64. Minimum Path Sum

Solved

Medium | Topics | Companies

Given a `m x 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.

dp[i][j] represents min Path Sum from i,j to m-1, n-1.

$dp[i][j] = a_{ij} + \min\{dp[i+1][j], dp[i][j+1]\}$

$dp[m-1][n-1] = 1$

$dp[m+1][n+1] = \infty$

for (i: m-1 to 0)
for (j: n-1 to 0)
$dp[i][j] = a_{ij} + \min\{dp[i+1][j], dp[i][j+1]\}$

Example 1:


```
Input: grid = [[1,3,1],[1,5,1],[4,2,1]]
Output: 7
Explanation: Because the path 1 → 3 → 1 → 1 → 1 minimizes the sum.
```

Example 2:

```
Input: grid = [[1,2,3],[4,5,6]]
Output: 12
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 200`
- `0 <= grid[i][j] <= 200`

Same as Unique Paths

Approach 1: Recursion

Approach 2: Memoization

Approach 3: Tabulation

dp[i][j] represents the minimum path Sum from cell [0][0] to cell [i][j]

```cpp
class Solution {
public:
    int minPathSum(vector<vector<int>>& grid) {
        int m = grid.size();
        int n = grid[0].size();

        vector<vector<int>> dp(m,vector<int>(n,0));

        for(int i =0;i<m;i++){
            for(int j=0;j<n;j++){
                if(i==0 && j==0)
                    dp[i][j] = grid[i][j];
                else if(i==0)
                    dp[i][j] = dp[i][j-1] + grid[i][j];
                else if(j==0)
                    dp[i][j] = dp[i-1][j] + grid[i][j];
                else
                    dp[i][j] = grid[i][j] + min(dp[i-1][j] , dp[i][j-1]);
            }
        }

        return dp[m-1][n-1];    }
};
```

} 1st cell in path

} only one way i.e. from left

} i.e. from Top

Top ↓ left ↓

$T(n) : O(m \times n)$

$S(n) : O(m \times n)$

Approach 4: Space Optimization on Tabulation

```cpp
class Solution {
public:
    int minPathSum(vector<vector<int>>& grid) {
        int m = grid.size();
        int n = grid[0].size();
```

```
        vector<int> prevRow(n,0);
        int prevCol = 0;
        for(int i =0;i<m;i++){
            for(int j=0;j<n;j++){
                if(i==0 && j==0)
                    prevCol = grid[i][j];
                else if(i==0)
                    prevCol = prevCol + grid[i][j];
                else if(j==0)
                    prevCol = prevRow[j] + grid[i][j];
                else
                    prevCol = grid[i][j] + min(prevRow[j] , prevCol);
                prevRow[j] = prevCol;
            }
        }

        return prevRow[n-1] ;
    }
};
```

} 1st cell in path

} executed for 1st row

} executed for 1st column

$T(n) : O(m \times n)$

$S(n) : O(n)$