Course: CS455 - Algorithms & Structured Programming
Faculty: Prof. Henry Chang

# Dijkstra’s Algorithm to find the shortest path of Maze

Signature Project
Name: Haripriya A
ID: 19579

# Table of Contents

# Introduction to Dijkstra’s Algorithm

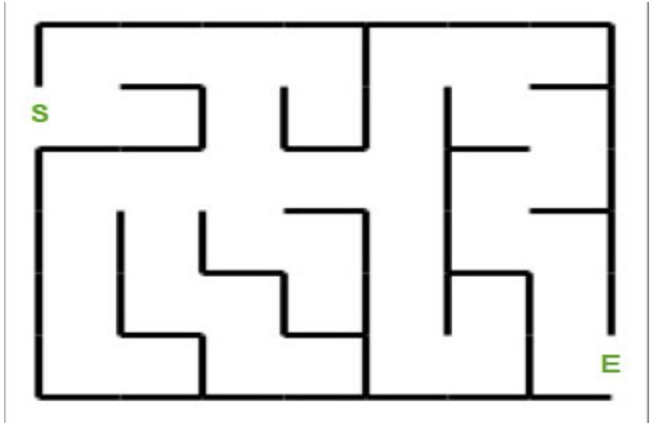
*Dijkstra’s algorithm solves the single-source shortest-paths problem on a directed weighted graph where all the nodes need to be non-negative.

*With the given maze we are going to implementing a tree of  shortest paths from source vertex to the target vertex and find the shortest path of the following maze using Dijkstra’s algorithm.

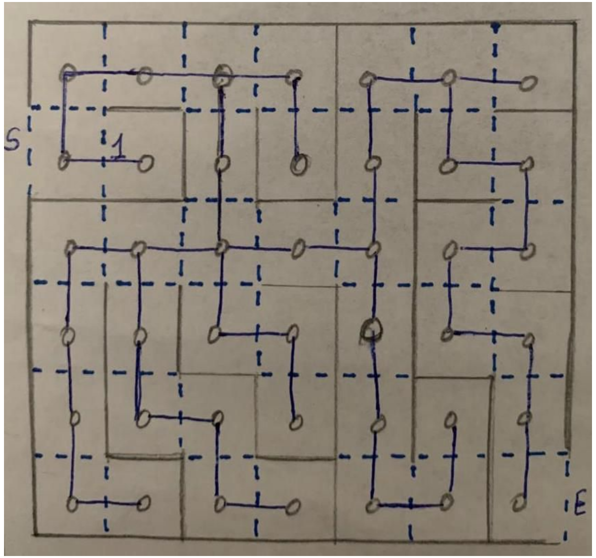
# Steps for Implementing a tree from given Maze

**A. Below are the detailed steps to implement a tree from source vertex S to the end E**

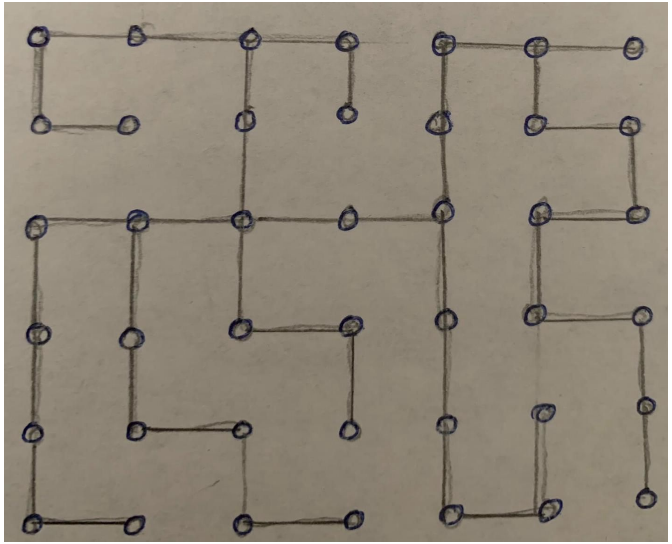
STEP - 1: Starting with the Initial Node S



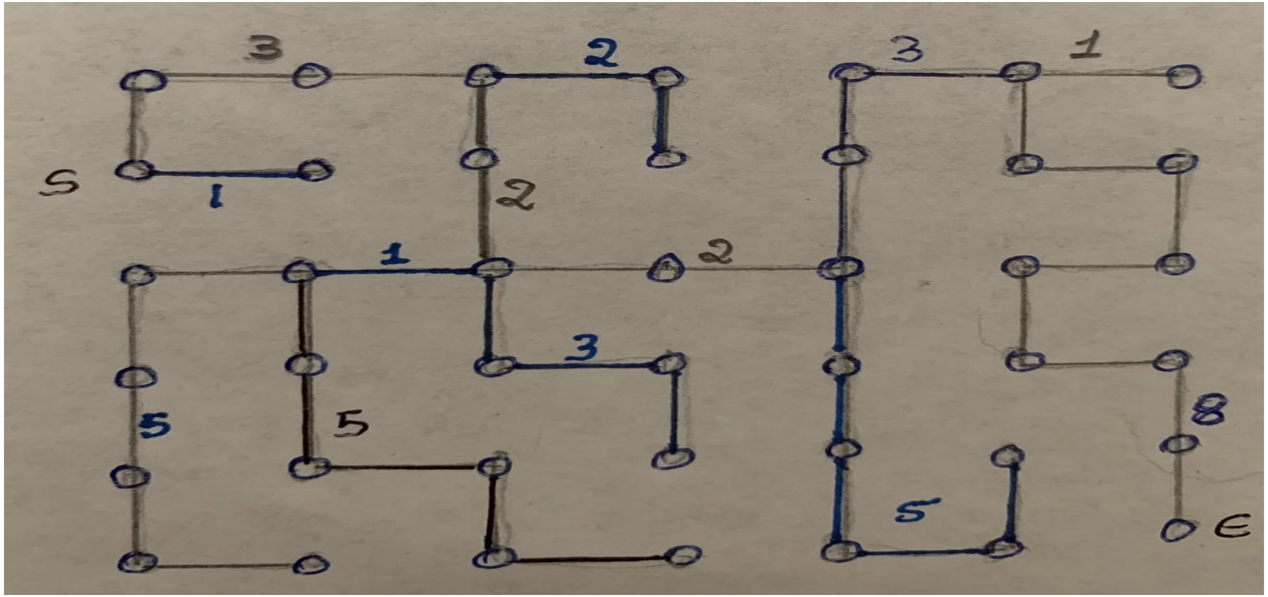
STEP - 2 : Draw the dotted lines and place the nodes in each box and draw the path from each node starting from S




STEP - 3 : Take out the maze and draw the nodes and path.

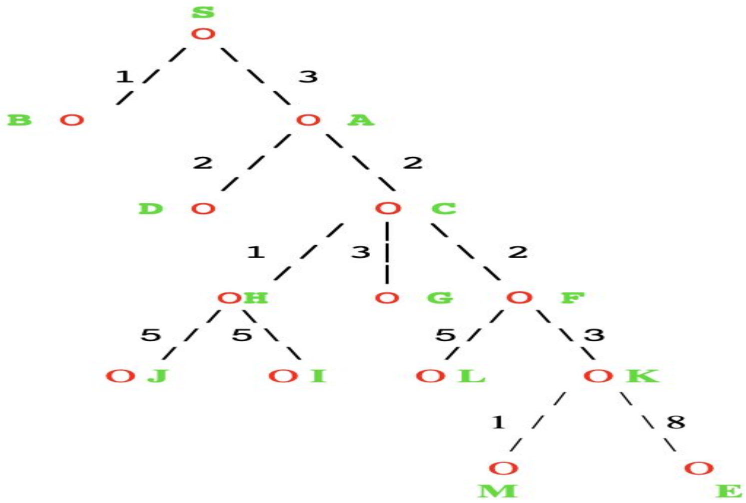


STEP - 4 : Identify the nodes end and find the distance to draw a tree




NORTHWESTERN POLYTECHNIC UNIVERSITY

STEP - 5: Implement tree with vertices and distance between the each vertices.



**B. Below are the detailed steps to find the shortest path from source vertex S to the target vertex E by developing a table**

Here we need to identify the shortest path between two vertices from source vertex which is shortest distance

Let G = (V, E) and vertices or nodes denoted as v or u. Here an edge is denoted as (u,v), and the distance or weight is denoted as w(u,v)

G=(V, E) be a non-negative edges (i.e., $w(u, v) \geq 0$ for each edge $(u, v) \in E$)

By using the function Extract-Min() we extract the node with the smallest key.

**Algorithm: Dijkstra's-Algorithm (G, w, s)**

```
for each vertex v Є G.V
   v.d := ∞
   v.∏ := NIL
s.d := 0
S := Φ
Q := G.V
while Q ≠ Φ
   u := Extract-Min (Q)
   S := S U {u}
   for each vertex v Є G.adj[u]
      if v.d > u.d + w(u, v)
         v.d := u.d + w(u, v)
         v.∏ := u
```

The complexity of this algorithm is fully dependent on the implementation of Extract-Min function. If extract min function is implemented using linear search, The complexity of this algorithm is $\boldsymbol{O(V^2 + E)}$.

For developing a table let us consider vertex S and E as the start and target destination vertex respectively.

Initially, all the vertices except the start vertex S are marked by ∞ and the start vertex S is marked by 0.

# STEPS to implement table and find the shortest path

We are going to determine the path based on predecessor information.
Initial : 0 is smallest cost on Initial step. Thus, S is selected as the starting point for Step 1.

Step-1: S is selected as the starting point for Step 1.

- From S, one can go to B or A
  - The accumulated cost on S is not changed. It is still 0.
  - The accumulated cost on B is 1.
  - The accumulated cost on A is 3.
  - 1 is smaller than 3.
    - Thus, B is selected as the starting point for Step 2.

Step-2: B is selected as the starting point for Step 2.

- From B, one can go to A  as the B is already visited and does not have any further extended path
  - The accumulated cost on S is not changed. It is still 0.
  - The accumulated cost on B is 1.
  - The accumulated cost on A is 3.
  - 3 is smaller as B is already visited and remaining nodes are infinity.
    - Thus, A is selected as the starting point for Step 3.

Step-3: A is selected as the starting point for Step 3.

- From A, one can go to D or C
  - The accumulated cost of D is 5 (S => A => D).
  - The accumulated cost on C is 5(S => A => C).
  - Here both D and C are 5. We can select any of the node and we are taking D
    - Thus, D is selected as the starting point for Step 4.

Step-4: D is selected as the starting point for Step 4.

- From D, cannot go to any other node as none is point away from D so selected C as in the table only C is small and all other are visited nodes.
  - The accumulated cost on C is 5(S => A => C).
  - Here C is the next smallest having 5.
    - Thus, C is selected as the starting point for Step 5.

Step-5: C is selected as the starting point for Step 5.

- From C, one can go to H or G or F
  - The accumulated cost of H is 6(S => A => C => H).
  - The accumulated cost on G is 8(S => A => C => G).
  - The accumulated cost on F is 7(S => A => C => F).
  - Here H is the smallest having 6 among G and F.
    - Thus, H is selected as the starting point for Step 6.

Step-6: H is selected as the starting point for Step 6.

- From H, one can go to I or J
  - The accumulated cost of I is 11(S => A => C => H => I).
  - The accumulated cost on J is 11(S => A => C => H => J).
  - The accumulated cost on G is 8(S => A => C => G).
  - The accumulated cost on F is 7(S => A => C => F).
  - We have F as the next smallest node in the table having 7.
    - Thus, F is selected as the starting point for Step 7.

Step-7: F is selected as the starting point for Step 7.

- From F, one can go to L or K
  - The accumulated cost of I is 11(S => A => C => H => I).
  - The accumulated cost on J is 11(S => A => C => H => J).
  - The accumulated cost on G is 8(S => A => C => G).
  - The accumulated cost on L is 12(S => A => C => F => L).
  - The accumulated cost on K is 10(S => A => C => F => K)
  - We have G as the next smallest node in the table having 8.
    i. Thus, G is selected as the starting point for Step 8.

Step-8: G is selected as the starting point for Step 8.

- From G, one cannot go to any other node as none is point away from G so selected K as in the table only K is small among all unvisited nodes.
  - The accumulated cost of I is 11(S => A => C => H => I).
  - The accumulated cost on J is 11(S => A => C => H => J).
  - The accumulated cost on L is 12(S => A => C => F => L).
  - The accumulated cost on K is 10(S => A => C => F => K)
  - We have K as the next smallest node in the table having 10.
    - Thus, K is selected as the starting point for Step 9.

Step-9: K is selected as the starting point for Step 9.

- From K, one can go to M or E.
  - The accumulated cost of I is 11(S => A => C => H => I).
  - The accumulated cost on J is 11(S => A => C => H => J).
  - The accumulated cost on L is 12(S => A => C => F => L).
  - The accumulated cost on M is 11(S => A => C => F => K => M)
  - The accumulated cost on E is 18(S => A => C => F => K => E)
  - We have M, I, J as the next smallest node in the table having 11. We can select any of the node and we are taking M.
    - Thus, M is selected as the starting point for Step 10.

Step-10: M is selected as the starting point for Step 10.

- From M, one cannot go to any other node as none is point away from M, we can select I or J, as in the table I and J are small among all unvisited nodes but alphabetically we are selecting I.
  - The accumulated cost of I is 11(S => A => C => H => I).
  - The accumulated cost on J is 11(S => A => C => H => J).
  - The accumulated cost on L is 12(S => A => C => F => L).
  - The accumulated cost on E is 18(S => A => C => F => K => E)
  - We have I, J as the next smallest nodes in the table having 11. We can select any of the node and we are taking I.
    - Thus, I is selected as the starting point for Step 11.

Step-11: I is selected as the starting point for Step 11.

- From M, one cannot go to any other node as none is point away from I so selected J, as in the table only J is small among all unvisited nodes.
  - The accumulated cost on J is 11(S => A => C => H => J).
  - The accumulated cost on L is 12(S => A => C => F => L).
  - The accumulated cost on E is 18(S => A => C => F => K => E)
  - J is the next smallest nodes in the table having 11. We can select J.
    - Thus, J is selected as the starting point for Step 12.

Step-12: J is selected as the starting point for Step 12.

- From J, one cannot go to any other node as none is point away from J so selected L, as in the table only L is small among all unvisited nodes.
  - The accumulated cost on L is 12(S => A => C => F => L).
  - The accumulated cost on E is 18(S => A => C => F => K => E)
  - L is the next smallest nodes in the table having 12. We can select L.
    - Thus, L is selected as the starting point for Step 13.

Step-13: L is selected as the starting point for Step 13.

- From L, one cannot go to any other node as none is point away from L so selected E, as in the table onlv E is left.
  - The accumulated cost on E is 18(S => A => C => F => K => E).

# Implemented table to find shortest path

| Vertex | Initial | Step 1 S — Next Step B | Step 2 B (S) — Next Step A | Step3 A (S, B) — Next Step D | Step4 D (S, B, A) — Next Step C | Step5 C (S, B, A, D) — Next Step H | Step6 H (S, B, A, D) — Next Step F | Step7 F (S, B, A, D, H) — Next Step G | Step8 G (S, B, A, D, H, F) — Next Step K | Step9 K (S, B, A, D, H, F, G) — Next Step M | Step10 M (S, B, A, D, H, F, G, K) — Next Step I | Step11 I (S, B, A, D, H, F, G, K, M) — Next Step J | Step12 J (S, B, A, D, H, F, G, K, M, I) — Next Step L | Step13 L \| (S, B, A, D, H, F, G, K, M, I, J, L) — Ends at Step E |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | 0 | 0 | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ |
| A | ∞ | 3 | 3 | 3 | ~~3~~ | ~~3~~ | ~~3~~ | ~~3~~ | ~~3~~ | ~~3~~ | ~~3~~ | ~~3~~ | ~~3~~ | ~~3~~ |
| B | ∞ | 1 | 1 | ~~1~~ | ~~1~~ | ~~1~~ | ~~1~~ | ~~1~~ | ~~1~~ | ~~1~~ | ~~1~~ | ~~1~~ | ~~1~~ | ~~1~~ |
| C | ∞ | ∞ | ∞ | 5 | 5 | 5 | ~~5~~ | ~~5~~ | ~~5~~ | ~~5~~ | ~~5~~ | ~~5~~ | ~~5~~ | ~~5~~ |
| D | ∞ | ∞ | ∞ | 5 | 5 | ~~5~~ | ~~5~~ | ~~5~~ | ~~5~~ | ~~5~~ | ~~5~~ | ~~5~~ | ~~5~~ | ~~5~~ |
| F | ∞ | ∞ | ∞ | ∞ | ∞ | 7 | 7 | 7 | ~~7~~ | ~~7~~ | ~~7~~ | ~~7~~ | ~~7~~ | ~~7~~ |
| G | ∞ | ∞ | ∞ | ∞ | ∞ | 8 | 8 | 8 | 8 | ~~8~~ | ~~8~~ | ~~8~~ | ~~8~~ | ~~8~~ |
| H | ∞ | ∞ | ∞ | ∞ | ∞ | 6 | 6 | ~~6~~ | ~~6~~ | ~~6~~ | ~~6~~ | ~~6~~ | ~~6~~ | ~~6~~ |
| I | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 11 | 11 | 11 | 11 | 11 | 11 | ~~11~~ | ~~11~~ |
| J | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 11 | 11 | 11 | 11 | 11 | 11 | 11 | ~~11~~ |
| K | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 10 | 10 | 10 | ~~10~~ | ~~10~~ | ~~10~~ | ~~10~~ |
| L | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| M | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 11 | 11 | ~~11~~ | ~~11~~ | ~~11~~ |
| E | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 18 | 18 | 18 | 18 | 18 |

- V: the current visiting node
- V: the next node to visit
- ~~V~~: this node has been visited

# Conclusion

Hence, the minimum distance from S to E in the maze is 18

*The path is S → A→ C→ F→ K→ E

* Distance is 3 → 2 → 2 → 3 → 8

Note : The shortest path can be found by either Dijkstra's Algorithm or Bellman Ford Algorithm.

- Dijkstra's algorithm only works for the edges which are non-negative whereas Bellman Ford Algorithm works for both negative and non-negative edges.

# References


- https://npu85.npu.edu/~henry/npu/classes/algorithm/tutorialpoints_daa/slide/shortest_paths.html
- https://npu85.npu.edu/~henry/npu/classes/algorithm/graph_alg/slide/maze.html


Google slides URL :

https://docs.google.com/presentation/d/1JbDMnSiinAii8PDmttBEmT9bl0mP7SUp18YGTKBCWYw/edit#slide=id.gcbd2d89a56_0_376