56) Count square submatrices with all ones

14 December 2024 03:02

Given a m * n matrix of ones and zeros, return how many square submatrices have all ones.

From <https://leetcode.com/problems/count-square-submatrices-with-all-ones/description/>

```
int helper(int i, int j, vector<vector<int>>& matrix, vector<vector<int>>& dp){
    if(i >= matrix.size() || j >= matrix[0].size()) return 0;
    if(matrix[i][j] == 0) return 0;
    if(dp[i][j] != -1) return dp[i][j];
    int down = helper(i+1, j, matrix, dp);
    int diag = helper(i+1, j+1, matrix, dp);
    int right = helper(i, j+1, matrix, dp);
    return dp[i][j] = 1 + min({down, diag, right});
}
int countSquares(vector<vector<int>>& matrix) {
    int m = matrix.size();
    int n = matrix[0].size();
    vector<vector<int>>dp(m, vector<int>(n, -1));
    int result = 0;
    for(int i = 0; i < m; i++){
        for(int j = 0; j < n; j++){
            if(matrix[i][j] == 1){
                result += helper(i, j, matrix, dp);
            }
        }
    }
    return result;
}
```

57) Maximise the cut segment

13 February 2025 16:50

Given an integer **n** denoting the Length of a line segment. You need to cut the line segment in such a way that the cut length of a line segment each time is either **x** , **y** or **z**. Here x, y, and z are integers.
After performing all the cut operations, your total number of cut segments must be maximum. Return the maximum number of cut segments possible.
Note: if no segment can be cut then return 0.

From <https://www.geeksforgeeks.org/problems/cutted-segments1642/1>

```
int helper(int n, int x, int y, int z, vector<int> &dp){
    if(n == 0) return 0;
    if(n < 0) return INT_MIN;
    if(dp[n] != -1) return dp[n];

    int a = 1+helper(n-x, x, y, z, dp);
    int b = 1+helper(n-y, x, y, z, dp);
    int c = 1+helper(n-z, x, y, z, dp);
    return dp[n] = max(a, (max(b, c)));
}
int maximizeTheCuts(int n, int x, int y, int z) {
    vector<int> dp(n+1, -1);
    if(helper(n, x, y, z, dp) < 0) return 0;
    else return helper(n, x, y, z, dp);

}
```

58) Number of dice rolls with target sum

14 February 2025 17:16

You have n dice, and each dice has k faces numbered from 1 to k.
Given three integers n, k, and target, return *the number of possible ways (out of the* k^n *total ways) to roll the dice, so the sum of the face-up numbers equals* target. Since the answer may be too large, return it **modulo** $10^9 + 7$.

From <https://leetcode.com/problems/number-of-dice-rolls-with-target-sum/description/>

```
int helper(int n, int k, int target, vector<vector<int>>& dp) {
    if (target < 0) return 0;
    if (target == 0 && n != 0) return 0;
    if (target != 0 && n == 0) return 0;
    if (target == 0 && n == 0) return 1;
    if (dp[n][target] != -1) return dp[n][target];
    int ans = 0;
    for (int i = 1; i <= k; i++) {
        ans = (ans + helper(n - 1, k, target - i, dp)) % 1000000007;
    }
    return dp[n][target] = ans;
}
int numRollsToTarget(int n, int k, int target) {
    vector<vector<int>> dp(n + 1, vector<int>(target + 1, -1));
    return helper(n, k, target, dp);
}
```

59) Perfect Sq

14 February 2025 17:35

Given an integer n, return *the least number of perfect square numbers that sum to* n.
A **perfect square** is an integer that is the square of an integer; in other words, it is the product of some integer with itself. For example, 1, 4, 9, and 16 are perfect squares while 3 and 11 are not.

From <https://leetcode.com/problems/perfect-squares/description/>

```
int helper(int n, vector<int> &dp){
    if(n == 0) return 0;
    int minCount = INT_MAX;
    if(dp[n] != -1) return dp[n];
    for(int i = 1; i*i <= n; i++){
        int result = 1 + helper(n-i*i, dp);
        minCount = min (minCount, result);
    }
    return dp[n] = minCount;
}
int numSquares(int n) {
    vector<int>dp(n+1, -1);
    return helper(n, dp);
}
```

60) Min cost for tickets

14 February 2025 17:46

You have planned some train traveling one year in advance. The days of the year in which you will travel are given as an integer array days. Each day is an integer from 1 to 365.
Train tickets are sold in **three different ways**:

- a **1-day** pass is sold for costs[0] dollars,
- a **7-day** pass is sold for costs[1] dollars, and
- a **30-day** pass is sold for costs[2] dollars.

The passes allow that many days of consecutive travel.

- For example, if we get a **7-day** pass on day 2, then we can travel for 7 days: 2, 3, 4, 5, 6, 7, and 8.

Return *the minimum number of dollars you need to travel every day in the given list of days*.

From <https://leetcode.com/problems/minimum-cost-for-tickets/description/>

```
int helper(vector<int>& days, vector<int>& costs, int i, vector<int>&dp){
    if(i >= days.size()) return 0;
    if(dp[i] != -1) return dp[i];
    int cost1 = costs[0] + helper(days, costs, i+1, dp);
    int j = i;
    int endDay = days[i] + 7 - 1;
    while(j < days.size() && days[j] <= endDay){
        j++;
    }
    int cost7 = costs[1] + helper(days, costs, j, dp);
    j = i;
    endDay = days[i] + 30 - 1;
    while(j < days.size() && days[j] <= endDay){
        j++;
    }
    int cost30 = costs[2] + helper(days, costs, j, dp);
    return dp[i] = min(cost1, min(cost7, cost30));
}
int mincostTickets(vector<int>& days, vector<int>& costs) {
    vector<int> dp(days.size(), -1);
    return helper(days, costs, 0, dp);
}
```

61) Painting the Fence

14 February 2025 18:30

Given a fence with **n** posts and **k** colours, find out the number of ways of painting the fence so that **not more than two** consecutive posts have the same colours.
Answers are guaranteed to be fit into a 32 bit integer.

From <https://www.geeksforgeeks.org/problems/painting-the-fence3727/1>

```
int helper(int n, int k, vector<int> &dp){
    if(n == 1) return k;
    if(n == 2) return k*(k-1) + k;
    if(dp[n] != -1) return dp[n];
    return dp[n] = (k-1)*(helper(n-1, k, dp) + helper(n-2, k, dp));
}
int countWays(int n, int k) {
    vector<int> dp(n+1, -1);
    return helper(n, k, dp);
}
```

62) Maximal Square

15 February 2025 14:13

Given an m x n binary matrix filled with 0's and 1's, *find the largest square containing only* 1's *and return its area*.

From <https://leetcode.com/problems/maximal-square/description/>

```
    int solve(vector<vector<char>>& matrix, int i, int j, int row, int col, int&
maxi, vector<vector<int>>&dp){
        //base case
        if(i >= row || j >= col) return 0;
        if(dp[i][j] != -1) return dp[i][j];
        //explore all 3 directions
        int right = solve(matrix, i, j+1, row, col, maxi, dp);
        int diagonal = solve(matrix, i+1, j+1, row, col, maxi, dp);
        int down = solve(matrix, i+1, j, row, col, maxi, dp);
        //check can we build square from current position
        if(matrix[i][j] == '1'){
            int ans = 1 + min(right, min(down, diagonal));
            //ye imp h
            maxi = max(maxi, ans);
            return dp[i][j] = ans;
        }
        else return dp[i][j] = 0; //agr zero pe hi khade h to answer bhi 0 hoga
    }
    int maximalSquare(vector<vector<char>>& matrix) {
        int i = 0, j = 0;
        int row = matrix.size();
        int col = matrix[0].size();
        vector<vector<int>>dp(row+1, vector<int>(col+1, -1));
        int maxi = 0;
        int ans = solve(matrix, i , j, row, col, maxi, dp);
        return maxi*maxi;
    }
```