Kruskal's Algorithm

1 Algorithm

Basically, add edges to a subset of the viable solution by increasing weight, given that no cycles are induced by the addition. Merges disjoint forests into a single MST.

```
def kruskalMST(G = (V, E)):
    A = {}
    Place each vertex in its own set
    Sort E by increasing order by weight

    for (u, v) in sorted(E):
        if find(u) != find(v):  # different trees
            A += (u, v)
            union(u, v)    # merge trees
```

2 Proof

Consider at any point in the algorithm, with the partial solution A', that we are considering the edge $(u, v) \mid u \in A' \subset A, v \notin A'$ and (u, v) does not induce any cycles in A'. Take the cut $(A', V - A')$:

- Every crossing edge is not in A' by definition of the cut
- (u, v) must cross the cut by definition of the edge being considered. Because edges are considered by weight order, (u, v) must be the cheapest such crossing edge.

By the safe spanning edge lemma, (u, v) is a safe edge. Therefore, Kruskal's algorithm correctly generates an MST.

3 Runtime

The initial sorting of edges is $O(m \log m)$. Given an efficient Union Find structure, there are $\Theta(n)$ operations on a total input size of n elements to give $\Theta(m \log n)$ for the loop (theoretical lower bound of $\Theta(m \cdot \alpha(n))$. Therefore, the algorithm is $\Theta(m \log n)$.