Dijkstra's algorithm

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

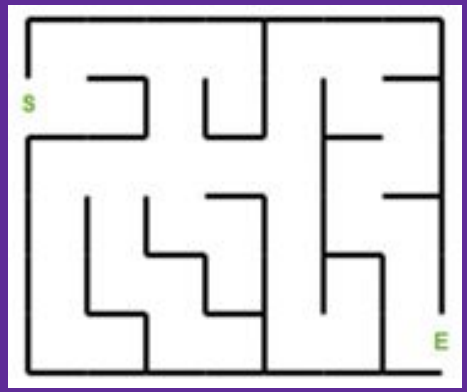
Table of Contents

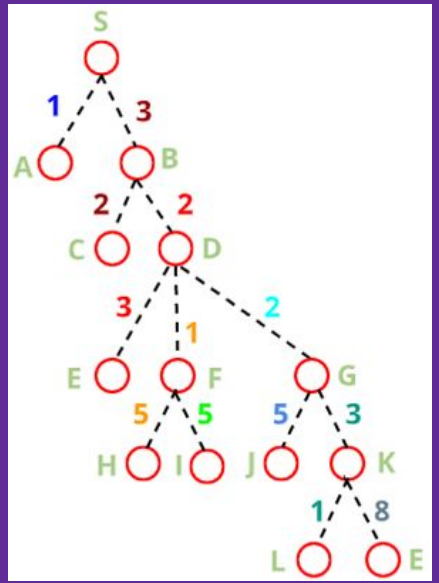
Introduction

- The algorithm is created by Edgar Dijkstra
- It finds the shortest distance from the start vertice, adding the edge/distance until you reach the end vertice
- The distance has to be nonnegative
- It is used many applications that benefits in finding the shortest path (e.g.: map app)

Design

The algorithm initially starts with a maze.
The maze turns into a tree graph which is used to solve using Dijkstra's algorithm for this project



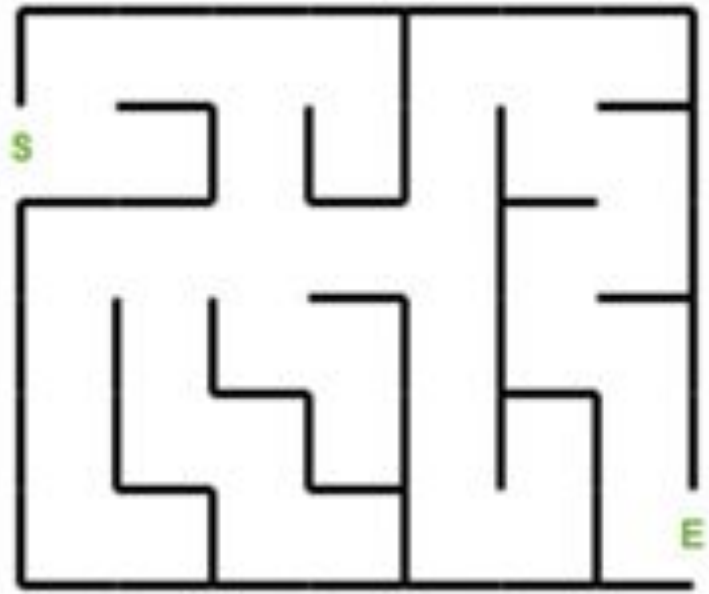


Implementation

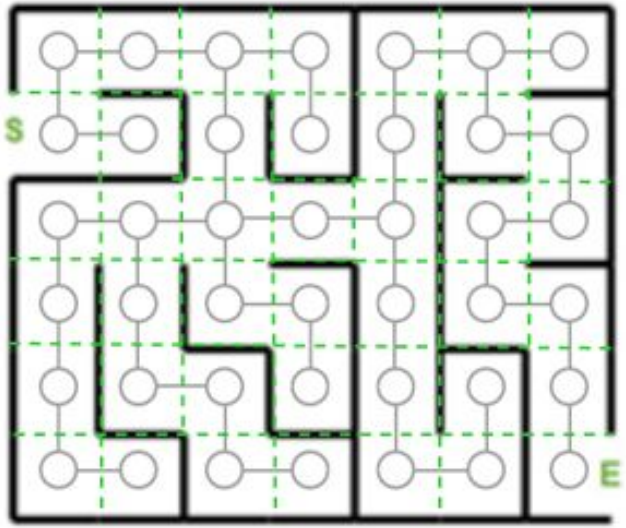
1. Consider vertices in increasing order of distance from S
2. Add vertex to the tree and relax all edges pointing from that vertex
    a. Relax means adding the distance from the current node to the next vertex and assigning that value to that vertex. If there is more than one, assign the lower value.
3. Mark the current vertex as visited
4. Choose the next vertex with the lowest valued edge/distance and repeat until all vertices are visited
5. The shortest path is the lowest distance from the start vertex to the end vertex

Test

1. Problem

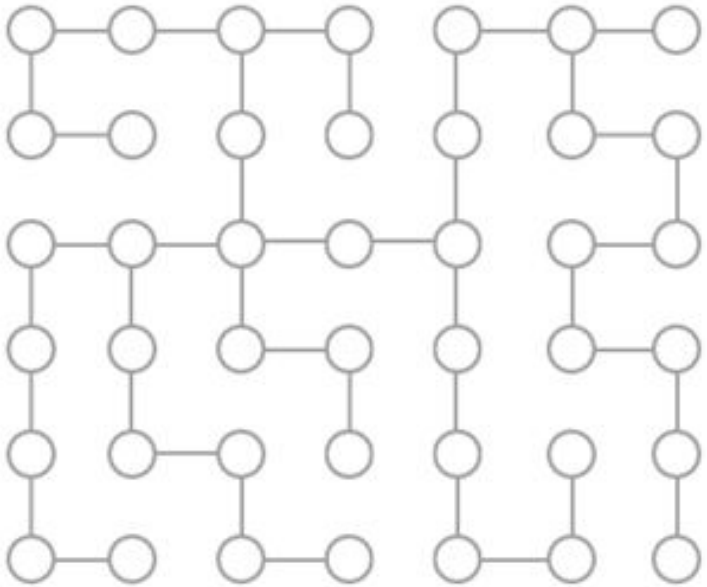


2. Create a graph over the maze and add nodes and edges

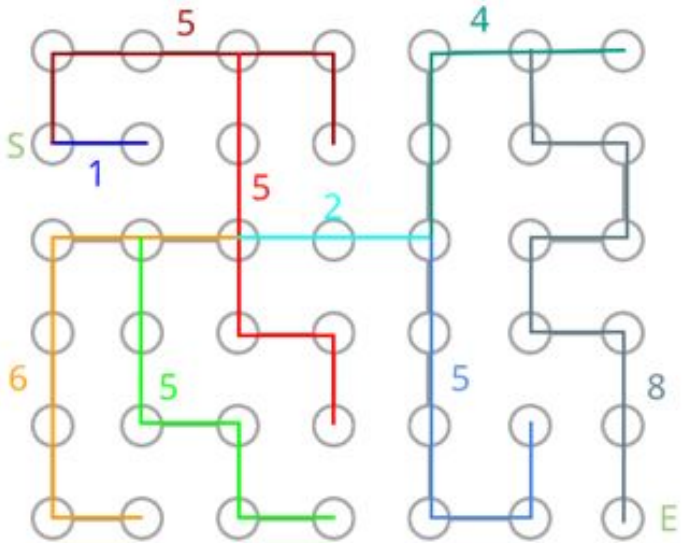


Test

3. Extract the nodes and edges from the maze

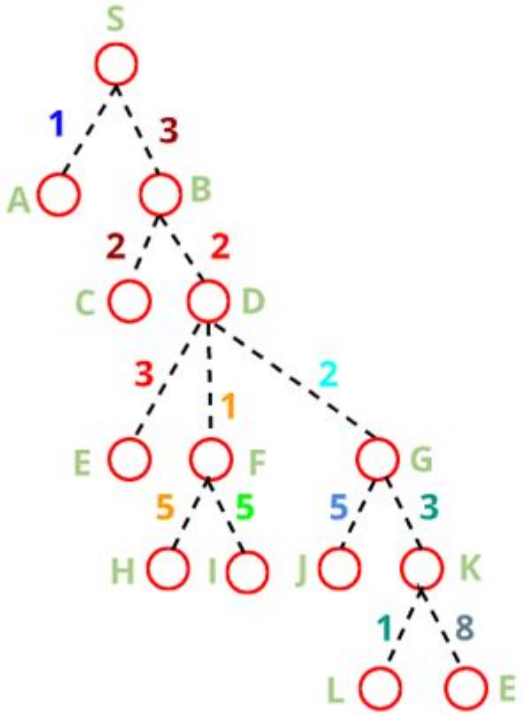


4. Count the number of nodes until it reaches a vertice. That will be the edge value



Test

5. Convert into a tree



6. Use Dijkstra's algorithm to find the shortest path from start vertice (S) to end vertice (E)

   a. Image on the next slide

Test

Vertex	Initial	Step 1: S (S)	Step 2: A (S, A)	Step 3: B (S, B)	Step 4: C (S, B, C)	Step 5: D (S, B, D)	Step 6: F (S, B, D, F)	Step 7: H (S, B, D, F, H)	Step 8: I (S, B, D, F, I)	Step 9: G (S, B, D, G)	Step 10: K (S, B, D, G, K)	Step 11: L (S, B, D, G, K, L)	Step 12: E (S, B, D, G, K, E)
Next step	S	A	B	C	D	F	H	I	G	K	L	E	
S	0	0	0	0	0	0	0	0	0	0	0	0	0
A	∞	1	1	1	1	1	1	1	1	1	1	1	1
B	∞	3	3	3	3	3	3	3	3	3	3	3	3
C	∞	∞	∞	3+2 = 5	5	5	5	5	5	5	5	5	5
D	∞	∞	∞	3+2 = 5	5	5	5	5	5	5	5	5	5
E	∞	∞	∞	∞	∞	5+3 = 8	8	8	8	8	8	8	8
F	∞	∞	∞	∞	∞	5+1 = 6	6	6	6	6	6	6	6
G	∞	∞	∞	∞	∞	5+2 = 7	7	7	7	7	7	7	7
H	∞	∞	∞	∞	∞	∞	6+5 = 11	11	11	11	11	11	11
I	∞	∞	∞	∞	∞	∞	6+5 = 11	11	11	7+5 = 12	12	12	12
J	∞	∞	∞	∞	∞	∞	∞	∞	∞	7+3 = 10	10	10	10
K	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	10+1 = 11	11	11
L						∞	∞	∞	∞	∞	10+8 = 18	18	18
E	∞	∞	∞	∞	∞	∞	∞	∞	∞				

Test

Shortest distance: S -> B -> D -> G -> K -> E = 18

Enhancement ideas

- There are two ways in finding the shortest path: Dijkstra’s algorithm and Bellman Ford’s algorithm. It would be beneficial to compare the two algorithms.
- This slide shows the traditional/graphical way of solving Dijkstra’s algorithm. It would be interesting to code this algorithm into a program similar to a map.

Conclusion

- Dijkstra’s algorithm is proven to be correct in finding the shortest path from start vertex to end vertex
- Dijkstra’s algorithm is similar to another algorithm which is Prim’s Minimum Spanning Tree algorithm
- Main difference:
  - Prim’s MST algorithm chooses the closest vertex to the tree through an undirected edge
  - Dijkstra’s algorithm chooses the closest vertex to the source

Bibliography

Sedgwick, R., Wayne, K. (ND). Algorithms: 24-Part Lecture Series [Dijkstra’s Algorithm]. Retrieved from https://learning.oreilly.com/videos/algorithms-24-part-lecture/9780134384528/