

Project Report on

**Rat in a maze**

Course Title – Data Structures

Course Code – CSE207

Section - 06

**Submitted To:**

Md. Manowarul Islam

Adjunct Faculty

Department of Computer Science and Engineering

**Submitted By:**

Shamil Bin Hossain Noor (**2021-2-60-102**)

Redown Ahmed (**2022-1-60-159**)

Tasnuva Tasnim Nova (**2022-1-60-266**)

Abdul Mumeet Pathan (**2022-1-60-267**)

Department of Computer Science and Engineering

**Date of Submission:** 16 September 2023

**Declaration**

We, hereby declare that the work demonstrated in this assessment has been done by us and it represents our own work.

**Acknowledgment**

We would like to extend our heartfelt gratitude to our course instructor, Md. Manowarul Islam sir, for his guidance and mentorship throughout this project, and our peers for their unwavering support during the completion of the Boost Converter Circuit project. Additionally, we appreciate the resources provided by our institution, the online communities, and component suppliers. This collaborative effort has been instrumental in the successful execution of the project, and we look forward to applying the knowledge gained in future endeavors.

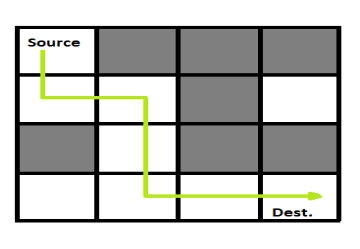
**Introduction**

A "rat in a maze" is a classic metaphorical scenario often used in various contexts, including computer science, psychology, and decision-making studies. It refers to a simple yet illustrative problem where a hypothetical rat is placed within a maze and must navigate through it to reach a specific goal or exit point.

**Project Description**

A Maze is given as N\*N binary matrix of blocks where source block is the upper left most block i.e., maze [0][0] of the above figure and destination block is lower rightmost block i.e., maze [N-1][N-1]. A rat starts from source and has to reach destination. The rat can move in four directions: right, left, top and down. In the maze matrix, 0 means the block is dead end and 1 means the block can be used in the path from source to destination. We have used matrix for the representation of the maze and graph to figure out the movement of the rat where the rat cannot move toward the dead block.

This project shows the simulation of a mouse’s movement through the maze and shows the path through the maze moved by the mouse.



**Explanation of code**

1. It includes necessary header files: <bits/stdc++.h> and <windows.h>.
2. The di and dj arrays define the possible directions (down, left, right, up) for movement in the maze.
3. The pos\_way variable is used to keep track of the number of possible paths found in the maze.
4. The maze array represents the maze grid, where 1 represents a usable block, and 0 represents a dead end.
5. The vis array is used to mark visited cells in the maze.
6. The ans vector will store the paths found from the source to destination.
7. The solve function is a recursive function that explores all possible paths in the maze using backtracking.
8. Inside the solve function, it checks if the current cell is the destination. If so, it displays the path, updates the pos\_way counter, and adds the path to the ans vector.
9. It then recursively explores the adjacent cells in all four directions, marking visited cells and continuing the search.
10. The main function initializes the maze, takes input for the maze size, and populates the maze grid.
11. It provides a visual representation of the maze layout with specific colors and symbols for different cell types (current position, visited position, dead end).
12. It calls the solve function to find paths from the source (0,0) to the destination (n-1, n-1).
13. Finally, it prints the number of paths found or a message if no path exists.
14. The program also uses Windows console commands to display colorful output for better visualization of the maze and the rat's path. The colors and symbols used are explained in comments in the code.

Overall, this code is a maze-solving program that finds and displays all possible paths from the source to the destination ina maze grid.

**Source Code**

#include <bits/stdc++.h>

#include <windows.h>

using namespace std;

int di[] = {1, 0, 0, -1};

int dj[] = {0, -1, 1, 0};

int pos\_way = 0;

int maze[1000][1000];

bool vis[1000][1000];

vector<string> ans;

void solve(int i, int j, int n, string move)

{

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

if (i == n - 1 && j == n - 1)

{

// Simulation Display

pos\_way++;

cout << "----------------------------\n";

cout << "Path " << pos\_way << " : \n";

SetConsoleTextAttribute(h, 185);

cout << "\t" << move;

SetConsoleTextAttribute(h, 240);

cout << '\n';

for (int i1 = 0; i1 < n; i1++)

{

cout << "\t";

for (int j1 = 0; j1 < n; j1++)

{

if (maze[i1][j1] == 0)

{

SetConsoleTextAttribute(h, 271);

cout << " 0 ";

}

else

{

if (i1 == n - 1 && j1 == n - 1)

{

SetConsoleTextAttribute(h, 47);

cout << " 1 ";

continue;

}

if (vis[i1][j1])

{

SetConsoleTextAttribute(h, 207);

cout << " 1 ";

}

else

{

SetConsoleTextAttribute(h, 240);

cout << " 1 ";

}

}

}

SetConsoleTextAttribute(h, 240);

cout << '\n';

}

cout << "----------------------------\n";

cout << '\n';

// End

ans.push\_back(move);

return;

}

string dir = "DLRU";

for (int ind = 0; ind < 4; ind++)

{

int nexti = i + di[ind];

int nextj = j + dj[ind];

if (nexti >= 0 && nextj >= 0 && nexti < n && nextj < n && !vis[nexti][nextj] && maze[nexti][nextj] == 1)

{

vis[i][j] = 1;

// Simulation Display

SetConsoleTextAttribute(h, 185);

cout << "\t" << move;

SetConsoleTextAttribute(h, 240);

cout << '\n';

for (int i1 = 0; i1 < n; i1++)

{

cout << "\t";

for (int j1 = 0; j1 < n; j1++)

{

if (maze[i1][j1] == 0)

{

SetConsoleTextAttribute(h, 271);

cout << " 0 ";

}

else

{

if (vis[i1][j1])

{

if (i1 == i && j1 == j)

{

SetConsoleTextAttribute(h, 47);

cout << " 1 ";

}

else

{

SetConsoleTextAttribute(h, 207);

cout << " 1 ";

}

}

else

{

SetConsoleTextAttribute(h, 240);

cout << " 1 ";

}

}

}

SetConsoleTextAttribute(h, 240);

cout << '\n';

}

cout << '\n';

// End

solve(nexti, nextj, n, move + dir[ind]);

vis[i][j] = 0;

}

}

}

int main()

{

system("color f0");

HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);

int n;

cout << "Enter the Maze Size (N) : ";

cin >> n;

cout << '\n';

cout << "Enter '0' or '1' to the Cell. ( '1' = Usable Block, '0' = Dead End )\n\n";

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

while (1)

{

bool flag = 0;

string temp;

cout << "( " << i + 1 << ", " << j + 1 << " ) Cell : ";

cin >> temp;

for (int k = 0; k < temp.size(); k++)

{

if (!isdigit(temp[k]))

{

flag = 1;

break;

}

}

if (flag == 1)

{

cout << "Invalid Input! Enter the Correct Value.\n\n";

}

else

{

int num = stoi(temp);

if (num == 0 || num == 1)

{

maze[i][j] = num;

break;

}

else

{

cout << "Invalid Input! Enter the Correct Value.\n\n";

}

}

}

}

}

// Simulation Start

cout << '\n';

cout << "# For Simulation Purpose: \n\n";

SetConsoleTextAttribute(h, 47);

cout << "\t 1 ";

SetConsoleTextAttribute(h, 240);

cout << " = Current Position of Rat\n\n";

SetConsoleTextAttribute(h, 207);

cout << "\t 1 ";

SetConsoleTextAttribute(h, 240);

cout << " = Visited Position of Rat\n\n";

SetConsoleTextAttribute(h, 271);

cout << "\t 0 ";

SetConsoleTextAttribute(h, 240);

cout << " = Dead End Position\n\n\n";

cout << "The Maze Layout : \n\n";

for (int i = 0; i < n; i++)

{

cout << "\t";

for (int j = 0; j < n; j++)

{

if (maze[i][j] == 0)

{

SetConsoleTextAttribute(h, 271);

cout << " 0 ";

}

else

{

SetConsoleTextAttribute(h, 240);

cout << " 1 ";

}

SetConsoleTextAttribute(h, 240);

}

cout << '\n';

}

cout << '\n';

// End

if (maze[0][0] == 1)

{

cout << "Simulation Starts !\n\n";

solve(0, 0, n, "");

}

if (ans.size() == 0)

{

cout << "There is No Path Between Source to Destination.\n";

}

else

{

cout << "Total Path from Source to Destination = " << ans.size() << '\n';

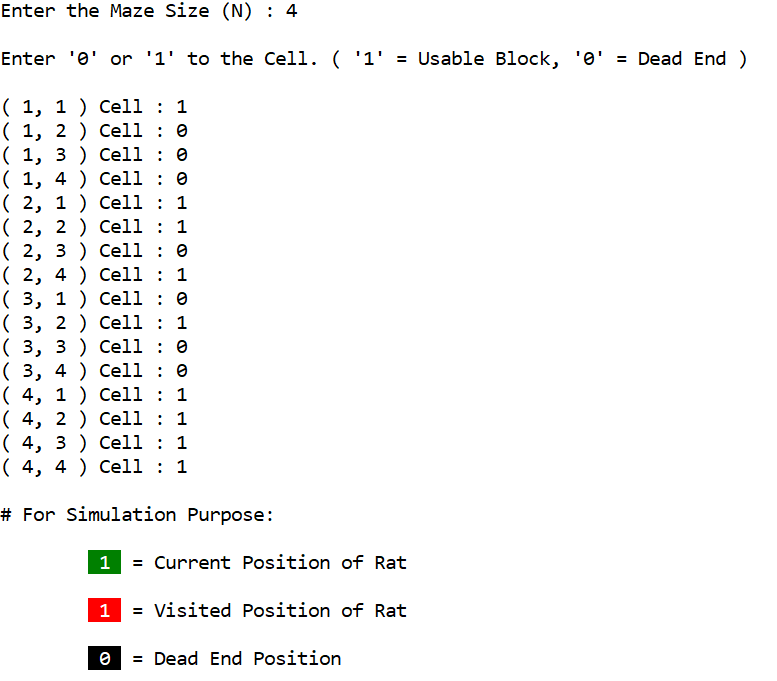
}

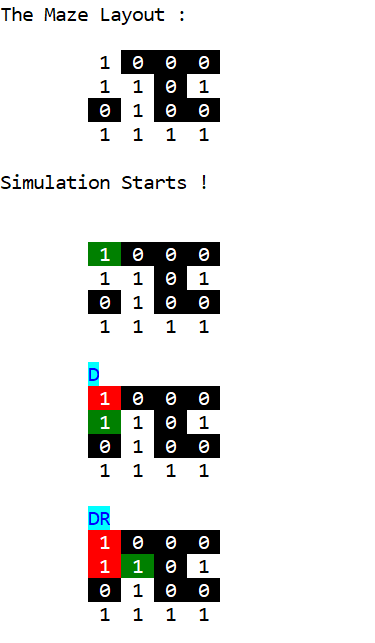
return 0;

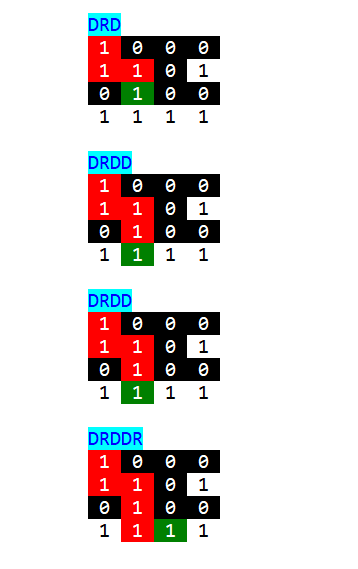
}

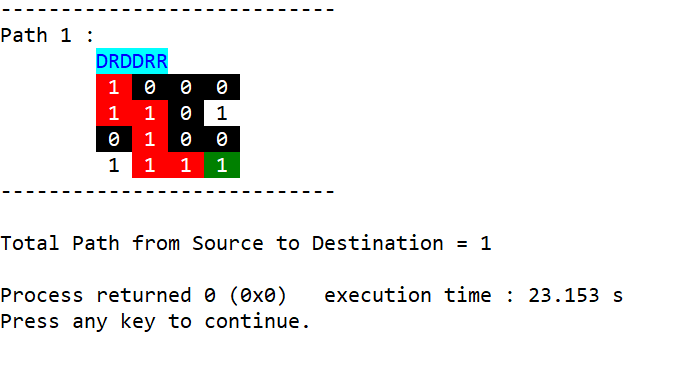
**Output**

**In Question:**

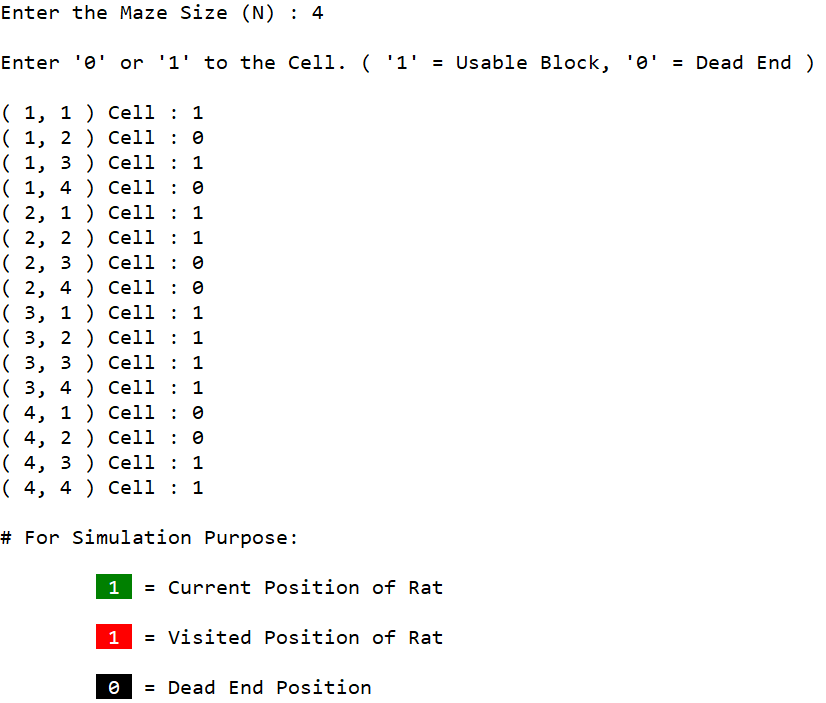
****

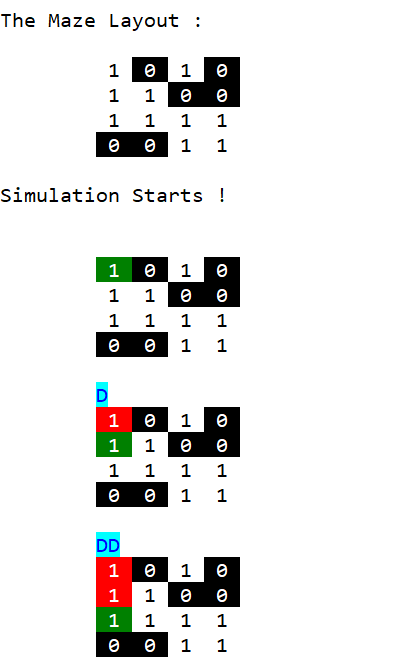
****

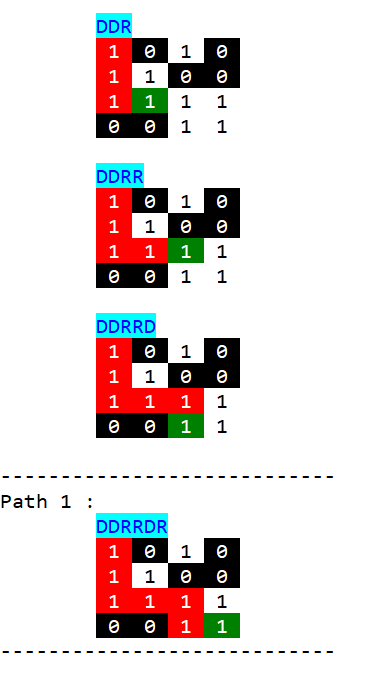
****

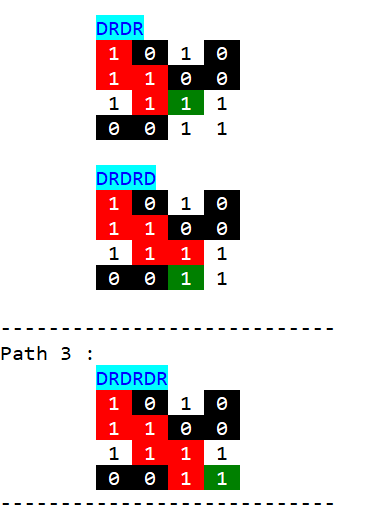
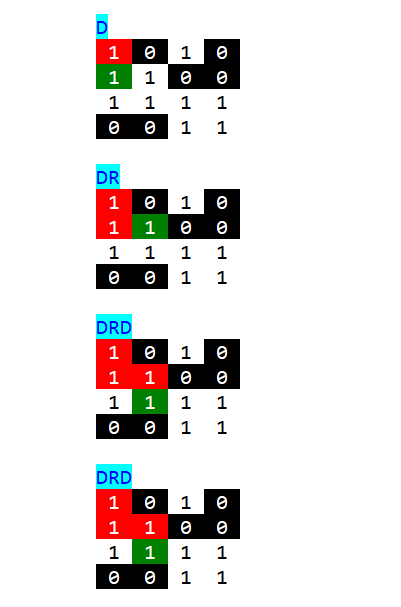
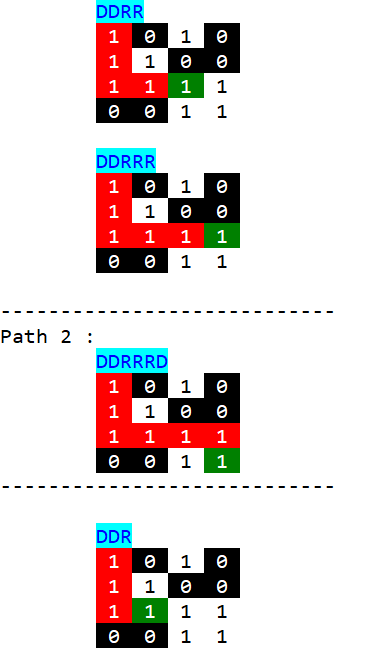


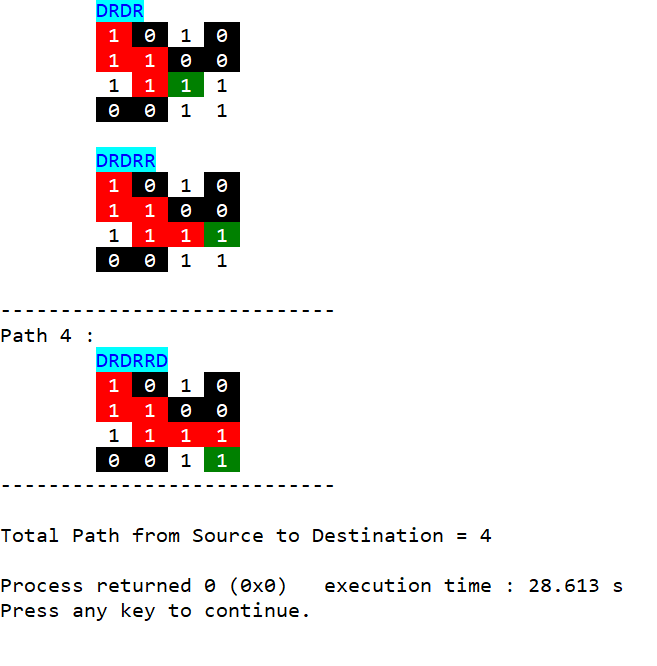
**More than one path:**











**Conclusion:**

The "Rat in a maze" problem helps us analyze graphs in depth because of DFS (depth first search). Also, it strengthens our knowledge of data structure.

As we conclude this project, it is evident that the "Rat in a Maze" scenario serves as a valuable educational tool for understanding fundamental concepts in computer science, artificial intelligence, and decision-making. It challenges us to think critically, experiment with algorithms, and appreciate the significance of optimization in problem-solving in future work.