Theory

3.

A tree is an undirected graph that is connected and acyclic. This graph with *n* nodes has *n-1* edges and is characterized by having a unique path between any pair of nodes on the tree. A spanning tree of an undirected graph *G* is a subset of the graph’s edges that form a tree containing (spanning) all nodes.

The optimal min/max spanning tree could be found using the naïve algorithm which consists of checking all possible spanning trees in a graph. However, if we have n nodes there are *2^n* possible spanning trees thus the algorithm has an exponential running time.

Another alternative for finding the min/max spanning tree is through a greedy algorithm (such as Prim’s or Kruskal algorithms). A greedy algorithm consists of picking up the next best solution while maintaining feasibility. In other words, it works in finding the next best local choice with the hope of finding the global optimum.

In the case of finding the optimal spanning tree, the greedy algorithm is the best approach as it does not only find the local best solution but gives as well a global optimal solution. This can be shown by the cut property which indicates that in any cut (*S, V-S*) in a graph *G*, any least weight crossing edge (for a min case) such as edge *e* incident to nodes *u* and *v* with *u ∈ S* and *V ∉ S* is in some minimum spanning tree of graph G. This can be proven as follows:

Consider a minimum spanning tree *T* of *G* where *e* is not in *T* and has a path between nodes *u* and *v*. Another minimum spanning tree *T’* that contains edge *e* is constructed based on the previous tree *T*. Since *T*  has a already unique path between *u* and *v*  we cannot add edge *e*  to *T* as it creates a cycle (tree should be acyclic). Thus, we assume that there is another edge *e’* that connect the cut to *T*. Thus, for *T’* we would have

and *weight (T’) = Weight (T) + W(e) – W(e’).* Since *e* is the lowest edge then the weight of *T’*  is less than the weight of *T.*  Thus, if T is a minimum spanning tree then *T’ =T* and *W(e) = W(e’)* proving that *T’* is also an MST where the least weight crossing edge is chosen (this can also be shown with a maximum spanning tree taking the highest weight crossing edge).

Both Prim’s and Kruskal are based on the cut property to find the min/max spanning tree. The Prim’s algorithm adds edges that have the lowest/highest weight to gradually build up the spanning tree. In other words, at every step, we need to find a cut and select the lowest/highest weight edge and include it in an empty set containing the spanning tree. This algorithm is very similar to Djikstra as it uses also a priority queue to find the lowest/highest edge and has a running time depending on the type of priority queue used. A running time of O (V^2) using an array priority queue. This running time is enhanced by using a binary heap and adjacency data structure with a running time of O(E log V) with E for edges and V for vertex.

The Kruskal’s algorithm is another approach to solve the min/max spanning tree problem but starts with the globally lowest / highest weight edge where we repeatedly add to the empty set the next lightest/heavier edge that does not produce a cycle. Sorting of the edges in this algorithm has a total running time of O (E log E). After sorting of the edges, we need to iterate through all edges and apply the union and find operations with a running time of *2E·T(Find)+V·T(Union) = O((E+V)log V) = O(E log V)*

Thus, the total running time is O (E log E) or O (E log V) which are similar. This running time can be also improved by using a randomized algorithm for minimum cut.

*References:*

* Massachusetts Institute of Technology. (2015). Lecture 12: Greedy Algorithms and Minimum Spanning Tre. *Spring Lecture 12*, [online]. Available at: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-design-and-analysis-of-algorithms-spring-2015/lecture-notes/MIT6\_046JS15\_writtenlec12.pdf [Accessed 2 Dec 2017].
* S Dasgupta, C.H. Papadimitriou, and U.V. Vazirani. Greedy Algorithms. P. 143 to 151. Available at: file:///C:/Users/Joanna%20Andari/AppData/Local/Packages/Microsoft.MicrosoftEdge\_8wekyb3d8bbwe/TempState/Downloads/greedy.pdf [Accessed 2 Dec 2017].