花束摆放问题解题报告

POJ 1157 **LITTLE SHOP OF FLOWERS**

Description

You want to arrange the window of your flower shop in a most pleasant way. You have F bunches of flowers, each being of a different kind, and at least as many vases ordered in a row. The vases are glued onto the shelf and are numbered consecutively 1 through V, where V is the number of vases, from left to right so that the vase 1 is the leftmost, and the vase V is the rightmost vase. The bunches are moveable and are uniquely identified by integers between 1 and F. These id-numbers have a significance: They determine the required order of appearance of the flower bunches in the row of vases so that the bunch i must be in a vase to the left of the vase containing bunch j whenever i < j. Suppose, for example, you have bunch of azaleas (id-number=1), a bunch of begonias (id-number=2) and a bunch of carnations (id-number=3). Now, all the bunches must be put into the vases keeping their id-numbers in order. The bunch of azaleas must be in a vase to the left of begonias, and the bunch of begonias must be in a vase to the left of carnations. If there are more vases than bunches of flowers then the excess will be left empty. A vase can hold only one bunch of flowers.

Each vase has a distinct characteristic (just like flowers do). Hence, putting a bunch of flowers in a vase results in a certain aesthetic value, expressed by an integer. The aesthetic values are presented in a table as shown below. Leaving a vase empty has an aesthetic value of 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

According to the table, azaleas, for example, would look great in vase 2, but they would look awful in vase 4.

To achieve the most pleasant effect you have to maximize the sum of aesthetic values for the arrangement while keeping the required ordering of the flowers. If more than one arrangement has the maximal sum value, any one of them will be acceptable. You have to produce exactly one arrangement.

Input

- The first line contains two numbers: F, V.

- The following F lines: Each of these lines contains V integers, so that A_{ij} is given as the j^{th} number on the $(i+1)^{st}$ line of the input file.
- $1 \leq F \leq 100$ where F is the number of the bunches of flowers. The bunches are numbered 1 through F.
- $F \leq V \leq 100$ where V is the number of vases.
- $-50 \leq A_{ij} \leq 50$ where A_{ij} is the aesthetic value obtained by putting the flower bunch i into the vase j.

Output

The first line will contain the sum of aesthetic values for your arrangement.

Sample Input

```
3 5
7 23 -5 -24 16
5 21 -4 10 23
-21 5 -4 -20 20
```

Sample Output

```
53
```

Source

IOI 1999

思路

题目中限定了编号较小的花永远在编号相对较大的花的左边，这是个很好的性质，我们可以利用这个性质来设计无后效性的状态，从而得出一个正确的状态转移方程，解决我们的问题。

由于是一个花和花瓶的匹配问题，只有两种不同物件，我们很直观的想到：可以设计一个二维的状态来保存某个状态的答案。

状态设计

我们设dp[i][j]为前i种花（包含）插到前j个瓶子（包含）中的最优答案。

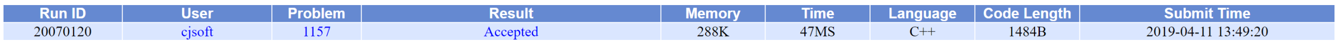
$$dp[i][j] = \begin{cases} a[1][1] & \text{i=1, j=1} \\ \max\{a[1][j], dp[1][j-1]\} & \text{k>1} \\ \max\{dp[i-1][j-1] + a[i][j], dp[i][j-1]\} & \text{i>1, j>i} \\ dp[i-1][i-1] + a[i][i] & \text{i=i} \end{cases}$$

证明

最优子结构证明：若dp[i][j]不是由从dp[i-1][j-1]或dp[i][j-1]转移过来。则无非两种情况：第一种是最后一个瓶子不插花，就用他的转移来源更新dp[i][j-1]，那么就违反了dp[i][j-1]是其代表状态的最优解的前提。第二种情况是最后一个瓶子插花，假设他不是由dp[i-1][j-1]转移过来，那么一定是从dp[i-1][k]（其中k小于j）转移过来，他比dp[i-1][j-1]更优，并且可以更新它，与假设矛盾。

测试

由于这道题是一个有着将近20年历史的老题了，POJ上就有成题，所以我就直接放到POJ上来测试算法正确性了。



Run ID	User	Problem	Result	Memory	Time	Language	Code Length	Submit Time
20070120	cjsoft	1157	Accepted	288K	47MS	C++	1484B	2019-04-11 13:49:20

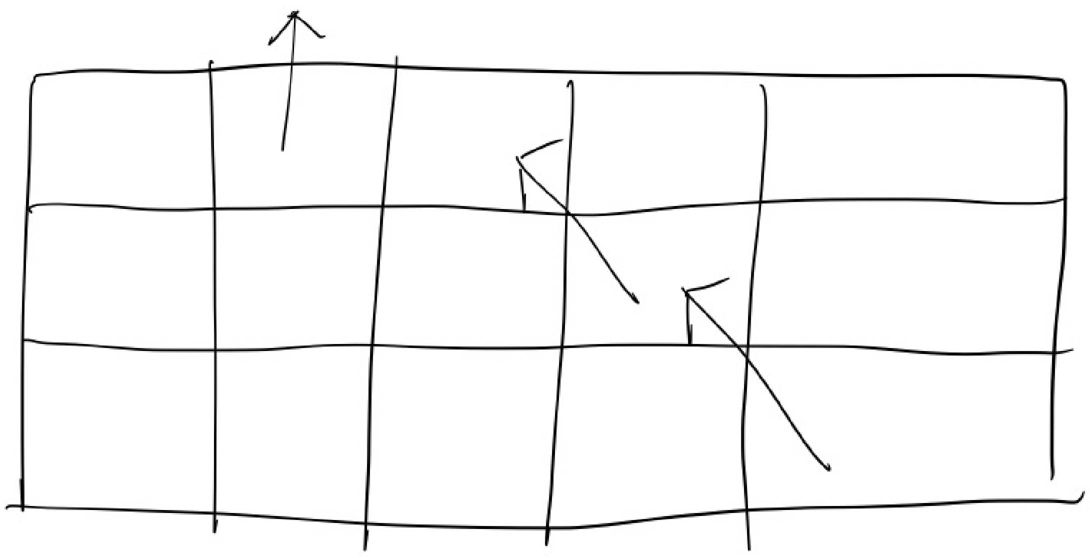
https://paste.ubuntu.com/p/SdbgrMm52T/

以上这个是提交的代码

作业代码为`dp.cpp`

作业中包含的代码文件进行了适当的代码拆分，并且支持输出方案。 输出方案的思路是新建一个revfinder数组，记录下他是从哪个(i, j)转移过来的，很显然，如果revfinder[i][j].first + 1 = i且revfinder[i][j].second + 1 = j，则说明第i支花应该插入到第j个花瓶当中。

示例



以下是对于样例输入的增强输出，其中中间的矩阵是最后的dp数组内容

```
53
   7   23   23   23   23
   0   28   28   33   46
   0    0   24   24   53
Put flower 3 in vase 5.
Put flower 2 in vase 4.
Put flower 1 in vase 2.
```

复杂度分析

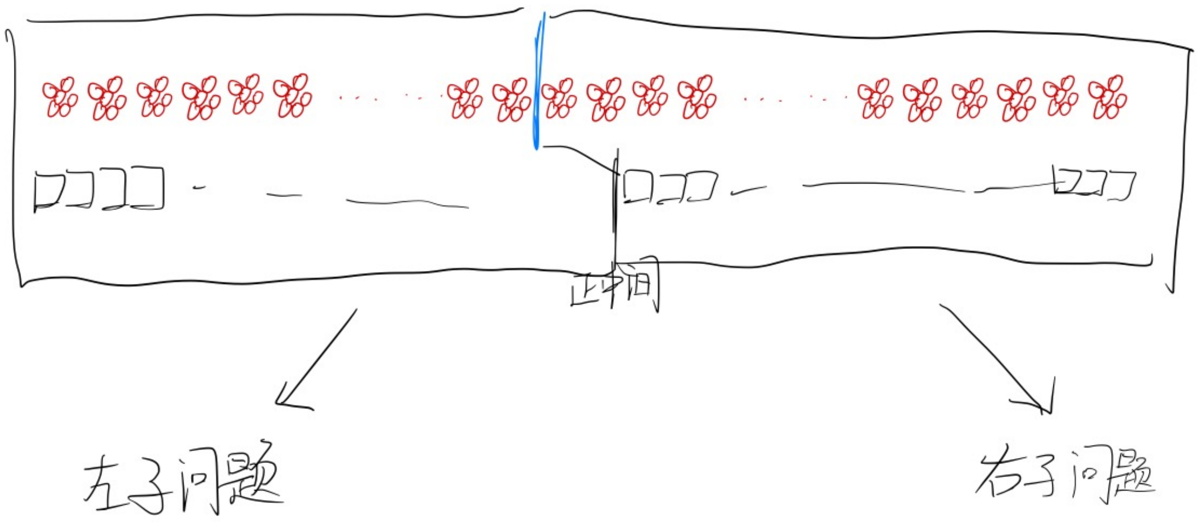
核心代码是一个二重循环，复杂度显而易见，是$O(FV)$，具体的循环次数约等于$\frac{FV}{2}$。

与分治法的对比

分治法算法设计

如果花的个数和花瓶个数相等，则方案确定且唯一。

我们每次将花瓶平均分成两半，然后枚举分配方案，例如共有n朵花，枚举分界点，分界点左边的花放在左边一半的花瓶，右边的在右边一半的花瓶。由于花的数量不超过花瓶的数量，所以我们可以以此为判断依据做剪枝。



示意图

复杂度分析

$$
\begin{aligned}
T(m,n) &= \sum_{k=0}^{m} T(k,\frac{n}{2}) + T(m-k,\frac{n}{2}) \\
&= 2\sum_{k=0}^{m} T(k,\frac{n}{2})
\end{aligned}
$$

然后不会推了，复杂度估算（猜的）是$O(F^{\log V})$，总之就是很糟糕了，比平方的算法差很多。

POJ上TLE（这不显然的嘛）

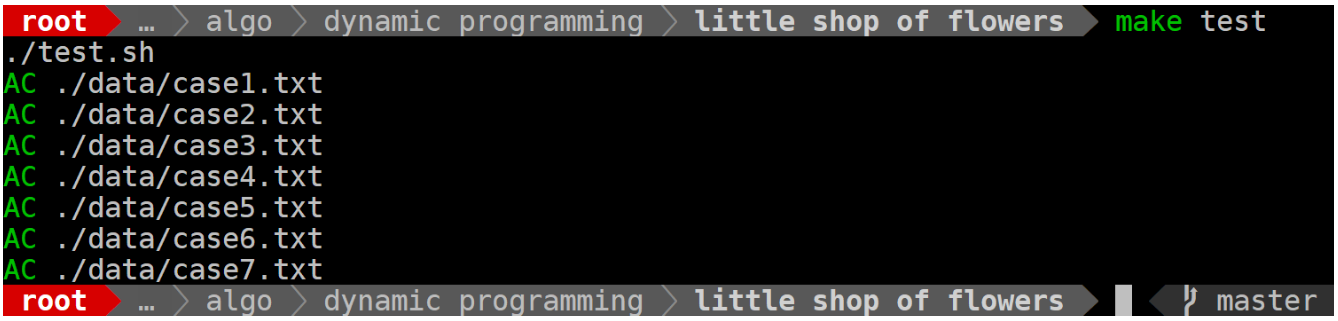
测试工具集

由于分治法复杂度比较高，POJ上过不了，所以我以dp作为std代码，编写了一套简单的数据生成及测试脚本（工作于Linux，需要g++、python2）

功能说明

```
make            # 执行编译、数据生成、使用两种算法跑数据，并比较答案是否相同
make dp         # 仅编译dp做法程序
make dnc        # 仅编译分治做法程序
make testcase   # 生成7组测试数据集
make test       # 使用两种不同算法跑数据，比较答案是否相同
```

测试结果



```
root  …  algo  dynamic programming  little shop of flowers  make test
./test.sh
AC ./data/case1.txt
AC ./data/case2.txt
AC ./data/case3.txt
AC ./data/case4.txt
AC ./data/case5.txt
AC ./data/case6.txt
AC ./data/case7.txt
root  …  algo  dynamic programming  little shop of flowers     master
```

附录

动态规划代码

```
#include <bits/stdc++.h>
#include "input.h"
using namespace std;
typedef pair<int, int> PII;
int arr[MXN][MXN];
int dp[MXN][MXN];
PII revfinder[MXN][MXN];
int f, v;

void work() {
    dp[1][1] = arr[1][1];
    for (int i = 2; i <= v; ++i) { // 初始化边界条件
        if (dp[1][i - 1] < arr[1][i]) {
            dp[1][i] = arr[1][i];
            revfinder[1][i] = make_pair(0, i - 1);
        } else {
            dp[1][i] = dp[1][i - 1];
            revfinder[1][i] = make_pair(1, i - 1);
        }
    }
    for (int j = 2; j <= v; ++j) {  // 状态转移，记下转移来源
        for (int i = 2; i <= j; ++i) {
            if (checkmax(dp[i][j], dp[i - 1][j - 1] + arr[i][j])) {
                revfinder[i][j] = make_pair(i - 1, j - 1);
            }
            if (checkmax(dp[i][j], dp[i][j - 1])) {
                revfinder[i][j] = make_pair(i, j - 1);
            }
        }
    }
}
void output() { // 方案输出
    PII tmp = make_pair(f, v);
    while (tmp.first && tmp.second) {
        PII rtmp = revfinder[tmp.first][tmp.second];
        if (rtmp.first == tmp.first - 1 && rtmp.second == tmp.second - 1) {
            printf("Put flower %d in vase %d.\n", tmp.first, tmp.second);
        }
        tmp = rtmp;
    }
}
int main() {
    input(arr, f, v);                // 读入数据
    work();
    printf("%d\n", dp[f][v]);        // 结果
    for (int i = 1; i <= f; ++i) {   // 输出最后的dp数组
        for (int j = 1; j <= v; ++j) {
            printf("%4d ", dp[i][j]);
        }
        putchar('\n');
    }
    output();                        // 输出方案
}
```

分治法代码

```
#include "input.h"
#include <bits/stdc++.h>
#define IMIN INT_MIN
using namespace std;

int arr[MXN][MXN], f, v;

int dnc(int fl, int fr, int vl, int vr) {
    if (fl > fr) return 0;
    if (fr - fl > vr - vl || vr < vl) {
        return IMIN;
    }
    if (fr - fl == vr - vl) {
        int rtn = 0;
        for (int i = fl, j = vl; i <= fr; ++i, ++j)
            rtn += arr[i][j];
        return rtn;
    }
    int m = (vl + vr) / 2;
    int rtn = IMIN;
    for (int i = 0; i <= fr - fl + 1; ++i) {
        int tl = dnc(fl, fl + i - 1, vl, m);
        if (tl == IMIN) continue;
        int tr = dnc(fl + i, fr, m + 1, vr);
        if (tr == IMIN) continue;
        checkmax(rtn, tl + tr);
    }
    return rtn;
}

int main() {
    input(arr, f, v);
    printf("%d\n", dnc(1, f, 1, v));
}
```

读入库代码

```
/*input.h*/
#ifndef INPUT_H
#define INPUT_H
const int MXN = 107;
void input(int[MXN][MXN], int&, int &);
template <typename T>
bool checkmax(T &a, T b);
template <typename T>
bool checkmin(T &a, T b);
template <typename T>
bool checkmax(T &a, T b) {
    if (b > a) {
        a = b;
        return true;
    }
    return false;
}

template <typename T>
bool checkmin(T &a, T b) {
    if (b < a) {
        a = b;
        return true;
    }
    return false;
}
#endif


/*input.cpp*/
#include "input.h"
#include <bits/stdc++.h>
using namespace std;
void input(int arr[MXN][MXN], int &f, int &v) {
    scanf("%d %d", &f, &v);
    for (int i = 1; i <= f; ++i) {
        for (int j = 1; j <= v; ++j) {
            scanf("%d", &arr[i][j]);
        }
    }
}
```