CIS 1210—Data Structures and Algorithms—Fall 2024

Minimum Spanning Trees—Tuesday, November 5 / Wednesday, November 6

Readings

- Lecture Notes Chapter 21: Minimum Spanning Trees
- Lecture Notes Chapter 22: Union Find

Review: Minimum Spanning Trees (MSTs)

A spanning tree of a graph $G = (V, E)$ is a subgraph that is a tree and which "spans" (includes) all vertices in V. Given an undirected, connected, and weighted graph G, a minimum spanning tree (MST) of G is a spanning tree whose edges have a minimum sum out of all possible spanning trees of G.

Prim's Algorithm

At a high-level, Prim's algorithm starts with a single vertex and grows the MST by greedily adding the vertex that can be added the most "cheaply" to the partial tree. In other words, at each step, we greedily add the vertex that has the minimum weight edge to some vertex in the partial tree. As seen in the pseudocode, Prim's algorithm is very similar to Dijkstra's, so we can see that the runtime is $O(m \log n)$ since the input graph is connected and thus $n = O(m)$.

Kruskal's Algorithm

At a high-level, Kruskal's algorithm finds an MST by first starting on the graph with no edges. We sort the edges in increasing order of weight and then iterate through the edges in this order, adding an edge to the MST if it does not create a cycle since we want the minimum total sum. In contrast to Prim's, Kruskal's starts with every vertex as its own tree and then gradually connects these "subtrees" into an MST.

Since Kruskal's relies on the ability to detect cycles efficiently, as seen in the pseudocode, we use the **Union-Find** data structure, where the MakeSet operation runs in $O(1)$ time and the Union/Find operations run in $O(\log n)$ time if we just use Union by Rank and in $O(\log^* n)$ — barely more than $O(1)$ — time if we use Path Compression with Union by Rank. If we do not know anything about the edge weights, then the runtime of Kruskal's will be the time it takes to sort the edges, which is $O(m \log m) = O(m \log n)$, so it technically would not matter whether or not we optimize with Path Compression since both would not exceed the overall $O(m \log n)$ runtime. However, for example, if the edges are given to us sorted, then the UF data structure can become the bottleneck for Kruskal's runtime, so it is important to use Path Compression then.

Reverse Delete Algorithm

Reverse Delete leverages the Cycle Property by removing edges in decreasing order of weight, ensuring that only the heaviest edges in any cycle are deleted, which guarantees they are not part of any MST—though it's slower due to the repeated connectivity checks, making its runtime $O(m^2)$ compared to Prim's and Kruskal's.

Correctness

The correctness and intuition behind Prim's and Kruskal's is derived from the cut property. The correctness of another MST algorithm, Reverse-Delete, is derived from the cycle property. Both are summarized below:

Cut Property: Let $e = (u, v)$ be a minimum cost edge "crossing the cut" with one end in S and the other in $V - S$. Then some MST of G contains e. Moreover, if $e = (u, v)$ is the unique minimum cost edge crossing this cut, then e belongs to *every* MST of G.

Cycle Property: Let C be any cycle in G and let $e = (u, v)$ be a heaviest edge in C. Then e does not belong to some MST of G. Moreover, if $e = (u, v)$ is the unique heaviest edge on the cycle, then e belongs to *no* MST of G.

Problem 1: True or False

1. Kruskal's and Prim's algorithms work on a graph with negative edge weights.
2. Suppose that we have an MST T of a graph G but are told that an edge not in T has a lower weight than originally specified and so T is now an invalid MST. It is guaranteed that we can fix our tree by removing an edge and adding a different one.

Problem 2

Suppose we have some MST, T, in a graph G with positive edge weights. Construct a graph G' where for any weight $w(e)$ for edge e in G, we have weights $w(e)^2$ in G'. Is T still a MST in G'? Prove your answer.

If G also had some negative edge weights, would your answer change from above change? Prove your answer.