PRIM'S ALGORITHM

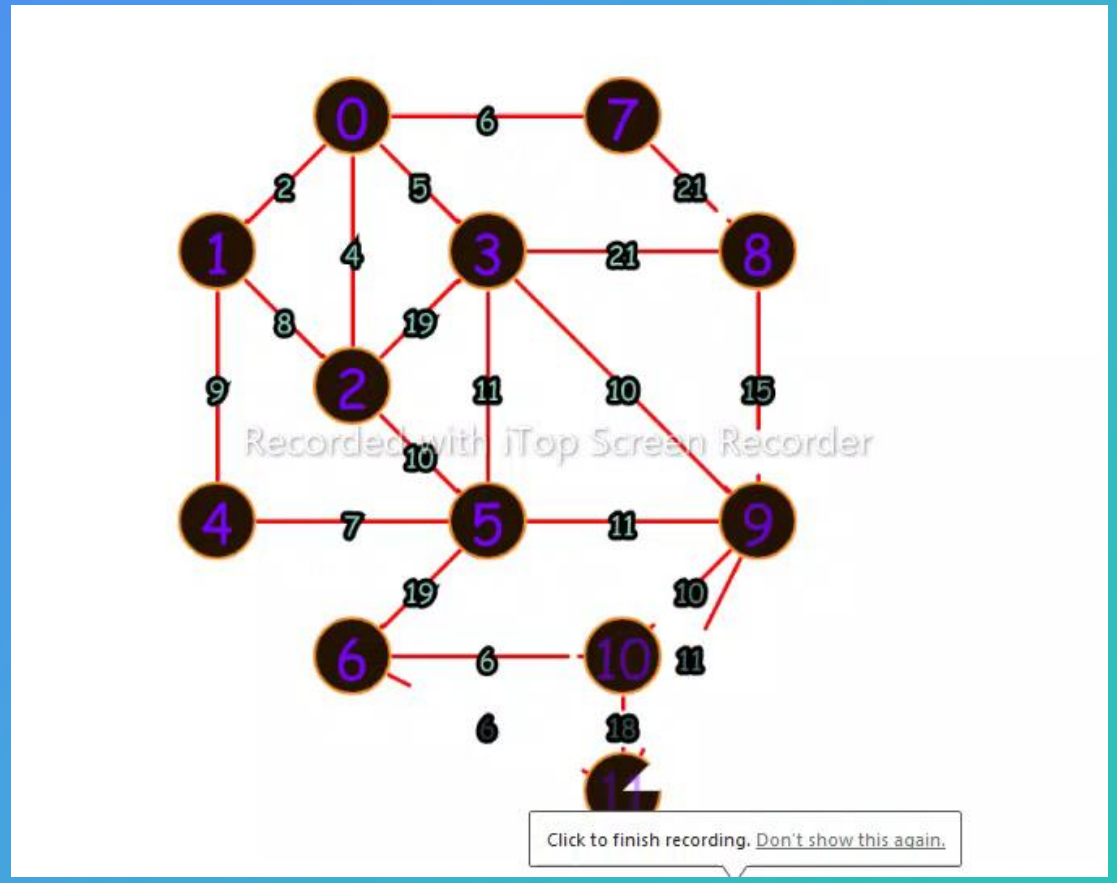
Rishi Gupta
002921999

What is Prim's Algorithm

- **Prim's Algorithm** was created to discover the MST (minimum spanning tree) across a connected, weighted undirected graph. This is based on a greedy algorithm. This signifies that the technique finds the minimal weight "tree" (a structure without cycles) that connects all of the vertices via a subset of all available edges.

- **Time and Space complexity of the Algorithm –**
  - The time complexity of Prim's method using the adjacency matrix, is O(V^2), where V is the number of vertices in the graph. By employing Fibonacci heaps, this time complexity may be reduced to O(E + Vlog V). By using the binary heap and adjacency list this time complexity may be reduced to O(E log V)
  - The total complexity of space is of the order O(V+E).

Prim's Algorithm to find Minimum Spanning Tree



1.Create a tree with a single vertex picked at random from the graph.
2.Grow the tree by one edge: of the edges that connect the tree to vertices that aren't yet in the tree, pick the one with the lowest weight and add it to the tree.
3.Recursively call step 2 (until the tree covers all the nodes).

Prim's Algorithm psuedo code :-

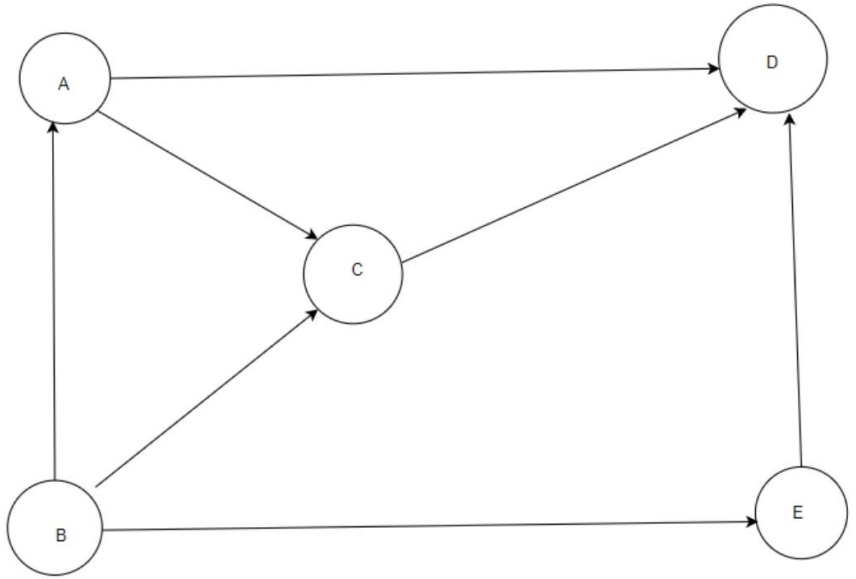
```
prims_algo(Graph pa) {
for vertexes v in pa:
    v.dist = INF
St = pa.startingnode()
St.dist = 0
 mst = {}
 heap = buildHeap(pa.vertices – {St})
 for each vertex v in St.neighbors():
     v.dist = pa.weight(St, v)
     v.previous = St
     heap.decreaseKey(v, v.dist)
 while (! heap.empty()):
     v = heap.deleteMin()
     mst.addEdge(v, v.previous)
     for edges (v, u) in pa:
         d1 = v.dist
         d2 = u.dist
         if (d1 < d2):
             u.previous = v
```

Limitation Prim's Algorithm:

The time it takes to find the least weight arc slows it down for big numbers of nodes.

Equal weighted edge can add complexity if one of the weights in the cycle is removed and the vertices are connected.

Prim’s algorithm can fails for directed graph : In a directed graph not every node is reachable from every other node. So Prim's algorithm fails for this reason. For example, in below-directed graph let’s assume the starting node is “D”.



Applications of Prim's Algorithm:

- City center distance for transit minimum route calculation.
- In a network setup, cables play an important role in determining the minimum number of cables required to cover an entire area.
- Cluster Analysis: The k-clustering problem can be viewed as taking the MST and removing the k-1 most expensive edges.
- It can be used in the path-finding algorithm which is used in AI.
- Prim's Algorithm can be used in game development like maze games.
- It helps to create a log on the network cycle.

Conclusion

We saw how Prim's algorithm uses the greedy approach to build a minimum spanning tree. A greedy algorithm makes it easy to pick the edge with the lowest weight. We also analyzed how the minimum heap is chosen and the tree is formed. The time complexity of this algorithm has also been discussed and we have also seen how this algorithm is achieved. This makes the Prims algorithm a good greedy approach to finding a minimum spanning tree.

References

- https://www.geeksforgeeks.org/difference-between-prims-and-kruskals-algorithm-for-mst/#:~:text=Prim's%20algorithm%20has%20a%20time,works%20only%20on%20connected%20graph.
- https://www.softwaretestinghelp.com/minimum-spanning-tree-tutorial/
- https://github.com/JazonJiao/Manim.js/tree/master/Graph%20Algorithms
- https://www.geeksforgeeks.org/why-prims-and-kruskals-mst-algorithm-fails-for-directed-graph/
- https://www.youtube.com/watch?v=tDj9BkaQDO8