Shortest Path Algorithms 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

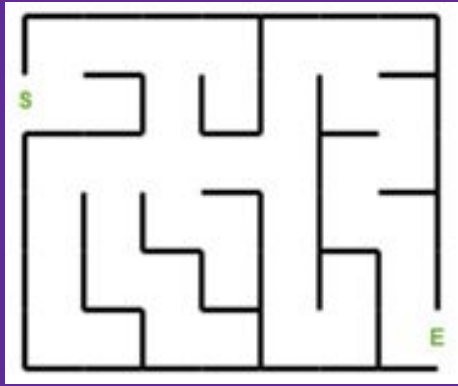
Table of Contents

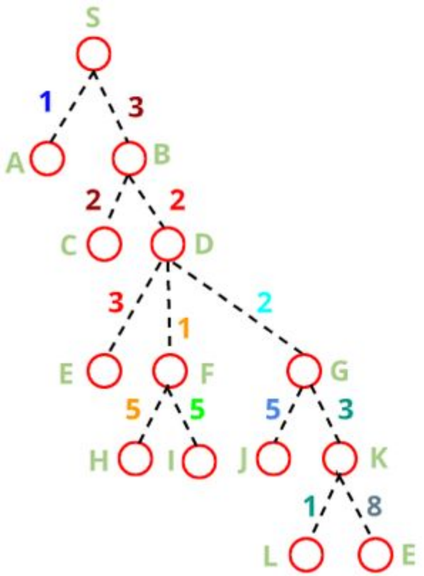
Introduction

- This document compares two shortest path algorithms:
    - Dijkstra’s algorithm
        - Only works on positive edge weights
    - Bellman-Ford’s algorithm
        - Can have a positive or negative edge weights
- Similarities: It finds the shortest distance from the start vertice, adding (or subtracting) the edge/distance until you reach the end vertice

Design

The algorithm initially starts with a maze. The goal is to start on letter S and reach the end on letter E.



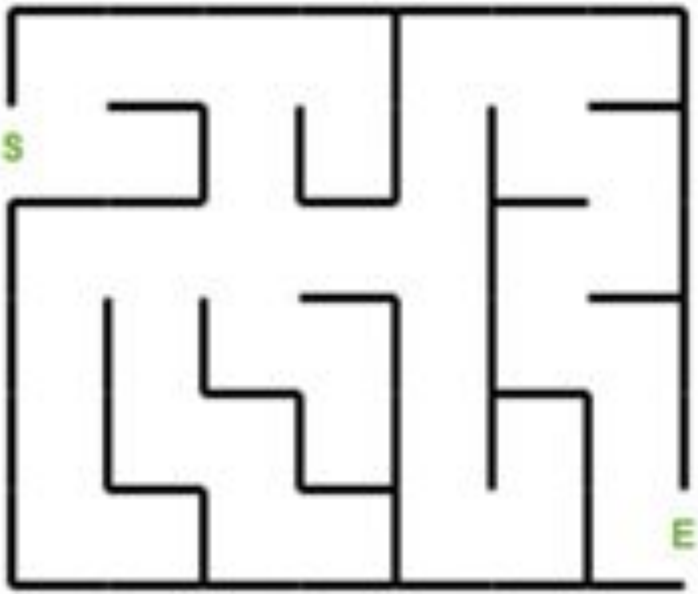


Implementation: Dijkstra’s Algorithm

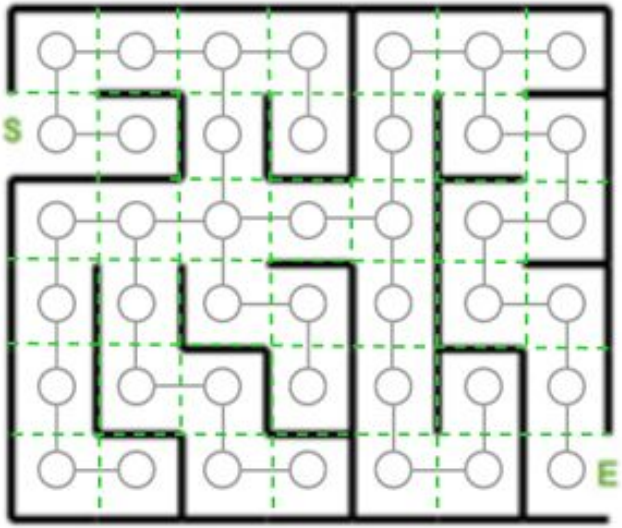
1. Consider vertices in increasing order of distance from S
2. Add vertex to the tree and relax all edges pointing from that vertex
   a. Relax means adding the distance from the current node to the next vertex and assigning that value to that vertex. If there is more than one, assign the lower value.
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: Maze Problem

1. Problem

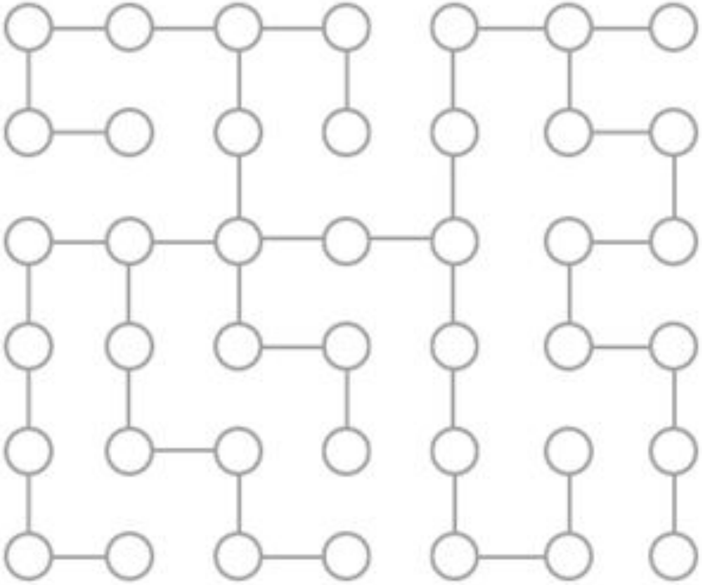


2. Create a graph over the maze and add nodes and edges

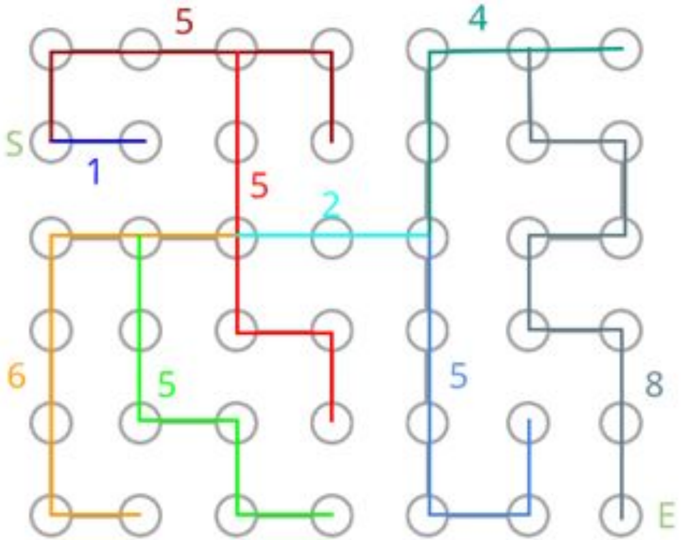


Test: Maze Problem

3. Extract the nodes and edges from the maze

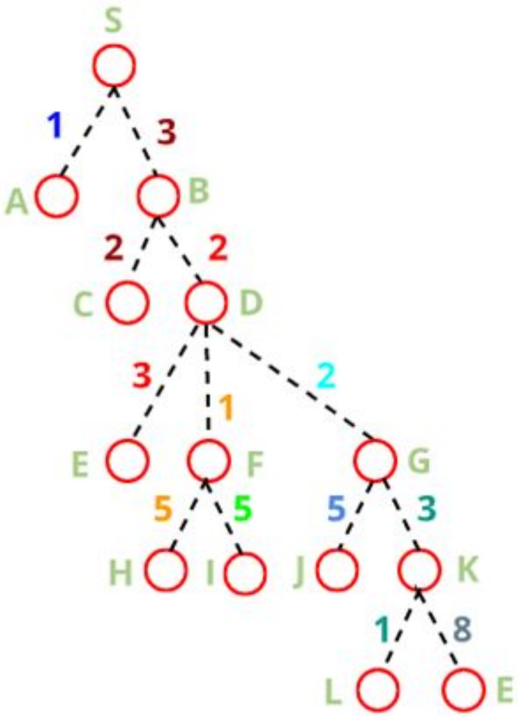


4. Count the number of nodes until it reaches a vertice. That will be the edge value



Test: Maze Problem

5. Convert into a tree



Test: Dijkstra’s Algorithm

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	11	11	11	11
J	∞	∞	∞	∞	∞	∞	∞	∞	∞	7+5 = 12	12	12	12
K	∞	∞	∞	∞	∞	∞	∞	∞	∞	7+3 = 10	10	10	10
L						∞	∞	∞	∞	∞	10+1 = 11	11	11
E	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	10+8 = 18	18	18

Test: Dijkstra's Algorithm

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

Implementation: Bellman-Ford’s Algorithm

- Similar to Dijkstra’s Algorithm
- Difference
  - Can have multiple cycles
  - Relaxing edges can mean adding or subtracting the distance from the current node
  - The process stops when...
    - You get a negative weight cycle
    - The minimum distance did not change for the cycle

Test: Bellman-Ford’s Algorithm

Cycle 1														
Initial	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
S	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
A	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
B	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	3+2=5	3+2=5	∞	∞	∞	∞	∞	∞	∞	∞	∞
C	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	∞	∞	∞	∞	∞	∞	∞	∞	∞
D	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	5+3=8	5+1=6	5+2=7	∞	∞	∞	∞	∞	∞
E	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	∞	∞	∞	∞	∞	∞
F	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	6+5=11	6+5=11	∞	∞	∞	∞
G	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	7+5=12	7+3=10	∞	∞
H	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	∞	∞
I	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	∞	∞
J	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	∞	∞
K	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	10+1=11	10+8=18
L	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
E	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18

Cycle 2														
Initial	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
S	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
A	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
B	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
C	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
D	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
E	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
F	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
G	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
H	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
I	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
J	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
K	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
L	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18
E	S	A	B	C	D	E	F	G	H	I	J	K	L	E
	0	1	3	5	5	8	6	7	11	11	12	10	11	18

Enhancement ideas

- Real-life application of the algorithms
    - Dijkstra’s on map route algorithm
        - Take the speed limit as the edge and the speed limit changes or turns as the vertices
    - Bellman-Ford’s on exchange rates arbitrage detection
        - Take the transaction percentage as the edge and the currency as the vertices

Conclusion

Dijkstra's Algorithm

- Binary heap
- Shorter time (visit vertices once)
    - 12 steps
- Big-O is O(V + E * log(V))
    - V for vertices
    - E for edges

Bellman-Ford's Algorithm

- Queue-based
- Takes twice as long (at least two cycles)
    - 14 steps per cycle
- Big-O is O(V * E)
    - V for vertices
    - E for edges

Bibliography

Sedgewick, R., & Wayne, K. (2015). Algorithms, Fourth Edition: Book and 24-Part Lecture Series (1st ed.). Addison-Wesley Professional.

Sryheni, S. (2020, September 9). Dijkstra’s vs Bellman-Ford Algorithm. Baeldung on Computer Science. https://www.baeldung.com/cs/dijkstra-vs-bellman-ford