Week 7 21.08.2025

Graphs

Question 1: Maze Solver - BFS vs. DFS

You are given a 2D grid representing a maze.

- 0 represents a walkable path.
- 1 represents an impassable wall.
- S represents the Start point.
- E represents the Exit point.

Your task is to write a program that:

1. **Reads the maze layout from the user.**
2. **Finds a path from S to E using BFS.**
    - Implement BFS using a queue to explore the grid level-by-level.
    - The algorithm must track the parent of each node to reconstruct the shortest path (in terms of steps) upon finding the exit.
3. **Finds a path from S to E using DFS.**
    - Implement DFS using a stack (iterative approach).
    - The algorithm will find *a* path, but it may not be the shortest.
4. **Output:**
    - Print the maze, visually marking the discovered path for both BFS and DFS

Assumptions:

- Maze is represented as a 2D grid.
- 'S' = Start, 'E' = Exit, '0' = Path, '1' = Wall.
- Movements are allowed only in 4 directions: Up, Down, Left, Right.

Test Case 1: Simple Straight Path (Sanity Check)

Purpose: To verify basic functionality and correct pathfinding in a trivial case.
Maze:

```
S 0 0 E
```

Expected BFS Path: (0,0) -> (0,1) -> (0,2) -> (0,3) Length: 3 steps
Expected DFS Path: (0,0) -> (0,1) -> (0,2) -> (0,3) Length: 3 steps
Expected Output: Both algorithms should find the same, shortest path. The path should be visually printed on the grid.

Test Case 2: Maze with a Dead End

Purpose: To test if the algorithms can backtrack correctly. DFS should go into the dead end, while BFS should not.
Maze:

```
S 0 0 0
1 1 1 0
E 0 0 0
```

Layout:

```
Row0: S, 0, 0, 0
Row1: 1, 1, 1, 0
Row2: E, 0, 0, 0
```

Expected BFS Path (Shortest):
(0,0) -> (0,1) -> (0,2) -> (0,3) -> (1,3) -> (2,3) -> (2,2) -> (2,1) Length: 7 steps

Expected DFS Path (May be longer):
(0,0) -> (0,1) -> (0,2) -> (0,3) -> (1,3) -> (2,3) -> (2,2) -> (2,1) Length: 7 steps
OR (depending on movement priority, e.g., Right before Down)
(0,0) -> (0,1) -> (0,2) -> (0,3) -> (1,3) -> (2,3) -> (2,2) -> (2,1) Length: 7 steps

Test Case 3: Maze with a Clear Fork and Dead End
Purpose: To clearly demonstrate the difference in behavior between BFS and DFS. BFS should find the shortest path. DFS will likely find a longer path by exploring the dead end first.
Maze:
```
S 0 1 0 0 0
0 1 1 1 1 0
0 0 0 0 0 E
```
Layout:
```
Row0: S, 0, 1, 0, 0, 0
Row1: 0, 1, 1, 1, 1, 0
Row2: 0, 0, 0, 0, 0, E
```
Expected BFS Path (Shortest): BFS will expand uniformly. The path going down immediately will reach the exit fastest.
(0,0) -> (1,0) -> (2,0) -> (2,1) -> (2,2) -> (2,3) -> (2,4) -> (2,5) Length: 7 steps
Expected DFS Path (Likely longer): If movement priority is Down > Right > Up > Left, DFS will go right first into the dead end, backtrack, and then go down.
(0,0) -> (0,1) -> (0,3) -> (0,4) -> (0,5) -> (1,5)
...hits a wall, must backtrack to (0,0)...
(0,0) -> (1,0) -> (2,0) -> (2,1) -> (2,2) -> (2,3) -> (2,4) -> (2,5) Length: 13+ steps

Test Case 4: No Solution
Maze:
```
S 0 1
1 1 1
0 0 E
```
Expected BFS Output: A message like "BFS: No path exists to the exit."
Expected DFS Output: A message like "DFS: No path exists to the exit."

Question 2: Network Router Simulation (Dijkstra's Algorithm)
You are to simulate a network router that calculates the shortest path for data packets based on link latency (delay). The network is represented as a graph where:

- **Routers** are nodes (labeled alphabetically or numerically).
- **Network links** are edges.
- **Latency** (in milliseconds) is the weight of each edge.

Your task is to write a program that:

1. Represents the weighted network graph using an adjacency matrix or list. You can hardcode the following network:
    - A connected to B with latency 4, and to C with latency 2.
    - B connected to C with latency 1, and to D with latency 5.
    - C connected to D with latency 8, and to E with latency 10.
    - D connected to E with latency 2.
    - E has no outgoing connections.

2. Implements Dijkstra's algorithm to compute the shortest path (lowest total latency) from a given source router (e.g., A) to all other routers in the network.

3. Output:

- For each router, print the shortest latency from the source and the full path taken.
- Example output: Shortest path to D: A -> B -> D (Total latency: 9 ms)

****