1 实验六 贪心算法——背包问题和最优装载问题

1-1 一、实验目的

理解贪心算法的解题思想；
掌握贪心算法解题步骤；
学会使用贪心算法求解背包问题；
学会使用贪心算法求解最优装载问题。

1-2 二、实验内容

I （一） 背包问题

1. 问题描述

书上实例，3种物品，背包容量50公斤，物品1中10公斤，价值60元，物品2重20公斤，价值100元，物品3重30公斤，价值120元。

输入格式

`输入的第一行包含两个整数n, m，分别表示物品的个数和背包能装重量。以后两行分别为每个物品的重量，价值`

输出格式

`输出2行，第一行包含一个整数，表示最大价值 接下来依次输出装入的物品编号及装入比例`

样例输入

```
3 50 w[]={10,20,30} v[]={60,100,120}
```

样例输出

```
背包中物品的最大价值为:240.0
依次装入背包的物品为:
第1个物品: 1.0个
第2个物品: 1.0个
第3个物品: 0.6666667个
```

2. 要求

（1） 写出问题的分析过程 （2） 写出程序代码 （3） 贴出程序结果

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
struct Goods //定义一个物品的信息结构体
{
    int weight; //物品的重量
    int value;  // 物品的价值
    float P;    // 权重=价值/重量
    float N;    //物品装入背包的部分，如果全部装入则为1，装入一半为0.5
};
bool compare(Goods &a, Goods &b) //编写sort函数的比较函数
{
    return a.P > b.P; //采用升序排序
}

/*自己编写的直接插入排序
 void InsertSort(Goods goods[],int n){
    int j;
    for(int i=2;i<=n;++i)
    if(goods[i].P>goods[i-1].P)
    {
        goods[0]=goods[i];
    goods[i]=goods[i-1];
        for(j=i-2;goods[0].P>goods[j].P;--j)
        goods[j+1]=goods[j];
        goods[j+1]=goods[0];
}

 } */

void Greedy(Goods goods[], int n, int c) //贪心算法
{
    for (int i = 0; i < n; i++)
    {
        if (c > goods[i].weight) //如果背包足够装下整个物品
        {
            c -= goods[i].weight;
            goods[i].N = 1; //该物品全部装入记为1
        }
        else if (c > 0)
        {                                                   //如果背包不足以装下整个物品，就装入物品的一部分
            goods[i].N = c / (goods[i].weight * 1.0); //计算物品装入背包的部分
            c = 0;                                          //背包容量置0
        }
    }
}
int main()
{
    int n; //物品的数量
    int v; //背包的容量
    float total_value = 0;
    float total_weight = 0;
    // cout << "请输入背包的容量:" << endl;
    // cin >> v;
    // cout << "请输入物品的数量: " << endl;
    cin >> n >> v;
    Goods goods[3];
    // cout << "请分别输入物品的重量和价值: " << endl;
```

```
    for (int i = 0; i < n; i++)
    {
        cin >> goods[i].weight; //>> goods[i].value;//输入重量和价值
        goods[i].N = 0;         // N置0
    }
    for (int i = 0; i < n; i++)
    {
        cin >> goods[i].value;
    }
    sort(goods, goods + n, compare); //调用C++内置的sort函数 ， 当然也可以自己编写比较函数
    // InsertSort(goods,n);
    Greedy(goods, n, v);

    for (int i = 0; i < n; i++)
    {
        if (goods[i].N == 0.0)
            break;
        total_value += (goods[i].value * goods[i].N);   //装入背包的物品总价值
        total_weight += (goods[i].weight * goods[i].N); //装入背包的物品总重量
        // cout << "weight: " << goods[i].weight << "  " << "value: " <<
goods[i].value << "  the part of goods: " << goods[i].N << endl;//输出装入背包的物品信息
    }
    printf("背包中物品的最大价值为:%.1f\n", total_value);
    // cout << "背包中物品的最大价值为:" << total_value << endl;//输出装入物品的总价值
    // cout << "背包的容量为: " << v << endl;//输出背包容量
    // cout << "装入背包中的物品的总重量为: " << total_weight << endl;//输出装入物品的总重量
    cout << "依次装入背包的物品为:" << endl;
    for (int i = 0; i < n; i++)
    {
        cout << "第" << i + 1 << "个物品:" << goods[i].N << "个" << endl;
    }
    return 0;
}
```

II （二）最优装载问题

1. 问题描述

题目

有一批集装箱要装上一艘载重量为c的轮船。其中集装箱i的重量为wi。最优装载问题要求确定在装载体积不受限制的情况下，将尽可能多的集装箱装上轮船。

输入格式

`输入的第一行包含两个整数n, m，分别表示集装箱的个数和背包能装重量。　　以后一行为每个集装箱的重量`

输出格式

`输出2行，第一行包含一个整数，表示最优装载量 接下来依次输出装入的物品编号`

样例输入

`4 5 w[]={2,3,1,1}`

样例输出

```
最优装载重量为: 4.0
最优装载下被选中的集装箱序号为:  0 2 3
```

2. 要求

（1） 写出问题的分析过程 （2） 写出程序代码 （3） 贴出程序结果

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

struct load
{
    int index;
    int w;
} box[1001];

bool cmp(load a, load b)
{
    if (a.w < b.w)
        return true;
    else
        return false;
}

int main()
{
    int c, n;
    int x[1001];
    while (scanf("%d%d", &c, &n) != EOF)
    {
        memset(box, 0, sizeof(box));
        memset(x, 0, sizeof(x));
        for (int i = 1; i <= n; i++)
        {
            scanf("%d", &box[i].w);
            box[i].index = i;
        }
        stable_sort(box, box + n + 1, cmp);
        if (box[1].w > c)
        {
            printf("No answer!\n");
            continue;
        }
        int i;
        for (i = 1; i <= n && box[i].w <= c; i++)
        {
            x[box[i].index] = 1;
            c -= box[i].w;
        }
        printf("%d\n", i - 1);
        for (i = 1; i <= n; i++)
            if (x[i])
                printf("%d ", i);
        printf("\n");
    }
    return 0;
}
```

1-3 三、实验总结（写出本次实验的收获，遇到的问题等）