222018321062006　　宋行健

算法实验题 4-1 会场安排问题

源代码

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int startTime[10005],endTime[10005];
    for(int i = 0; i < n; i++) {
        cin >> startTime[i] >> endTime[i];
    }
    sort(startTime,startTime+n);
    sort(endTime,endTime+n);
    int room = 0;
    int j = 0;
    for(int i = 0; i < n; ++i) {
        if(startTime[i] < endTime[j]) room++;
        else j++;
    }
    cout << room << endl;
    return 0;
}
```

程序运行截图

图 1 会场问题运行结果截图

实验心得

解决本问题在于贪心算法的核心思想。按照开始时间排序，来了一个活动就把它做了，如果没有空的会场就增加一个会场（room++），过程中维护一下会场的状态。这里实际维护了这样一个东西——在开的所有会场中，里面正在进行的会议结束时间最早的时间（endTime[j]）。

具体做法是把开始时间和结束时间分别装在两个数组中，startTime[] 和 endTime[]，从早到晚排序，循环遍历开始时间 startTime[i]，设置位置标记 j，如果 startTime[i] < endTime[j]，说明当前没有空会场可用了，要新开一个会场，room++；否则 j++，表示已经有会场能空出来了，更新当前所有会场结束的最早时间。

PPT 补充题目：贪心算法实现最少硬币找零问题

源代码

```
#include <iostream>
#include <fstream>
#include <algorithm>
using namespace std;

int Coins[6];                                       // 各面值硬币数
int CoinNum = 0;                                    // 需要的最少硬币数
int Pay;                                            // 需要支付的钱数
int CoinValue[6] = {5, 10, 20, 50, 100, 200};       // 硬币面值，以分为单位
int ChangeValue[7] = {0, 5, 10, 20, 50, 100, 200}; // 用来找零的硬币面值，0 为不找零的情况
int RealMoney = CoinValue[0] - ChangeValue[0];      // 实际支付金额 = 支付硬币 - 找零硬币 【贪心变量】
ifstream input("input.txt");
ofstream output("output.txt");

/*****************************************************************
* 函数描述： 从文件中读取测试数据
*****************************************************************/
void getData()
{
    for (int i = 0; i < 6; ++i)
        input >> Coins[i];
    double costt;
    input >> costt;
    cout << "\n 目标金额：" << costt << "元"
         << "\n 各币值数量：" << endl;
    Pay = (int)(costt * 100); //将输入的钱转为分为单位
    for (int i = 0; i < 6; ++i)
        cout << Coins[i] << " ";
}

/*****************************************************************
```

```
32. * 函数描述：  格式化输出结果
33. ****************************************************************/
34. void outputResult()
35. {
36.     if (CoinNum == 0 || Pay != 0)
37.     {
38.         cout << "\n============== impossible" << endl;
39.         output << "impossible" << endl;
40.     }
41.     else
42.     {
43.         cout << endl
44.              << "各币值剩余数量:" << endl;
45.         for (int i = 0; i < 6; i++)
46.             cout << Coins[i] << " ";
47.         cout << endl
48.              << "============== 支付和找零需要的最少总硬币数量：" << CoinNum << endl;
49.         output << CoinNum << endl;
50.     }
51. }
52.
53. /****************************************************************
54. * 函数描述：查看顾客手中是否还有 a 面值的硬币，
55. *           如果有则返回面值索引，否则返回 -1
56. * 函数返回：面值索引 or -1
57. ****************************************************************/
58. int contains(int a)
59. {
60.     for (int i = 0; i < 6; ++i)
61.         if (CoinValue[i] == a && Coins[i] > 0)
62.             return i;
63.     return -1;
64. }
65.
66. /****************************************************************
67. * 函数描述：  贪心法实现硬币找零问题，以实际支付金额作为贪心变量
68. ****************************************************************/
69. void Greed()
70. {
71.     // 顾客手中的硬币面值从大到小遍历选取
72.     for (int i = 5; i >= 0; --i)
73.     {
74.         if (Coins[i] > 0) // 如果顾客手中当前面值硬币有剩余
75.         {
76.             // 商家手中的硬币从小到大遍历选取，进行找零操作
77.             for (int j = 0; j <= i; ++j)
78.             {
79.                 RealMoney = CoinValue[i] - ChangeValue[j]; // 实际支付金额 = 支付硬币 - 找零硬币
80.                 // 只有当实际支付金额小于目标金额的时候才进行选取，否则需要增加找零硬币的金额（即，继续本层循环 j）
```

```
                if (Pay >= RealMoney)
                {
                    if (Coins[i] >= Pay / RealMoney) //如果当前面值硬币找零后足够支付余额
                    {
                        int TempCoinNum = Pay / RealMoney; // 此步骤用户消耗硬币数量
                        CoinNum += TempCoinNum * 2;
                        // 如果顾客具有该硬币，则无需使用找零，因为找零需要耗费 2 个硬币，而支付只需要一个硬币
                        if (contains(RealMoney) != -1)
                        {
                            TempCoinNum = min(TempCoinNum, Coins[contains(RealMoney)]);
                            CoinNum -= TempCoinNum;
                            Coins[contains(RealMoney)] -= TempCoinNum; // 顾客手中相应硬币数减少
                        }
                        else
                            Coins[i] -= Pay / RealMoney; // 顾客手中当前面值硬币数减少
                        Pay = Pay % RealMoney;           // 更新还需支付的金额
                        if (contains(CoinValue[i]) == -1)
                            break; // 如果顾客手中该硬币用完，直接用下一个小面值
                    }
                    else // 如果当前币值不够支付，则用完该币值
                    {
                        CoinNum += Coins[i];
                        Pay = Pay - CoinValue[i] * Coins[i];
                        Coins[i] = 0;
                    }
                }
            }
        }
    }
}

int main()
{
    while (!input.eof())
    {
        CoinNum = 0;
        getData();      // 读取测试数据
        Greed();        // 贪心法实现硬币找零问题
        outputResult(); // 输出结果
    }
    input.close();
    output.close();
}
```

程序运行截图

图 2 最少硬币问题运行结果截图

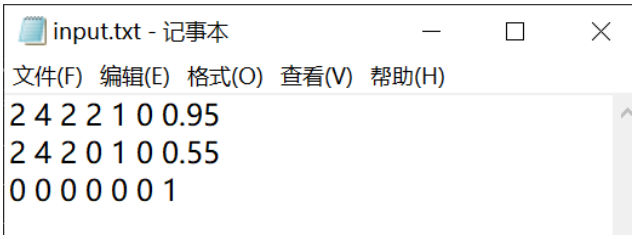

图 3 input.txt 输入文件截图

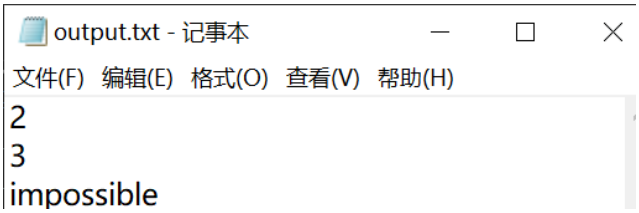

图 4 output.txt 输出文件截图

实验心得

解决本问题在于贪心变量的找取。原始的硬币问题的贪心变量很直观，就是硬币从大到小选取。而加上了找零的操作后，贪心变量就不那么直观了。本问题中的贪心变量是每个硬币的实际支付金额，实际支付金额 = 支付硬币 - 找零硬币。加上找零的本质就是通过找零的排列组合，将硬币的面值变多了。

其中有一点需要注意的就是，如果找零的排列组合（当前硬币的实际支付金额）存在等面值的真实硬币，则不用找零的操作，直接取硬币，因为找零需要耗费 2 枚硬币，而支付只需要一枚硬币。