

**Experiment No. : 3**

**Title:** Virtual lab on Kruskal’s algorithm to construct minimum spanning trees

# Batch:B2 Roll No.:1914078 Experiment No.:3

**Aim:** Explore the virtual lab on Kruskal’s algorithm to construct minimum spanning trees.

# Resources needed: Google Chrome

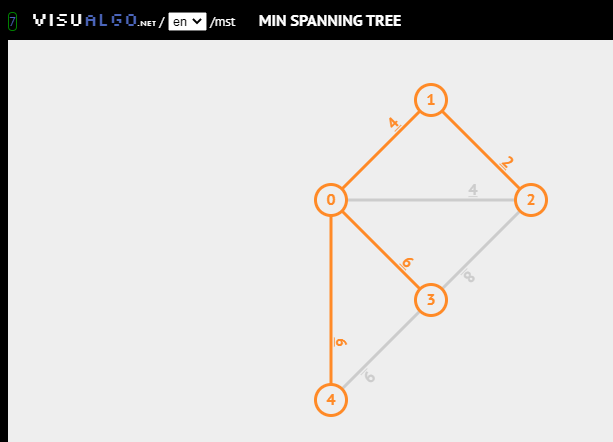
**Kruskal Algorithm Introduction :**

In this experiment, we will see a famous problem in graphs, finding the Minimum Spanning Tree. Let *G = (V, E, W)* be a weighted graph. Find a subgraph *G'* of *G* that is connected and has the smallest cost, where cost is defined as the sum of the edge weights of all edges in *G'*. Observe that if *G'* has a cycle, we can remove one of the edges along a cycle and still the resultant graph will remain connected and has smaller cost than *G'*. Hence, *G'* cannot be have a cycle and as it has to be connected, *G'* must be a tree. We define *G'* as a spanning subgraph of *G* iff *V(G) = V(G')* and a spanning subgraph that is also a tree is called a spanning tree of *G*. Our aim is to find a spanning tree of *G* that has the least cost and such a spanning tree is called as minimum spanning tree (MST) of *G*.

**Algorithm/Procedure :**

* Kruskal's algorithm: An O(E log V) greedy MST algorithm that grows a forest of minimum spanning trees and eventually combine them into one MST.
* Kruskal's requires a good sorting algorithm to sort edges of the input graph by increasing weight and another data structure called Union-Find Disjoint Sets (UFDS) to help in checking/preventing cycle.
* Kruskal's algorithm first sort the set of edges E in non-decreasing weight (there can be edges with the same weight), and if ties, by increasing smaller vertex number of the edge, and if still ties, by increasing larger vertex number of the edge.
* Then, Kruskal's algorithm will perform a loop through these sorted edges (that already have non-decreasing weight property) and greedily taking the next edge e if it does not create any cycle w.r.t edges that have been taken earlier.

**Observations from Simulation:**

****

**Self evaluation :** Solve both Kruskal Quiz and Analysis quiz and Display the result of your first attempt.

**Q.2. In a graph G, let the edge uv have the least weight. Is it true that uv is always part of any minimum spanning tree of G? Is it true that uv is always part of some minimum spanning tree of G? Justify your answers.**

Ans: Consider a graph G with V vertices and E edges. Suppose it’s possible to make an MST in the graph without using the shortest edge uv.

Now the MST has V-1 vertices and no cycles. Now if the edge uv is added to the MST, a cycle forms.

This cycle includes uv and other edges that are naturally heavier than uv.

In this cycle, we select the edge with the least edge which is uv and make another MST say Tmin.

Thus, Weight(Tmin) < Weight(T)

**4. Let G be a graph and T be a minimum spanning tree of G. Suppose that the weight of an edge e that also belongs to T is increased. How can you and the minimum spanning tree of the modifed graph. What is the runtime of your solution?**

Let e= (u,v) and let ‘Tu’ and ‘Tv’ be the subtrees obtained by removing ‘e’ .By doing BFS (ignoring edge weights) from ‘u’ and from ‘v’, we can determine which vertices are in ‘Tu’ and which are in ’Tv’ in time O(|V|+|E|). Assume we have marked each node with its membership. Now examine each edge, and keep the minimum weight edge e′(e` = edge e with increased weight) with one end point in ’Tu’ and the other in ’Tv’. This can be done in O(|E|)time. The total time is thus O(|V|+|E|).

**6. Let G be a graph and e be a maximum weight edge on some cycle in G. Let G' be the graph obtained by removing the edge e from G. Show that G' has a minimum spanning tree T' that is also a minimum spanning tree of G.**

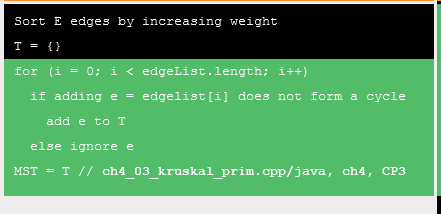
Ans. Let graph G have V vertices and E edges. G` is graph with edge ‘e’ removed. G` has a minimum spanning tree T`. It proves that the edge ‘e’ is not needed to form an MST in graph G`. Thus, the MST T` is also a minimum spanning tree for G as it will be ignored because of its maximum weight.

**8. Generalize the above question where the edge weights are between 1 and W.**

The time complexity for Dijkstra’s algorithm is O(E + VLogV) where E is edges and V is vertices.

When all the edges are either 1 or 2 we can modify the Dijkstra’s algorithm to get the time complexity of O(V+E). The idea is to use BFS. One important observation about BFS is, the path used in BFS always has least number of edges between any two vertices. So if all edges are of same weight, we can use BFS to find the shortest path. For this problem, we can modify the graph and split all edges of weight 2 into two edges of weight 1 each. In the modified graph, we can use BFS to find the shortest path. Same for W edges weights

# Analysis of Algorithm (Time and Space complexity) :



Initialising T takes O(1) time. Sorting the edges by weight takes Elog(E). Find set operation and Union operation for each edge in G, taking a time of O(ElogV). Since, E can be atmost V^2. Thus logV and LogE are same.

Thus the time complexity of Kruskal’s algorithm is O(ElogE).

The algorithm needs a set for storing the vertices V and an array to store the edges present in the MST.

Time Complexity: O( |E| log|V| ), in worst case we would have to do find or union operation for all the edges. Find and Union both are O( log|V| ) operation, which can be optimized to O(1) if we use path compression, in which case the time complexity of the Kruskal’s Algorithm will be O(|E|). If we consider the sorting of the edges then we would have another O(|E|log|E|).

Space Complexity: O(|E| + |V|), since Disjoint Set Data Structure takes O(|V|) space to keep track of the roots of all the vertices and another O(|E|) space to store all edges in sorted manner.

**Activity:** Search IIITH virtual lab, study and write some primary observation. Complete the writeup.

# Outcome: Implement Greedy and Dynamic Programming algorithms

**Conclusion:** We learned about Kruskal’s algorithm using Virtual lab and analysed it.

**References:**

**Books/ Journals/ Websites:**

1. Richard E. Neapolitan, " Foundation of Algorithms ", 5th Edition 2016, Jones & Bartlett Students Edition
2. T.H. Coreman ,C.E. Leiserson,R.L. Rivest, and C. Stein, " Introduction to algorithms", 3rd Edition 2009, Prentice Hall India Publication
3. https://ds2-iiith.vlabs.ac.in/exp/min-spanning-trees/exp.html#MST%20Experiment