

// <https://leetcode.com/problems/triangle/>

120. Triangle
Medium 2248 261 Add to List Share

Given a triangle, find the minimum path sum from top to bottom. Each step you may move to adjacent numbers on the row below.

For example, given the following triangle

```
[
  [2],
  [3,4],
  [6,5,7],
  [4,1,8,3]
]
```

The minimum path sum from top to bottom is 11 (i.e. $2 + 3 + 5 + 1 = 11$).

Note:
Bonus point if you are able to do this using only $O(n)$ extra space, where n is the total number of rows in the triangle.

Accepted 261,402 Submissions 586,364

See this question in a real interview before?

Companies

Related Topics

```
1 public class Solution {
2     public int MinimumTotal(IList<IList<int>> triangle) {
3         int smallestSum=Int32.MaxValue;
4         var len = triangle.Count;
5         for(int i=1;i<len;i++) // iterate thru all the Lists in Triangle and calculate min value possible at each index in that List
6         {
7             var listLen = triangle[i].Count;
8             for(int j=0;j<listLen;j++)
9             {
10                 if(j==0)
11                     triangle[i][j]+=triangle[i-1][0];
12                 else if (j==listLen-1)
13                     triangle[i][j]+=triangle[i-1][j-1];
14                 else
15                     triangle[i][j]+=Math.Min(triangle[i-1][j-1],triangle[i-1][j]);
16             }
17         }
18         for(int i=0;i<triangle[len-1].Count;i++)
19             smallestSum = Math.Min(smallestSum,triangle[len-1][i]);
20         return smallestSum;
21     }
22 }
23
```

Your previous code was restored from your local storage. [Reset to default](#)

Testcase	Run Code Result
Accepted	Runtime: 112 ms
Your input	[[2],[3,4],[6,5,7],[4,1,8,3]]
Output	11
Expected	11

```
public class Solution {

    public int MinimumTotal(IList<IList<int>> triangle) {

        int smallestSum=Int32.MaxValue;

        var len = triangle.Count;

        for(int i=1;i<len;i++) // iterate thru all the Lists in Triangle and calculate min value possible at
each index in that List

        {

            var listLen = triangle[i].Count;

            for(int j=0;j<listLen;j++)

            {

                if(j==0)

                    triangle[i][j]+=triangle[i-1][0];

                else if (j==listLen-1)

                    triangle[i][j]+=triangle[i-1][j-1];

                else

                    triangle[i][j]+=Math.Min(triangle[i-1][j-1],triangle[i-1][j]);

            }

        }

    }

}
```

```

    }
}
for(int i=0;i<triangle[len-1].Count;i++)
    smallestSum = Math.Min(smallestSum,triangle[len-1][i]);
return smallestSum;

}
}

```

Success [Details >](#)

Runtime: **96 ms**, faster than **92.00%** of C# online submissions for Triangle.

Memory Usage: **24.8 MB**, less than **18.00%** of C# online submissions for Triangle.

Next challenges:

[Super Washing Machines](#)

[Numbers At Most N Given Digit Set](#)

[Count All Valid Pickup and Delivery Options](#)

Show off your acceptance:



Time Submitted	Status	Runtime	Memory	Language
09/18/2020 23:23	Accepted	96 ms	24.8 MB	csharp
09/18/2020 22:55	Wrong Answer	N/A	N/A	csharp
09/18/2020 22:48	Runtime Error	N/A	N/A	csharp
09/18/2020 22:29	Wrong Answer	N/A	N/A	csharp