贪心算法

算法定义

人心不足蛇吞相

贪心算法（又称贪婪算法）是指，在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，算法得到的是在某种意义上的局部最优解。

贪心算法

贪心算法总是作出在当前看来最好的选择。也就是说贪心算法并不从整体最优考虑，它所作出的选择只是在某种意义上的局部最优选择.
当然，希望贪心算法得到的最终结果也是整体最优的.虽然贪心算法不能对所有问题都得到整体最优解，但对许多问题它能产生整体最优解.
而在一些情况下，即使贪心算法不能得到整体最优解，其最终结果却是最优解的很好近似.
贪心算法不需要回溯。

贪心算法求解步骤

1.初始化:已知问题有n个输入，置问题的解集合J为空;
2.选度量标准:选取一种度量标准，按照这种度量标准对n个输入排序;
3.考察输入:按序-次输入一个量，看该量能否和J中已选出来的元素(该量度意义下的部分最优解)加在一起构成新的可行解:如果可以，则把该量并入J集合，从而得到一个新的部分解集合;如果不可以，则丢弃该量，J集合保持不变。之后，继续上述过程，考察下一输入量，直到所有的输入都考察完毕。
4.获得贪心解:所有的n个输入都被考虑完毕，则被记入到集合J中的输入量构成了这种量度意义下的问题的最优解。
贪心方法的求解关键：选取能够得到问题最优解的度量标准。

贪心算法的关键步骤

(1)决定问题的最优子结构;

(2) 设计出一个递归解; .

(3) 证明递归的任一阶段，最优的选择总是贪心选择，即做出贪心选择以后所有其他子问题都为空;

(4)将递归转化为迭代;

加勒比海盗船——最优装载问题

问题描述

在北美洲东南部，有一片神秘的海域，那里碧海蓝天、阳光明媚，这，正式传说中海盗最活跃的加勒比海(Caribbean Sea)。17世纪时，这里更是欧洲大陆的商旅舰队到达美洲的必经之地，所以当时的海盗活动非常猖獗，海盗们不仅攻击过往商人，甚至攻击英国皇家舰..... 有一天，海盗们截获了一艘装满各种各样古董的货船，每一件古董都价值连城，一旦打碎就失去了它的价值。虽然海盗船足够大，载重量为C，每件古董的重量为Wi，海盗们如何把尽可能多数量的宝贝装上海盗船呢？

加勒比海盗船——最优装载问题

问题分析

首先要确定贪心策略，选择一个你认为最好的方案，运用贪心算法的缺点就是可能得不到最优解。

这道题要求古董的数量尽可能多，而船的容量是固定的，那么优先把重量小的物品放进去，在容量固定的情况下，装的物品最多，这里我就采用这种策略。

加勒比海盗船——最优装载问题

输入
重量C及古董个数、每个古董的重量，用空格分开
输出
装入的古董最大数

输入案例
30 8 4 10 7 11 3 5 14 2
输出案例
5

加勒比海盗船——最优装载问题

```
#include <iostream>
#include <algorithm>
using namespace std;
const int N = 101;//数组不用定义太多
double w[N];
int main()
{
    double c;
    int n;
    cout << "请输入载重量C及古董个数:" << endl;
    cin >> c >> n;
    cout << "请输入每个古董的重量，用空格分开:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> w[i];
    }
    sort(w, w + n);
    double = tmp = 0.0;
    int ans = 0;
    for (int i = 0; i < n; i++) {
        tmp += w[i];
        if (tmp <= c) {
            ans ++;
        } else {
            break;
        }
    }
    cout << "能装入的古董最大数为Ans=";
    cout << ans << endl;
    return 0;
}
```

阿里巴巴与四十大盗--背包问题

问题描述
阿里巴巴闯进了四十大盗的宝藏山洞，山东一共有n种宝物，每种宝物有一定的重量w和相应的价值v，毛驴只能运走m重量的宝物，一种宝物只能拿一样，宝物可以分割，怎样才使价值最大呢？

阿里巴巴与四十大盗--背包问题

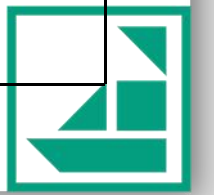
问题分析

（1）每次挑选价值最大的宝物装入背包，得到的结果是否最优？

（2）每次挑选重量最小的宝物装入背包，能否得到最优解？

（3）每次选择单位重量价值最大的宝物，能否使价值最高？

如果选择价值最大的宝物，但重量非常大，也是不行的，因为运载能力是有限的，所以舍弃策略（1）；如果选重量小的物品装入，那么其价值不一定高，所以不能再总重限制的情况下保证价值最大，所以舍弃策略（2）；而第三种是每次选择单位重量价值最大的宝物，也就是说每次选择性价比（价值/重量）最高的宝物，如果可以达到运载重量m，那么一定能使得价值最大，因此采用策略（3），每次从剩下的宝物中选择性价比最高的宝物。



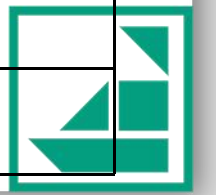
阿里巴巴与四十大盗--背包问题

问题分析
有两个条件，一个是价值与重量，宝物本身，还有一个限制条件，驴的容量。这里多了一个前提，宝物可以分割，因此，我们不选择最轻重量或者最贵宝物，而是单位价值最高的宝物先装。 创造一个结构体，存入，w,v,单位价值。按单位价值对结构体排序。 cmp(struct 1 ,struct 2)

阿里巴巴与四十大盗--背包问题

输入
毛驴的承重能力和宝物的数量、每个宝物的重量和价格
输出
毛驴载重宝物的最大价值

输入案例
10 4 2 4 2 6 4 9 6 12
输出案例
23



阿里巴巴与四十大盗--背包问题

```
#include<iostream>
#include<algorithm>
using namespace std;
const int M=10000000;
struct treasure{
    double w;
    double v;
    double p;
}s[M];
bool cmp(treasure a, treasure b){
    return a.p>b.p;
}
int main(){
    double m;
    int n;
    cin>>n>>m;
    for(int i=0;i<n;i++){
        cin>>s[i].w>>s[i].v;
        s[i].p=s[i].v/s[i].w;
    }
    sort(s,s+n,cmp);
    double sum=0.0;
    for(int i=0;i<n;i++){
        if(m>s[i].w){
            m-=s[i].w;
            sum+=s[i].v;
        }
        else{
            sum+=m*s[i].p;
            break;
        }
    }
    cout<<sum<<endl;
    return 0;
}
```

阿里巴巴与四十大盗--背包问题

课后练习
一本通基础算法、动态规划、背包问题 1267、1268、1269、1270

高级钟点秘书——会议安排

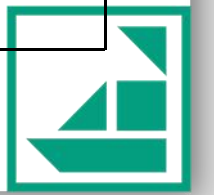
问题描述

所谓“钟点秘书”,是指年轻白领女性利用工余时间为客户提供秘书服务,并按钟点收取酬金。“钟点秘书”为客户提供有偿服务的方式一般是:采用电话、电传、上网等“遥控”式服务,或亲自到客户公司处理部分业务。

简而言之：最之间段内开最多的会议，但是会议和会议之间不能相交

高级钟点秘书——会议安排

问题分析
会议安排问题是典型的贪心算法，要得到最多安排的会议数，我们有3种贪心策略，按会议最早开始，按会议持续时间最短，按会议最早结束，按会议最早开始可能会有一个会议从早上8点到晚上，持续时间最短，有可能会议开始时间晚导致得不到最优解，按第三种最早结束策略，能得到最优解，如果两个会议结束时间相等，就按最晚开始排序，这样持续时间最短。最优解包括了子问题的最优解，符合最优子结构的性质。



高级钟点秘书——会议安排

问题分析

两个会议之间不能相交，不能有交集。

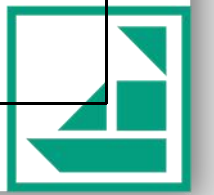
贪心策略：每次从剩下的会议中选择具有最早结束时间且与已安排的会议相容的会议安排

也就是说我们先把会议的结束时间从小到大排序，然后再每次从会议的结束时间和后面没有安排的会议进行安排，但是不能和后面的会议相交。

高级钟点秘书——会议安排

输入
会议的个数、会议的开始时间和结束时间
输出
选择的会议和总安排的会议数

输入案例
5　1 4、 2　6、34　98 、4　7、2　9
输出案例
您选择了第1会议 您选择了第4会议 您选择了第3会议 一共可以开3个会议



高级钟点秘书——会议安排

课后练习
一本通1422：【例题1】活动安排