474. Ones and Zeroes

Add to List

Description Hints Submissions Solutions

- Total Accepted: **8205**
- Total Submissions: **21886**
- Difficulty: **Medium**
- Contributors: piy9

In the computer world, use restricted resource you have to generate maximum benefit is what we always want to pursue.

For now, suppose you are a dominator of **m** `0s` and **n** `1s` respectively. On the other hand, there is an array with strings consisting of only `0s` and `1s`.

Now your task is to find the maximum number of strings that you can form with given **m** `0s` and **n** `1s`. Each `0` and `1` can be used at most **once**.

Note:

1. The given numbers of `0s` and `1s` will both not exceed `100`
2. The size of given string array won't exceed `600`.

This problem is a typical 0-1 knapsack problem, we need to pick several strings in provided strings to get the maximum number of strings using limited number 0 and 1. We can create a three dimensional array, in which dp[i][j][k] means the maximum number of strings we can get from the first i argument strs using limited j number of '0's and k number of '1's.

For dp[i][j][k], we can get it by fetching the current string i or discarding the current string, which would result in dp[i][j][k] = dp[i-1][j-numOfZero(strs[i])][i-numOfOnes(strs[i])] and dp[i][j][k] = dp[i-1][j][k]; We only need to treat the larger one in it as the largest number for dp[i][j][k].

```
//c++
void calculate(string s, vector<int> &count)
{
    int zero=0,one=0;
    for(char c:s)
    {
        if(c=='0') zero++;
        else if (c=='1') one++;
    }
    count[0] = zero;
    count[1] = one;
}

int findMaxForm(vector<string>& strs, int m, int n) {
    int l = strs.size();
    //int dp[l+1][m+1][n+1];
    vector<vector<vector<int>>>
dp(l+1,vector<vector<int>>(m+1,vector<int>(n+1,0)));
    for(int i=0;i<l+1;i++)
    {
```

```
        //cout<< i<< endl;
        vector<int> num = {0,0};
        if(i>0)
        {
            calculate(strs[i-1],num);
        }
        for(int j=0;j<m+1;j++)
        {
            for(int k=0;k<n+1;k++)
            {
                if(i==0)
                {
                    dp[i][j][k]=0;
                }else if (j>=num[0] && k>=num[1])
                {
                    dp[i][j][k] =
max(dp[i-1][j][k],dp[i-1][j-num[0]][k-num[1]]+1);
                }else{
                    dp[i][j][k] = dp[i-1][j][k];
                }
            }
        }
    }
    return dp[l][m][n];
}
```

```java
//java
public static int findMaxForm(String[] strs, int m, int n) {
        int[][] dp = new int[m + 1][n + 1];
        for(String str : strs){
            int one = 0;
            int zero = 0;
            for(char c : str.toCharArray()){
                if(c == '1')
                    one++;
                else
                    zero++;
            }
            for(int i = m; i >= zero; i--){
                for(int j = n; j >= one; j--){
                    if(one <= j && zero <= i)
                        dp[i][j] = Math.max(dp[i][j],dp[i - zero][j - one] +
1);
                }
            }
        }
        return dp[m][n];
    }
```