**2100030132(SECTION-10)**

# GLOBAL CHALLENGE-1

1.BOT SAVES PRINCESS

#include <stdio.h>

#include <string.h> #include <math.h>

**void** displayPathtoPrincess(**int** n, **char** grid[101][101]){

}

**int** main(**void**) { **int** m; scanf("%d", &m); **char** grid[101][101]={}; **char** line[101]; **for**(**int** i=0; i<m; i++) { scanf("%s", line); strcpy(grid[i], line);

}

displayPathtoPrincess(m,grid); **return** 0; }

## 2.BOT SAVES PRINCESS-2

#include <stdio.h>

#include <string.h> #include <math.h>

**void** nextMove(**int** n, **int** r, **int** c, **char** grid[101][101]){

}

**int** main(**void**) { **int** n, r, c; scanf("%d", &n); scanf("%d", &r); scanf("%d", &c);

**char** grid[101][101];

**for**(**int** i=0; i<n; i++) { scanf("%s[^\n]%\*c", grid[i]);

}

nextMove(n, r, c, grid); **return** 0; }

## 3.PACMAN-DFS

#include <stdio.h>

#include <string.h> #include <math.h>

**void** dfs(**int** r, **int** c, **int** pacman\_r, **int** pacman\_c, **int** food\_r, **int** food\_c, **char** grid[r][c]){

}

**int** main(**void**) { **int** r, c; **int** pacman\_r, pacman\_c; **int** food\_r, food\_c;

scanf( "%d %d", &pacman\_r, &pacman\_c); scanf( "%d %d", &food\_r, &food\_c); scanf( "%d %d", &r, &c); **char** grid[r][c]; **for**( **int** i=0; i<r; i++) { scanf("%s[^\\n]%\*c", grid[i]);

}

dfs( r, c, pacman\_r, pacman\_c, food\_r, food\_c, grid); **return** 0; }

## 4.PACMAN-A\*

**private** Node UCS() {

Queue<Node> queue = **new** PriorityQueue<Node>(X \* Y, **new** Comparator<Node>() {

@Override

**public** **int** compare(Node node1, Node node2) {

**return** node1.distance() + cost(node1) - (node2.distance() + cost(node2));

}

});

**boolean**[][] explored = **new** **boolean**[X][Y];

queue.add(**new** Node(pacman));

**while** (!queue.isEmpty()) { Node node = queue.poll();

**int** x = node.position()[0]; **int** y = node.position()[1];

**if** (explored[x][y]) { **continue**; }

explored[x][y] = **true**;

**if** (node.isFood()) { **return** node;

}

**if** (isValidMove(x - 1, y) && !explored[x - 1][y]) { queue.add(**new** Node(**new** **int**[] {x - 1, y}, node));

}

**if** (isValidMove(x, y - 1) && !explored[x][y - 1]) { queue.add(**new** Node(**new** **int**[] {x, y - 1}, node));

}

**if** (isValidMove(x, y + 1) && !explored[x][y + 1]) { queue.add(**new** Node(**new** **int**[] {x, y + 1}, node));

}

**if** (isValidMove(x + 1, y) && !explored[x + 1][y]) { queue.add(**new** Node(**new** **int**[] {x + 1, y}, node));

} }

**return** **null**;

}

**private** **boolean** isValidMove(**int** x, **int** y) {

**return** x >= 0 && y >= 0 && x < X && y < Y && board[x][y] != WALL;

}

**private** **int** cost(Node node) {

**return** (node.position()[0] > food[0] ? node.position()[0] - food[0] : food[0] - node.position()[0]) +

(node.position()[1] > food[1] ? node.position()[1] - food[1] : food[1]

- node.position()[1]);

}

**private** Node(**int**[] position, Node parent) { **this**.position = position; **this**.parent = parent;

**this**.distance = parent.distance() + 1;

}

## 5.PACMAN-BFS

#include <stdio.h>

#include <string.h> #include <math.h>

**void** nextMove(**int** r, **int** c, **int** pacman\_r, **int** pacman\_c, **int** food\_r, **int** food\_c, **char** grid[r][c]){

}

**int** main(**void**) { **int** r, c; **int** pacman\_r, pacman\_c; **int** food\_r, food\_c;

scanf( "%d %d", &pacman\_r, &pacman\_c); scanf( "%d %d", &food\_r, &food\_c); scanf( "%d %d", &r, &c); **char** grid[r][c]; **for**( **int** i=0; i<r; i++) { scanf("%s[^\\n]%\*c", grid[i]);

}

nextMove( r, c, pacman\_r, pacman\_c, food\_r, food\_c, grid); **return** 0;

}