花束摆放问题

一、问题描述

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 $(F \leq V)$ 。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，必须在保持花束顺序的前提下，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。

二、实验要求

- 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；
- 编程序实现上述算法，并完成测试；
- 分析算法的时间复杂度

三、实验原理

用 $N[i][j]$ 表示第 i 束花放到第 j 个瓶子中的美学值，用$(x_1, x_2, \ldots, x_F)$表示最大美学值总和下的摆放方式，即第 i 束花被摆放到 x_i 花瓶中，$1 \leq x_i \leq V$ 且序列中各元素互不相等。

证明最优子结构

用反证法证明：

设$(x_1, x_2, \ldots, x_F)$是所给花束摆放问题的一个最优解，则$(x_1, x_2, \ldots, x_{F-1})$是下面一个子问题的最优解：

> *$F-1$ 束花被摆放到$V-1$ 个花瓶中，求最大美学值总和*

如若不然，设$(y_1, y_2, \ldots, y_{F-1})$是上述子问题的一个最优解，及

$$\sum_{i=1}^{F-1} N[i][x_i] < \sum_{i=1}^{F-1} N[i][y_i]$$

$$\sum_{i=1}^{F} N[i][x_i] = \sum_{i=1}^{F-1} N[i][x_i] + N[F][x_F] < \sum_{i=1}^{F-1} N[i][y_i] + N[F][x_F]$$

得到$(x_1, x_2, \ldots, x_F)$ 不 是所给花束摆放问题的一个最优解，与题设矛盾。

因此，该问题满足最优子结构。

求解递推式

用 $M[i][j]$ 表示将前 i 束花插到前 j 个花瓶中能够产生的最大美学值总和。

考虑两种情况：

- 当前第 i 束花摆放到第 j 个花瓶中，则

  $M[i][j] = M[i-1][j-1] + N[i][j]$
- 当前第 i 束花不放到第 j 个花瓶中，则

  $M[i][j] = M[i][j-1]$

综合以上两种情况，可以递推式
$M[i][j] = Max\{M[i][j-1], M[i-1][j-1] + N[i][j]\}$

特别的，当$i=0$或$j=0$，$M[i][j]=0$

记录解向量

要记录最大美学值的解向量$X=(x_1,x_2,\ldots,x_F)$。我们可以考虑从$M[F][V]$出发，倒退求解向量：

- 初始$i=F,j=V$
- 若$M[i][j]>M[i-1][j]$，则花束 i 被摆放在 花瓶 j，用于更新$M[i][j]$，$i=i-1,j=j-1$
- 若$M[i][j]=M[i-1][j]$，则花束 i **没有** 被摆放在花瓶 j，$j=j-1,i$ 保持不变

四、算法实现

DP初始化

```
/*DP初始化*/
Algorithm1::Algorithm1(int f, int v, int** n)
{
    F = f;
    V = v;
    N = n;
    M = new int* [F + 1];

    for (int i = 0; i <= F; ++i)
        M[i] = new int[V + 1];
    for (int i = 0; i <= F; ++i)
        for (int j = 0; j <= V; ++j)
            M[i][j] = 0;

    Result = new int [F + 1];
```

```
    for (int i = 1; i <= F; ++i)
        Result[i] = 0;
}
```

DP计算

```
/*DP计算*/
int Algorithm1::Run()
{

    /*动态规划过程*/
    for (int i = 1; i <= F; ++i) //遍历花束
        for (int j = i; j <= V; ++j)    //遍历花瓶，第i束花必须摆放在第i个花瓶之后
            M[i][j] = gmax(M[i][j - 1], M[i - 1][j - 1] + N[i][j]); //抉择当前花瓶摆还是不摆

    /*倒推求解解向量过程*/
    for (int i = F, j = V; i >= 1; j--)
    {
        if (M[i][j] > M[i][j - 1])  //如果当前在花束下，前j个花瓶方案比前j-1个花瓶方案更优，则表明第j个花瓶被选中
        {
            Result[i] = j;  //记录解向量
            i --;   //继续考虑上一个花束
        }
    }

    return M[F][V];
}
```

时间统计

```
#pragma once
#ifdef _WIN32
#include <windows.h>
#else
#include <time.h>
#endif  // _WIND32

// 定义64位整形
#if defined(_WIN32) && !defined(CYGWIN)
typedef __int64 int64_t;
#else
typedef long long int64t;
#endif  // _WIN32

int64_t GetSysTimeMicros();

#include "ustime.h"

// 获取系统的当前时间，单位微秒(us)
int64_t GetSysTimeMicros()
{
#ifdef _WIN32
    // 从1601年1月1日0:0:0:000到1970年1月1日0:0:0:000的时间(单位
100ns)
#define EPOCHFILETIME   (116444736000000000UL)
    FILETIME ft;
    LARGE_INTEGER li;
    int64_t tt = 0;
    GetSystemTimeAsFileTime(&ft);
    li.LowPart = ft.dwLowDateTime;
    li.HighPart = ft.dwHighDateTime;
    // 从1970年1月1日0:0:0:000到现在的微秒数(UTC时间)
    tt = (li.QuadPart - EPOCHFILETIME) / 10;
    return tt;
#else
    timeval tv;
    gettimeofday(&tv, 0);
```

```
    return (int64_t)tv.tv_sec * 1000000 +
(int64_t)tv.tv_usec;
#endif // _WIN32
    return 0;
}
```

数据生成

```
DataCreate::DataCreate(int f, int v)
{
    F = f;
    V = v;
    data = new int* [F + 1];
    for (int i = 0; i <= F; ++i)
        data[i] = new int[V + 1];
}

int** DataCreate::GetData()
{
    srand(time(NULL));
    for (int i = 1; i <= F; i++)
        for (int j = 1; j <= V; j++)
        {
            data[i][j] = rand() % 100;
        }
    return data;
}
```

美学值表打印

```
void DataCreate::Print()
{
    cout << "美学值表: " << endl;
    cout << setw(8) << "花束\\花瓶";
    for (int i = 1; i <= V; ++i)
        cout << setw(5) << i;
    cout << endl;
    for (int i = 1; i <= F; ++i)
    {
        cout << setw(9) << i;
        for (int j = 1; j <= V; ++j)
            cout << setw(5) << data[i][j];
        cout << endl;
    }
}
```

DP结果打印

```
void Algorithm1::Print()
{
#ifdef DEBUG
    cout << "DP表: " << endl;
    cout << setw(8) << "花束\\花瓶";
    for (int i = 1; i <= V; ++i)
        cout << setw(5) << i;
    cout << endl;
    for (int i = 1; i <= F; ++i)
    {
        cout << setw(9) << i;
        for (int j = 1; j <= V; ++j)
            cout << setw(5) << M[i][j];
        cout << endl;
    }
```

```
#endif // DEBUG



    cout << "最大美学值为: " << M[F][V] << endl;
    cout << "摆花方案为: " << endl;
    for (int i = 1; i <= F; ++i)
        cout << "第 " << i << " 束花被摆放到第 " << Result[i]
<< " 个花瓶" << endl;
    cout << "运行时间: " << RunTime / 1000000.0 << " s" <<
endl;
}
```

测试程序

```
#include "DataCreate.h"
#include "ustime.h"
#include "Algorithm1.h"
#include <iostream>
using namespace std;
int main()
{
    while (1)
    {
        int F, V;
        cout << "请输入花束总数F: ";
        cin >> F;
        cout << "请输入花瓶总数V: ";
        cin >> V;
        DataCreate Data(F, V);
        int** N = Data.GetData();
        Data.Print();
        Algorithm1 Algo(F, V, N);
        Algo.Run();
        Algo.Print();
    }
```

```
}
```

五、算法分析

时间复杂度

在动态规划求解过程中，DP部分进行 $O(FV)$ 的二重循环

```
for (int i = 1; i <= F; ++i)
        for (int j = i; j <= V; ++j)
            M[i][j] = gmax(M[i][j - 1], M[i - 1][j - 1] +
N[i][j]);
```

在求解解向量的过程中，进行 $O(V)$ 的单重循环

故时间复杂度为$O(FV)$

空间复杂度

DP数组$M[F][V]$， 空间复杂度为$O(FV)$

六、算法测试

10束花，15个花瓶

```
请输入花束总数F: 10
```

请输入花瓶总数V: 15
美学值表:

花束\花瓶	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	30	60	24	81	11	77	20	96	32	83	78	80	63	2	75
2	29	25	37	54	35	80	23	55	37	50	77	61	87	90	48
3	45	75	64	74	31	90	77	99	55	96	0	22	44	29	45
4	67	7	33	38	87	40	19	37	74	36	61	86	68	11	67
5	37	84	36	25	5	37	69	41	94	25	81	66	25	63	23
6	49	51	27	69	46	83	5	44	3	78	94	29	27	75	66
7	5	22	43	39	97	54	11	40	25	53	65	18	78	62	64
8	27	46	73	7	19	39	79	79	36	98	25	56	56	9	31
9	17	62	63	16	3	78	91	0	25	12	72	78	25	60	19
10	88	92	5	57	63	82	71	63	63	28	27	75	19	91	88

DP表:

花束\花瓶	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	30	60	60	81	81	81	81	96	96	96	96	96	96	96	96
2	0	55	97	114	116	161	161	161	161	161	173	173	183	186	186
3	0	0	119	171	171	206	238	260	260	260	260	260	260	260	260
4	0	0	0	157	258	258	258	275	334	334	334	346	346	346	346
5	0	0	0	0	162	295	327	327	369	369	415	415	415	415	415

```
        6    0    0    0    0    0  245  300  371  371  447
463  463  463  490  490
        7    0    0    0    0    0    0  256  340  396  424
512  512  541  541  554
        8    0    0    0    0    0    0    0  335  376  494
494  568  568  568  572
        9    0    0    0    0    0    0    0    0  360  388
566  572  593  628  628
       10    0    0    0    0    0    0    0    0    0  388
415  641  641  684  716
最大美学值为: 716
摆花方案为:
第 1 束花被摆放到第 4 个花瓶
第 2 束花被摆放到第 6 个花瓶
第 3 束花被摆放到第 7 个花瓶
第 4 束花被摆放到第 8 个花瓶
第 5 束花被摆放到第 9 个花瓶
第 6 束花被摆放到第 10 个花瓶
第 7 束花被摆放到第 11 个花瓶
第 8 束花被摆放到第 12 个花瓶
第 9 束花被摆放到第 14 个花瓶
第 10 束花被摆放到第 15 个花瓶
运行时间: 0 s
```

20束花，30个花瓶

```
请输入花束总数F: 20
请输入花瓶总数V: 30
美学值表:
花束\花瓶     1    2    3    4    5    6    7    8    9   10
11   12   13   14   15   16   17   18   19   20   21   22
23   24   25   26   27   28   29   30
        1   87   95   20   93   24   63    7   29   62   69
  65   98   65   92   83   47   52    1    7    0   74   15
  96   68   72    1   21   46   35    9
```

2 97 75 67 95 98 91 92 44 34 98
16 28 10 21 45 70 33 25 8 86 86 24
82 55 38 89 70 31 7 88
3 96 99 14 6 0 43 72 22 6 94
27 17 89 54 48 12 5 66 90 91 9 11
65 96 4 61 99 16 64 30
4 87 57 69 72 89 65 35 91 6 17
94 57 16 82 89 49 96 48 61 54 97 1
48 53 68 68 5 91 34 25
5 89 12 21 4 94 1 3 71 43 44
83 17 29 5 70 77 17 67 21 88 78 1
40 66 79 6 6 33 96 71
6 62 11 45 67 21 85 56 20 40 65
29 84 9 34 94 46 29 12 36 42 45 89
63 10 69 22 52 11 76 59
7 1 20 6 62 61 9 82 1 86 30
52 10 87 81 6 55 18 21 33 54 6 24
42 22 0 15 5 30 79 29
8 30 53 89 30 91 60 92 25 58 24
25 99 62 63 11 81 51 44 7 64 93 25
79 12 73 97 21 51 40 42
9 98 98 68 64 71 84 17 70 22 90
83 97 91 28 64 3 21 86 8 2 47 12
27 41 36 54 18 94 35 50
10 2 97 11 76 82 16 3 72 80 55
96 94 18 45 50 47 88 67 63 74 24 73
34 38 49 75 21 14 13 75
11 61 8 55 96 14 2 79 36 57 45
51 65 52 8 2 92 46 22 31 83 15 43
5 72 42 40 48 23 36 71
12 42 65 32 57 34 26 61 27 93 96
78 87 35 72 61 52 37 72 29 47 65 34
48 4 5 24 25 20 47 68
13 15 56 44 74 23 93 8 36 74 57
34 9 25 15 77 45 52 30 69 32 1 52
25 97 95 85 33 61 15 8

```
      14   8  91  85  10  16  14  22  44  85  61
  96  11  75  23  67  43  51   4  15  26  35  50
   6  31  57   3  83  29  85  54
      15  50   6  61  91  66  26   3  69  29   2
  41  98  72  21  36  33  86  20  41  28  37   9
  59  51  62  46  35  77  32  94
      16  76  15  20  77  72  61  43  80   2  34
  21  64  24  31  69  71  92   4   9  58  33  89
  98  93  49  66  71  20  12  30
      17  95  23   8  33  88  81  43  49  59  96
  68  77  53  84  49  67  19  61  41  12   5   6
  13  49  75  79  69  32  12  45
      18  36   3  14  72  35  68  35  50  82  94
  23  15  72  71  93  55  38  43  46  39  93   0
  24  70   7  57   3  80  66  28
      19   6  45  95  55  31  14  66  19  44  76
  31  60  33  91  73  68  86  34  77  14  39  33
  53  60  93  32   9  56  46  58
      20   9  39  56  76  48   7  82  67  61  63
  51  19  36  37  19  61  16  87  87   8  27  61
  13  91  73  53  84  98  16  80
```

DP表：

```
花束\花瓶    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30
       1   87   95   95   95   95   95   95   95   95   95   95   98   98   98   98   98   98   98   98   98   98   98   98   98   98   98   98   98   98
       2    0  162  162  190  193  193  193  193  193  193  193  193  193  193  193  193  193  193  193  193  193  193  193  193  193  193  193  193  193
       3    0    0  176  176  190  236  265  265  265  287  287  287  287  287  287  287  287  287  287  287  287  287  287  289  289  289  292  292  292  292
       4    0    0    0  248  265  265  271  356  356  356  381  381  381  381  381  381  383  383  383  383  384  384  384  384  384  384  384  384  384  384
```

```
       5    0    0    0    0  342  342  342  342  399  400
439  439  439  439  451  458  458  458  458  471  471  471
471  471  471  471  471  471  480  480
       6    0    0    0    0    0  427  427  427  427  464
464  523  523  523  533  533  533  533  533  533  533  560
560  560  560  560  560  560  560  560
       7    0    0    0    0    0    0  509  509  513  513
516  516  610  610  610  610  610  610  610  610  610  610
610  610  610  610  610  610  639  639
       8    0    0    0    0    0    0    0  534  567  567
567  615  615  673  673  691  691  691  691  691  703  703
703  703  703  707  707  707  707  707
       9    0    0    0    0    0    0    0    0  556  657
657  664  706  706  737  737  737  777  777  777  777  777
777  777  777  777  777  801  801  801
      10    0    0    0    0    0    0    0    0    0  611
753  753  753  753  756  784  825  825  840  851  851  851
851  851  851  852  852  852  852  876
      11    0    0    0    0    0    0    0    0    0    0
662  818  818  818  818  848  848  848  856  923  923  923
923  923  923  923  923  923  923  923
      12    0    0    0    0    0    0    0    0    0    0
   0  749  853  890  890  890  890  920  920  920  988  988
988  988  988  988  988  988  988  991
      13    0    0    0    0    0    0    0    0    0    0
   0    0  774  868  967  967  967  967  989  989  989 1040
1040 1085 1085 1085 1085 1085 1085 1085
      14    0    0    0    0    0    0    0    0    0    0
   0    0    0  797  935 1010 1018 1018 1018 1018 1024 1039
1046 1071 1142 1142 1168 1168 1170 1170
      15    0    0    0    0    0    0    0    0    0    0
   0    0    0    0  833  968 1096 1096 1096 1096 1096 1096
1098 1098 1133 1188 1188 1245 1245 1264
      16    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0  904 1060 1100 1105 1154 1154 1185
1194 1194 1194 1199 1259 1259 1259 1275
```

```
       17    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0  923 1121 1141 1141 1159 1160
1198 1243 1269 1273 1273 1291 1291 1304
       18    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0  966 1167 1180 1234 1234
1234 1268 1268 1326 1326 1353 1357 1357
       19    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0 1043 1181 1219 1267
1287 1294 1361 1361 1361 1382 1399 1415
       20    0    0    0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0    0 1051 1208 1280
1280 1378 1378 1414 1445 1459 1459 1479
最大美学值为: 1479
摆花方案为:
第 1 束花被摆放到第 1 个花瓶
第 2 束花被摆放到第 2 个花瓶
第 3 束花被摆放到第 3 个花瓶
第 4 束花被摆放到第 4 个花瓶
第 5 束花被摆放到第 5 个花瓶
第 6 束花被摆放到第 6 个花瓶
第 7 束花被摆放到第 7 个花瓶
第 8 束花被摆放到第 9 个花瓶
第 9 束花被摆放到第 10 个花瓶
第 10 束花被摆放到第 11 个花瓶
第 11 束花被摆放到第 12 个花瓶
第 12 束花被摆放到第 14 个花瓶
第 13 束花被摆放到第 15 个花瓶
第 14 束花被摆放到第 16 个花瓶
第 15 束花被摆放到第 17 个花瓶
第 16 束花被摆放到第 23 个花瓶
第 17 束花被摆放到第 26 个花瓶
第 18 束花被摆放到第 28 个花瓶
第 19 束花被摆放到第 29 个花瓶
第 20 束花被摆放到第 30 个花瓶
```

附录 完整代码

Homework5.cpp

```cpp
#include "DataCreate.h"
#include "ustime.h"
#include "Algorithm1.h"
#include <iostream>
using namespace std;
int main()
{
    while (1)
    {
        int F, V;
        cout << "请输入花束总数F: ";
        cin >> F;
        cout << "请输入花瓶总数V: ";
        cin >> V;
        DataCreate Data(F, V);
        int** N = Data.GetData();
        Data.Print();
        Algorithm1 Algo(F, V, N);
        Algo.Run();
        Algo.Print();
    }
}
```

ustime.h

```cpp
#pragma once
#ifdef _WIN32
#include <windows.h>
#else
#include <time.h>
```

```
#endif  // _WIND32

// 定义64位整形
#if defined(_WIN32) && !defined(CYGWIN)
typedef __int64 int64_t;
#else
typedef long long int64t;
#endif  // _WIN32

int64_t GetSysTimeMicros();
```

ustime.cpp

```
#include "ustime.h"

// 获取系统的当前时间，单位微秒(us)
int64_t GetSysTimeMicros()
{
#ifdef _WIN32
    // 从1601年1月1日0:0:0:000到1970年1月1日0:0:0:000的时间(单位
100ns)
#define EPOCHFILETIME   (116444736000000000UL)
    FILETIME ft;
    LARGE_INTEGER li;
    int64_t tt = 0;
    GetSystemTimeAsFileTime(&ft);
    li.LowPart = ft.dwLowDateTime;
    li.HighPart = ft.dwHighDateTime;
    // 从1970年1月1日0:0:0:000到现在的微秒数(UTC时间)
    tt = (li.QuadPart - EPOCHFILETIME) / 10;
    return tt;
#else
    timeval tv;
    gettimeofday(&tv, 0);
```

```
    return (int64_t)tv.tv_sec * 1000000 +
(int64_t)tv.tv_usec;
#endif // _WIN32
    return 0;
}
```

DateCreate.h

```
#pragma once
class DataCreate
{
public:
    DataCreate(int, int);
    ~DataCreate();
    int** GetData();
    void Print();
private:
    int F, V;   //需要随机生成的数据量
    int** data; //随机生成的数据数组
};
```

DataCreate.cpp

```
#include "DataCreate.h"
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
```

```cpp
using namespace std;

DataCreate::DataCreate(int f, int v)
{
    F = f;
    V = v;
    data = new int* [F + 1];
    for (int i = 0; i <= F; ++i)
        data[i] = new int[V + 1];
}

int** DataCreate::GetData()
{
    srand(time(NULL));
    for (int i = 1; i <= F; i++)
        for (int j = 1; j <= V; j++)
        {
            data[i][j] = rand() % 100;
        }
    return data;
}

void DataCreate::Print()
{
    cout << "美学值表: " << endl;
    cout << setw(8) << "花束\\花瓶";
    for (int i = 1; i <= V; ++i)
        cout << setw(5) << i;
    cout << endl;
    for (int i = 1; i <= F; ++i)
    {
        cout << setw(9) << i;
        for (int j = 1; j <= V; ++j)
            cout << setw(5) << data[i][j];
        cout << endl;
    }
}
```

```
DataCreate::~DataCreate()
{
    for (int i = 0; i <= F; ++i)
        delete[] data[i];
    delete[] data;
}
```

Algorithm1.h

```
#pragma once
#include "ustime.h"
using namespace std;
class Algorithm1
{
public:
    Algorithm1(int, int, int**);
    ~Algorithm1();
    int Run();
    void Print();
    long long RunTime;
private:
    int F, V;
    int** N;
    int** M;
    int* Result;
};
```

Algorithm1.cpp

```
#include "Algorithm1.h"
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
#define DEBUG

#define gmax(_a, _b) ((_a) > (_b) ? (_a) : (_b))
Algorithm1::Algorithm1(int f, int v, int** n)
{
    F = f;
    V = v;
    N = n;
    M = new int* [F + 1];

    for (int i = 0; i <= F; ++i)
        M[i] = new int[V + 1];
    for (int i = 0; i <= F; ++i)
        for (int j = 0; j <= V; ++j)
            M[i][j] = 0;

    Result = new int [F + 1];
    for (int i = 1; i <= F; ++i)
        Result[i] = 0;
}

int Algorithm1::Run()
{
    long long StartTime, EndTime;
    StartTime = GetSysTimeMicros();

    for (int i = 1; i <= F; ++i)
        for (int j = i; j <= V; ++j)
            M[i][j] = gmax(M[i][j - 1], M[i - 1][j - 1] +
N[i][j]);

    for (int i = F, j = V; i >= 1; j--)
```

```
    {
        if (M[i][j] > M[i][j - 1])
        {
            Result[i] = j;
            i --;
        }
    }

    EndTime = GetSysTimeMicros();
    RunTime = EndTime - StartTime;
    return M[F][V];
}

void Algorithm1::Print()
{
#ifdef DEBUG
    cout << "DP表: " << endl;
    cout << setw(8) << "花束\\花瓶";
    for (int i = 1; i <= V; ++i)
        cout << setw(5) << i;
    cout << endl;
    for (int i = 1; i <= F; ++i)
    {
        cout << setw(9) << i;
        for (int j = 1; j <= V; ++j)
            cout << setw(5) << M[i][j];
        cout << endl;
    }
#endif // DEBUG


    cout << "最大美学值为: " << M[F][V] << endl;
    cout << "摆花方案为: " << endl;
    for (int i = 1; i <= F; ++i)
        cout << "第 " << i << " 束花被摆放到第 " << Result[i]
<< " 个花瓶" << endl;
    cout << "运行时间: " << RunTime / 1000000.0 << " s" <<
endl;
```

```
}

Algorithm1::~Algorithm1()
{
    for (int i = 0; i <= F; ++i)
        delete[] M[i];
    delete[] M;

    delete[] Result;
}
```