Solution Set

CIS 121—Data Structures and Algorithms—Fall 2020

Minimum Spanning Trees—Monday, November 9 / Tuesday, November 10

Readings

- Lecture Notes Chapter 21: Minimum Spanning Trees

Problems

Problem 1

Does Kruskal's algorithm work on a graph with negative weights? How about Prim's?

Solution

Yes. Kruskal's algorithm will still select the lowest weight edges in order (more negative edges first). Similarly, since Prim's is selecting the lowest weight edge that crosses the cut (from the cut property), the algorithm works with negative edge weights.

Interestingly, you can also find the most negatively weighted edge in the graph, and add that weight to every edge in the graph to make all the edge weights non-negative. Think about why adding a constant to all the edge weights broke Dijkstra's algorithm but not Prim's, despite their similarities.

Problem 2

Say we have some MST, T, in a positively weighted graph G. Construct a graph G' where for any weight $w(e)$ for edge e in G, we have weights $(w(e))^2$ in G'. Does T still remain an MST in G'? Prove your answer. Now if G also had negative weights, would your answer change from the previous part? Prove your answer.

Solution

If G only has positive weights, then this claim holds. Proof by contradiction: assume the claim does not hold and let T' be an MST of G'. Because $T \neq T'$, it must be that for some cut in the graph T chooses edge e while T' chooses some other edge e' such that $w(e)^2 \neq w(e')^2$, and $w(e')^2 < w(e)^2$ (since we assume T is no longer an MST in G'). However, since we know that T is a proper MST in G, that means that the edge e must have been the minimum edge spanning that cut in G, and therefore $w(e) \leq w(e')$. It is not possible for both of these statements to be true if we have positive edge weights, so we have reached a contradiction.

If G can have negative weights, then it is very simple to construct a graph such that the claim does not hold. For example, a graph with three vertices and edges weights of -1, -2, and -3.

Note that in general, any change to the edge weights that preserves their relative ordering suffices."

Problem 3

Imagine we have a graph G where all edge weights are equal. Design an algorithm to efficiently find an MST of G. Analyze the running time.

Solution

The intuition here is that we can find any spanning tree, and it will be the minimum spanning tree. Thus, our optimal algorithm would be to run DFS and only keep track of the tree edges (so we don't introduce any cycles). Notice at any step we can choose any edge, since the edge weights are all equal. You can do a similar thing with BFS. The running time is therefore $O(|E| + |V|)$.

Problem 4

Suppose that we have found an MST T of a graph G, but soon after, we are told that an edge not in T has a lower weight than we at first thought, and as such our MST is now invalid. Is it guaranteed that we can fix our tree by removing an edge and adding a different one? If so, explain how. If not, provide a counterexample.

Solution

If we can be sure that our MST is invalid because of this change, it follows that the modified edge e exists in all possible MSTs of the new graph. Let us add e to our MST. This has now created a cycle, and we no longer have a tree. We may fix this by removing a particular edge.
The cycle must have some maximum-weight edge. This is the edge that we would not have encountered in our MST creation in the new graph, as it will be the last of the edges in the cycle to be added. Since we are given that the new edge is necessary, the maximum-weight edge must be distinct from e. Thus we may remove this edge, after which we will again have a valid MST.