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” (or 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 $G$.

Kruskal’s Algorithm

At a high-level, running Kruskal’s Algorithm on an input graph $G = (V, E)$ 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. The pseudocode is as follows:

```
Kruskal(G)
    $T = \emptyset$
    Sort the edges in $E$ in increasing order of weight.
    for each $v \in V$ do
        MakeSet($v$)
    for each $e = (u, v) \in E$ do
        if Find($u$) $\neq$ Find($v$)
            $T = T \cup \{e\}$
            Union($u$, $v$)
    return $T$
```

Since Kruskal’s relies on the ability to detect cycles efficiently, 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.

Non-Distinct Edge Weights

Note that in lecture, we first proved the correctness of Kruskal’s in the case where edge weights were all distinct. However, we can still extend the proof of correctness to the case where edge weights are non-distinct: we arbitrarily break ties by modifying the edge weights so that they maintain their relative ordering but
are now distinct (full details on how to do this are in Section 21.4 of the lecture notes). As you will see in some of the problems below, any modifications to edge weights that maintain their relative ordering are valid transformations. Similarly, we can modify the cut and cycle property as follows:

**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 \( e \).

**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 \( G \).

---

### Problems

**Problem 1**

Does Kruskal’s algorithm work on a graph with negative weights?

**Solution**

Yes, Kruskal’s algorithm still works on a graph with negative weights because it relies on a “pre-processing” step where edges are sorted in increasing order by their weights — this sorting is still done correctly even if some edge weights are negative. Kruskal’s operates by going through the sorted edge weights in increasing order while trying to add each edge to the MST, so it will still correctly select edges in order of increasing weight by selecting the more negative weighted edges first.

**Note:** Although Kruskal’s algorithm will still correctly find a MST when a graph has negative weights, observe that the output MST may not be the minimum spanning subgraph, since we could potentially add more negative weighted edges to the output MST to get a “more” minimum spanning subgraph that is not a tree.

Generally, any transformation to the graph that preserves relative edge weight ordering is valid. For example, you can find the most negative weighted edge in the graph, add that weight to every edge in the graph to make all the edge weights non-negative, and then call Kruskal’s algorithm on this modified graph.

**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.

**Solution**

If \( G \) only has positive weights, then after squaring edge weights, \( T \) remains a valid MST in the transformed graph \( G' \). We will prove this via contradiction. Assume the claim does not hold and that \( T \) is no longer an MST in \( G' \). Instead, 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 \( w(e')^2 < w(e)^2 \). However, since \( T \) is a proper MST in \( G \), the edge \( e \) must have been a minimum weight edge spanning that cut in \( G \), and therefore \( w(e) \leq w(e') \). Note that since we have positive edge weights, it is not possible to have both \( w(e')^2 < w(e)^2 \) and \( w(e) \leq w(e') \), so we have reached a contradiction. Therefore, squaring edge weights in a positively weighted graph is a valid transformation.

On the other hand, note that if we have negative edge weights, we can still have both \( w(e')^2 < w(e)^2 \) and \( w(e) \leq w(e') \). Hence, the contradiction from above does not hold, and the claim is actually false. As a
counterexample, consider the graph below. The MST $T$ consists of the edges $b - c$ and $a - c$, with edge weights of $-3$ and $-2$, respectively. After squaring, however, we have edge weights of $1$, $4$, and $9$, so the new MST $T'$ consists of the edges $a - b$ and $a - c$, with edge weights of $1$ and $4$, respectively. Therefore, squaring edge weights when some are negative is not a valid transformation, as shown in this example where the MSTs have changed.

Again, generally any transformation/change to edge weights that preserves their relative ordering is valid.

**Problem 3**

Suppose we have a connected graph $G$ where all edge weights are equal. Design an efficient algorithm find an MST of $G$. What is the running time of your algorithm?

**Solution**

**Algorithm:** Run BFS/DFS on $G$ and output the edges in the BFS/DFS tree.

**Proof of Correctness:** Observe that because all edge weights are equal, any spanning tree of $G$ will be a minimum spanning tree. We know that running BFS/DFS on $G$ will output a BFS/DFS tree; in particular, we have a DFS tree instead of a forest because $G$ is connected. Additionally, since $G$ is connected, we know that our BFS/DFS tree must be spanning because the traversals visit every vertex in the graph. Therefore, since the BFS/DFS tree is a spanning tree, it is also a minimum spanning tree, so our algorithm is correct.

**Runtime Analysis:** Since we only run BFS/DFS, the running time of our algorithm is $O(|V| + |E|)$.

**Problem 4**

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. 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**

Since we are given that our MST $T$ is invalid because of this change, observe that the modified edge $e$ does not exist in $T$ or else $T$ would still be a valid MST. Furthermore, $e$ must exist in all possible MSTs of the new graph. So, add $e$ to the original MST $T$. Since all trees are maximally acyclic, in our new graph of $T + \{e\}$, we have formed a cycle $C$ that contains the edge $e$. By cycle property, a maximum-weight edge in $C$ does not exist in some MST. Note that since the modified edge $e$ is necessary, a maximum-weight edge $e'$ must also be distinct from $e$. Therefore, removing $e'$ from the graph of $T + \{e\}$ yields a valid MST again.