题目：

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`.

Example 1:

```
Input: Array = {"10", "0001", "111001", "1", "0"}, m = 5, n = 3
Output: 4


Explanation: This are totally 4 strings can be formed by the using of 5 0s a
nd 3 1s, which are “10,”0001”,”1”,”0”
```

Example 2:

```
Input: Array = {"10", "0", "1"}, m = 1, n = 1
Output: 2


Explanation: You could form "10", but then you'd have nothing left. Better f
orm "0" and "1".
```

1.时间：O（ ）；空间：O（ ）

```
class Solution {
    /* dp[i][j][k]：选择第 i 个字符串时，'0'的数目为 j，'1'的数目为 k 时所能构成的字符串的最大数量；
        dp[i][j][k] = max(dp[i-1][j][k], dp[i-1][j-zeros][k-ones]), 0 <= i < strs.size(), j >= zeros(strs[i]), k >= ones(strs[i]) */
public:
    /* 以下是经过优化的 dp，通过滚动数组，将 3 维 dp 将为 2 维 dp */
    int findMaxForm(vector<string>& strs, int m, int n) {
        if (strs.empty()) return 0;
        std::vector<std::vector<int>> dp(m + 1, std::vector<int>(n + 1, 0));
        for (int i = 0; i < strs.size(); ++i){
            int ones = 0, zeros = 0;
            calcOneAndZero(strs[i], ones, zeros);
            for (int j = m; j >= ones; --j){
                for (int k = n; k >= zeros; --k){
                    dp[j][k] = std::max(dp[j][k], dp[j - ones][k - zeros] + 1);
                }
            }
        }
        return dp[m][n];
    }
```

```
private:
    void calcOneAndZero(const std::string& str, int& ones, int &zeros){
        for (int i = 0; i < str.size(); ++i){
            str[i] == '0' ? ++ones : ++zeros;
        }
    }
};
```