

## Backtracking | Set 2 (Rat in a Maze)

We have discussed Backtracking and Knight's tour problem in [Set 1](#). Let us discuss Rat in a [Maze](#) as another example problem that can be solved using Backtracking.

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]` 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 only in two directions: forward 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. Note that this is a simple version of the typical Maze problem. For example, a more complex version can be that the rat can move in 4 directions and a more complex version can be with limited number of moves.

Following is an example maze.

Gray blocks are dead ends (value = 0).

Source			
			Dest.

Following is binary matrix representation of the above maze.

Google™ Custom Search



GeeksforGeeks



53,525 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

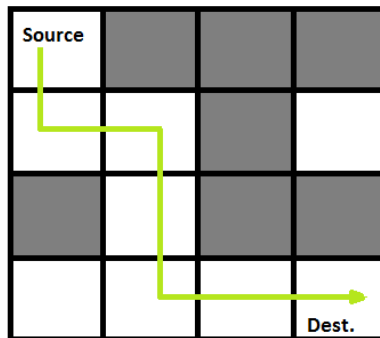
[Geometric Algorithms](#)

```

{1, 0, 0, 0}
{1, 1, 0, 1}
{0, 1, 0, 0}
{1, 1, 1, 1}

```

Following is maze with highlighted solution path.



Following is the solution matrix (output of program) for the above input matrix.

```

{1, 0, 0, 0}
{1, 1, 0, 0}
{0, 1, 0, 0}
{0, 1, 1, 1}

```

All enteries in solution path are marked as 1.

## Naive Algorithm

The Naive Algorithm is to generate all paths from source to destination and one by one check if the generated path satisfies the constraints.

```

while there are untried paths
{
    generate the next path
    if this path has all blocks as 1
    {
        print this path;
    }
}

```



## Popular Posts

[All permutations of a given string](#)

[Memory Layout of C Programs](#)

[Understanding "extern" keyword in C](#)

[Median of two sorted arrays](#)

[Tree traversal without recursion and without stack!](#)

[Structure Member Alignment, Padding and Data Packing](#)

[Intersection point of two Linked Lists](#)

[Lowest Common Ancestor in a BST.](#)

[Check if a binary tree is BST or not](#)

[Sorted Linked List to Balanced BST](#)

## Backtracking Algorithm

If destination is reached

print the solution matrix

Else

- a) Mark current cell in solution matrix as 1.
- b) Move forward in horizontal direction and recursively check if this move leads to a solution.
- c) If the move chosen in the above step doesn't lead to a solution then move down and check if this move leads to a solution.
- d) If none of the above solutions work then unmark this cell as 0 (BACKTRACK) and return false.

## Implementation of Backtracking solution

```
#include<stdio.h>

// Maze size
#define N 4

bool solveMazeUtil(int maze[N][N], int x, int y, int sol[N][N]);

/* A utility function to print solution matrix sol[N][N] */
void printSolution(int sol[N][N])
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
            printf(" %d ", sol[i][j]);
        printf("\n");
    }
}

/* A utility function to check if x,y is valid index for N*N maze */
bool isSafe(int maze[N][N], int x, int y)
{
    // if (x,y outside maze) return false
    if(x >= 0 && x < N && y >= 0 && y < N && maze[x][y] == 1)
        return true;

    return false;
}
```



## Recent Comments

Abhi You live US or India?

[Google \(Mountain View\) interview](#) · 8 minutes ago

[Aman](#) Hi, Why arent we checking for conditions...

[Write a C program to Delete a Tree.](#) · 47 minutes ago

kzs please provide solution for the problem...

[Backtracking | Set 2 \(Rat in a Maze\)](#) · 51 minutes ago

[Sanjay Agarwal](#) bool

tree::Root\_to\_leaf\_path\_given\_sum(tree...


Root to leaf path sum equal to a given number · 1 hour ago

[GOPI GOPINATH @admin](#) Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

[newCoder3006](#) If the array contains negative numbers also. We...

[Find subarray with given sum](#) · 1 hour ago

AdChoices 

[► C++ Code](#)

[► Programming C++](#)

[► Rat Maze](#)

AdChoices 

```
/* This function solves the Maze problem using Backtracking. It mainly
solveMazeUtil() to solve the problem. It returns false if no path is possible
otherwise return true and prints the path in the form of 1s. Please note that
there may be more than one solutions, this function prints one of the solutions.*/
```

```
bool solveMaze(int maze[N][N])
{
    int sol[N][N] = { {0, 0, 0, 0},
                      {0, 0, 0, 0},
                      {0, 0, 0, 0},
                      {0, 0, 0, 0}
    };

    if(solveMazeUtil(maze, 0, 0, sol) == false)
    {
        printf("Solution doesn't exist");
        return false;
    }

    printSolution(sol);
    return true;
}

/* A recursive utility function to solve Maze problem */
bool solveMazeUtil(int maze[N][N], int x, int y, int sol[N][N])
{
    // if (x,y is goal) return true
    if(x == N-1 && y == N-1)
    {
        sol[x][y] = 1;
        return true;
    }

    // Check if maze[x][y] is valid
    if(isSafe(maze, x, y) == true)
    {
        // mark x,y as part of solution path
        sol[x][y] = 1;

        /* Move forward in x direction */
        if (solveMazeUtil(maze, x+1, y, sol) == true)
            return true;

        /* If moving in x direction doesn't give solution then
        Move down in y direction */
        if (solveMazeUtil(maze, x, y+1, sol) == true)
```

```

        return true;

        /* If none of the above movements work then BACKTRACK:
           unmark x,y as part of solution path */
        sol[x][y] = 0;
        return false;
    }

    return false;
}

// driver program to test above function
int main()
{
    int maze[N][N] = { {1, 0, 0, 0},
                        {1, 1, 0, 1},
                        {0, 1, 0, 0},
                        {1, 1, 1, 1}
    };

    solveMaze(maze);
    getchar();
    return 0;
}

```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

► [Java C++](#)

► [C++ Example](#)

► [Graph C++](#)

AdChoices ►

► [C++ Program](#)

► [Test C++](#)

► [Int](#)



## Related Topics:

- Backtracking | Set 8 (Solving Cryptarithmic Puzzles)
- Tail Recursion
- Find if two rectangles overlap
- Analysis of Algorithm | Set 4 (Solving Recurrences)
- Print all possible paths from top left to bottom right of a mXn matrix
- Generate all unique partitions of an integer
- Russian Peasant Multiplication
- Closest Pair of Points | O(nlogn) Implementation



6



Tweet

0



0

Writing code in comment? Please use [ideone.com](https://ideone.com) and share the link here.

33 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion



com the discussion...



**kzs** • an hour ago

please provide solution for the problem including 1.down 2.left 3.right possible

^ | v •



**Amit** • 2 months ago

Very well explained but it will fail for the case in which rat has to travel left or up  
ex

1 1 1 0 0

0 0 1 0 0

0 1 1 0 0

0 1 0 0 0

0 1 1 1 1

^ | v •



**Kartik** → **Amit** • 2 months ago

Amit, please take a closer look at the problem statement. It is clearly m  
forward (or right) and down.

^ | v •



**amit** → **Kartik** • 2 months ago

oops.

kartik Thanku buddy

^ | v •



**shiv** • 3 months ago

here is one more sipmle solution of maze

[technofrendzz.blogspot.in](http://technofrendzz.blogspot.in)

^ | v •



**nakul gowda** • 5 months ago

here is the bfs implementation for the same, which is much quicker and gives

```
#include<stdio.h>
```

```
#define SIZE 7
```

```
int a[SIZE][SIZE],q[SIZE*SIZE],visited[SIZE][SIZE],n,i,j,f=0,r=-1;
```

```
int parent[SIZE*SIZE], x_dest, y_dest, x_temp, y_temp;
```

```
int flag =0;
```

```
void find_neighbours(int x, int y)
```

```
{
```

```
if (( (y+1)<n) &&"" (a[x][y+1])="" &&"" !visited[x][y+1])="" {="" q[++r]="(x" *=""  
n="" +="" (y+1)]="x" *="" n="" +y;="" visited[x][y+1]="1;" }="" if="" ((y-1)="">=0 ;
```

```
{
```

[see more](#)

1 ^ | v .



**Raj** • 6 months ago

Here is the java version of rat in a maze. Code is really easy to understand.

<http://techlovejump.in/2013/11...>

Demo of code :-

```
boolean solveMazeUtil(int matrix[],int x,int y,int sol[],int n){
```

```
//path complete (if we are at our goal then path complete)
```

```
if(x == n-1 && y == n-1){
```

```
sol[x][y] = 1;
```



```

return true;
}
if(isSafe(matrix,x,y,n) == true)
{
sol[x][y] = 1;
// try right
if(solveMazeUtil(matrix,x+1,y,sol,n))
return true;
//try down
if(solveMazeUtil(matrix,x,y+1,sol,n))

```

[see more](#)

^ | v .



**GeeksforGeeks** • 10 months ago

There is always a path from source to source itself, right?

6 ^ | v .



**Mahendra Mundru** • 10 months ago

It fails when we given destination as 0 (at that time also it is giving output as th

^ | v .



**Suryaamsh** • a year ago

This is quite long comment/reply.

I think performance of the solution posted can be improved by memorizing wh destination).

Say, we are at  $a[i][j]$  and if both recursive calls of solveMazeUtil return false, it doesn't exist a path leading to destination. Set a flag say,  $flags[i][j]$  to false. So, recursive call (arriving from different path), we can altogether ignore the subse found to be false. There is an overhead of  $n^2$  bits though. In my opinion, it car redundant recursive calls on an average. Please point the mistakes if found

redundant recursive calls on an average. Please point the mistakes if found.

^ | v .



**Aditya** → Suryaamsh · 9 months ago

so you are saying false => no path , true => path exists, what about the path exists or not, i.e. we haven't calculated yet?  
its better if we use an int array with 0,1,2 values.

correct me if I'm mistaken.

^ | v .



**Kavish Dwivedi** · a year ago

Here is my code for the problem:

```
#include<stdio.h>
#define N 4
int sol[N][N],maze[N][N];
int check(int x,int y)
{
    if( x>=0 && x<N && y>=0 && y<N && maze[x][y]!=0 && sol[x][y]!=0)
        return 1;
    return 0;
}
int solve(int x,int y,int move)
{
    if(x==N-1&&y==N-1)
    {
        return 1;
    }
    else
```

see more

1 ^ | v .



**Ravishankar Narayanan** · 2 years ago

This prints no. of solutions from 0,0 to n-1,n-1

```
[sourcecode language="C++"]
#include<stdio>
int v[10][10],m[10][10],c=0,n,i,j,dx[4]={1,0,-1,0},dy[4]={0,1,0,-1};
int s(int x,int y){
if(x>=0&&x<n&&y>=0&&y<n&&m[x][y]&&!v[x][y]){
if(x==n-1&&y==n-1)return 1;
v[x][y]=1;
for(int i=0;i<4;i++)c+=s(x+dx[i],y+dy[i]);
v[x][y]=0;
}
return 0;
}
int main(){
scanf("%d",&n);
for(i=0;i<n;i++)for(j=0;j<n;j++)scanf("%d",&m[i][j]);
s(0,0);
printf("%d\n",c);
return 0;
}
```

^ | v ·



**hariprasath** → Ravishankar Narayanan · a year ago

does this implementation correct??

```
#include
#include
#include
```

```
int a[9][9];
```

```

int b[9][9];

int check(int x,int y,int val)
{
    int i,j;

    //Scanning its own Row and Column

    for(i=0;i<9;i++)
    {
        if(val==b[i][y] || val==b[x][i]) return 0;
    }
}

```

see more

^ | v .



**Ravishankar Narayanan** · 2 years ago

Thanks

^ | v .



**sachin** · 2 years ago

I have basic C /recursion noob question.Can Someone please explain- when \n As sol[x][y] should not already be set to 0 when return to previous state of recu

^ | v .



**monika** → sachin · a year ago

It is because, we were making changes in the same sol[][] matrix, and undo the changes we had made.

^ | v .



**sachin** · 2 years ago

How to find out the shortest path among all the paths?

^ | v .



**vivek** · 2 years ago

where is the backtracking step? i mean in backtracking,if subsequent computation fails, we revert the state of a system to a previous stable state...right? so i don't get where the backtracking is in condition checking .. please explain

^ | v .



**Guddu sharma** · 2 years ago

Here is my code which prints all the solutions of the maze.

```
void Maze(int (*maze)[N],int x,int y,int (*sol)[N])
{
    static int i,j;
    if(x==N-1 && y==N-1)
    {
        printf("\n\nSolution\n\n");
        sol[x][y]=1;
        for(i=0;i<N;i++)
        {
            for(j=0;j<N;j++)
                printf("%d ",sol[i][j]);
            printf("\n");
        }
    }
    else
    {
```

see more

^ | v .



**Ajinkya** · 2 years ago



Execute this modified code to print all possible paths and not just one path... A procedure....

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>

// Maze size
#define N 4

void solveMazeUtil(int maze[N][N], int x, int y, int sol[N][N]);

/* A utility function to print solution matrix sol[N][N] */
void printSolution(int sol[N][N],int i,int j)
{
    /*for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
```

see more

^ | v .



**Agniswar** • 3 years ago

Hey this is my solution..please tell me if there is any wrong in the code somew

```
#include
#include

#define N 4

int is_safe(int a[N][N],int sol[][N],int x1,int y1)
{
    if((x1>=0 && y1>=0) && (x1<=N-1 && y1<=N-1))
```

```

{
if(a[x1][y1]==1 && sol[x1][y1]==0)
return 1;
}
return 0;
}

```

```

void init(int sol[N][N])
{

```

see more

2 ^ | v .



**karthiga** · 3 years ago

can somebody pls help me how dis part working ..i cant understand dis !!!!!

```

bool solveMazeUtil(int maze[N][N], int x, int y, int sol[N][N])
{
    // if (x,y is goal) return true
    if(x == N-1 && y == N-1)
    {
        sol[x][y] = 1;
        return true;
    }

    // Check if maze[x][y] is valid
    if(isSafe(maze, x, y) == true)
    {
        // mark x,y as part of solution path
        sol[x][y] = 1;

        /* Move forward in x direction */

```

see more

^ | v .



**PolaNix** → karthiga · 2 years ago

If you cant understand this clearly written code along with the descriptive flag.

^ | v .



**xor** → PolaNix · 2 years ago

The code is clear to you because you wrote it.

^ | v .



**nitishgarg** · 3 years ago

Err..The code given above is not printing the correct solution, check here: [http](#). Can anyone check why?

^ | v .



**Sandeep** → nitishgarg · 3 years ago

@nitishgarg: Thanks for pointing this out. The original code was working there was a small issue which was causing problems on Ideone. I have solved solveMaze(). It should work on all compilers now.

^ | v .



**atul007** → Sandeep · 2 years ago

Here is the code which will work for all cases :-

```
#include<stdio.h>
#include<stdlib.h>
#define N 9
```



```

int validPath(int maze[][N],int row,int col)
{
    if(maze[row][col])
        return 1;

    return 0;
}

void printSolution(int path[][N])
{
    int i,j;

```

[see more](#)

^ | v .



**atul** → Sandeep · 2 years ago

it will fail for following test case :-

```

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

```

```

/* Paste your code here (You may delete these lines if r

```

^ | v .



**atul** → Sandeep · 2 years ago

Are you a developer? Try out the [HTML to PDF API](#)



code does not work for below test case :-

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

```
/* Paste your code here (You may delete these lines if r
```

^ | v .



**Aps** → atul · 3 months ago

It doesn't work because we defined 'N' value to 9 so the compiler considers it as 9X9 by inserting zeros(as we n changing N value to 4 we can get a correct path

^ | v .



**yu** · 3 years ago

we can also build a binary tree from the matrix representation and do a depth- destination.

^ | v .



**Sandeep** → yu · 3 years ago

Backtracking solution does DFS traversal only. The point to note is tha always traverse the complete tree. Backtracking stops at an internal n feasible from this node and starts exploring other possibilities.

^ | v .

 [Subscribe](#)

 [Add Disqus to your site](#)

@geeksforgeeks, **Some rights reserved**

[Contact Us!](#)

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team