

# Minimum Path Sum

Solution

Given a  $m \times n$  grid filled with non-negative numbers, find a path from top left to bottom right which *minimizes* the sum of all numbers along its path.

**Note:** You can only move either down or right at any point in time.

**Example:**

**Input:**

```
[
  [1,3,1],
  [1,5,1],
  [4,2,1]
]
```

**Output:** 7

**Explanation:** Because the path 1→3→1→1→1 minimizes the sum.

Java



```
1 // 2D dp
2 /*class Solution {
3     public int minPathSum(int[][] grid) {
4         int[][] dp = new int[grid.length][grid[0].length];
5         dp[0][0] = grid[0][0];
6         for(int i = 1; i < dp[0].length; i++) {
7             dp[0][i] = grid[0][i] + dp[0][i-1];
8         }
9
10        for(int i = 1; i < dp.length; i++) {
11            dp[i][0] = grid[i][0] + dp[i-1][0];
12        }
13
14        for(int i = 1; i < dp.length; i++) {
15            for(int j = 1; j < dp[0].length; j++) {
16                dp[i][j] = grid[i][j] + Math.min(dp[i-1][j], dp[i][j-1]);
17            }
18        }
19    }
20
21    return dp[dp.length-1][dp[0].length-1];
22 }
23 }*/
24
25 // 1D dp
26 class Solution {
27     public int minPathSum(int[][] grid) {
28         int[] dp = new int[grid[0].length];
29         dp[0] = grid[0][0];
30         for(int i = 1; i < dp.length; i++) {
31             dp[i] = grid[0][i] + dp[i-1];
32         }
33
34         for(int i = 1; i < grid.length; i++) {
35             for(int j = 0; j < grid[0].length; j++) {
36                 if (j == 0) dp[j] += grid[i][j];
37                 else {
38                     dp[j] = grid[i][j] + Math.min(dp[j-1], dp[j]);
39                 }
40             }
41         }
42
43         return dp[dp.length-1];
44     }
45 }
46 }
```

☐ Custom Testcase ( [Contribute](#) )


Run Code

Submit