ADS Tutorial 4 – Solutions*

Instructor: Heng Guo Tutors: Georgios Kalantzis and Dhairya Mangla

October 22, 2026

Problem 1

Let $G = (V, E)$ be a connected graph with positive edge costs, and let G_T be a Minimum Spanning Tree (MST) on G. Let G' be a connected graph that is the same as G, but with different edge costs c'_e rather than c_e for each edge $e \in E$. In which of the following cases will G_T still be a minimum spanning tree of the connected graph G'?

(a) When $c'_e = c_e + 17$.

(b) When $c'_e = 17c_e$.

(c) When $c'_e = c_e^2$.

(d) All of the above.

(e) None of the above.

Solution

All of the above. The key observation is that none of the transformations above alters the relative ordering of the edges; i.e.,

$$c_1 < c_2 < \cdots < c_m \iff c'_1 < c'_2 < \cdots < c'_m,$$

for each of the definitions of c'_e above. We showed in lecture that Kruskal's algorithm produces a MST by repeatedly adding the next cheapest edge that doesn't produce a cycle. Thus, running Kruskal's algorithm on the set of altered edges with costs $\{c'_e\}_{e\in E}$ produces the same MST as running it on the set of edges with costs $\{c_e\}_{e\in E}$. If the costs are distinct, the MST is unique (Problem 2(b)), so this tree is G_T.

If some costs are equal, there can be several MSTs, and G_T need not be the one that Kruskal's algorithm finds. So break ties in favour of G_T: sort the edges by cost and, among equal costs, put the edges of G_T first. With this order, Kruskal's algorithm returns G_T itself. Suppose instead it returns $K \neq G_T$, and let e be the first edge in the order that lies in only one of the two trees.

- If $e \in G_T$, the algorithm rejected e because it closes a cycle with earlier edges of K. Those come before e, so they are in G_T too, and G_T would contain a cycle.
- If $e \in K$, adding e to G_T closes a cycle with an edge $f \notin K$. That f is in G_T, so it comes after e; and since $f \in G_T$ and $e \notin G_T$, the tie-breaking rules out $c_f = c_e$, so $c_f > c_e$. Swapping f for e would make G_T cheaper.

The same order also sorts the costs c', so on c' Kruskal's algorithm makes the same choices and returns G_T again. By Lecture 7, Kruskal's algorithm with any consistent tie-breaking returns a minimum spanning tree. So G_T is a minimum spanning tree of G'.

*The solutions contain additional explanations that are not necessary, if you were to answer such a question in an exam.

Problem 2

In this problem we will consider the *uniqueness* of MSTs on connected graphs.

(a) First consider connected graphs in which the edge costs might not be distinct. Draw an example of such a graph that has two different MSTs.

(b) Now consider any connected graph G for which the edge costs are all distinct. Prove that there is a unique MST on G.

Solution

(a) Example: A triangle with equal edge costs has 3 distinct MSTs.

(b) Proof by contradiction: Suppose our graph contains two distinct MSTs, T_1 and T_2. Consider some edge $\{u, v\}$ that's in T_1 but not in T_2. (Since the MSTs are distinct, such an edge must exist.) Let $T_1(u)$, $T_1(v)$ be the two components of T_1 after deleting the edge $\{u, v\}$, with $u \in T_1(u)$ and $v \in T_1(v)$, and consider the path $p_{u,v}(T_2)$ between u and v in T_2. Let E' be the edges of $p_{u,v}(T_2)$ that cross the cut $T_1(u)$, $T_1(v)$. (We are guaranteed that $|E'| \geq 1$: the path starts at u, on one side of the cut, and ends at v, on the other side.)

Take any $e \in E'$. As $\{u, v\} \notin T_2$, we have $e \neq \{u, v\}$, and the costs are distinct, so there are two possible cases:

- Suppose $c_e > c_{uv}$. Then define $T_2' = (T_2 \setminus \{e\}) \cup \{\{u, v\}\}$. This is a spanning tree of cost strictly less than T_2. Contradiction!
- Suppose $c_e < c_{uv}$. Then define $T_1' = (T_1 \setminus \{\{u, v\}\}) \cup \{e\}$. This is a spanning tree of G with cost strictly less than T_1. Contradiction!

In either case we prove that one of T_1, T_2 was not a MST.

Problem 3*

Suppose you are given a connected graph $G = (V, E)$ with edge costs which are all distinct, with $|V| = n$ and $|E| = m$, and a specified edge $e^* \in E$. Design an algorithm with running time $O(m + n)$ which returns "yes" if e^* is contained in a MST of G and "no" otherwise.
To prove the correctness of your algorithm, you may first want to prove the following statement:

> *An edge $e = \{v, w\}$ does not belong to a MST of G if and only if there is a path from v to w which consists entirely of edges that have smaller cost than e.*

Hint: You may want to use the Cut Property and the Cycle Property to prove the statement.

Solution

[From KT chapter 4, solved exercise 3.]

From Lecture 7, we know of two rules by which we can conclude whether an edge e belongs to a minimum spanning tree: the Cut Property says that e is in every minimum spanning tree when it is the cheapest edge crossing from some set S to the complement $V \setminus S$; and the Cycle Property says that e is in no minimum spanning tree if it is the most expensive edge on some cycle C. Let's see if we can make use of these two rules as part of an algorithm that solves this problem in linear time.

Both the Cut and Cycle Properties are essentially talking about how e relates to the set of edges that are *cheaper* than e. The Cut Property can be viewed as asking: Is there some set $S \subseteq V$ so that in order to get from S to $V \setminus S$ without using e, we need to use an edge that is more expensive than e? And if we think

about using the cycle C in the statement of the Cycle Property, going the "long way" around C (avoiding e) can be viewed as an alternate route between the endpoints of e that only uses cheaper edges.

Putting these two observations together suggests that we should try proving the following statement.

Claim: Edge $e = \{v, w\}$ does not belong to a minimum spanning tree of G if and only if v and w can be joined by a path consisting entirely of edges that are cheaper than e.

Proof. First suppose that P is a v-w path consisting entirely of edges cheaper than e. If we add e to P, we get a cycle on which e is the most expensive edge. Thus, by the Cycle Property, e does not belong to a minimum spanning tree of G.

On the other hand, suppose that v and w cannot be joined by a path consisting entirely of edges cheaper than e. We will now identify a set S for which e is the cheapest edge with one end in S and the other in $V \setminus S$; if we can do this, the Cut Property will imply that e belongs to every minimum spanning tree. Our set S will be the set of all nodes that are reachable from v using a path consisting only of edges that are cheaper than e. By our assumption, we have $w \in V \setminus S$. Also, by the definition of S, there cannot be an edge $f = \{x, y\}$ that is cheaper than e, and for which one end x lies in S and the other end y lies in $V \setminus S$. Indeed if there were such an edge f then since the node x is reachable from v using only edges cheaper than e, the node y would be reachable as well. Hence e is the cheapest edge with one end in S and the other in $V \setminus S$, as desired, and so we are done. □

Given this fact, our algorithm is now simply the following. Let $e^* = \{v, w\}$. We form a graph G' by deleting from G all the edges of cost greater than c_{e^*} (as well as deleting e^* itself). We then use breadth-first or depth-first search (Lecture 7; time $O(n + m)$) to determine whether there is a path from v to w in G'. The claim says that e^* belongs to a minimum spanning tree if and only if there is no such path.

The running time of this algorithm is $O(m + n)$ to build G' and $O(m + n)$ to test for a path from v to w.

Problem 4

Consider an alternative algorithm for computing an MST G_T on a connected graph $G = (V, E)$ with distinct edge costs. At iteration t of the algorithm, let G_T^t be the graph consisting of the set of nodes V and the set of edges that have been added by the algorithm to the developing MST in iterations $1, \ldots, t-1$. Let $\mathcal{C}^t$ be the set of connected components of G_T^t. For each connected component $C \in \mathcal{C}^t$, add to the MST the edge with the smallest cost $e = \{v, u\}$ where v is in C and u is not in C. Update the graph to G_T^{t+1} and the set of connected components to $\mathcal{C}^{t+1}$, and repeat until there is a single connected component. Before the first iteration, G_T^1 contains no edges and $\mathcal{C}^1 = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$, i.e., every node is a connected component on its own.
Show that this algorithm terminates in $O(m \log n)$ time and that in the end it produces a MST of G. It might be useful to first prove the following statement, using the Cut Property for the proof:

> *Let $u \in V$ be any node in G. The MST of G must contain edge $\{u, w\}$, which has the minimum cost amongst all edges incident to u.*

Note: This algorithm was first proposed by Otakar Borůvka in 1926, in response to a request to construct an electrical network connecting several cities using the least amount of wire. The algorithm was rediscovered multiple times, in the late 1930s, the early 1950s and the mid-1960s.

Solution

First, to show the claim that the MST of G contains the minimum-cost edge incident to u for all vertices $u \in V$, we simply apply the Cut Property to the cut $\{u\}$ and $V \setminus \{u\}$. Now we argue that the algorithm described above should indeed produce an MST. To prove this, we show that an edge is added by the above algorithm if and only if it is in the MST; i.e., that any edge added by the above algorithm is in the MST, and, moreover, that the algorithm eventually adds all the edges in the MST.

For the first direction, we note that at the first step, when the connected components are simply individual nodes, the edges added in an iteration of the algorithm all belong to the MST by the claim we proved. For subsequent iterations t, the algorithm adds to the tree the smallest edge leaving each connected component. Consider some connected component $C_i \in \mathcal{C}^t$, and let e be the smallest edge across the cut $(C_i, V \setminus C_i)$. We know that algorithm above will add e to the tree it's constructing. And by the Cut Property, this edge belongs to the original graph's MST. Thus, all the edges added by the algorithm are in the original graph's MST. To show that all the necessary edges have been added, we simply note that the algorithm stops as soon as there is a single connected component, at which point the constructed tree spans the original graph.

The algorithm has runtime $O(m \log n)$. To see this, we observe that each iteration of the algorithm will reduce the number of connected components by at least half, since the algorithm chooses at least one edge from each connected component and, in the worst case, each edge chosen is chosen by another component. (I.e., the smallest edge in C_1 is also the smallest edge in some other component C_i; and so on for every edge that the algorithm selects.) Therefore, we need $O(\log n)$ iterations to get to a single connected component, and each step considers $O(m)$ edges. This gives us an overall runtime of $O(m \log n)$.