Minimum Spanning Tree Comparison

By Mikaela Montaos
Prepared under the direction of Professor Henry Chang

School of Engineering
Northwestern Polytechnic University
117 Fourier Ave, Fremont, CA 94539
April 2021

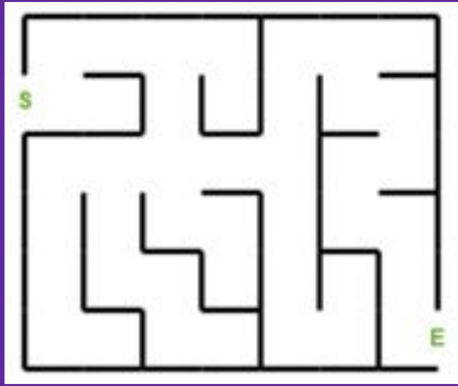
Table of Contents

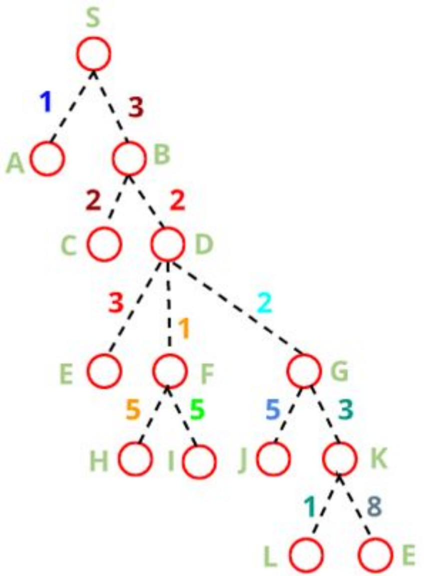
Introduction

- A Minimum Spanning Tree (MST) is finding the minimum weight in a spanning tree. A spanning tree is both a tree (connected) and spanning (includes all vertices).
- This document compares two MST algorithms:
  - Prim’s MST
  - Kruskal’s MST

Design

The problem is a maze. The goal is to start on letter S and reach the end on letter E while using the minimum weight/fastest path.



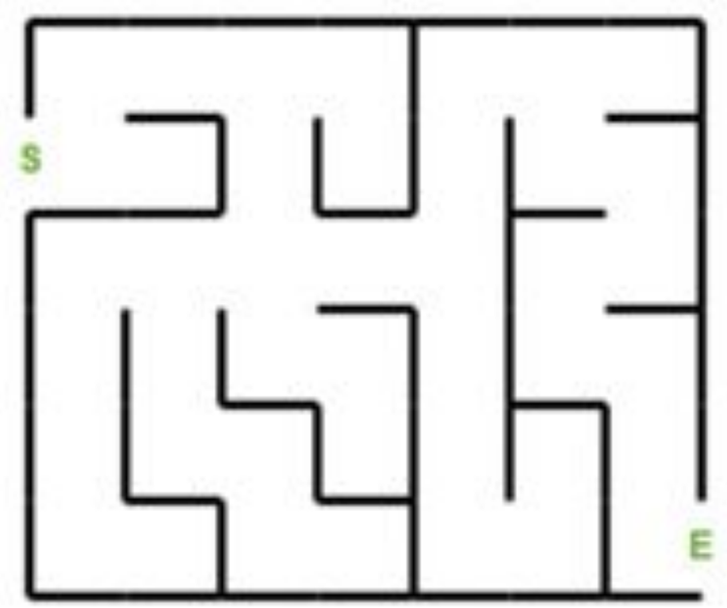


Implementation: Prim’s MST

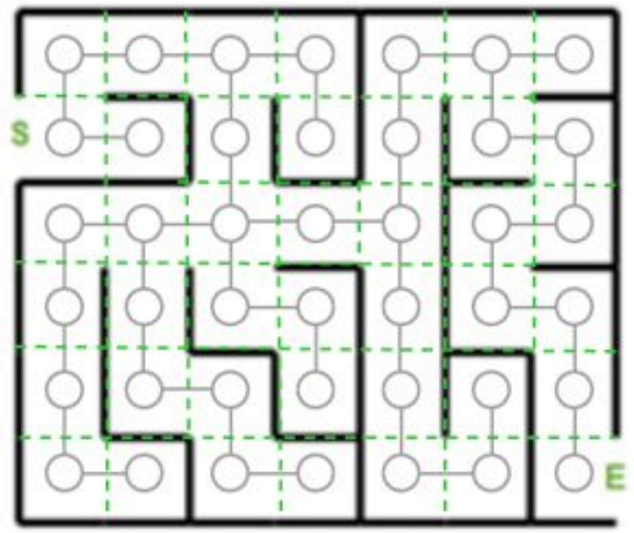
1. Add 0 next to the starting node and mark it as visited
2. Add the weight value next to the nodes connected to the starting node
3. The next node to visit is the node with the lowest weight value
4. Continue until all nodes are visited
    a. If there is more than one weight on a node, choose the lowest value

Test: Maze Problem

1. Problem

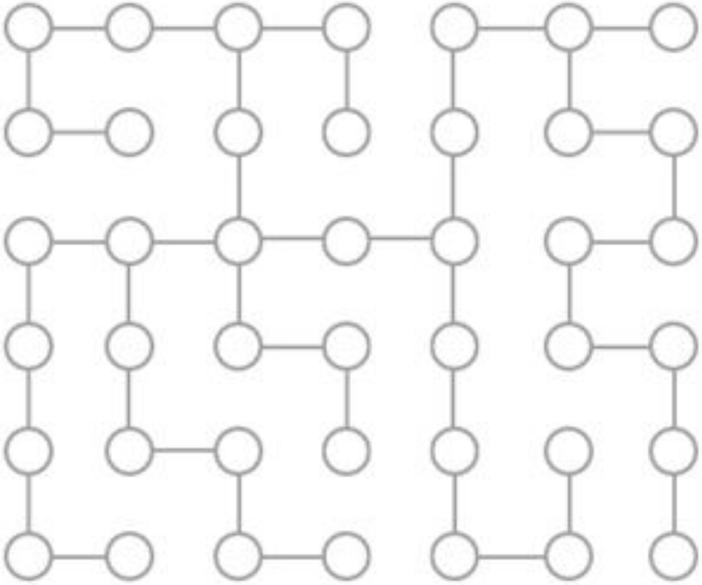


2. Create a graph over the maze and add nodes and edges

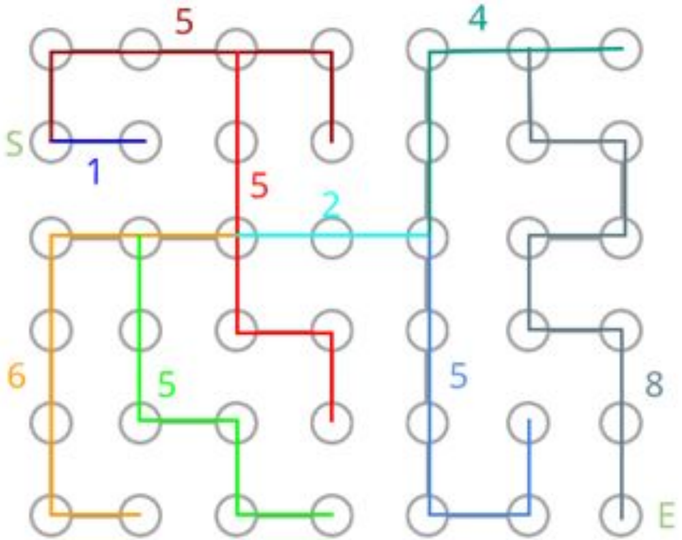


Test: Maze Problem

3. Extract the nodes and edges from the maze

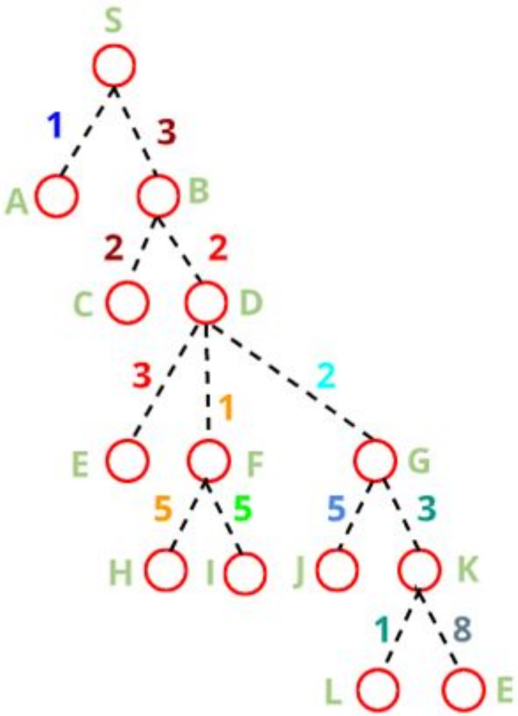


4. Count the number of nodes until it reaches a vertice. That will be the edge value



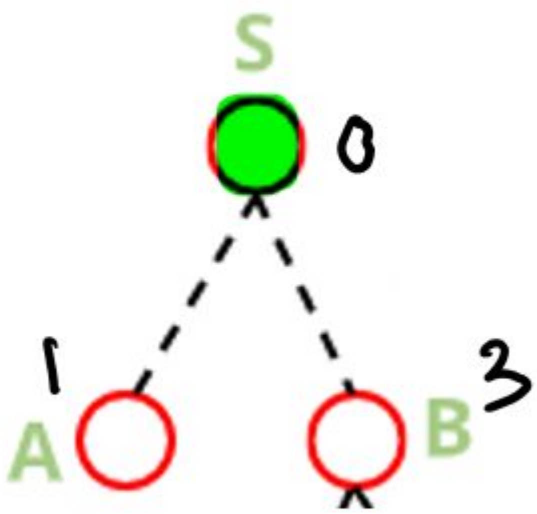
Test: Maze Problem

5. Convert into a tree

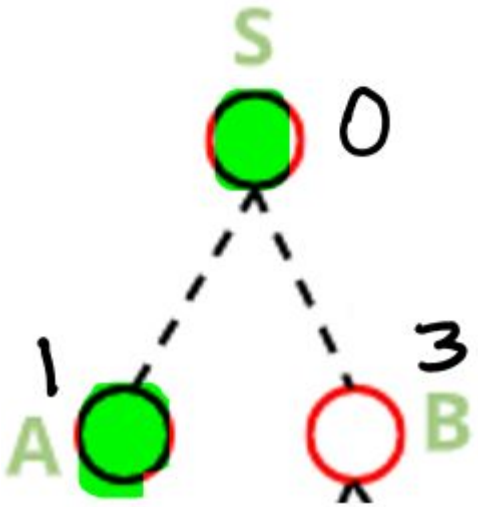


Test: Prim's MST

1. Node Start

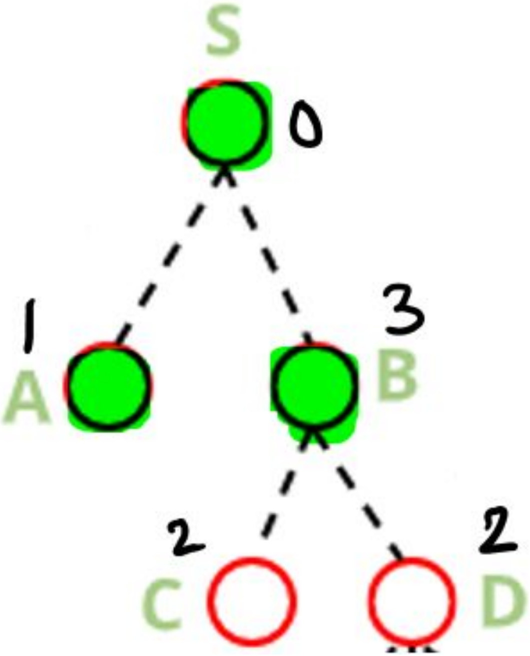


2. Node A

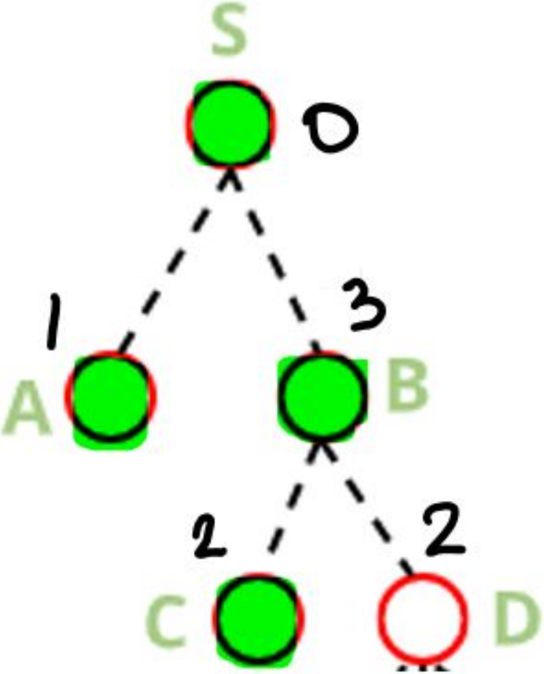


Test: Prim's MST

3. Node B

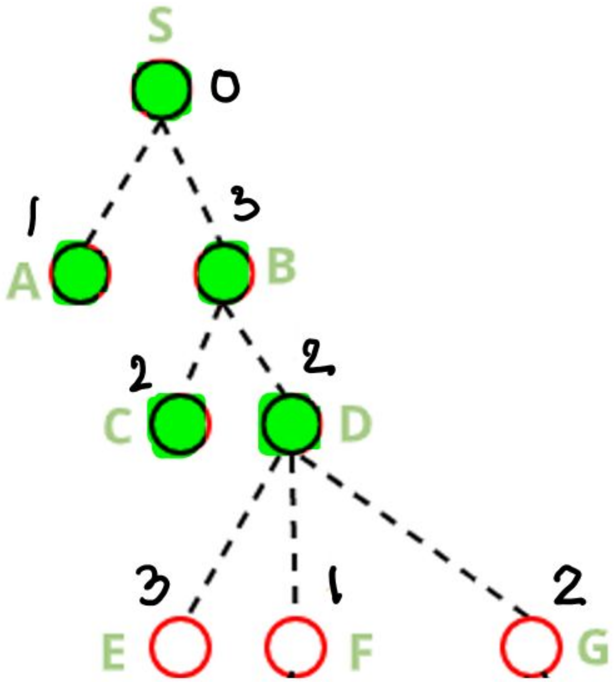


4. Node C

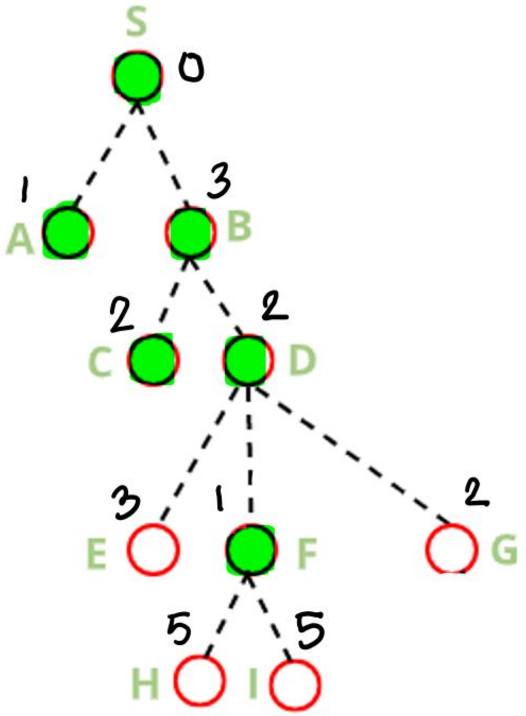


Test: Prim's MST

5. Node D

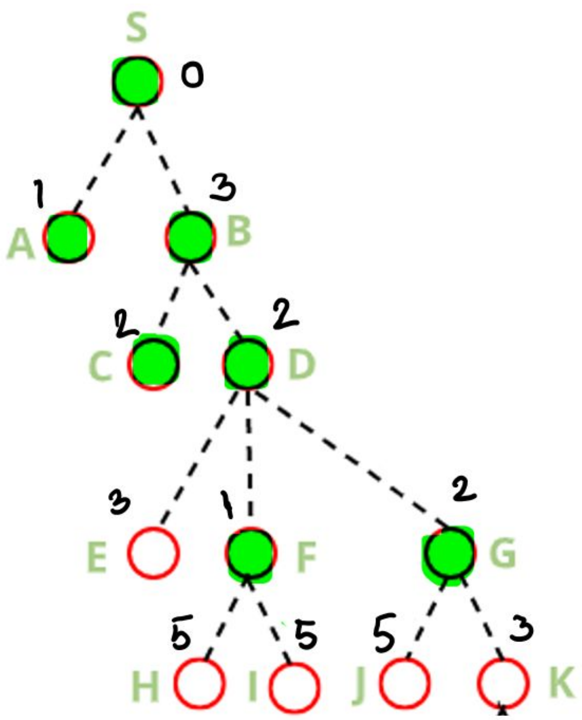


6. Node F

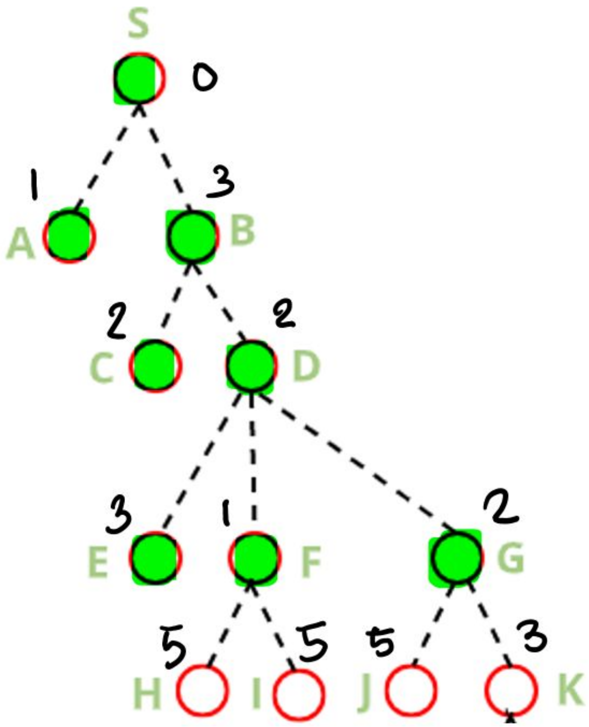


Test: Prim's MST

7. Node G

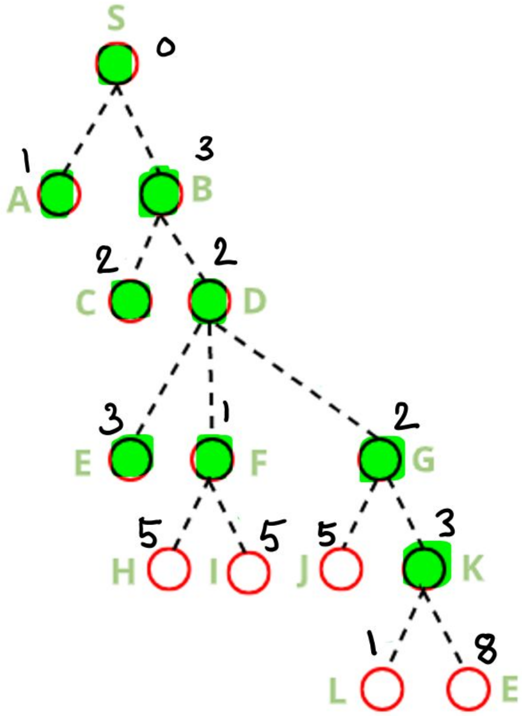


8. Node E

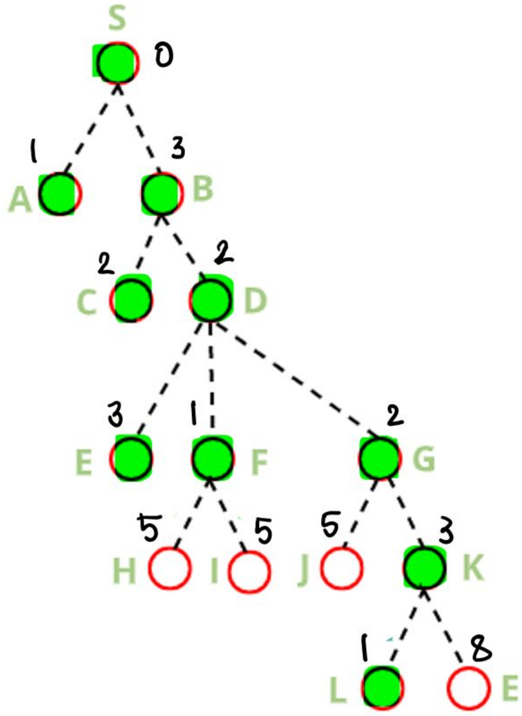


Test: Prim’s MST

9. Node K

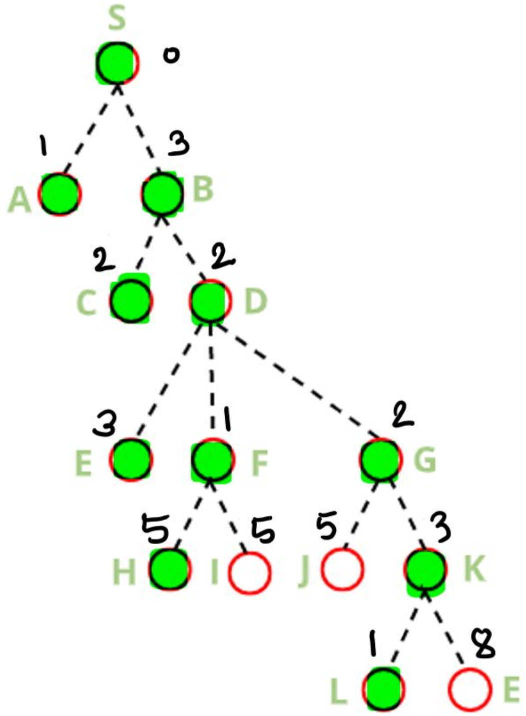


10. Node L

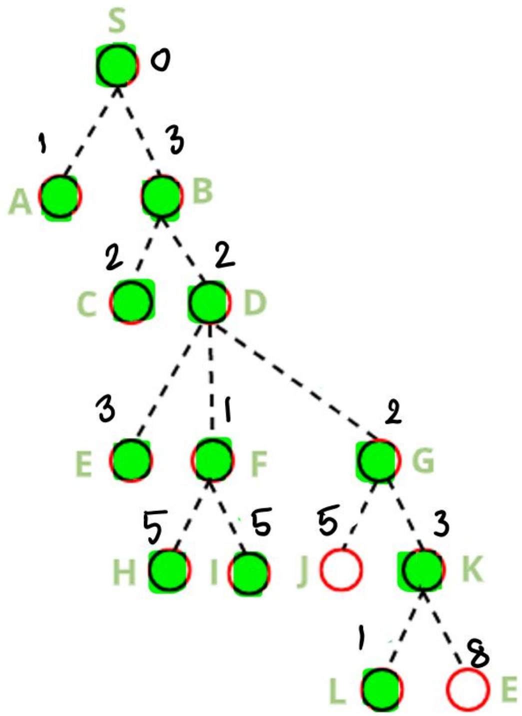


Test: Prim's MST

11. Node H

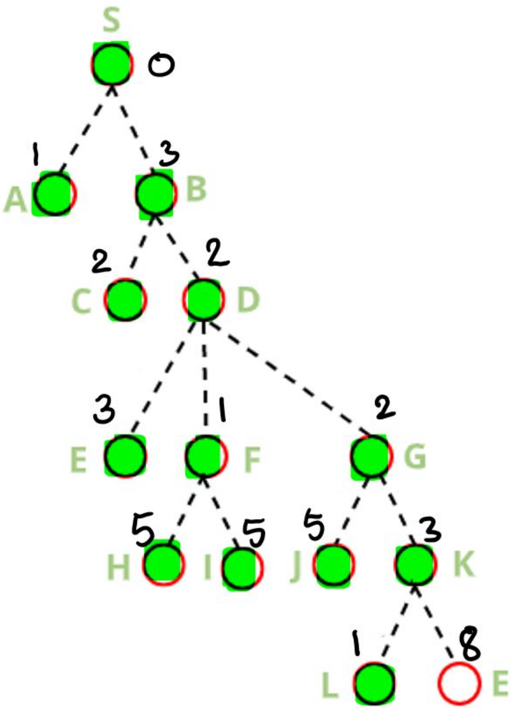


12. Node I

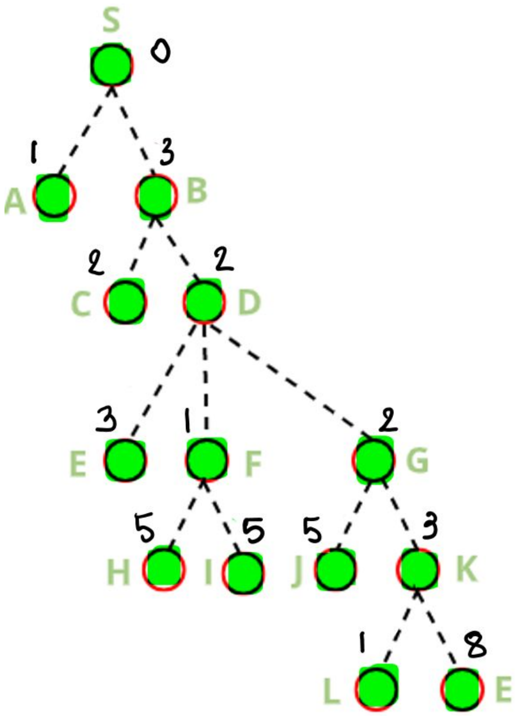


Test: Prim's MST

13. Node J



14. Node End



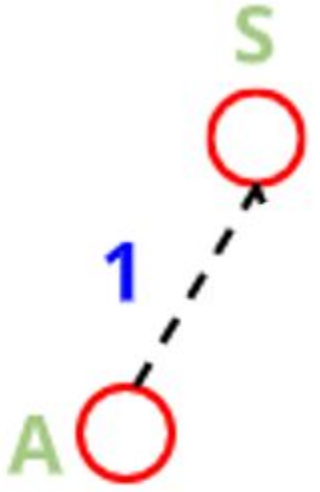
Implementation: Kruskal’s MST

1. Sort the edges in an increasing order according to weight
2. Start with the smallest edge
3. Continue until all edges are visited
    a. If a cycle/loop is formed, do not include the edge
    b. If there is an edge already connected to the node, disregard. Only keep the smaller edge.

Test: Kruskal's MST

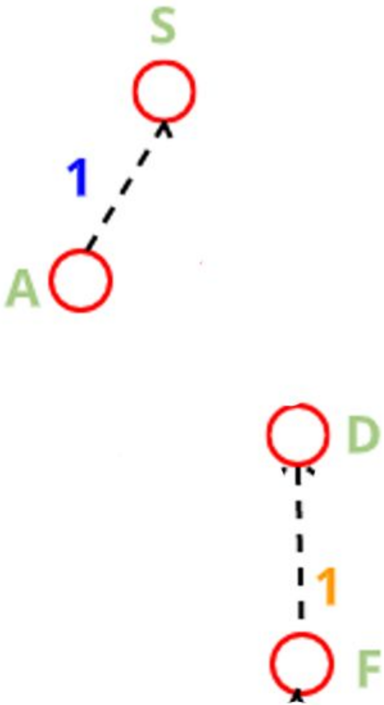
Weight	Source -> Destination
1	S -> A
1	D -> F
1	K -> L
2	B -> C
2	B -> D
2	D -> G
3	S -> B
3	D -> E
3	G -> K
5	F -> H
5	F -> I
5	G -> J
8	K -> E (End)

S -> A

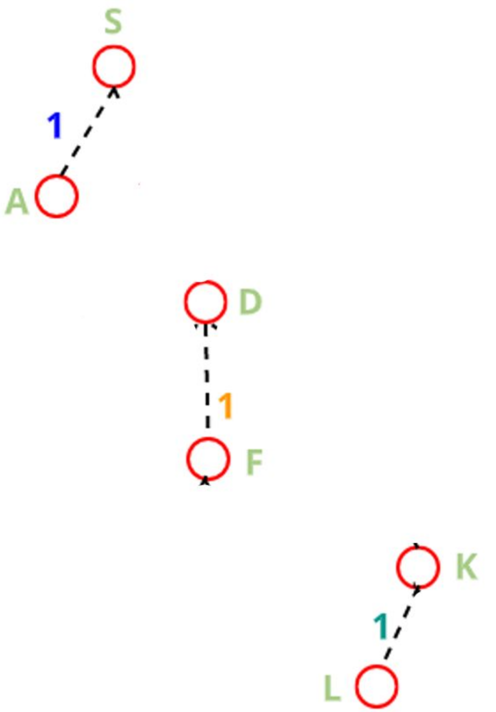


Test: Kruskal’s MST

D -> F

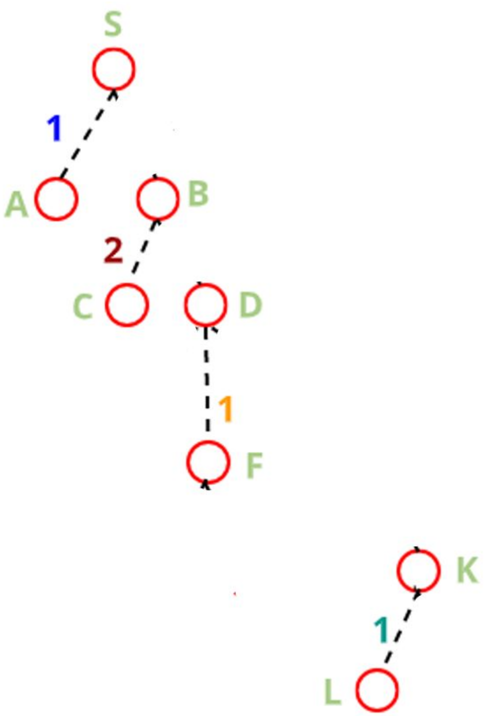


K -> L

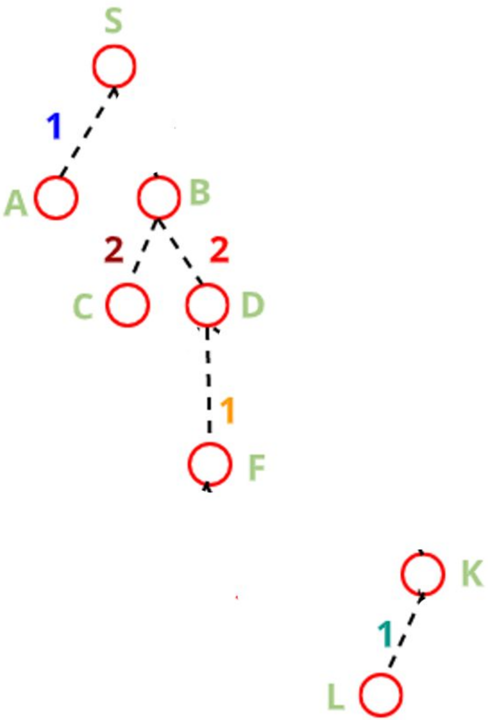


Test: Kruskal's MST

B -> C

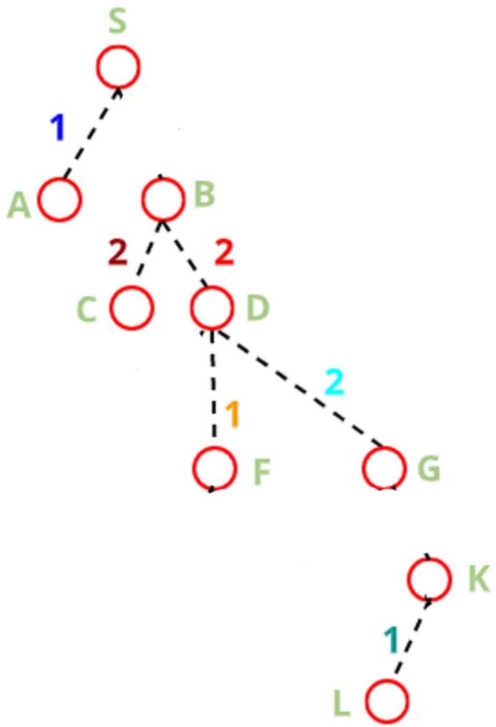


B -> D

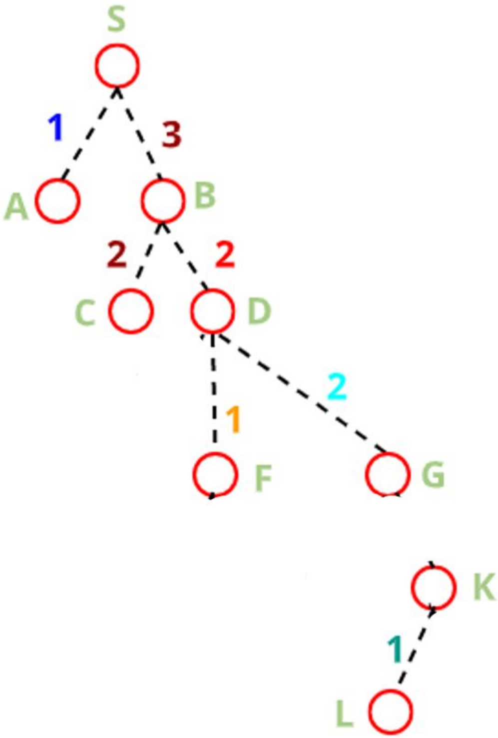


Test: Kruskal's MST

D -> G

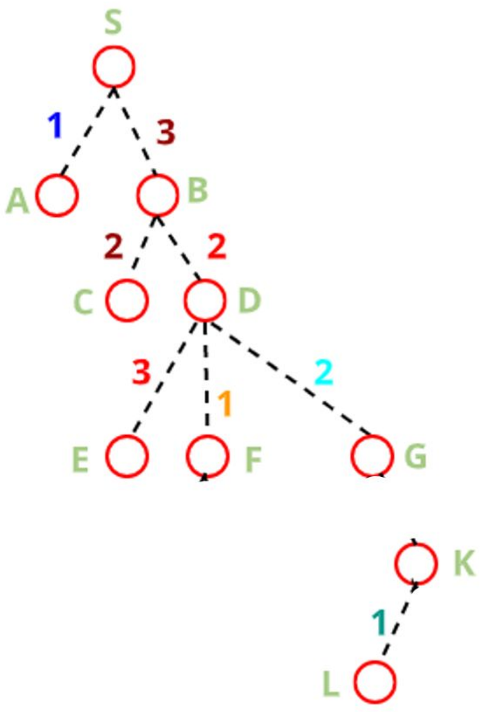


S -> B

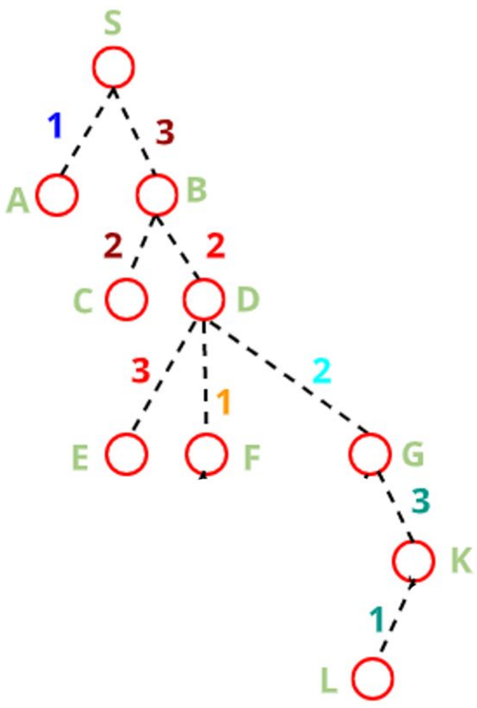


Test: Kruskal's MST

D -> E

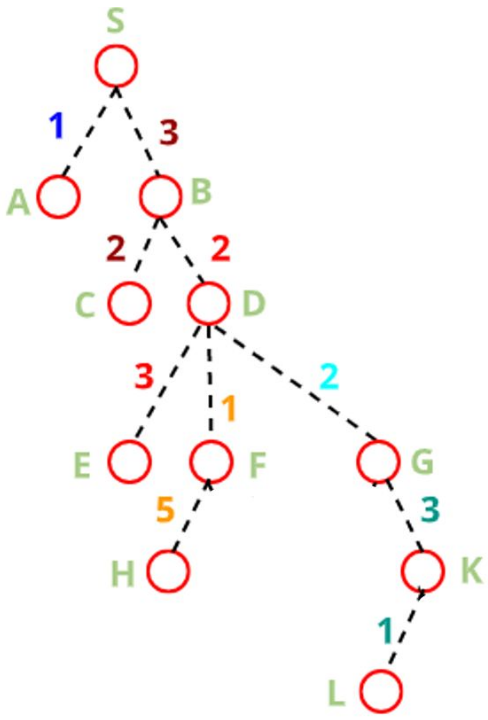


G -> K

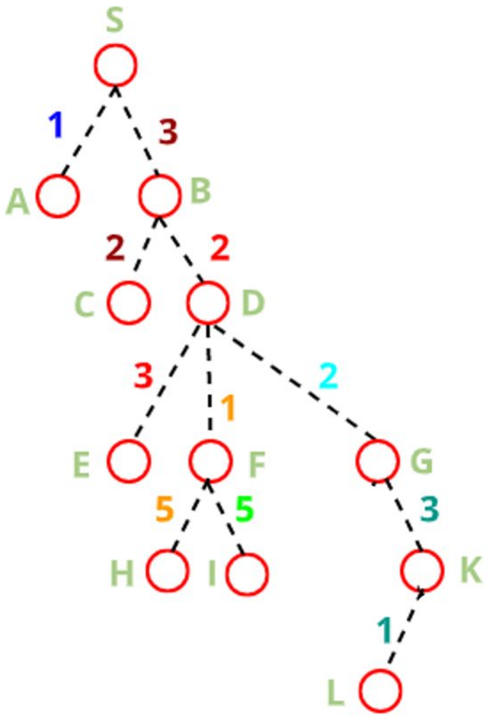


Test: Kruskal's MST

F -> H

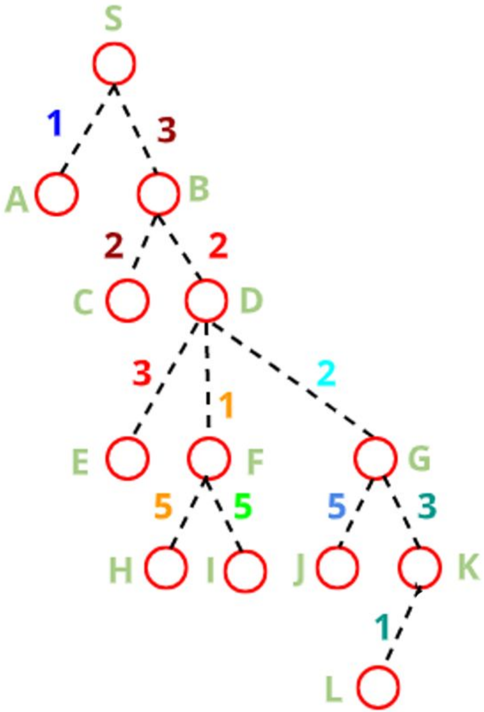


F -> I

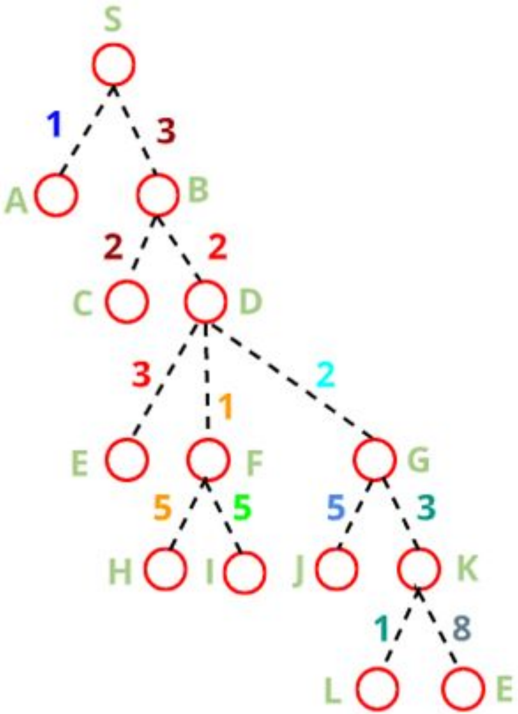


Test: Kruskal's MST

G -> J



K -> E (End)



Enhancement ideas

- Real-life application of the algorithms
  - Prim’s MST
    - AI pathfinding
  - Kruskal’s MST
    - Gas piping

Conclusion

Prim's MST

- Big-O is O(E + log V)
  - E = Edges, V = Vertex
- Same performance for the maze problem
- Runs faster in dense graphs (several edges)

Kruskal's MST

- Big-O is O(E log V)
  - E = Edges, V = Vertex
- Same performance for the maze problem
- Runs faster in sparse graphs (few edges)

Bibliography

GeeksforGeeks. (2020, October 20). Difference between Prim's and Kruskal's algorithm for MST. https://www.geeksforgeeks.org/difference-between-prims-and-kruskals-algorithm-for-mst/

Mitanshupbhoot. (2020, April 4). Comparative applications of Prim's and Kruskal's algorithm in real-life scenarios. Medium. https://medium.com/@mitanshupbhoot/comparative-applications-of-prims-and-kruskal-s-algorithm-in-real-life-scenarios-4aa0f92c7abc

Sedgewick, R., & Wayne, K. (2015). Algorithms, Fourth Edition: Book and 24-Part Lecture Series (4th ed.). Addison-Wesley Professional.

user340082710. (2016, March 17). A Graph's Density and Sparsity [Online forum post]. Computer Science Stack Exchange. https://cs.stackexchange.com/questions/54575/a-graphs-density-and-sparsity