S.R.M.Nagar, Kattankulathur, Chengalpattu District

JUNE2022

RAT IN A MAZE

AMINI PROJECTREPORT

Submittedby

Krishitaa Balamurali [Reg No: RA2011003010115]

Mohnaa Ghosh [Reg No: RA2011003010108]

Putta Sri Naga Sanjana [Reg No:RA2011003010123]

for the course 18CSC204J Design and Analysis of Algorithms

Under the guidance of

Dr. Sindhuja M

(Assistant Professor, Department of Computing Technologies)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

S.R.M. NAGAR, KATTANKULATHUR -603 203

KANCHEEPURAM DISTRICT

BONAFIDE CERTIFICATE

Register No ________________

Certified to be the bonafide record of work done by

____

of COMPUTER SCIENCE AND ENGINEERING , B.Tech Degree course in the Practical ***18CSC204J – DESIGN AND ANALYSIS OF ALGORITHMS*** *in* ***SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Kattankulathur*** *during the academic year 2022-2023*

Lab Incharge

Date: **Year Co-ordinator**

Submitted for University Examination held in__________________,

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,

Kattankulathur.

Date: **Examiner-1** **Examiner-2**

CERTIFICATE

This is to certify that the project entitled “**Rat in a maze**” carried out by “____” under my supervision at Department of Computing and Technology, SRM Institute of Technology, Kattankulathur, Chennai.

The work is original, as it has not been submitted earlier either in part or full for any purpose before.

Dr. M Sindhuja (Asst. Professor)

DECLARATION

I, hereby declare that the work presented in this dissertation entitled “**Rat in a maze**” has been done by me and my team, and this dissertation embodies my own work.

Approved By:

Dr. M Sindhuja

ACKNOWLEDGEMENTS

I would like to thank **Dr. M SINDHUJA** (Asst. Professor) who has been a great inspiration and who have provided sufficient background knowledge and understanding of this subject.

Our humble prostration goes to her, for providing all the necessary resources and environment, which have aided me to complete this project successfully.

CONTENT

CONTRIBUTION TABLE

NAME	WORK CONTRIBUTED
Krishitaa Balamurali	Gathering the required information and segregation and arrangement of the information as per the report format.
Mohnaa Ghosh	Working on the code and calculating the analysis.
Putta Sri Naga Sanjana	Arrangement of the information according to the report format and managing the overall

	project and report work.

PREFACE

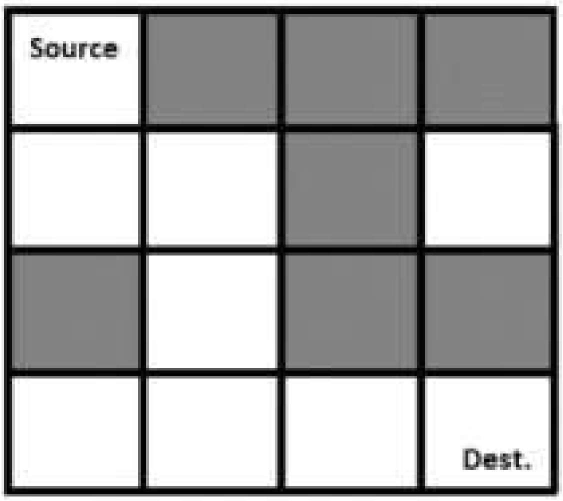
This report is about the problem Rat in a maze. We have used C++ programming here. The report focuses on the algorithm used for solving the problem Rat in a maze. The objective of this problem is that the rat will be at a particular cell and we have to find all the possible paths that the rat can take to reach the destination cell from the given source cell. We have taken references from various sites on internet and youtube videos.

Problem Definition

In this problem, there is a given maze of size N x N. The source and the destination location is top-left cell and bottom right cell respectively. Some cells are valid
to
move and some cells are blocked. If one rat starts moving from start vertex to destination vertex, we have to find that is there any way to complete the path, if it is
possible then mark the correct path for the rat. The maze is given using a binary matrix, where it is marked with 1, it is a valid path, otherwise 0 for a blocked cell.

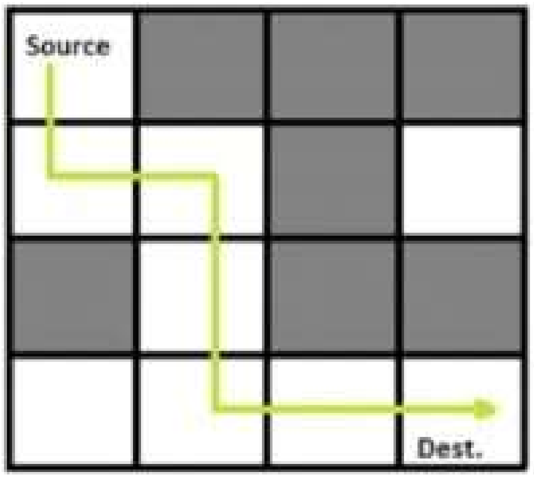
NOTE: The rat can only move in two directions, either to the right or to the down.

A maze is in the form of a 2D matrix in which some cells/blocks are blocked. One of the cells is termed as a source cell, from where we have to start. And another one of them is termed as a destination cell, where we
have to reach. We have to find a path from the source to the destination without
moving into any of the blocked cells. A picture of an unsolved maze is shown below, where grey cells denote the dead ends and white cells denote the cells which can be accessed.



To solve these types of puzzle, we first start with the source cell and move in a direction where the path is not blocked. If the path taken makes us reach the destination, then the puzzle is solved. Otherwise, we come back and change our direction of the path taken.

The solution of this maze would look like:



Design Techniques used

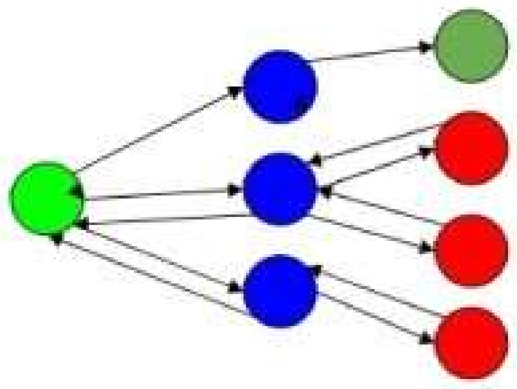
- The Algorithm used in this project is Backtracking.
- Backtracking is a famous algorithmic-technique for solving/resolving problems recursively by trying to build a solution incrementally. Solving one piece at a time, and removing those solutions that fail to satisfy the constraints of the problem at any point of time is the process of backtracking.

∎ According to the wiki definition, Backtracking can be termed as a general algorithmic technique that considers searching every possible combination in order to solve a computational problem.

General Technique of Backtracking

∎ In backtracking problem, the algorithm tries to find a sequence path to the solution
which has some small checkpoints from where the problem can backtrack if no feasible solution is found for the problem. ∎ For example,

∎ Here, green is the start point, blue is the intermediate point, red are points with no feasible solution, dark green is end solution.



∎ Here, when the algorithm propagates to an end to check if it is a solution or not, if it
is then returns the solution otherwise backtracks to the point one step behind it to
find track to the next point to find solution.

∎ ALGORITHM:-

- Step 2 ---- else, • Step 1 if current_position is goal, return success
- Step 3if current_position is an end point, return failed.
- Step 4 else, if current_position is not end point, explore and repeat above steps.

Why was Backtracking used...?

∎ A classic example of backtracking is solving a maze: if you go down one path and it isn't the correct path, then you backtrack to your last decision point to try an alternate path. If you are using an object passed by reference you need to either
undo (or "un-choose") paths that fail, or somehow mark them in your object. ■ For a maze, you don't want to try and traverse the same path twice, so you need to mark whether you have been down that path before.

Algorithm for backtracking

∎ Create a solution matrix, initially filled with 0’s.
∎ Create a recursive function, which takes initial matrix, output matrix and position of rat (i, j).
∎ if the position is out of the matrix or the position is not valid then return. ■ Mark the position output[i][j] as 1 and check if the current position is destination or not. If destination is reached print the output matrix and return. ■ Recursively call for position (i-1,j), (I,j-1), (i+1, j) and (i, j+1). ■ Unmark position (i, j), i.e output[i][j] = 0.

Explanation of algorithm with example

Write a boolean function that will declare whether the next move for the rat is free from obstacles or not.

Next, start writing a function that will trace the path for rat from the source to the destination.

Write base condition which will state that if the rat has reached the destination.

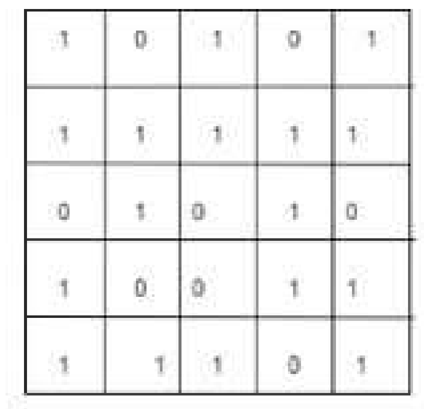
Now, if the next move for the rat is safe, the value should be initialized to 1. And we also have to note whether the rat has taken action in the forward or downward direction. If the rat is in a state where it can’t take any further action to reach the destination, the value should be initialized to 0. This is backtracking.

Now, declare the main function, which will have dynamic declaration and initialization of the input matrix.

Similarly, we will have a solution matrix to evaluate the output and print it on the screen.

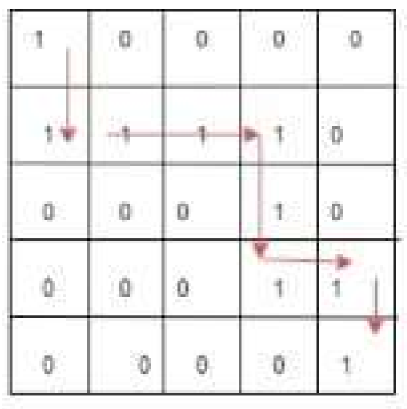
This is the most optimized approach using backtracking.

Example:



1	0	1	0	1
1	1	1	1	1
0	1	0	1	0
1	0	0	1	1
1	1	1	0	1

Input Matrix



1	0	0	0	0
1	1	1	1	0
0	0	0	1	0
0	0	0	1	1
0	0	0	0	1

Solution Matrix

PSEUDOCODE

```
Begin
  if (x,y) is the bottom right corner, then
    mark the place as 1
    return true
  if isValidPlace(x, y) = true, then
    mark (x, y) place as 1
```

```
      if solveRatMaze(x+1, y) = true, then //for forward movement
         return true
      if solveRatMaze(x, y+1) = true, then //for down movement
         return true
      mark (x,y) as 0 when backtracks
      return false
   return false
End
```

CODE

We have used C++ to implement the code for the problem.

```
#include<iostream>
#define N 5 using
namespace std;

int maze[N][N] = {
   {1, 0, 0, 0, 0},
   {1, 1, 0, 1, 0},
   {0, 1, 1, 1, 0},
   {0, 0, 0, 1, 0},
   {1, 1, 1, 1, 1}
};

int sol[N][N]; //final solution of the maze path is stored here
void showPath() {
   for (int i = 0; i < N; i++) {
      for (int j = 0; j < N; j++)
      cout << sol[i][j] << " ";
      cout << endl;
   }
}
```

```
bool isValidPlace(int x, int y) { //function to check place is inside the maze and
have value 1
  if(x >= 0 && x < N && y >= 0 && y < N && maze[x][y] == 1)
    return true;
  return false;
} bool solveRatMaze(int x, int
y) {
  if(x == N-1 && y == N-1) { //when (x,y) is the bottom right room
    sol[x][y] = 1;
    return true;
}

  if(isValidPlace(x, y) == true) { //check whether (x,y) is valid or not
    sol[x][y] = 1; //set 1, when it is valid place if (solveRatMaze(x+1, y) ==
    true) //find path by moving right direction return true;


      if (solveRatMaze(x, y+1) == true) //when x direction is blocked, go for
bottom direction
       return true;
    sol[x][y] = 0; //if both are closed, there is no path
    return false;
  }
  return false;
}

bool findSolution() {
  if(solveRatMaze(0, 0) == false) {
    cout << "There is no path";
    return false;
  }
  showPath();
   return true;
}

int main() {
   findSolution();
}
```

OUTPUT

```
1 0 0 0 0
1 1 0 0 0
0 1 1 1 0
0 0 0 1 0
0 0 0 1 1
```

ANALYSIS

1) Time Complexity :

Worst Case Time : O(2^(N^2))

Explanation : Since at each position rat has two possibilities, either to move forward or downward.

2)Space Time Complexity : O(N^2)

Explanation: since an n X n matrix is used.

Conclusion

Thus we have successfully solved the problem using backtracking. We have learnt the general method of backtracking ,the reason why we have used here backtracking. We have run the code and also calculated the time complexity of the code.

References

- Data Structures And Algorithms By Cormen
- Wikipedia
- Tutorialspoint
- Geeksforgeeks
- Javatpoint
- Coding Ninjas
- Youtube