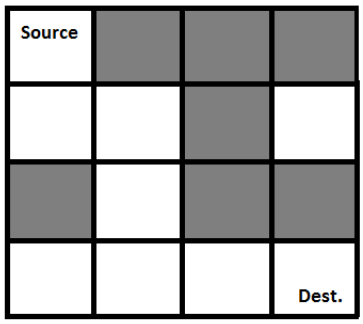
Backtracking | Set 2 (Rat in a Maze)

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.

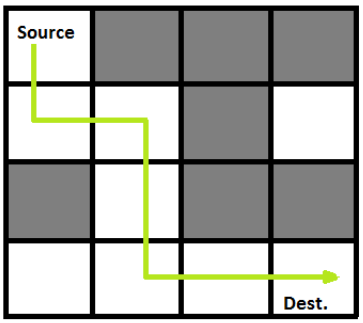
```
Gray blocks are dead ends (value = 0).
```



Following is binary matrix representation of the above maze.

```
{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 matrx.

```
{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;
   }
}
```

Backtrackng 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;
}

/* This function solves the Maze problem using Backtracking.  It mainly uses
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 feasible
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.

Related Topics:

- Linearity of Expectation

- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Backtracking

Writing code in comment? Please use **ideone.com** and share the link here.

Vikram Ojha • a month ago

I think u dont need to chech for x>=0 it will be olways greater than equal to zero in isSafe()

N also u dont need to write
if (solveMazeUtil(maze, x, y+1, sol) == true)
return true;
even

if (solveMazeUtil(maze, x, y+1, sol))
return true;

its ambiguous, u r checking true == true which adds to ur cpu cycle

Jasprit • a month ago

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

{

if (x == N-1 && y == N-1) {

sol[x][y] = 1;

return true;

}

sol[x][y] = 1;

```
if (!isSafe(mat, x, y)) {

sol[x][y] = 0;

return false;

}

return solveMazeUtil(mat, sol, x, y + 1) ||

solveMazeUtil (mat, sol, x+1, y);

}
```

• Reply • Share ›

let_us_c • 3 months ago

can easily done by dp

• Reply • Share ›

Sasidhar Koti • 4 months ago

if (solveMazeUtil(maze, x+1, y, sol) == true) it basically moving next row, if (solveMazeUtil(maze, x, y+1, sol) == true) is moving to next column with in the same row..

• Reply • Share ›

Sanket Patel ➚ Sasidhar Koti • 2 months ago

Yes, that is correct. They should swap the lines, or change the comments.

• Reply • Share ›

shanky • 6 months ago

Is the recurrence relation is T(m,n)=T(m-1,n)+T(m,n-1) , with T(0,1)=T(1,0)=1

• Reply • Share ›

Saksham Gupta • 6 months ago

The most general solution with all 8 moves.

http://ideone.com/cuuncc

3 • Reply • Share ›

Fernando Ferreira ➚ Saksham Gupta • 4 months ago

Saksham, your code permits diagonal movements. In this example we can only move forward or move down. But your solution is interesting, because we can have several types of movements in a table and your code is prepared for it.

• Reply • Share ›

Sumit Kesarwani ➚ Fernando Ferreira • 3 months ago

for that we need to add one condition only

if(solveMazeUtil(...,x+1,y+1,..)){do sth}...
right @Fernando Ferreira.... and if we are trying to move int all eight direction then we have to maintain a boolean two2D mattrix where we will store which cell we have visited allready so that we can avoid two revisite aggain....

please correct me if i am wrong@Fernando Ferreira

Reply • Share ›

Saksham Gupta • 6 months ago

If anyone is facing problems while compiling this code in codeblocks use
#include<stdbool.h>

Reply • Share ›

Sajal Sharma • 6 months ago

Tried a more general solution with all four possible moves...
here's my solution... if there are any bugs, please let me know...
http://ideone.com/kk4fzz
and for finding the shortest path
http://ideone.com/TqdXWx

Reply • Share ›

vivek • 7 months ago

mention complexity also...

Reply • Share ›

ashish • 7 months ago

for all the direction go

http://ideone.com/4zH7q0

2 Reply • Share ›

ashish • 7 months ago

For all four cardinal direction u may use this.....

```
#include<stdio.h>

#include<iostream>

using namespace std;

#define N 6

int visit[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]);
```

see more

1 • Reply • Share ›

jai ashish • 3 months ago
when you are making sol[x][y]=0 shouldnt visit[x][y] also be made =0
• Reply • Share ›

Rachit jai • 2 months ago
I dont think visit[x][y] needs to be made 0. Because Visit matrix only tells us what indexes have been visited. and in this case visit[x][y] was visited.
• Reply • Share ›

bhopu • 8 months ago
#include<stdio.h>
#define n 4

print(int sol[n][n]){
int i,j;
printf("\n");

for(i=0;i<n;i++){ for(j="0;j<n;j++)" printf("%d="" ",sol[i][j]);="" printf("\n");="" }="" }="" ratmaze(int="" maze[n][n],int="" sol[n][n],int="" r,int="" c){="" int="" i,j;="" for(i="c;i<n;i++){" if(maze[r][i]="=1&r<n){" sol[r][i]="1;" if(r="=n-1&i==n-1){" return="" 1;="" }else="" if(ratmaze(maze,sol,r+1,i))="" return="" 1;="" }else{="" for(j="i;j">=0&&r<n;j--) sol[r] [j]="0;" return="" 0;="" }="" }="" }="" main(){="" int="" maze[n][n]="{" {1,="" 1,="" 1,="" 0},="" {1,="" 1,="" 0,="" 1},="" {0,="" 1,="" 1,="" 0},="" {1,="" 1,="" 1,="" 1}="" };="" int="" sol[n][n]="{" {0,="" 0,="" 0,="" 0},="" {0,="" 0,="" 0,="" 0},="" {0,="" 0,="" 0,="" 0},="" {0,="" 0,="" 0,="" 0}="" };="" ratmaze(maze,sol,0,0);="" print(sol);="" }="">
• Reply • Share ›

Shrinivas • 8 months ago
What would be Time Complexity of above algorithm?
• Reply • Share ›

<HoldOnLife!#> • 10 months ago
complexity?

complexity ?

Reply • Share ›

Gaurav Gupta <HoldOnLife!#> • 10 months ago

I think it would be O(n*m) n being rows and m being columns.

Reply • Share ›

Ashish • 10 months ago

I am curious why do we need a another solution matrix. I think we can solve it without using additional solution matrix. please correct me if i am wrong.

Reply • Share ›

Ajinkya Kher MI Ashish • 9 months ago

You do not need any extra space if you just wish to know whether it possible to reach the end from start following ANY path, or not. If you want to actually know the path or paths, you would need extra space. (Although if you with to know just one path, it can be achieved with a clever tail recursion too, without the need of extra space)

Reply • Share ›

kzs • a year ago

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

1 Reply • Share ›

Amit • a year ago

Very well explained but it will fail for the case in which rat has to travel left or up and then the normal path

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

Reply • Share ›

Kartik Amit • a year ago

Amit, please take a closer look at the problem statement. It is clearly mentioned that the rat can only move forward (or right) and down.

Reply • Share ›

amit Kartik • a year ago

oops.

kartik Thanku buddy

Reply • Share ›

shiv • a year ago

here is one more sipmle solution of maze

technofrendzz.blogspot.in

• Reply • Share ›

nakul gowda • a year ago

here is the bfs implementation for the same, which is much quicker and gives you always the shortest path

#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);="" parent[x="" *="" n="" +="" (y+1)]="x" *="" n="" +y;="" visited[x][y+1]="1;" }="" if="" ((y-1)="">=0 && (a[x][y-1]) && !visited[x][y-1])

see more

4 • Reply • Share ›

Raj • 2 years 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

• Reply • Share ›

GeeksforGeeks • 2 years ago

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

7 • Reply • Share ›

Mahendra Mundru • 2 years ago

It fails when we given destination as 0 (at that time also it is giving output as there is a path).

• Reply • Share ›

Suryaamsh • 2 years ago

This is quite long comment/reply.

I think performance of the solution posted can be improved by memorizing whether a path exists from given node (to destination).

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

• Reply • Share ›

Aditya ➚ Suryaamsh • 2 years ago

so you are saying false => no path , true => path exists, what about the third case, where we do not know if 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.

• Reply • Share ›

Kavish Dwivedi • 2 years 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]==-1)
        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 • Reply • Share ›

Ravishankar Narayanan • 3 years ago

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

[sourcecode language="C++"]

```
#include<cstdio>
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;
}
```

• Reply • Share ›

hariprasath ➦ Ravishankar Narayanan • 2 years 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

1 • Reply • Share ›

Ravishankar Narayanan • 3 years ago

Thanks

• Reply • Share ›

sachin • 3 years ago

I have basic C /recursion noob question.Can Someone please explain- when we backtrack,why we set sol[x][y] =0 .
As sol[x][y] should not already be set to 0 when return to previous state of recursion?

• Reply • Share ›

monika sachin • 2 years ago

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

• Reply • Share ›

sachin • 3 years ago

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

• Reply • Share ›

vivek • 3 years ago

where is the backtracking step? i mean in backtracking,if subsequent computation don't provide a valid solution we revert the state of a system to a previous stable state...right? so i don't get where is that done here?here its just a condition checking .. please explain

• Reply • Share ›

Guddu sharma • 3 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

• Reply • Share ›

Ajinkya • 3 years ago

Execute this modified code to print all possible paths and not just one path... Also a new more clear printPath 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

• Reply • Share ›

Agniswar • 4 years ago

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

```
#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 • Reply • Share ›

karthiga • 4 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 &amp;amp;&amp;amp; 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

• Reply • Share ›

PolaNix ➚ karthiga • 3 years ago

If you cant understand this clearly written code along with the descriptive comments, this is a serious RED flag.

• Reply • Share ›

xor ➚ PolaNix • 3 years ago

The code is clear to you because you wrote it.

• Reply • Share ›

nitishgarg • 4 years ago

Err..The code given above is not printing the correct solution, check here:
https://ideone.com/Ffssm
Can anyone check why?

• Reply • Share ›

Sandeep ➚ nitishgarg • 4 years ago

@nitishgarg: Thanks for pointing this out. The original code was working on my compiler (Code Blocks) but there was a small issue which was causing problems on Ideone. I have added "return false" at the end of solveMaze(). It should work on all compilers now.

• Reply • Share ›

atul007 ➚ Sandeep • 3 years ago

Here is the code which will 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

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy DISQUS

Google™ Custom Search

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

- # 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

- Follow @GeeksforGeeks

- # Recent Comments

  - Ashish Aggarwal

    Try Data Structures and Algorithms Made Easy -...

    Algorithms · 17 minutes ago

  - Vlad

    Thanks. Very interesting lectures.

    Expected Number of Trials until Success · 1 hour ago

  - cfh

    My implementation which prints the index of the...

    Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

  - Gaurav pruthi

    forgot to see that part ;)

    Bloomberg Interview | Set 1 (Phone Interview) · 1 hour ago

  - saeid aslami

    thanks

    Greedy Algorithms | Set 7 (Dijkstra’s shortest path algorithm) · 1 hour ago

  - Cracker

    Implementation:...

    Implement Stack using Queues · 2 hours ago