

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)`?

1	2	3
4	8	2
1	5	3

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`).

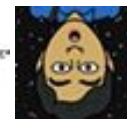
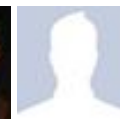
Google™ Custom Search



GeeksforGeeks



53,524 people like GeeksforGeeks.



Interview Experiences

Advanced Data Structures

Dynamic Programming

Greedy Algorithms

Backtracking

1	2	3
4	8	2
1	5	3

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

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]".

$$\text{minCost}(m, n) = \min(\text{minCost}(m-1, n-1), \text{minCost}(m-1, n), \text{minCost}(m, n-1)) + \text{cost}[m][n]$$

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;
}
```

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

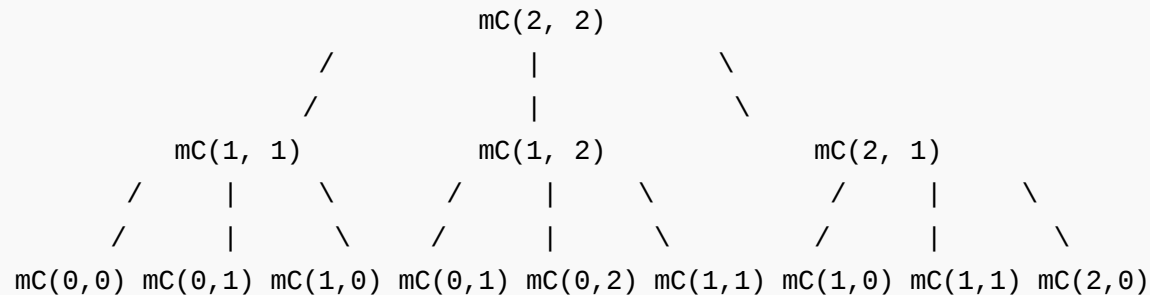
```

/* 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 appear more than once. Time complexity of this naive recursive solution is exponential and it is terribly slow.

mC refers to minCost()



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 memory to save space. The following line
    // used to keep the program simple and make it working on all comp
    int tc[R][C];
}

```



Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 38 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 42 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

newCoder3006 Code without using while loop. We can do it..

Find subarray with given sum · 1 hour ago

AdChoices 

► [C++ Vector](#)

► [Matrix in Java](#)

```

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.



► [C++ Code](#)

AdChoices ►

► [Log Min](#)

► [Min Min](#)

► [Get Matrix](#)

AdChoices ►

► [Time Matrix](#)

► [Int](#)

► [Min String](#)

Related Tpoics:

- [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](#)



8



Tweet

0



0

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

42 Comments

GeeksforGeeks

Sort by Newest ▼



with the recursion...



Ekta Goel · 17 days ago

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

^ | v ·



Ekta Goel → Ekta Goel · 17 days ago

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

^ | v ·



Swarup Mallick · 2 months 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 th

Please correct me if I am wrong ?

^ | v ·



dmr · 4 months ago

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

^ | v ·



Paparao Veeragandham → dmr · 3 months ago

DP method does not work If you are allowed to Traverse all possible p

^ | v .



prashant jha · 5 months 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](#)

^ | v .



Tanmay · 5 months ago

@Tulsi das khan: your memoization will not work becoz ... look carefully dp[0,1] but we would want dp(2,2) to do that.. try running your code and print dp array

^ | v .



Animesh Pratap Singh Sikarwar · 6 months ago

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 0thcolour

1 ^ | v .



Mojo → Animesh Pratap Singh Sikarwar · 5 months ago

I think what you are not taking into account is the fact that you are not a want to know the cost[2][0] the answer is and should be 12 and not 8 v to move 4->1->3.

1 ^ | v .



Animesh Pratap Singh Sikarwar → Mojo · 5 months ago

ohh yes....!!

^ | v .



lakshay → Animesh Pratap Singh Sikarwar · 5 days ago

how about moving from 4->1->1->3 then the asnwer sh

^ | v .



giri → Animesh Pratap Singh Sikarwar · 5 months ago

the code is perfectly fine. check your algo again.

^ | v .



pavansrinivas • 6 months ago

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

^ | v .



Tulsi das khan • 6 months 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

^ | v .



shdas • 8 months 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(i  
col,="" int="" r,="" int="" c){="" if(dist[r][c]!="-1)" return="" dist[r][c];="" int="" val
```

```
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];
```

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

see more

^ | v .



udit • 8 months 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=  
n)="" {="" int="" i,j;="" int="" tc[][]="new" int[m+1][n+1];="" tc[0][0]="cost[0][0];" i  
for(i="1;i<=m;i++)" tc[i][0]="tc[i-1][0]+cost[i][0];" insitainzg="" first="" row="" f  
1]+cost[0][j];" filling="" all="" other="" for(i="1;i<=m;i++)" {="" for(j="1;j<=n;j  
1],tc[i-1][j])+cost[i][j];" }="" }="" return="" tc[m][n];="" }="" public="" static="" voi  
cost[][]="{{1,2,3},{4,8,2},{1,5,3}};" system.out.println(mincost(cost,1,1));="" }
```

1 ^ | v .



Guduru Siva Reddy • 8 months 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<l;j++){ if(i==0&&j==0){ b[i][j]=''  
        b[i][j]="b[i][j-1]+a[i][j];" }="" if(j==0&&i!=0){ b[i][j]="b[i-1][j]+a[i][j];" }else  
        min=min(b[i-1][j-1],b[i][j-1]);  
        min=min(min, b[i-1][j]);  
        b[i][j]=min+a[i][j];
```

[see more](#)

^ | v •



nahar.abhishek9 • 9 months ago

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

^ | v •



mani • 10 months ago

here memoisation works better.correct?

1 ^ | v •



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

```
/* 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];
```

^ | v .



bhuv → sandy · 2 years ago

Sandy,

Read this line "You can only traverse down, right and diagonally lower cell (i, j), cells (i+1, j), (i, j+1) and (i+1, j+1) can be traversed". I think no makes sense

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

^ | v .



nikhil → bhuv · a year ago

What if we can traverse diagonally upper cells..?

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

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

^ | v .



Anil Kag → nikhil · 10 months ago

Will you really like to choose the upper diagonal while fir assumed to be positive here. And you need to find path to go up diagonally?

^ | v .



sneakysnoopy → Anil Kag · 6 months ago

it will depend on the cost matrix , right ..imagine a cost r

1 8 9 9 9 9

1 9 9 9 9 9

1 9 9 9 9 9

1 1 1 1 1 1

1 1 1 1 1 1

In this , the optimal way to reach to a 9 of third row is gc
?

^ | v .



kunal · 2 years ago

This is an straight forward application of : Maximum size square sub-matrix wi

^ | v .



Simple_thinking → kunal · 10 months ago

You don't really see the Dynamic programming solution don't you ?

^ | v .



jedimaster760 · 2 years ago

Provided that we have a 16x16 matrix, would this algorithm work? I was thinkir

and then synchronizing them back which would yield the best path.. Am I wro

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

^ | v .



kartikaditya · 2 years ago

In the question it is mentioned "... You may assume that all costs are positive i
assumption? Don't the method work even with negative values since we have

^ | v .



Geek · 2 years ago

How can we print the entire path in addition to the min cost.
plzz help

^ | v .



anmol · 3 years ago

You guys rock. provide excellent free content :) so happy to find your site, mar
don't even stand for what you provide free.

1 ^ | v .



GeeksforGeeks → anmol · 3 years ago

@anmol: Thanks for the nice comments. Keep visiting us and keep co

^ | v .




simon · 3 years ago

what if you can also go UP? What will it change the problem, and what is the t
approach?

^ | v .



kartik → simon · 3 years ago

 See [this](#) comment for time complexity. Considering that all costs are p if are allowed to?

^ | v .



simon → kartik · 3 years ago

can you plz also expand to neg cost, and go up is possible? Th interesting. will the time complexity be exponential?

^ | v .



kartik → simon · 3 years ago

Sure, sounds interesting. One way to solve is to apply E a thread on [forum](#) (using the [Ask a Question](#) page) for r

^ | v .



praveen · 3 years ago

This is similar to Dijkstra's shortest path algo. Isn't it?

^ | v .



Venki → praveen · 3 years ago

@praveen, It is more analogous to edit distance problem. For complete Programming problems list.

^ | v .



kartik → praveen · 3 years ago

You can say similar as both are DP based and both are for min cost p: this. Dijkstra is for a any graph with non-negative edges. Let me know :

^ | v .



kartikaditya → kartik · 2 years ago

Adding to it, the time complexity if done using Dijkstra based on => $O(3*mn \log mn)$, in the above case.

Correct me if I'm wrong.

^ | v .



kamal · 3 years ago

Good one. What is the time complexity of naive recursive solution?

^ | v .



kartik → kamal · 3 years ago

Time complexity is $O(3^n)$. The recursion tree is a complete ternary tree

2 ^ | v .



kamal → kartik · 3 years ago

Thanks Kartik

^ | v .



Subscribe



Add Disqus to your site

@geeksforgeeks, **Some rights reserved**

Contact Us!

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