# 1926. Nearest Exit from Entrance in Maze

You are given an m x n matrix maze (0-indexed) with empty cells (represented as '.') and walls (represented as '+'). You are also given the entrance of the maze, where entrance = [entrancerow, entrancecol] denotes the row and column of the cell you are initially standing at.

In one step, you can move one cell up, down, left, or right. You cannot step into a cell with a wall, and you cannot step outside the maze. Your goal is to find the nearest exit from the entrance. An exit is defined as an empty cell that is at the border of the maze. The entrance does not count as an exit.

Return the number of steps in the shortest path from the entrance to the nearest exit, or -1 if no such path exists

.

## SOLUTION IN JAVA

class Solution {

public int nearestExit(char[][] maze, int[] entrance) {

final int[][] dirs = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};

final int m = maze.length;

final int n = maze[0].length;

int ans = 0;

Queue<int[]> q = new ArrayDeque<>(Arrays.asList(new int[] {entrance[0], entrance[1]}));

boolean[][] seen = new boolean[m][n];

seen[entrance[0]][entrance[1]] = true;

while (!q.isEmpty()) {

++ans;

for (int sz = q.size(); sz > 0; --sz) {

final int i = q.peek()[0];

final int j = q.poll()[1];

for (int[] dir : dirs) {

final int x = i + dir[0];

final int y = j + dir[1];

if (x < 0 || x == m || y < 0 || y == n)

continue;

if (seen[x][y] || maze[x][y] == '+')

continue;

if (x == 0 || x == m - 1 || y == 0 || y == n - 1)

return ans;

q.offer(new int[] {x, y});

seen[x][y] = true;

}

}

}

return -1;

}

}