546. Remove Boxes

Add to List

Description HintsSubmissionsSolutions

- Total Accepted: **1144**
- Total Submissions: **3997**
- Difficulty: **Hard**
- Contributors:ckcz123

Given several boxes with different colors represented by different positive numbers.

You may experience several rounds to remove boxes until there is no box left. Each time you can choose some continuous boxes with the same color (composed of k boxes, k >= 1), remove them and get `k*k` points.

Find the maximum points you can get.

Example 1:

Input:

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

Output:

```
23
```

Explanation:

```
[1, 3, 2, 2, 2, 3, 4, 3, 1]
----> [1, 3, 3, 4, 3, 1] (3*3=9 points)
----> [1, 3, 3, 3, 1] (1*1=1 points)
----> [1, 1] (3*3=9 points)
```

```
----> [] (2*2=4 points)
```

Note: The number of boxes `n` would not exceed 100.

Subscribe to see which companies asked this question.

When facing this problem, I am keeping thinking how to simulate the case when `boxes[i] == boxes[j]` when `i` and `j` are not consecutive. It turns out that the dp matrix needs one more dimension to store such state. So we are going to define the state as

```
dp[i][j][k] represents the max points from box[i] to box[j] with k boxes whose values eq ual to box[i]
```

The transformation function is as below

```
dp[i][j][k] = max(dp[i+1][m-1][1] + dp[m][j][k+1]) when box[i] = box[m]
```

```
class Solution {
public:
    #define maxn  100
    int dp[maxn][maxn][maxn];

    int dfs(vector<int>& boxes,int i,int j,int k)
    {
        if(i>j) return 0;
        else if(i==j) return k*k;
        else if (dp[i][j][k]!=0)
        {
            return dp[i][j][k];// has been visited
        }else
        {
            int temp = dfs(boxes,i+1,j,1)+k*k;
            for(int m=i+1;m<=j;++m)
            {
                if(boxes[i]==boxes[m])
                {
                    temp                                                    =
max(temp,dfs(boxes,i+1,m-1,1)+dfs(boxes,m,j,k+1));
```

```
                }
            }
            dp[i][j][k]=temp;
            return temp;
        }


    }

    int removeBoxes(vector<int>& boxes) {
        if(boxes.size()==0) return 0;
        memset(dp,0,sizeof(dp));
        int ans = dfs(boxes,0,boxes.size()-1,1);
        return ans;
    }
};
```