 is connected infers there exists a path P in

T from v to w. Starting at v, suppose first node in $V - S$ of P in sequence $v \to w$ is w' and $v' \in S$ be previous node of w'. Denote $e' = (v', w')$, an edge of T_1 with one end in S and the other in $V - S$.

Exchange of e and e' produces $T_2 = (T_1 - \{e'\}) \cup \{e\}$. T_2 is also a spanning subgraph since:

(a) Clearly it is connected since (V, T_1) is connected and that any path in (V, T_2) that used the edge e' can now be "rerouted" in (V, T_2);

(b) It is also acyclic since the only cycle in $(V, T_2 \cup \{e'\}$ is the one composed of e and the path P and this cycle is not present in (V, T_2) due to the deletion of e'.

Note that $a_e \leq a_{e'}$ implies altitude of (V, T_2) is less equal to that of (V, T_1); specifically, (V, T_2) is also a minimal–altitude connected subgraph and contains e. This leads to a contradiction!

Therefore, there exists a minimum spanning tree containing the edge e.

2. *Not true. See the graph below.*

Question (4-21) *Let us say that a graph $G = (V, E)$ is a near-tree if it is connected and has at most $n + 8$ edges, where $n = |V|$. Give an algorithm with running time $O(n)$ that takes a near-tree G with costs on its edges, and returns a minimum spanning tree of G. You may assume that all the edge costs are distinct.*

Question (4-28)

Answer