GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

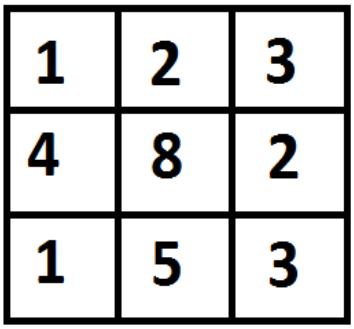
- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

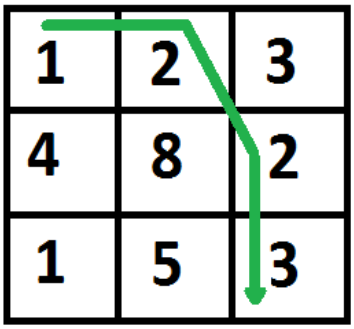
Dynamic Programming | Set 6 (Min Cost Path)

Given a cost matrix cost[][] and a position (m, n) in cost[][], write a function that returns cost of minimum cost path to reach (m, n) from (0, 0). Each cell of the matrix represents a cost to traverse through that cell. Total cost of a path to reach (m, n) is sum of all the costs on that path (including both source and destination). You can only traverse down, right and diagonally lower cells from a given cell, i.e., from a given cell (i, j), cells (i+1, j), (i, j+1) and (i+1, j+1) can be traversed. You may assume that all costs are positive integers.

For example, in the following figure, what is the minimum cost path to (2, 2)?



The path with minimum cost is highlighted in the following figure. The path is (0, 0) –> (0, 1) –> (1, 2) –> (2, 2). The cost of the path is 8 (1 + 2 + 2 + 3).



1) Optimal Substructure
The path to reach (m, n) must be through one of the 3 cells: (m-1, n-1) or (m-1, n) or (m, n-1). So minimum cost to reach (m, n) can be written as "minimum of the 3 cells plus cost[m][n]".

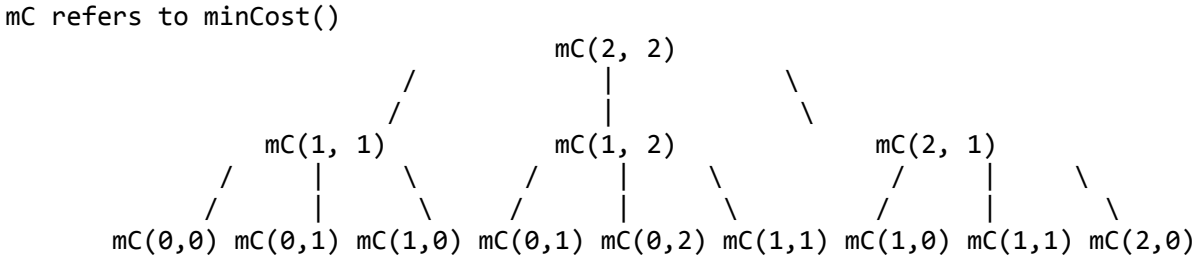
minCost(m, n) = min (minCost(m-1, n-1), minCost(m-1, n), minCost(m, n-1)) + cost[m][n]

2) Overlapping Subproblems
Following is simple recursive implementation of the MCP (Minimum Cost Path) problem. The implementation simply follows the recursive structure mentioned above.

```
/* A Naive recursive implementation of MCP(Minimum Cost Path) problem */
#include<stdio.h>
#include<limits.h>
#define R 3
#define C 3

int min(int x, int y, int z);

/* Returns cost of minimum cost path from (0,0) to (m, n) in mat[R][C]*/
int minCost(int cost[R][C], int m, int n)
{
   if (n < 0 || m < 0)
      return INT_MAX;
   else if (m == 0 && n == 0)
      return cost[m][n];
   else
      return cost[m][n] + min( minCost(cost, m-1, n-1),
                               minCost(cost, m-1, n),
                               minCost(cost, m, n-1) );
}
```

```
/* A utility function that returns minimum of 3 integers */
int min(int x, int y, int z)
{
   if (x < y)
      return (x < z)? x : z;
   else
      return (y < z)? y : z;
}

/* Driver program to test above functions */
int main()
{
   int cost[R][C] = { {1, 2, 3},
                      {4, 8, 2},
                      {1, 5, 3} };
   printf(" %d ", minCost(cost, 2, 2));
   return 0;
}
```

It should be noted that the above function computes the same subproblems again and again. See the following recursion tree, there are many nodes which apear more than once. Time complexity of this naive recursive solution is exponential and it is terribly slow.



So the MCP problem has both properties (see this and this) of a dynamic programming problem. Like other typical Dynamic Programming(DP) problems, recomputations of same subproblems can be avoided by constructing a temporary array tc[][] in bottom up manner.

```
/* Dynamic Programming implementation of MCP problem */
#include<stdio.h>
#include<limits.h>
#define R 3
#define C 3

int min(int x, int y, int z);

int minCost(int cost[R][C], int m, int n)
{
     int i, j;

     // Instead of following line, we can use int tc[m+1][n+1] or
     // dynamically allocate memoery to save space. The following line is
     // used to keep te program simple and make it working on all compilers.
     int tc[R][C];
```

```
    tc[0][0] = cost[0][0];

    /* Initialize first column of total cost(tc) array */
    for (i = 1; i <= m; i++)
       tc[i][0] = tc[i-1][0] + cost[i][0];

    /* Initialize first row of tc array */
    for (j = 1; j <= n; j++)
       tc[0][j] = tc[0][j-1] + cost[0][j];

    /* Construct rest of the tc array */
    for (i = 1; i <= m; i++)
       for (j = 1; j <= n; j++)
           tc[i][j] = min(tc[i-1][j-1], tc[i-1][j], tc[i][j-1]) + cost[i][j]

    return tc[m][n];
}

/* A utility function that returns minimum of 3 integers */
int min(int x, int y, int z)
{
   if (x < y)
      return (x < z)? x : z;
   else
      return (y < z)? y : z;
}

/* Driver program to test above functions */
int main()
{
   int cost[R][C] = { {1, 2, 3},
                      {4, 8, 2},
                      {1, 5, 3} };
   printf(" %d ", minCost(cost, 2, 2));
   return 0;
}
```

Time Complexity of the DP implementation is O(mn) which is much better than Naive Recursive implementation.

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: Dynamic Programming

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

Deepak • 13 days ago

It should be Cost[m-1][n-1] and not Cost [m][n] in recurssion function.

trend_setter • 5 months ago

I am getting 'time limit exceeded error' while executing this code. Plz tell me how to rectify it

```
#include <stdio.h>

int m=0,n=0;

int sum=0,i,j,k=2,l=2;

int main()

{ int mat[][3] ={ {1,2,3},

{4,8,2},

{1,5,3} };

/*for(int i=0;i<3;i++)

{ for(int j=0;j<3;j++)

{
```


Ankit Gupta • 6 months ago

What if restrictions on traversing the matrix is removed, earlier, only east, south and south east

directions were allowed only. What if all the directions are allowed?

• Reply • Share ›

sk ➚ Ankit Gupta • a month ago

What you mean by all direction ? going back or what ?
If it's allowed in 8 directions then you have to remember (i,j) as visited like DFS.

• Reply • Share ›

Ankit Gupta ➚ sk • a month ago

I am talking about a matrix like this:

1 9 9 1 1 9 9
9 1 9 1 8 1 9
9 1 9 1 9 1 9
9 9 1 1 9 1 9
9 9 9 9 1 9 9
9 9 9 9 9 1 9
9 9 9 9 9 9 1

In the above matrix, if we want to find an optimal path from top left corner to bottom right, we have to traverse through all the 1s right? So what approach will be required to solve this problem?

• Reply • Share ›

sk ➚ Ankit Gupta • a month ago

Still above algorithm will work . 1->1->1->1->1->1->1->1. And you will get 8 as cost .

• Reply • Share ›

rahulgiri.bhumca • 7 months ago

http://ideone.com/bVbbgn

• Reply • Share ›

sukanya • 8 months ago

http://ideone.com/bqt4md

• Reply • Share ›

BATMAN • 8 months ago

Can someone please explain why we need to initialize the first row and column ? I don't think it would be needed

7 • Reply • Share ›

Jerry Goyal ➚ BATMAN • 5 months ago

while calculating cost of 1st row or 1st column we are not allowed to move up or left, so

while calculating cost of 1st row or 1st column we are not allowed to move up or left, so we just added up current cost of row/column value with previous row/column cost.

• Reply • Share ›

BATMAN ➚ Jerry Goyal • 3 months ago

But can we not go to (1,0) from (0,0) by taking the following path - (0,0)->(0,1)->(1,1)->(1,0) assuming this can have a lower cost ... ?

• Reply • Share ›

Shivam Garg ➚ BATMAN • 2 months ago

then y not move (0,0)->(1,0) .. that would have even smaller cost? Gaurav goyal gave crrct explanation to your query

• Reply • Share ›

anon ➚ BATMAN • 6 months ago

In the cost matrix, for formula "min(tc[i-1][j-1], tc[i-1][j], tc[i][j-1])" we need to have i-1 and j-1 values. Thats why we need to initialize the first row and column.

• Reply • Share ›

don • 8 months ago

1 3 6
5 9 5
6 10 8

• Reply • Share ›

Guest • 9 months ago

Couldn't you just apply a greedy algorithm and select the next lowest number at each stage?

• Reply • Share ›

a ➚ Guest • 8 months ago

it may not give you the optimal solution.
Consider
1 3 3
1 30 20
20 20 20 ==> greedy algorithm give u a path: 1->1->20->20->20 (62)
Optimal: 1->3->3->20->20 (47)

2 • Reply • Share ›

django ➚ a • 2 months ago

It should be 42
optimal: 1->1->20->20

• Reply • Share ›

as ➚ a • 8 months ago

**corrected
it may not give you the optimal solution.
Consider
1 3 3
1 30 20
20 50 20 ==> greedy algorithm give u a path: 1->1->20->50->20 (92)
Optimal: 1->3->3->20->20 (47)

1 • Reply • Share ›

Anon as • 6 months ago
I think its 1 -> 3 -> 20 -> 20.

• Reply • Share ›

ch Anon • 5 months ago
Applying Dijkstra's algorithm which is greedy also gives the right solution here which is 1 -> 3 -> 20 -> 20.

• Reply • Share ›

Jun • 9 months ago
Method 2
http://ideone.com/2GzP71

• Reply • Share ›

Jun • 9 months ago
Method 1
http://ideone.com/2GzP71

• Reply • Share ›

rahul • 10 months ago
Here in DP solution R and C both are 3,

```
for (i = 1; i <= m; i++)
tc[i][0] = tc[i-1][0] + cost[i][0];

/* Initialize first row of tc array */
for (j = 1; j <= n; j++)
tc[0][j] = tc[0][j-1] + cost[0][j];
```

in both loops i and j runs till i=3 and j=3,how it is acess cost[3][0] and cost[0][3]. why this not array bound,please explain.

• Reply • Share ›

shantanu rahul • 10 months ago
see arguments passed are 2, 2

• Reply • Share ›

epsIon • a year ago

Here is a solution Using Memoization

```
#include<stdio.h>
#define M 3
#define N 3
#define I 999
int table[M][N];

int min(int a,int b,int c)
{
if(a==I && b==I && c==I)
return 0;
else
{
if(a<=b)
{
if(a<=c)
return a;
```

see more

• Reply • Share ›

Ekta Goel • a year ago

In order to print the path also, we can keep the track of which element returned minimum out of three using arrows and then backtrack from cell (m,n) .. (much the same way it is done in longest common subsequence problem). But is there any way out to do without this..?

• Reply • Share ›

Ekta Goel ➚ Ekta Goel • a year ago

I did this way: http://ideone.com/i3AtVG

• Reply • Share ›

Swarup Mallick • a year ago

Here in the naive approach there is a condition like

```
else if (m == 0 && n == 0)
return cost[m][n];
```

It should be the sum of that row or column.

If I will enter only one row or one column then the mentioned code will return the last element which is wrong.

Please correct me if I am wrong ?

• Reply • Share ›

dmr • a year ago

If we were allowed to traverse in all possible directions rather than the three allowed here, could we have still used DP ? Personally I think we can as still we would have optimal and overlapping subproblems.

1 • Reply • Share ›

Paparao Veeragandham dmr • a year ago

DP method does not work If you are allowed to Traverse all possible paths . Because it will be Infinity loop. All DP problems will be map to Short path from source to Dest. In above case it was not possible.

2 • Reply • Share ›

Mukesh M Paparao Veeragandham • a year ago

Considering (0,0) as source A and (m,n) as destination B given problems is similar to any shortest path algorithm for A to B. DP can be used to solve in case all directions are unblocked to traverse with loop coming only in case of negative weights.

• Reply • Share ›

prashant jha • a year ago

```
/*

#include<iostream>
#define n 4
#define infinity 999999
using namespace std;
int min(int a,int b)
{
return (a>b)?b:a;
}
int min(int a,int b,int c)
{
return min(min(a,b),c);
}
int fun(int a[n][n],int b[n][n],int s1,int s2,int v1,int v2)
{
if((s1==v1)&&(s2==v2))
return a[s1][s2];
```

see more

• Reply • Share ›

Tanmay • a year ago

@Tulsi das khan: your memoization will not work becoz ... look carefully dp[0,0] will store the shortest route to (2,2).. but we would want dp(2,2) to do that.. try running your code and print dp array

• Reply • Share ›

Animesh Pratap Singh Sikarwar • a year ago

I dont find it correct.... as we copy the 0th row and 0th column, straight away..!!

what if matrix is...
4 4 4
5 1 1
3 1 1

your t[][] will look like this
4 8 12
9 x x
12 x x

so this algo will give wrong cost if destination d(m,n) liea in 0th row or 0thcolumn.

1 • Reply • Share ›

Mojo ➦ Animesh Pratap Singh Sikarwar • a year ago

I think what you are not taking into account is the fact that you are not allowed to move left or upwards. If you want to know the cost[2][0] the answer is and should be 12 and not 8 which you can get if you were allowed to move 4->1->3.

1 • Reply • Share ›

Animesh Pratap Singh Sikarwar ➦ Mojo • a year ago

ohh yes....!!

• Reply • Share ›

lakshay ➦ Animesh Pratap Singh Sikarwar • a year ago

how about moving from 4->1->1->3 then the asnwer should be 9. i think thats allowed?

• Reply • Share ›

giri ➦ Animesh Pratap Singh Sikarwar • a year ago

the code is perfectly fine. check your algo again.

• Reply • Share ›

pavansrinivas • a year ago

I think this can also be solved using greedy technique.....
Correct me if I am wrong..

• Reply • Share ›

Tulsi das khan • a year ago

```
#include <iostream>
#include <string.h>
#include <limits.h>
using namespace std;

int cost[100][100];
int m, n;
int dp[100][100];
int minSumPath(int i, int j)
{
    if(dp[i][j] != -1)
        return dp[i][j];


    if(i==  m-1 && j == n-1)
    {
        dp[i][j] = cost[i][j];
```

see more

• Reply • Share ›

goobs Tulsi das khan • 8 months ago

Why do u use this condition if(dp[i][j] != -1) ?

• Reply • Share ›

shdas • 2 years ago

Is this a DP soln??

#include <stdio.h>

#include <stdlib.h>

int **dist;

int min(int a, int b, int c){

if(a<b) return="" (a<c)?a:c;="" else="" return="" (b<c)?b:c;="" }="" int="" cost(int="" **mat,="" int="" row,="" int="" col,="" int="" r,="" int="" c){="" if(dist[r][c]!="-1)" return="" dist[r][c];="" int="" val;="" if(r="0" &&="" c="">0)

```
val = cost(mat,row,col,r,c-1)+mat[r][c];

else if(r>0 && c==0)

val = cost(mat,row,col,r-1,c)+mat[r][c];
```

see more

• Reply • Share ›

udit • 2 years ago

java solution:

public class Min_Cost_Path {
static int min(int x,int y,int z)
{
if(x<y) return="" (x<z)?x:z;="" else="" return="" (y<z)?y:z;="" }="" static="" int="" mincost(int="" cost[][],int="" m,int="" n)="" {="" int="" i,j;="" int="" tc[][]="new" int[m+1][n+1];="" tc[0][0]="cost[0][0];" initailizsing="" first="" column="" for(i="1;i<=m;i++)" tc[i][0]="tc[i-1][0]+cost[i][0];" insitainzg="" first="" row="" for(j="1;j<=n;j++)" tc[0][j]="tc[0][j-1]+cost[0][j];" filling="" all="" other="" for(i="1;i<=m;i++)" {="" for(j="1;j<=n;j++)" {="" tc[i][j]="min(tc[i-1][j-1],tc[i][j-1],tc[i-1][j])+cost[i][j];" }="" }="" return="" tc[m][n];="" }="" public="" static="" void="" main(string[]="" args)="" {="" int="" cost[][]="{{1,2,3},{4,8,2},{1,5,3}};" system.out.println(mincost(cost,1,1));="" }="" }="">

1 • Reply • Share ›

Guduru Siva Reddy • 2 years ago

Java Solution :::

```
public class MinCost {

public static int min(int a,int b){
if(a>b) return b;
else return a;
}
public static int mincost(int [][]a){
int l=a.length;
int min=0;

int b[][]=new int[l][l];

for(int i=0;i<l;i++){ for(int="" j="0;j&amp;lt;l;j++){" if(i="=0&amp;amp;amp;amp;j==0){" b[i][j]="a[i][j];" }="" if(i="=0&amp;amp;amp;amp;j!=0){" b[i][j]="b[i][j-1]+a[i][j];" }="" if(j="=0&amp;amp;amp;amp;i!=0){" b[i][j]="b[i-1][j]+a[i][j];" }else="" if(i="">0&&j>0){
min=min(b[i-1][j-1],b[i][j-1]);
min=min(min,b[i-1][j]);
```

```
min=min(min, b[i-1][j]);
```

see more

• Reply • Share ›

nahar.abhishek9 • 2 years ago

Can you provide a solution if we can travel diagonally upwards as well.

• Reply • Share ›

mani • 2 years ago

here memoisation works better.correct?

1 • Reply • Share ›

sandy • 3 years ago

What if the array is as defined below:

```
/* Paste your code here (You may delete these lines if not writing code) */
  int cost[R][C] = { {1, 2, 3},
                     {4, 1, 2},
                     {1, 5, 3} };
```

Then the path cost using the code below for the top row would be wrong as we have a shorter path (1,1,3) not (1,2,3)

```
/* Paste your code here (You may delete these lines if not writing code) */
    /* Initialize first row of tc array */
    for (j = 1; j <= n; j++)
       tc[0][j] = tc[0][j-1] + cost[0][j];
```

• Reply • Share ›

bhuvi ➦ sandy • 2 years ago

Sandy,
Read this line "You can only traverse down, right and diagonally lower cells from a given cell, i.e., from a given cell (i, j), cells (i+1, j), (i, j+1) and (i+1, j+1) can be traversed" . I think now initializing 1st row an d 1st col makes sense

```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

nikhil ➦ bhuvi • 2 years ago

What if we can traverse diagonally upper cells..?

What modification can be done in above solution to solve the problem

What modification can be done in above solution to solve the problem...

```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

Anil Kag nikhil • 2 years ago

Will you really like to choose the upper diagonal while finding the min-cost path? Costs are assumed to be positive here. And you need to find path from (0,0) to (m,n), why would you like to go up diagonally?

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

- 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

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 1 hour ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

because we first fill zero in first col and...

Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago