花盆算法实验报告

描述

- 你想把你的花店的橱窗布置成最悦目的样子，你有F束花，每一束都是不同的种类，而且至少有多少个花瓶被订购。你有F束花，每束花都是不同的种类，至少有同样多的花瓶排列成一排。花瓶被粘在架子上，从左到右连续编号为1到V，其中V是花瓶的数量，所以花瓶1是最左边的，花瓶V是最右边的花瓶。这些花束是可以移动的，并且由1和F之间的整数来唯一标识，这些id-numbers具有重要意义。它们决定了花束在这排花瓶中的出场顺序，所以当i<j时，花束i必须在装有花束j的花瓶的左边。 例如，假设你有一束杜鹃花(id -number=1)，一束海棠花(id -number=2)和一束康乃馨(id -number=3)。
- 现在，所有的花束都必须按序放入花瓶中。杜鹃花必须放在海棠花左边的花瓶中，而海棠花必须放在康乃馨左边的花瓶中。如果花瓶比花束多，那么多出来的花就会空着。一个花瓶只能装一束花。每个花瓶都有一个明显的特点（就像鲜花一样）。因此，把一束花放在一个花瓶里，就会产生一定的审美价值，用一个整数来表示。审美值用表格表示，如下图所示。让花瓶空着，审美值为0。

		V	**A**	**S**	**E**	**S**
		1	**2**	**3**	**4**	**5**
Bunches	**1 (azaleas)**	7	23	-5	-24	16
	2 (begonias)	5	21	-4	10	23
	3 (carnations)	-21	5	-4	-20	20

- 根据表格，例如杜鹃花，放在2号花瓶里会很好看，但放在4号花瓶里就会很难看。
- 為了達到最愉快的效果，你必須最大限度地提高安排的美學價值的總和，同時保持所需的花的排序。如果有超過一個安排具有最大的和值，任何一個都可以接受。你必須准确地制作一个插花。
- 第一行包含两个数字。`F, V`
  下面的 F 行。每一行都包含 V 个整数，所以`A[i][j]`是输入文件`(i+1)st`行的第 j 个数字。
  `1 <= F <= 100`，其中 F 是花束的数量。花束的编号为 1 到 F。
  `F <= V <= 100`，其中 V 是花瓶的数量。
  `-50 <= A[i][j] <= 50` 其中`A[i][j]`是将花束`i`放入花瓶`j`中得到的审美价值。

分析

- 我们设`dp[i][j]`表示处理到第i朵花，然后把第`i`朵花放到第`j`个花瓶里时所能获得的最大价值；用`dp[i][j]`表示第 i 束花插入第 j 个花瓶的最大价值，把第 i 束花插入第 j 个花瓶，那么前 i 束花取得的最大价值就是前 i-1 束花取得的最大价值加上第 i 束花插入花瓶 j 的价值。`dp[i][j] = max(dp[i - 1][k]) + val[i][j]`，其中`k< j val[i][j]`表示把第`i`朵花放到第`j`个花瓶里产生的价值。初始化问题，第一束花插入任意一个瓶子的价值就是瓶子的价值，其后的价值全部初始化为无穷小。

最优子结构证明

- 如果把 i 束花插到 j 个瓶的最优解是$p_1, p_2, p_3, \ldots, p_i$（$p_i$表示第 i 束花的摆放位置），其子问题：去掉某束花j以及装有该花的花瓶，求其最大美术价值。
- 假设花束摆放问题不满足最优子结构，那么其子问题的最优解为$p'_1, p'_2, \ldots, p'_{j-1}, p'_{j+1}, \ldots, p'_i$，那么把该束花连带花瓶放回去，则其最优解应该为$p'_1, p'_2, \ldots, p'_{j-1}, p_j, p'_{j+1}, \ldots, p'_i$，与假设矛盾，

所以该问题具有最优子结构。

复杂度分析

- 需要遍历可能的花种类、数量在花盆中摆放位置具有的美学价值，并且选出价值最高的方案。需要三重循环，设花朵种类为N,花盆数量为M时间复杂度应为$O(N \times M^2)$。

代码

```
//
// Created by Ridd on 2020/11/23.
//
#include <iostream>
#include <algorithm>

using namespace std;
const int maxn = 100 + 5;
const int INF = 0x3f3f3f3f;
int a[maxn][maxn];
int dp[maxn][maxn];
int n, m;

int main() {
    while (cin >> n >> m) {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++)
                cin >> a[i][j], dp[i][j] = -INF;
        }
        for (int i = 0; i <= m; i++)
            dp[0][i] = 0;
        dp[1][1] = a[1][1];
        for (int i = 1; i <= n; i++) {
            for (int j = i; j <= m; j++) {
                for (int k = 1; k < j; k++) {
                    dp[i][j] = max(dp[i][j], dp[i - 1][k] + a[i][j]);
                }
            }
        }
        int ans = -INF;
        for (int i = 1; i <= m; i++) {
            ans = max(ans, dp[n][i]);
        }
        cout << ans << endl;
    }
    return 0;
}
```

运行结果

```
G:\Courseware\Algorithm\AlgorithmFinal\Task6_Flower\cmake-build-debug\TSK6.exe
3 5
7 23 -5 -24 16
5 21 -4 10 23
-21 5 -4 -20 20

53
```