花束摆放实验报告

2018211302班 2018210074 熊宇

指导教师：王晓茹

一、实验内容

现在有F束不同品种的花束，同时有至少同样数量的花瓶被按顺序摆成一行，其位置固定于架子上，并从1至V按从左到右顺序编号，V是花瓶的数目（F<=V）。花束可以移动，并且每束花用1至F的整数唯一标识。标识花束的整数决定了花束在花瓶中排列的顺序，如果i<j，花束i必须放在花束j左边的花瓶中。每个花瓶只能放一束花。如果花瓶的数目大于花束的数目，则多余的花瓶空置。

每一个花瓶都具有各自的特点。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，并以一美学值（一个整数）来表示，空置花瓶的美学值为零。为取得最佳美学效果，必须在保证花束顺序的前提下，使花束的摆放取得最大的美学值。请求出具有最大美学值的一种摆放方式。

1.设计动态规划算法，描述最优子结构，在报告中写出设计思路；
2.编程序实现上述算法，并完成测试；

二、实验分析

1.最优子结构

审美值储存在value[] []中，最优子结构为

```
res[i][j] = MAX{ B[i-1][k], k<=j-1} + value[i][j]
```

也就是i个花束、j个花瓶的最大美学值摆放方案，等于i-1个花束在i-1到j-1个花瓶中的最大美学值摆放方案附加一个第i束花摆在第j个花瓶。

2.算法设计

(1)最优子结构问题

a.设计思路

审美值储存在value[] []中，最优子结构为

```
res[i][j] = MAX{ B[i-1][k], k<=j-1} + value[i][j]
```

也就是i个花束、j个花瓶的最大美学值摆放方案，等于i-1个花束在i-1到j-1个花瓶中的最大美学值摆放方案附加一个第i束花摆在第j个花瓶。

b.代码实现

```
int k = V - F;
        for (int i = 1; i <= F; i++)
        {
            for (int j = i; j <= V && j <= i + k; j++)
            {
                if (res[i - 1][j - 1] + value[i][j] > res[i][j -
1])
                {
                    res[i][j] = res[i - 1][j - 1] + value[i][j];
                }
                else
                {
                    res[i][j] = res[i][j - 1];
                }
            }
        }
```

(2)摆放方案存储问题

a.设计思路

花朵摆放方案存储在一个vector数组，每一个vector对应存储着每一束花在算法执行过程中将其摆放过的花瓶编号记录，每次i-1束花、j-1个花瓶附加第i束花摆放在第j号花瓶是最大美学值摆放方案时，就将j-1压入i-1的vector、将j压入i的vector；否则先将j压入i的vector，然后将从第1束到第i束花vector中顶端元素弹出。

算法结束后，每束花的vector的顶端花瓶号就是最大美学值摆放方案中她所摆放的位置。

b.代码实现

```
int k = V - F;
        for (int i = 1; i <= F; i++)
        {
            for (int j = i; j <= V && j <= i + k; j++)
            {
                if (res[i - 1][j - 1] + value[i][j] > res[i][j -
1])
                {
                    place[i].push_back(j);
                    place[i - 1].push_back(j - 1);
                }
                else
                {
                    place[i].push_back(j);
                    for (int w = 1; w <= i; w++)
                        place[w].pop_back();
                }
            }
        }
        cout << "实现此最大美学值的花束摆放方案是：" << endl;
        for (int i = 1; i <= F; i++)
        {
            int temp = place[i][place[i].size() - 1];

            cout << "第" << i << "束花束放在第" << temp << "个花瓶，美
学值为：" << value[i][temp] << endl;
        }
```

3.复杂度分析

很明显，该动态规划算法算法时间复杂度为O(F*V)

三、实验源代码

```
#include <iostream>
#include <algorithm>
#include <stdio.h>
#include <stdlib.h>
#include <iomanip>
#include <math.h>
#include <vector>

#define MaxNumOfBottle 200 //最大花瓶数目
#define MaxNUmOfFlower 100 //最大花束数目

using namespace std;

int main()
{
    cout << "No problem" << endl;
    cout<<endl;
    while (1)
    {

 cout<<"**********************************************************
*******************************"<<endl;
        int F, V; //F为花束数目，V为花瓶数目
        cout << "请输入花束数目和花瓶数目，以空格间隔：" << endl;
        cin >> F >> V;
        cout << "请输入每束花束在每个花瓶里产生的美学值：" << endl;
        int value[MaxNUmOfFlower][MaxNumOfBottle] = {0}; //每一个花
束放在每一个花瓶所产生的美学值
        int res[MaxNUmOfFlower][MaxNumOfBottle];          //存结果
        vector<int> place[MaxNUmOfFlower];                //每朵花的放
置情况
        for (int i = 1; i <= F; i++)
        {
            for (int j = 1; j <= V; j++)
                res[i][j] = INT_MIN;
        }
        for (int i = 1; i <= F; i++)
        {
            for (int j = 1; j <= V; j++)
                cin >> value[i][j];
        }
        /*
    cout << "每束花束在每个花瓶里产生的美学值是：" << endl;
    for (int i = 1; i <= F; i++)
    {
        for (int j = 1; j <= V; j++)
            cout << value[i][j] << " ";
        cout << endl;
```

```
    }
    */
        /*
    cout<<"计算前的结果数组为："<<endl;
    for(int i=1;i<=F;i++)
    {
        for(int j=1;j<=V;j++)
            cout<<res[i][j]<<" ";
        cout<<endl;
    }
    */
        int k = V - F;
        for (int i = 1; i <= F; i++)
        {
            for (int j = i; j <= V && j <= i + k; j++)
            {
                //place[i].push_back(j);
                if (res[i - 1][j - 1] + value[i][j] > res[i][j -
1])
                {
                    //cout<<res[i - 1][j - 1] + value[i][j] <<" "<<
res[i][j - 1]<<endl;
                    res[i][j] = res[i - 1][j - 1] + value[i][j];
                    place[i].push_back(j);
                    place[i - 1].push_back(j - 1);
                }
                else
                {
                    //cout<<res[i - 1][j - 1] + value[i][j] <<" "<<
res[i][j - 1]<<endl;
                    res[i][j] = res[i][j - 1];
                    place[i].push_back(j);
                    for (int w = 1; w <= i; w++)
                        place[w].pop_back();
                }
                /*
            cout<<"我们看看place"<<i<<endl;
            for(int ww=0;ww<place[i].size();ww++)
                cout<<place[i][ww]<<" ";
            cout<<endl<<endl;
            */
            }
        }
        /*
    cout<<"计算后的结果数组为："<<endl;
    for(int i=1;i<=F;i++)
    {
        for(int j=1;j<=V;j++)
            cout<<res[i][j]<<" ";
        cout<<endl;
    }
    */
```

```
        /*
    for(int i=1;i<=F;i++)
    {
        cout<<"我们看看place"<<i<<endl;
        for(int ww=0;ww<place[i].size();ww++)
                cout<<place[i][ww]<<" ";
            cout<<endl<<endl;
    }
    */

        cout << "能产生的最大美学值是：" << res[F][V] << endl;
        cout << "实现此最大美学值的花束摆放方案是：" << endl;
        for (int i = 1; i <= F; i++)
        {
            int temp = place[i][place[i].size() - 1];

            cout << "第" << i << "束花束放在第" << temp << "个花瓶，美
学值为：" << value[i][temp] << endl;
        }

 cout<<"**********************************************************
*******************************"<<endl;
    }
    system("pause");
    return 0;
}

/*
3 5

7 23 -5 -24 16
5 21 -4 10 23
-21 5 -4 -20 20
*/
```

四、实验结果

1.输入输出样例

```
No problem

************************************************************************
**********************
请输入花束数目和花瓶数目，以空格间隔：
3 5
请输入每束花束在每个花瓶里产生的美学值：
7 23 -5 -24 16
5 21 -4 10 23
-21 5 -4 -20 20
能产生的最大美学值是：53
实现此最大美学值的花束摆放方案是：
第1束花束放在第2个花瓶，美学值为：23
第2束花束放在第4个花瓶，美学值为：10
第3束花束放在第5个花瓶，美学值为：20
************************************************************************
**********************
************************************************************************
**********************
请输入花束数目和花瓶数目，以空格间隔：
1 2
请输入每束花束在每个花瓶里产生的美学值：
5 1
能产生的最大美学值是：5
实现此最大美学值的花束摆放方案是：
第1束花束放在第1个花瓶，美学值为：5
************************************************************************
**********************
************************************************************************
**********************
请输入花束数目和花瓶数目，以空格间隔：
4 4
请输入每束花束在每个花瓶里产生的美学值：
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
能产生的最大美学值是：4
实现此最大美学值的花束摆放方案是：
第1束花束放在第1个花瓶，美学值为：1
第2束花束放在第2个花瓶，美学值为：1
第3束花束放在第3个花瓶，美学值为：1
第4束花束放在第4个花瓶，美学值为：1
************************************************************************
**********************
```

2.算法正确性分析

测试了3束花、5个花瓶的情况，结果正确无误，满足序号不降条件且达到最大美学值；

测试了1束花、2个花瓶的情况，结果正确无误，满足序号不降条件且达到最大美学值；

测试了4束花、4个花瓶的情况，结果正确无误，满足序号不降条件且达到最大美学值。