Problem Submissions

Correct Answer

Report

Submitted on May 15, 2022, 5:40:58 PM

Penalty	Score	Test Cases
10%	72	8/8

Runtime	Language
351ms	C++ (g++ 5.4)

Previous Submissions

Submitted On	Status	Score	Penalty
6 mins ago	Correct Answer 8/8 Test cases Passed	80 / 80	0%
10 mins ago	Wrong Answer 6/8 Test cases Passed	60 / 80	0%
19 mins ago	Correct Answer 8/8 Test cases Passed	80 / 80	0%
36 mins ago	Correct Answer 8/8 Test cases Passed	80 / 80	0%
53 mins ago	Correct Answer 8/8 Test cases Passed	80 / 80	0%
1 hour ago	Correct Answer 8/8 Test cases Passed	80 / 80	0%
1 hour ago	Time Limit Exceeded	49.6 / 80	0%
1 hour ago	Time Limit Exceeded	40 / 80	0%
2 mins ago	Correct Answer 8/8 Test cases Passed	80 / 80	0%

C++ (g++ 5.4)

```cpp
int solve(int r, int c, vector<vector<int>> &matrix, int n, vector<vector<int>>& dp){
    if(c < 0 or c >= matrix[0].size())
        return -1e8;
    if(r == (n - 1))
        return dp[r][c] = matrix[r][c];

    if(dp[r][c] != -1)
        return dp[r][c];

    int d = solve(r + 1, c, matrix, n, dp) + matrix[r][c];
    int dLeft = solve(r + 1, c - 1, matrix, n, dp) + matrix[r][c];
    int dRight = solve(r + 1, c + 1, matrix, n, dp) + matrix[r][c];

    return dp[r][c] = max(d, max(dLeft, dRight));
}
int getMaxPathSum(vector<vector<int>> &matrix)
{
    //  Write your code here.
    /*
        //This is Recursive and Memoization solution
            Recursive
                //TC = O(3^n)
                //SP = O(1)
            Memoization
                //TC = O(n * n)
                //SP = O(n * n + St)

    int columns = matrix[0].size();
    int rows = matrix.size();
    int ans = -1e8;
    vector<vector<int>> dp(rows, vector<int>(columns, -1));
    for(int i = 0; i < columns; i++){
        ans = max(ans, solve(0, i, matrix, rows, dp));
    }
    return ans;
    */

    /*
    //This is Tabulation Approach
            //TC = O(n*n)
            //SP = O(n*n)
    int columns = matrix[0].size();
    int rows = matrix.size();
    vector<vector<int>> dp(rows, vector<int>(columns, -1));
    int ans = -1e8;
    for(int c = 0; c < columns; c++)
        dp[rows - 1][c] = matrix[rows - 1][c];

    for(int r = rows - 2; r >= 0; r--){
        for(int c = 0; c < columns; c++){
            if(c == 0){
                int d = matrix[r][c] + dp[r + 1][c];
                int dr = matrix[r][c] + dp[r + 1][c + 1];
                dp[r][c] = max(d, dr);
            }
            else if(c == (columns - 1)){
                int d = matrix[r][c] + dp[r + 1][c];
                int dl = matrix[r][c] + dp[r + 1][c - 1];
                dp[r][c] = max(d, dl);
            }
            else{
                int d = matrix[r][c] + dp[r + 1][c];
                int dl = matrix[r][c] + dp[r + 1][c - 1];
                int dr = matrix[r][c] + dp[r + 1][c + 1];
                dp[r][c] = max(d, max(dl, dr));
            }
        }
    }
    for(int c = 0; c < columns; c++)
        ans = max(ans, dp[0][c]);
    return ans;
    */

        //This is space optimized solution
            //TC = O(n^2)
            //SP = O(n)
    int columns = matrix[0].size();
    int rows = matrix.size();
```

Console

Previous Next Show Hint Last saved on 5:31:01 PM Run Code Submit