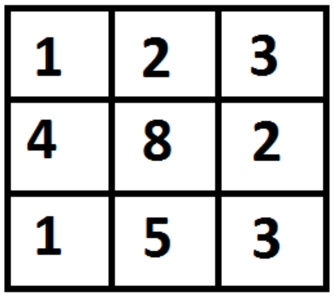
Dynamic Programming | Set 6 (Min Cost Path)

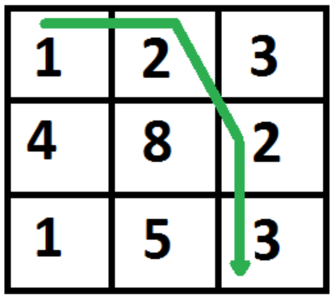
Given a cost matrix cost[][] and a position (m, n) in cost[][], write a function that returns cost of minimum cost path to reach (m, n) from (0, 0). Each cell of the matrix represents a cost to traverse through that cell. Total cost of a path to reach (m, n) is sum of all the costs on that path (including 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).



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;
}
```

Run on IDE

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.

```
mC refers to minCost()
                                    mC(2, 2)
                          /            |            \
                         /             |             \
                 mC(1, 1)           mC(1, 2)             mC(2, 1)
              /     |     \       /     |     \           /     |     \
             /      |      \     /      |      \         /      |       \
       mC(0,0) mC(0,1) mC(1,0) mC(0,1) mC(0,2) mC(1,1) mC(1,0) mC(1,1) mC(2,0)
```

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.

C++

```c
/* 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;
}
```

Run on IDE

Java

```java
/* Java program for Dynamic Programming implementation
   of Min Cost Path problem */
import java.util.*;

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

    private 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];

        /* 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];
    }

    /* Driver program to test above functions */
    public static void main(String args[])
    {
        int cost[][]= {{1, 2, 3},
                       {4, 8, 2},
                       {1, 5, 3}};
        System.out.println("minimum cost to reach (2,2) = " +
                                        minCost(cost,2,2));
    }
}
// This code is contributed by Pankaj Kumar
```

Run on IDE

Python

```
# Dynamic Programming Python implementation of Min Cost Path
# problem
R = 3
C = 3

def minCost(cost, m, n):

    # 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.
    tc = [[0 for x in range(C)] for x in range(R)]

    tc[0][0] = cost[0][0]

    # Initialize first column of total cost(tc) array
    for i in range(1, m+1):
        tc[i][0] = tc[i-1][0] + cost[i][0]
    # Initialize first row of tc array
    for j in range(1, n+1):
        tc[0][j] = tc[0][j-1] + cost[0][j]

    # Construct rest of the tc array
    for i in range(1, m+1):
        for j in range(1, n+1):
            tc[i][j] = min(tc[i-1][j-1], tc[i-1][j], tc[i][j-1]) + cost[i][j]

    return tc[m][n]

# Driver program to test above functions
cost = [[1, 2, 3],
        [4, 8, 2],
        [1, 5, 3]]
print(minCost(cost, 2, 2))

# This code is contributed by Bhavya Jain
```

Run on IDE

Output:

```
8
```

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.

157 Comments GeeksforGeeks shreejit ray chaud...

Recommend 9 Share Sort by Newest

Join the discussion…

Ameya Chikodi • a month ago

We dont need 'tc' matrix of size (m+1)*(n+1) instead it should be m*n. otherwise we get ArrayIndexOutOfBound exception

Reply • Share ›

Anil • 2 months ago

Hello

Can anyone tell me whats wrong with below approach:

At every node, i'll pick the least node. If i apply this logic to the below example, 1 -> 2 -> 2 -> 3 = 8, i can still find the min cost. I am sure i am missing some logic. Please help.
{1, 2, 3},
{4, 8, 2},
{1, 5, 3}
The difference between my approach and recursive one is, i am picking the minimum node and then going forward where as in recursive, we need to compute the whole chain.

Reply • Share ›

faizan Anil • a month ago

1, 4, 2
2, 20, 3
8, 6, 7
Get from top left to bottom right
The min path will be 1->4->3->7 Cost = 15
But choosing the min value each node will result in you choosing 1->2->8->6->7 Cost = 24. Hence greedy aint gonna work here.

Reply • Share ›

maverick • 3 months ago

For me it made more sense to update the right/diagonal/bottom with the min cost seen so far (as Dijkstra does) since we can assume that the current already has the minimum cost (0,0): http://code.geeksforgeeks.org/...

1 ^ | v • Reply • Share ›

Kiran Nalawade • 4 months ago

Check this its easier.
https://github.com/kiran-nalaw...

^ | v • Reply • Share ›

Rahul Kumar → Kiran Nalawade • 3 months ago

what is the complexity of recursive solution ?

^ | v • Reply • Share ›

Abhiram Shastri • 4 months ago

I don't see the point of going till m. (m-1) should be sufficient. Also cost[3][0] is not defined.

1 ^ | v • Reply • Share ›

Nik • 6 months ago

How do you analyze the Naive Recursive implementation??What is the time complexity?

^ | v • Reply • Share ›

baymax28 → Nik • 5 months ago

It' exponential of n. Similar to analysis of recursive fibonacci problem.

https://www.youtube.com/watch?...

^ | v • Reply • Share ›

Aritra Sinha • 6 months ago

Gives wrong answer(14) for minCost(cost, 0, 2) when cost[R][C] = { {1, 10, 3}, {4, 8, 2},{1, 5, 3} }, where the correct answer should be 12(1->8->3)

2 ^ | v • Reply • Share ›

thisisgaara → Aritra Sinha • 5 months ago

I believe a greedy implementation for this would give out 14. (1 -> 4 -> 1 -> 5->3). Otherwise, the DP solution discussed gives me 12.

^ | v • Reply • Share ›

Chinmay Pani → Aritra Sinha • 6 months ago

The correct answer is 14 since according to the constraints we can only move to the right, the down or diagonally down-right. so the only way to reach is (0,0) -> (0,1) -> (0,2). so ans is 1 + 10 +3 =14.

3 ^ | v • Reply • Share ›

Sha Terra • 6 months ago

Here is backtracking solution in C#
http://code.geeksforgeeks.org/...

Reply • Share ›

Poon • 6 months ago

why is there an initialization of 1st row and column as a sum of prev+current in method2?
Also, it is going from destination to source rather than source to destination, right?

Reply • Share ›

Chinmay Pani → Poon • 6 months ago

because thats the only way to go,,,when u are in the first row or column,,, u do not have three ways to go but just one

Reply • Share ›

Amit prajapati • 7 months ago

How to understand that how to approach such question in DP.... Before Reading the Post It seem difficult to make solution out of it,but after reading the post.It become very easy..but still in the next question same situation arises. Can anybody help!!

1 • Reply • Share ›

Qiang Li → Amit prajapati • 6 months ago

Just keep asking yourself, Can I derive the final result based on pre result? If positive, it is then dp-solvable and try to find the formula

Reply • Share ›

Amit prajapati • 7 months ago

I am getting wrong answer for this matrix: move to (2,2)

{ {1, 10, 3},
{1, 18, 5},
{3, 15, 3} };
Answer is coming: 19;

http://code.geeksforgeeks.org/...

Reply • Share ›

Chinmay Pani → Amit prajapati • 6 months ago

the answer is right the min Cost path is 1->10->5->3 which gives 19

Reply • Share ›

JeDi Man → Amit prajapati • 7 months ago

This is correct. 1 -> 10 -> 5 -> 3 = 19

Reply • Share ›

Amit prajapati ➦ JeDi Man • 7 months ago

thank you...I was stupit ..I didn't notice that.

Reply • Share ›

PINGPING WU ➦ Amit prajapati • 7 months ago

The given solution looks correct at first sight. I wonder how to fix it.

Reply • Share ›

Vardaan Sangar • 7 months ago

The Backtracking Solution
http://code.geeksforgeeks.org/...

Reply • Share ›

Lehar • 7 months ago

http://ideone.com/ZuS7R1

1 • Reply • Share ›

Shatakshi Agrawal ➦ Lehar • 7 months ago

Hi i saw ur code. 1 thing is not clear to me .. The question says calculate min cost path from (0,0) to any position (m,n) in the matrix.i see that u haven't taken anywhere m,n rather i think it's only (2,2) i.e. the last element. Pls can u explain more?Maybe i'm not getting something

Reply • Share ›

Karan Kapoor ➦ Lehar • 7 months ago

Nice :)

1 • Reply • Share ›

Karan Kapoor • 7 months ago

Both bottom up n top down

http://ideone.com/veRqKJ

Reply • Share ›

Solazy • 7 months ago

can somebody tell what will be the approach if cost is negative. I suppose it is backtracking.

Reply • Share ›

varun9 ➦ Solazy • 7 months ago

Yes, I believe it will work for negative cost as well. Think of this as shortest path distance problem with negative cost. Now to solve this we have Bellman–Ford algorithm, because Dijkstra algorithm is a greedy based approach, once it considers the vertex, it won't come again even if through some other path, the value to that vertex is smaller. Now here, you have directed graph where you can go either left[L], down[D] and diagonal[G]. That means you cannot reach a node apart from these steps [you cannot go up, right or anti diagonal]. Since DP equation here is taking care of all the paths from where a node can be reached, it should give correct result for whatever input you give. If **@bhavik gujarati** can vouch for it, we can then close it.

1 ^ | v • Reply • Share ›

bhavik gujarati • 8 months ago

Return when you reach the desired cell.

^ | v • Reply • Share ›

Shubham Chaudhary • 8 months ago

Somewhat a famous problem in placements ! :D

^ | v • Reply • Share ›

Avat

This comment was deleted.

Rajeev Ranjan ➦ Guest • 8 months ago

Problem statement says "You can only traverse down, right and diagonally lower cells from a given cell". So you can not go to (2,1) from (2,2).

^ | v • Reply • Share ›

Richa B • 8 months ago

Can anyone tell me why is the first row of tc[][] not getting initialized to the correct values in my code? Its the same as the one given here!

http://ideone.com/CVTUQN

^ | v • Reply • Share ›

.NetGeek ➦ Richa B • 8 months ago

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

Check condition of the above loop. It must be j instead of i :)

1 ^ | v • Reply • Share ›

Richa B ➦ .NetGeek • 8 months ago

oh.. that's a silly mistake.. n i couldn't find it even after going through the code again and again!..:p Thank u so much :)

^ | v • Reply • Share ›

.NetGeek • 8 months ago

C# Implementation: http://ideone.com/15Eo0q

1 ^ | v • Reply • Share ›

Shivam Maharshi • 9 months ago

Python Code: https://github.com/shivam-maha...

^ | v • Reply • Share ›

Lakhan Gite • 9 months ago

Check below link for better solution:
http://lakhangite.blogspot.in/...

^ | v • Reply • Share ›

amit • 9 months ago

If all 4 directions are allowed, can we use same approach ?

1 ^ | v • Reply • Share ›

Ambi ➦ amit • 8 months ago

No you can not use, i think in that case you should consider each block to be vertex and use dijkstra algorithm to solve the problem

^ | v • Reply • Share ›

.NetGeek • 9 months ago

C# Implementation: http://ideone.com/I5ZxKw

1 ^ | v • Reply • Share ›

Mahima • 10 months ago

can we also determine the path? like Home->Right->Diagonal->Down->down?

^ | v • Reply • Share ›

nishant sinha • a year ago

why initialisation of first row and column is required

^ | v • Reply • Share ›

Sally Talks ➦ nishant sinha • a year ago

To Stop Recursion.

1 ^ | v • Reply • Share ›

simar • a year ago

How to find the minimum cost path in a matrix if you can move left, right, up and down in the matrix?

^ | v • Reply • Share ›

Sally Talks ➦ simar • a year ago

I think it is same as these paths would always be costly than others. Your questions becomes valid in case of negative numbers.

1 ^ | v • Reply • Share ›

Murtaza Hasan • a year ago

if someone is interested in a more user based input answer, here's the code for that:https://ideone.com/PJxWGo

Reply • Share ›

sunil • a year ago

https://ideone.com/CyWG4N

Reply • Share ›

sunil • a year ago

```
#include<iostream>

#include<iomanip>

#include<climits>

using namespace std;

const int m=3,n=3;

int min_cost(int cost[][n],int m,int n,int r1,int c1,int r2,int c2,int **mc);

int min(const int a,const int b,const int c);

void print_matrix(int mat[][n],int m,int n);

int main(){

int cost[m][n]={

{1,2,3},
```

see more

Reply • Share ›

Load more comments