
NORTHWESTERN POLYTECHNIC
UNIVERSITY

# Maze Project

**Prepared For:**

Mr. Henry Chaung
Algorithms & Structured Programming CS455
Spring 2021
Northwestern Polytechnic University

**Prepared By:**

Ms. Nagalla, Santhi Sree ID:19568

# Table of Contents

# Introduction


- Maze solving is the act of finding a route through the maze from the start to finish. Some maze solving methods are designed to be used inside the maze by a traveler with no prior knowledge of the maze, whereas others are designed to be used by a person or computer program that can see the whole maze at once.

# Maze-solving algorithms


- There are a number of different maze-solving algorithms, that is, automated methods for the solving of mazes.
  - ➢ Wall follower
  - ➢ Random mouse algorithm
  - ➢ Pledge algorithm
  - ➢ Trémaux's algorithm
  - ➢ Dead-end filling

# Maze-solving algorithms - Continues

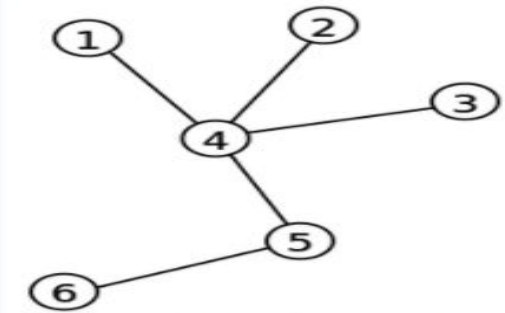

- ➢ Maze-routing algorithm
- ➢ Recursive algorithm
- ➢ Shortest path algorithm

- Mazes containing no loops are known as "simply connected", or "perfect" mazes, and are equivalent to a tree in graph theory. Thus many maze-solving algorithms are closely related to graph theory.

# What is a Graph Theory?

- In graph theory, a tree is an undirected graph in which any two vertices are connected by exactly one path, or equivalently a connected acyclic undirected graph. A forest is an undirected graph in which any two vertices are connected by at most one path, or equivalently an acyclic undirected graph, or equivalently a disjoint union of trees.

**Trees**




A labeled tree with 6 vertices and 5 edges.

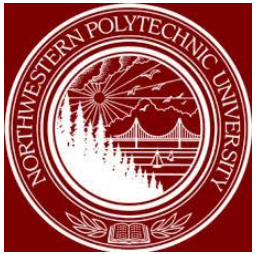
| | |
|---|---|
| **Vertices** | $v$ |
| **Edges** | $v - 1$ |
| **Chromatic number** | 2 if $v > 1$ |

Table of graphs and parameters

# Shortest path algorithm


- When a maze has multiple solutions, the solver may want to find the shortest path from start to finish. One such algorithm finds the shortest path by implementing a breadth-first search, while another, the A* algorithm, uses a heuristic technique. The breadth-first search algorithm uses a queue to visit cells in increasing distance order from the start until the finish is reached.

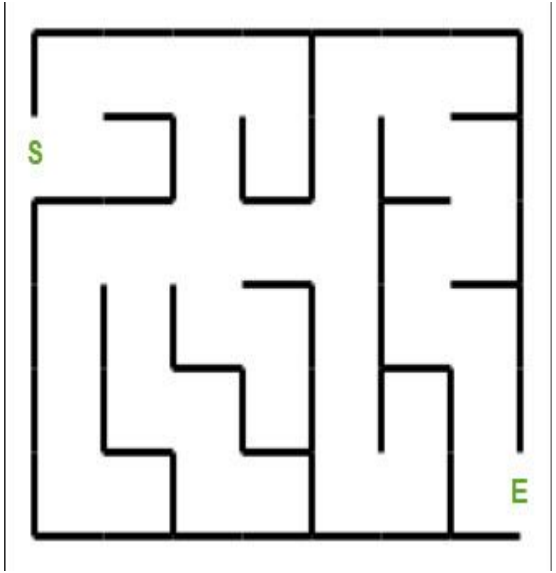
# Dijkstra’s Algorithm




- One algorithm for finding the shortest path from a starting node to a target node in a weighted graph is **Dijkstra's algorithm**.The algorithm creates a tree of shortest paths from the starting vertex, the source, to all other points in the graph.Each visited cell needs to keep track of its distance from the start or which adjacent cell nearer to the start caused it to be added to the queue. When the finish location is found, follow the path of cells backwards to the start, which is the shortest path. The breadth-first search in its simplest form has its limitations, like finding the shortest path in weighted graphs.

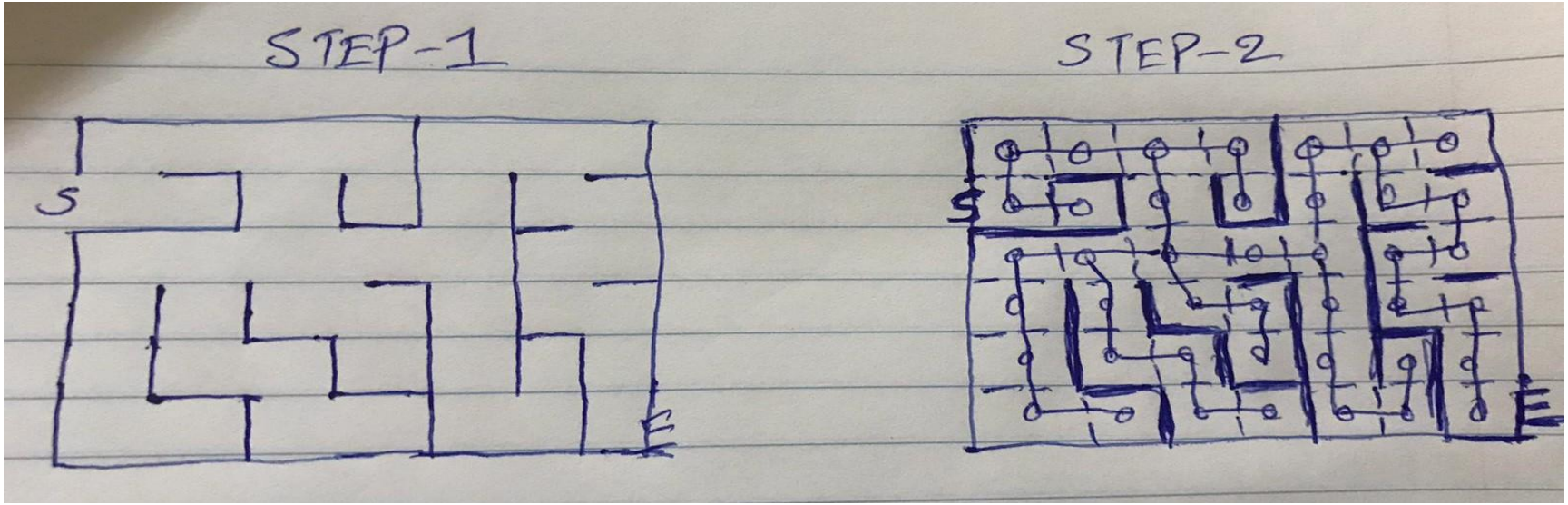
# Steps to follow Shortest-Path

- Initialize distances according to the **algorithm**.
- Pick first node and **calculate** distances to adjacent nodes.
- Pick next node with minimal **distance** repeat adjacent node **distance** calculations.
- Final result of **shortest-path** tree.



# Dijkstra's Algorithm to find the shortest path

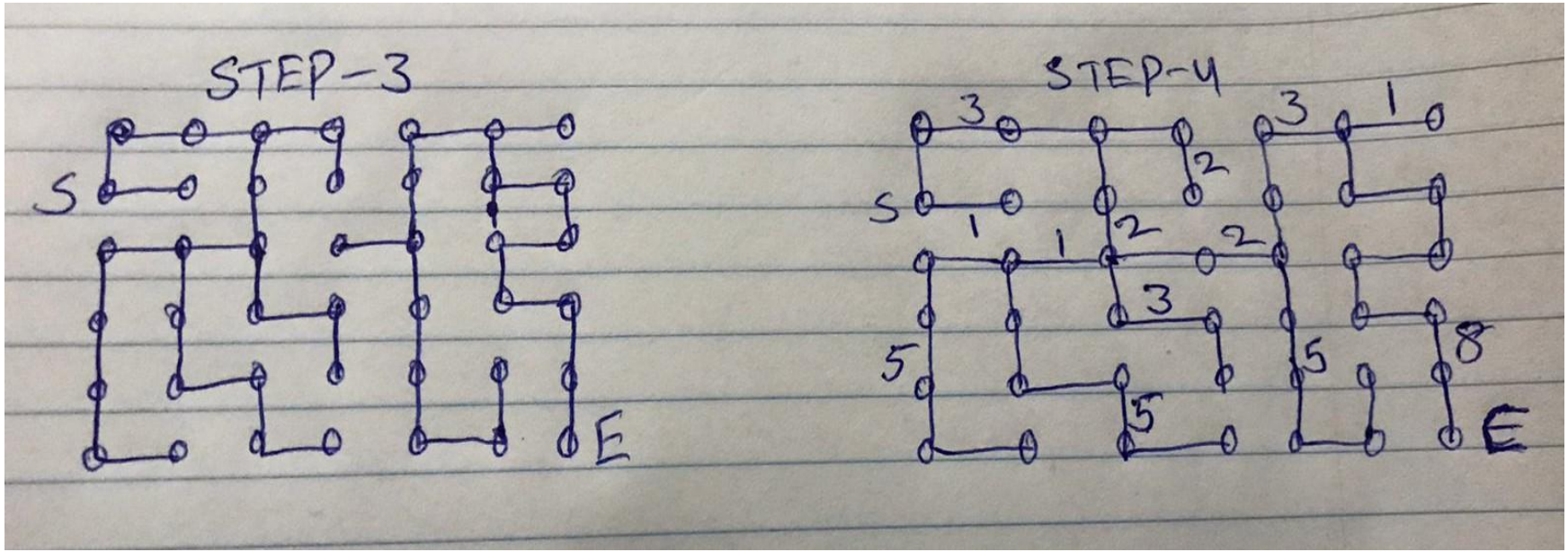
Step 1: Initialize distances according to the algorithm.



# Dijkstra's Algorithm to find the shortest path

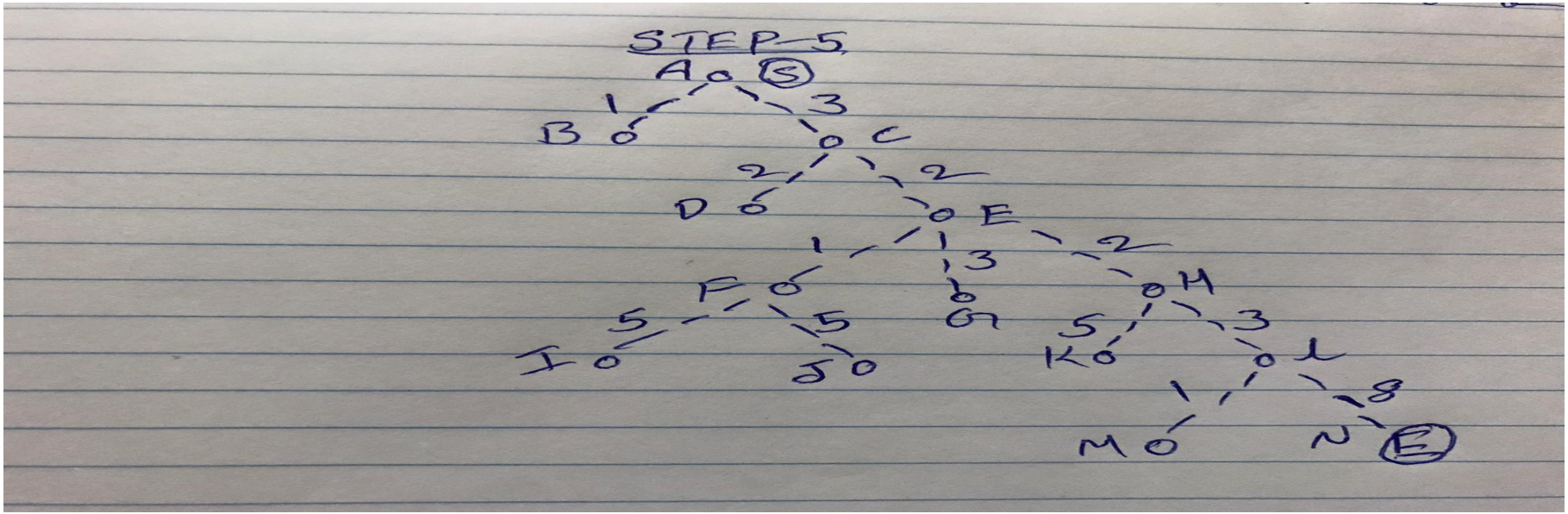
Step 2: Pick first node and calculate distances to adjacent nodes.

Step 3: Pick next node with minimal distance repeat adjacent node distance calculations.



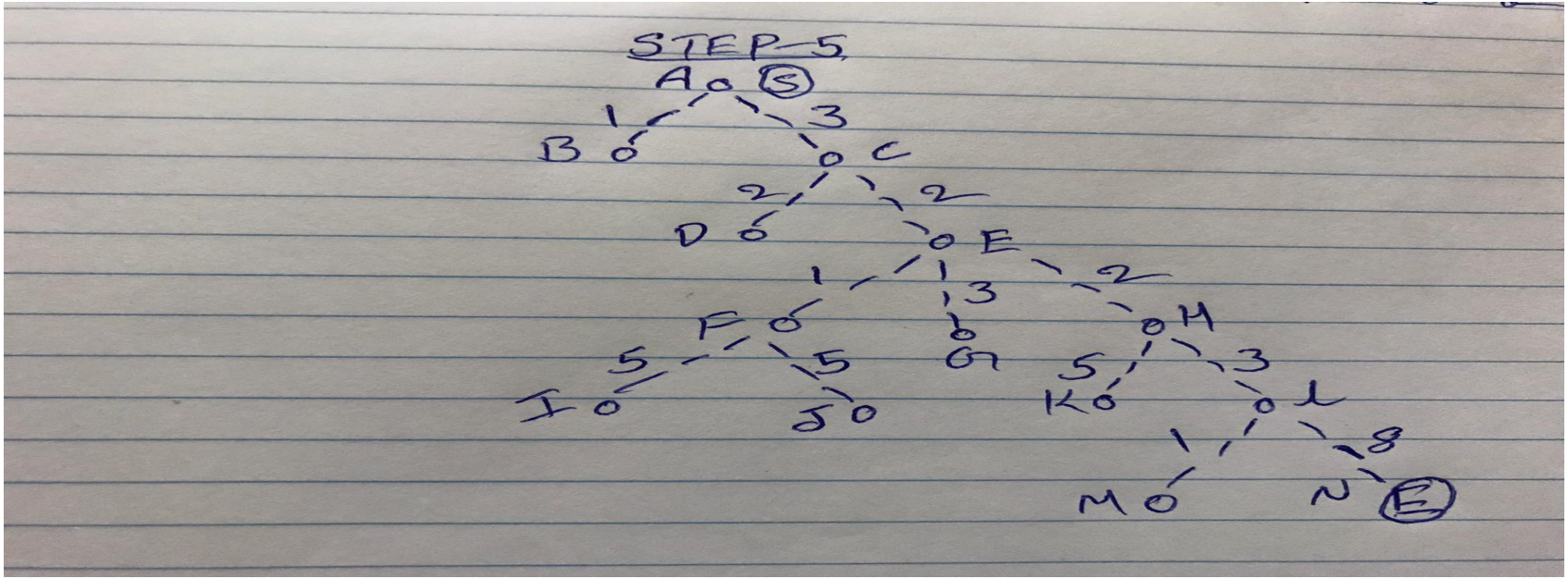
# Tree

- A tree is an undirected graph in which any two vertices are connected by exactly one path, or equivalently a connected acyclic undirected graph



# Find the Path and Distance using Tree

- This path is determined based on predecessor information.



# Find the minimum distance using Table

| Vertex (accumulated path) | Initial ( ) Next Step A | Step 1 -A (A) Next Step B | Step 2- B (A,B) Next Step C | Step 3 -C (A,B, C) Next Step D | Step 4- D (A,B, C,D) Next Step E | Step 5- E (A,B, C,D, E) Next Step F | Step 6 -F (A,B, C,D, E,F) Next Step H | Step 7 -H (A,B, C,D, E,F, H) Next Step G | Step 8- G(A, B,C, D,E, F,H, G) Next Step L | Step 9-L (A,B, C,D, E,F, G,H, L) Next Step M | Step 10-M (A,B, C,D, E,F, G,H, L,M) Next Step I | Step 11-I (A,B, C,D, E,F, G,H, L,M, I) Next Step K | Step 12- K(A, B,C, D,E, F,G, H,L, M,K, I) End at N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B | ∞ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| C | ∞ | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| D | ∞ | ∞ | ∞ | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| E | ∞ | ∞ | ∞ | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| F | ∞ | ∞ | ∞ | ∞ | ∞ | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| G | ∞ | ∞ | ∞ | ∞ | ∞ | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| H | ∞ | ∞ | ∞ | ∞ | ∞ | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| I | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| J | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| K | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 12 | 12 | 12 | 12 | 12 | 12 |
| L | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 10 | 10 | 10 | 10 | 10 | 10 |
| M | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 11 | 11 | 11 | 11 |
| N | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | 18 | 18 | 18 | 18 |

# Steps to follow Minimum Distance


- Let us consider vertex **A** and **N** as the start and destination vertex respectively.
- Initially, all the vertices except the start vertex **A** are marked by ∞ and the start vertex **A** is marked by ***0***.
  - **Step X** - represents: the current visiting node
  - **Next Step** represents: the next node to visit
  - ~~V~~: this node has been visited

  - Initial
    - 0 is smallest cost on Initial step.
      - Thus, **A** is selected as the starting point for Step 1.

# Steps - Continues

- Step 1
  - **A** is selected as the starting point for Step 1.
    - From **A**, one can go to **A or B or C**
      - The accumulated cost on **A** is not changed. It is still 0.
      - The accumulated cost on **B** is **1.**
      - The accumulated cost on **C** is **3.**
      - **1** is smaller than **3**.
        - Thus, **B** is selected as the starting point for Step 2.
  - Stop if the destination node **N** is reached
- Continue this process, you will find the minimum distance of **N** from ***A*** is **18**. And the path is

**3→ 2→ 2→ 3→ 8**

# CONCLUSION AND FUTURE WORK

- Maze solving problem used different algorithms to solve this problem. Using a good algorithm can get a high performance led to find a best shortest route in short time. The proposed technique based on the Flood fill algorithm works better and with less searching time. Using camera in this work avoids the algorithm step by executing them in the computer using image processing technique rather than using them by a robot.

# References:

- Maze

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

- Shortest Path

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

- Google Slides URL -

  https://docs.google.com/presentation/d/1l40XOId-ty1Abin3VaeqFXEESE9-xY1rFSoFSUXtlPQ/edit?usp=sharing