Algorithms

BFS Homework 1

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

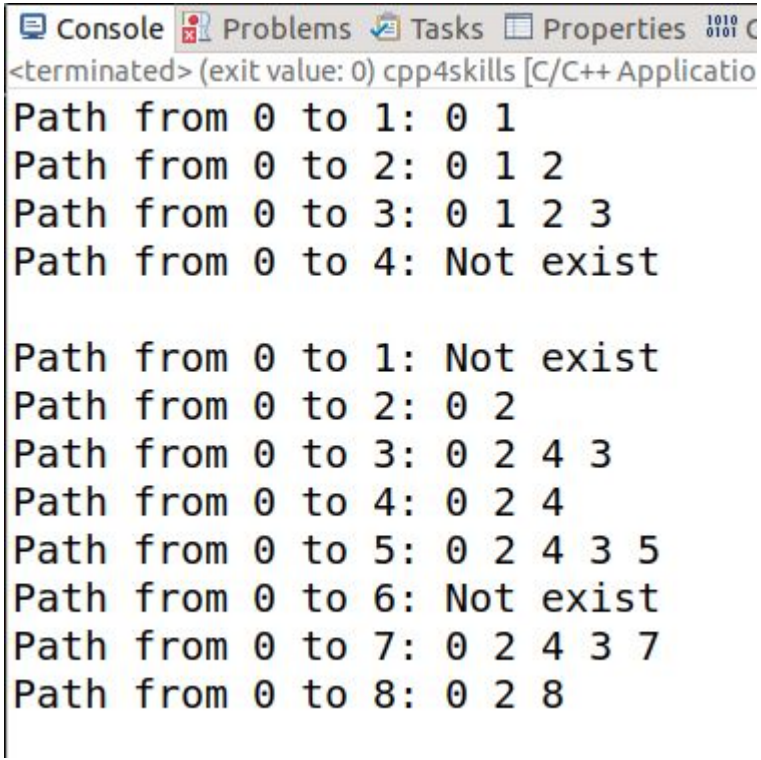
Problem #1: Print Path

- Modify the lecture code to print the path from the starting node (0) to every other node
- Reading
  - Read number of test cases
  - Read a directed graph as usual (nodes edges and list of edges
- Print as shown

```
2

5 4
0 1
1 2
2 3
4 3

9 13
1 3
1 5
1 6
3 5
4 3
3 7
5 4
6 0
2 4
2 8
0 2
2 8
2 2
```



Problem #2: LeetCode 261 - Graph Valid Tree

261. Graph Valid Tree

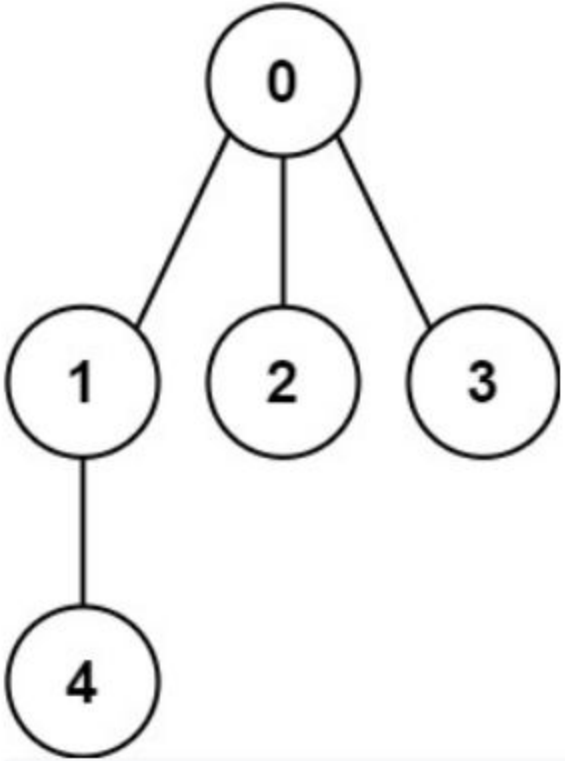
Medium 1891 53 Add to List Share

You have a graph of `n` nodes labeled from `0` to `n - 1`. You are given an integer n and a list of `edges` where `edges[i] = [a_i, b_i]` indicates that there is an undirected edge between nodes `a_i` and `b_i` in the graph.

Return `true` *if the edges of the given graph make up a valid tree, and* `false` *otherwise.*

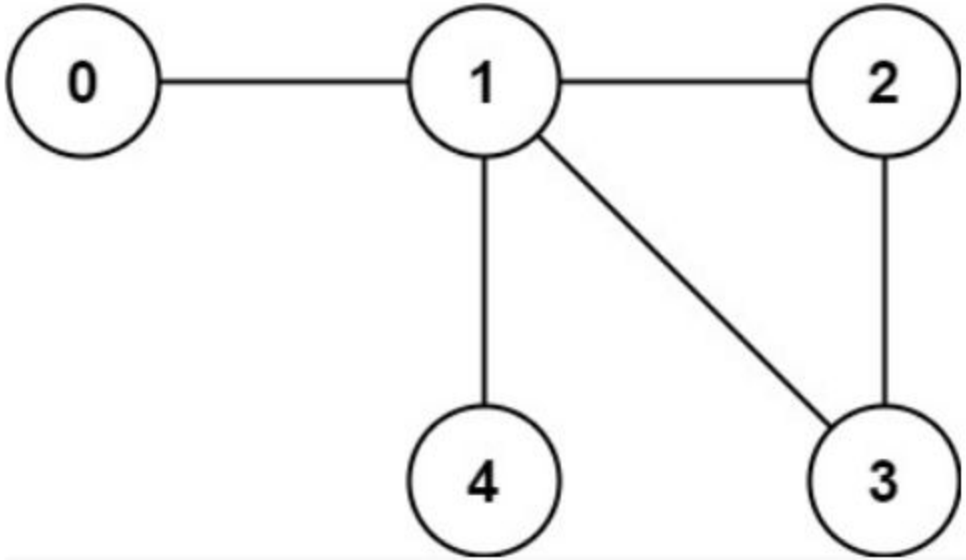
- C++: bool validTree(int nodes, vector<vector<int>> &edges)
- Java: public boolean validTree(int n, int[][] edges)
- Python: def validTree(self, n: int, edges: List[List[int]]) -> bool
- Javascript: var validTree = function(n, edges)
- no self-loops or repeated edges
- **Use BFS**

Example 1:




```
Input: n = 5, edges = [[0,1],[0,2],[0,3],[1,4]]
Output: true
```

Example 2:




```
Input: n = 5, edges = [[0,1],[1,2],[2,3],[1,3],[1,4]]
Output: false
```

Problem #3: LeetCode 1730 - Shortest Path to Get Food

You are starving and you want to eat food as quickly as possible. You want to find the shortest path to arrive at any food cell.

You are given an `m x n` character matrix, `grid`, of these different types of cells:

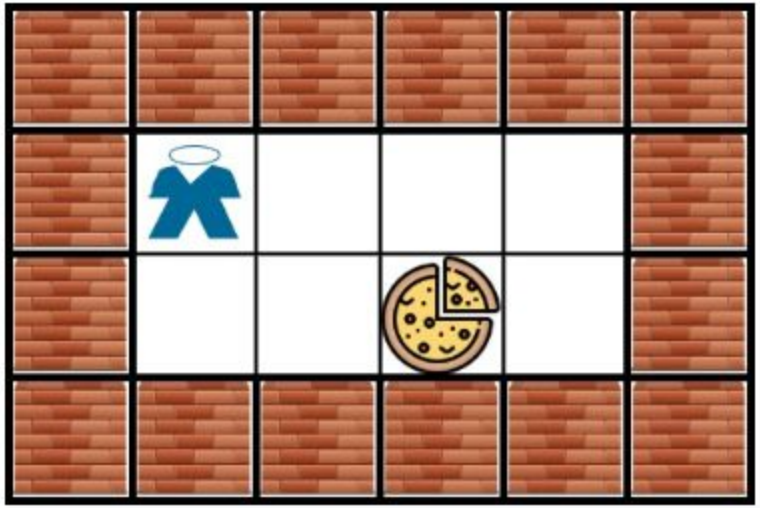
- `'*'` is your location. There is **exactly one** `'*'` cell.
- `'#'` is a food cell. There may be **multiple** food cells.
- `'O'` is free space, and you can travel through these cells.
- `'X'` is an obstacle, and you cannot travel through these cells.

You can travel to any adjacent cell north, east, south, or west of your current location if there is not an obstacle.

Return *the **length** of the shortest path for you to reach **any** food cell*. If there is no path for you to reach food, return `-1`.

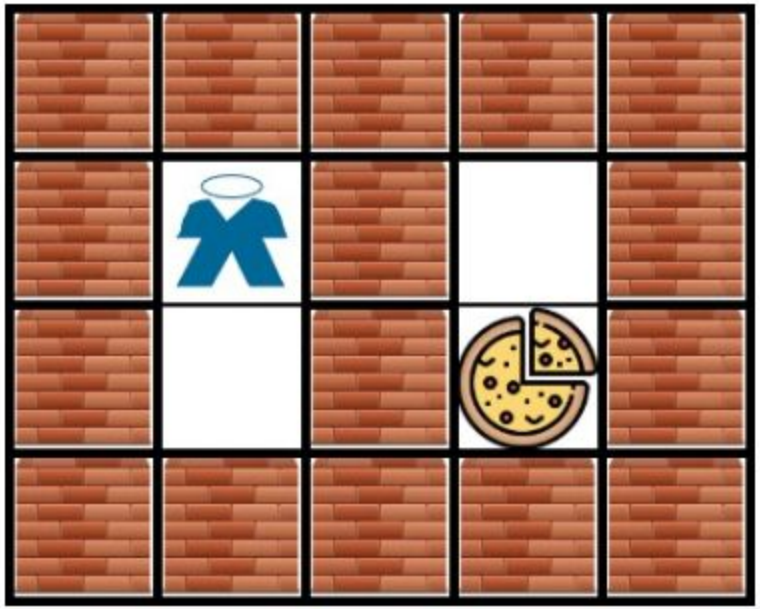
- C++: int getFood(vector<vector<char>> &matrix)
- Java: public int getFood(char[][] grid)
- Python: def getFood(self, grid: List[List[str]]) -> int:
- Javascript: var getFood = function(grid)

Example 1:



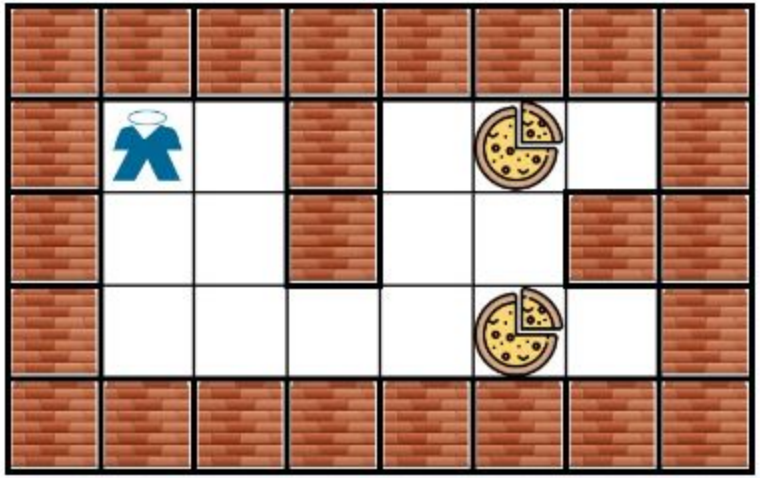
Input: grid = [["X","X","X","X","X","X"],["X","*","O","O","O","X"],
["X","O","O","#","O","X"],["X","X","X","X","X","X"]]
Output: 3
Explanation: It takes 3 steps to reach the food.

Example 2:



Input: grid = [["X","X","X","X","X"],["X","*","X","O","X"],["X","O","X","#","X"],["X","X","X","X","X"]]
Output: -1
Explanation: It is not possible to reach the food.

Example 3:



Input: grid = [["X","X","X","X","X","X","X","X"],["X","*","O","X","O","#","O","X"],
["X","O","O","X","O","O","X","X"],["X","O","O","O","O","#","O","X"],
["X","X","X","X","X","X","X","X"]]

Output: 6

Explanation: There can be multiple food cells. It only takes 6 steps to reach the bottom food.

Example 4:

```
Input: grid = [["O","*"],["#","O"]]
Output: 2
```

Example 5:

```
Input: grid = [["X","*"],["#","X"]]
Output: -1
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."