1279：橱窗布置(flower)

题目描述

假设以最美观的方式布置花店的橱窗，有F束花，每束花的品种都不一样，同时，至少有同样数量的花瓶，被按顺序摆成一行，花瓶的位置是固定的，并从左到右，从1到V顺序编号，V是花瓶的数目，编号为1的花瓶在最左边，编号为V的花瓶在最右边，花束可以移动，并且每束花用1到F的整数惟一标识，标识花束的整数决定了花束在花瓶中列的顺序即如果i<j，则花束i必须放在花束j左边的花瓶中。

例如，假设杜鹃花的标识数为1，秋海棠的标识数为2，康乃馨的标识数为3，所有的花束在放入花瓶时必须保持其标识数的顺序，即：杜鹃花必须放在秋海棠左边的花瓶中，秋海棠必须放在康乃馨左边的花瓶中。如果花瓶的数目大于花束的数目，则多余的花瓶必须空，即每个花瓶中只能放一束花。

每一个花瓶的形状和颜色也不相同，因此，当各个花瓶中放入不同的花束时会产生不同的美学效果，并以美学值(一个整数)来表示，空置花瓶的美学值为0。在上述例子中，花瓶与花束的不同搭配所具有的美学值，可以用如下表格表示。根据表格，杜鹃花放在花瓶2中，会显得非常好看，但若放在花瓶4中则显得很难看。

为取得最佳美学效果，必须在保持花束顺序的前提下，使花的摆放取得最大的美学值，如果具有最大美学值的摆放方式不止一种，则输出任何一种方案即可。题中数据满足下面条件：1≤F≤100，F≤V≤100，-50≤Aij≤50，其中Aij是花束i摆放在花瓶j中的美学值。输入整数F，V和矩阵(Aij)，输出最大美学值和每束花摆放在各个花瓶中的花瓶编号。

	花瓶1	花瓶2	花瓶3	花瓶4	花瓶5
杜鹃花	7	23	-5	-24	16
秋海棠	5	21	-4	10	23
康乃馨	-21	5	-4	-20	20

假设条件:

1≤F≤100，其中 F 为花束的数量，花束编号从 1 至 F。

F≤V≤100，其中 V是花瓶的数量。

-50≤Aij≤50，其中 Aij 是花束 i 在花瓶 j 中的美学值。

输入

第一行包含两个数：F，V。

随后的F行中，每行包含V个整数，Aij 即为输入文件中第（i+1）行中的第j个数。

输出

第一行是程序所产生摆放方式的美学值。

第二行必须用F个数表示摆放方式，即该行的第K个数表示花束K所在的花瓶的编号。

输入样例

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

输出样例

```
53
2 4 5
```

解析

对于这类问题，我们的第一反应应该是缩减规模。我无法直接计算3朵花5个瓶子，但是可以轻松的计算1朵花，5个瓶子。我们首先记录原始表：

	花瓶1	花瓶2	花瓶3	花瓶4	花瓶5
杜鹃花	7	23	-5	-24	16
秋海棠	5	21	-4	10	23
康乃馨	-21	5	-4	-20	20

然后分别计算不同的花在不同的瓶子中的最大值，首先是杜鹃花自己。

	花瓶1	花瓶2	花瓶3	花瓶4	花瓶5
杜鹃花	7	23	-5	-24	16

然后是杜鹃和海棠：

	花瓶1	花瓶2	花瓶3	花瓶4	花瓶5
杜鹃花	7	23	-5	-24	16
秋海棠	7	28	19	33	46

最后加入第三朵花：

	花瓶1	花瓶2	花瓶3	花瓶4	花瓶5
杜鹃花	7	23	-5	-24	16
秋海棠	7	28	19	33	46
康乃馨	7	12	24	-8	53

定义dp[i][j]表示前i朵花装载j个瓶子中的最大美学值，那么可以得到：

dp[i][j] = max(dp[i-1][j-1]+value[i][j],dp[i-1][j-1]+value[i][j],……,dp[i-1][1]+value[i][j])

编码

```
#include<bits/stdc++.h>

using namespace std;

int f, v;  //f朵花，v个瓶子
int baseValue[101][101];   //每朵花在不同瓶子的美学值
int dp[101][101];   //每朵花在不同瓶子的最大美学值
int maxV;  //全局最大的美学值
int minV = -0x3f3f3f3f;

//逆序打印路径
void print(int h) {
    //不存在花则直接返回
    if (h == 0) return;
    //从第一个花瓶开始尝试摆放
    int i = 1;
    //找到当前花装在第几个瓶子时取到最大值
    while (dp[h][i] != maxV) {
        i++;
    }
    //减掉当前花的美学值
    maxV -= baseValue[h][i];
    //继续找前一朵花
    print(h - 1);
    //打印当前瓶子的编号
    cout << i << " ";
}


int main() {
    cin >> f >> v;
    //将全部dp初始化为负数
    memset(dp, minV, sizeof(dp));
    //读入每朵花在不同瓶子中的美学值
    for (int i = 1; i <= f; ++i) {
        for (int j = 1; j <= v; ++j) {
            cin >> baseValue[i][j];
        }
    }
    //保证第一朵花在不同瓶子中的计算都是正确的
    for (int i = 0; i <= v; ++i) {
```

```
            dp[0][i] = 0;
        }
        //遍历每一种花
        for (int i = 1; i <= f; ++i) {
            //遍历花装在不同瓶子的情况下
            for (int j = i; j <= v; ++j) {
                //遍历前面花的组合
                for (int k = 1; k <= j; ++k) {
                    int value = dp[i - 1][k - 1] + baseValue[i][j];
                    dp[i][j] = max(value, dp[i][j]);
                }
            }
        }
        //输出最大值
        cout << maxV << endl;
        //递归打印每个花瓶的组合情况
        print(f);
}
```