Maximum path sum in a triangle.

We have given numbers in form of triangle, by starting at the top of the triangle and moving to adjacent numbers on the row below, find the maximum total from top to bottom.

Examples:

```
Input :
   3
  7 4
 2 4 6
8 5 9 3
Output : 23
Explanation : 3 + 7 + 4 + 9 = 23

Input :
   8
 -4 4
 2 2 6
1 1 1 1
Output : 19
Explanation : 8 + 4 + 6 + 1 = 19
```

Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

We can go through the brute force by checking every possible path but that is much time taking so we should try to solve this problem with the help of dynamic programming which reduces the time complexity.

If we should left shift every element and put 0 at each empty position to make it a regular matrix, then our problem looks like minimum cost path.

So, after converting our input triangle elements into a regular matrix we should apply the dynamic programmic concept to find the maximum path sum.

Applying, DP in bottom-up manner we should solve our problem as:

Example:

```
    3
   7 4
  2 4 6
 8 5 9 3

 Step 1 :
 3 0 0 0
 7 4 0 0
 2 4 6 0
 8 5 9 3

 Step 2 :
 3  0  0  0
 7  4  0  0
 10 13 15 0

 Step 3 :
 3  0  0  0
 20 19 0  0

 Step 4:
 23 0 0 0

 output : 23
```

```
/* Dynamic Programming implementation of
   Max sum problem in a triangle */
#include<bits/stdc++.h>
using namespace std;
#define N 3

//  Function for finding maximum sum
int maxPathSum(int tri[][N], int m, int n)
{
     // loop for bottom-up calculation
     for (int i=m-1; i>=0; i--)
     {
        for (int j=0; j<=i; j++)
        {
            // for each element, check both
            // elements just below the number
            // and below right to the number
            // add the maximum of them to it
            if (tri[i+1][j] > tri[i+1][j+1])
                tri[i][j] += tri[i+1][j];
            else
                tri[i][j] += tri[i+1][j+1];
        }
     }

     // return the top element
     // which stores the maximum sum
     return tri[0][0];
}

/* Driver program to test above functions */
int main()
```

```
{
   int tri[N][N] = {  {1, 0, 0},
                      {4, 8, 0},
                      {1, 5, 3} };
   cout << maxPathSum(tri, 2, 2);
   return 0;
}
```

Run on IDE

Output:

```
14
```

This article is contributed by **Shivam Pradhan (anuj_charm)**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

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

GATE CS Corner Company Wise Coding Practice

Dynamic Programming triangle

Recommended Posts:

Friends Pairing Problem
LCS (Longest Common Subsequence) of three strings
Dynamic Programming | Set 6 (Min Cost Path)
Maximum points collected by two persons allowed to meet once
Size of array after repeated deletion of LIS
Count ways to build street under given constraints
Maximum length subsequence with difference between adjacent elements as either 0 or 1
Unique paths in a Grid with Obstacles
Largest rectangular sub-matrix whose sum is 0
Coin game winner where every player has three choices

(Login to Rate and Mark)

3.1 Average Difficulty : **3.1/5.0**
Based on **12** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Share this post!

3 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

guestmi • 2 months ago

what is defined as the adjacent in the next row here? if we could choose any elemnt in the next row then we would be done by taking the maximum only of the next row

Reply • Share ›

Skartik • 5 months ago

My solution: Time: O(n) and Space: O(1)
http://ide.geeksforgeeks.or...

Reply • Share ›

Sharan Lobana ➚ Skartik • 2 hours ago

The above solution is O(n) in number of elements not number of rows, its O(n*n) in number of rows.

Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd Privacy