什么是贪心?

贪心的本质是选择每一阶段的局部最优，从而达到全局最优。

贪心也可以说不是一种算法 它可以是一种,思想贪心算法在有最优子结构的问题中尤为有效。最优子结构的意思是问题能够分解成子问题来解决，子问题的最优解能递推到最终问题的最优解。

例如 一堆硬币n枚，每一枚硬币价值可能不一样 你只能挑k枚 怎么挑才能让钱最多，很显然，我贪心一点，从价值最大的从价值次大的挑，这样就能保证我们挑到的价值最大。

贪心算法并没有固定的套路

所以唯一的难点就是如何通过局部最优，推出整体最优。

那么如何能看出局部最优是否能推出整体最优呢？有没有什么固定策略或者套路呢?

在想到贪心时，最好用的策略就是举反例，如果想不到反例，那么就试一试贪心吧。

刷题的时候，手动模拟一下感觉可以局部最优 推出整体最优，而且想不到反例，那么就试一试贪心

贪心算法一般分为如下四步（网上百度的）：

1.将问题分解为若干个子问题

2.找出适合的贪心策略

3.求解每一个子问题的最优解

4.将局部最优解堆叠成全局最优解

其实这个分的有点细，我们真正做题的时候很难分出这么详细的解题步骤。

在我看来，贪心没有套路，就是常识性推导加上举反例hack这个贪心是否成功。

常见

- 「我们将 XXX 按照某某顺序排序，然后按某种顺序（例如从小到大）选择。」。
- 「我们每次都取 XXX 中最大/小的东西，并更新 XXX。」（有时「XXX 中最大/小的东西」可以优化，比如用优先队列维护）

下面是一些简单题的题解

codeup题单的一些中文题:

A 看电视

题目:

> 暑假到了，小明终于可以开心的看电视了。但是小明喜欢的节目太多了，他希望尽量多的看到完整的节目。
> 现在他把他喜欢的电视节目的转播时间表给你，你能帮他合理安排吗?

思路:

> 小明想看最多的电视 ,小明只有一个人, 所有我们应该是 每一个时间段结束得越早越好, 从而来看得最多的电视
>
> 这道题属于区间贪心的一种 是线段覆盖问题 问n个区间 如何找出最大的且不重叠的区间个数
>
> 我们首先讲每一个区间的开始时间`l`和结束时间`r`读入进来,然后我们对每一个区间的结束时间排一个序 开始时间不用管 因为我们知道 结束得越早越好就行 排好序以后 从第二个时间段开始扫描 如果当前的时间已经大于了上次看电视的时间 那我们就直接把`pre`标记为本次看完的时间并且`ans++`即可

AC代码:

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N = 111;
int n;
struct E{int l,r;};
E g[N];
bool cmp(E a,E b){return a.r<b.r;}
int main(){
    while(cin>>n&&n){
        for(int i=1;i<=n;i++)
        cin>>g[i].l>>g[i].r;
        sort(g+1,g+1+n,cmp);
        int pre = g[1].r,ans = 1;
        for(int i=2;i<=n;i++){
            if(pre<=g[i].l)
                pre=g[i].r,ans++;
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

B 出租车费

题目:

> 某市出租车计价规则如下：起步4公里10元，即使你的行程没超过4公里；接下来的4公里，每公里2元；之后每公里2.4元。行程的最后一段即使不到1公里，也当作1公里计费。
> 一个乘客可以根据行程公里数合理安排坐车方式来使自己的打车费最小。

例如，整个行程为16公里，乘客应该将行程分成长度相同的两部分，每部分花费18元，总共花费36元。如果坐出租车一次走完全程要花费37.2元。
现在给你整个行程的公里数，请你计算坐出租车的最小花费。

思路:

为了使乘坐价钱最少 我们需要不断的换乘

我们可以枚举一下开始10公里之内所需要耗费的钱 :

1-4公里肯定是10元

5公里肯定是12元 直到8公里都是每一公里增加2元

公里数	换车	不换车	优劣
9	18+10	18+2.4	2
10	18+10	18+2.4*2	2
11	18+10	18+2.4*3	2
12	18+10	18+2.4*4	2
13	18+10+2	18+2.4*5	一样
14	18+10+2*2	18+2.4*6	1
15	18+10+2*3	18+2.4*7	1

找出贪心的规律

大于四公里时 当剩余的公里数除以8的零头公里数 <=4时 我们应该 不换车 否则我们应该换车

根据题目要求 要对浮点数就行一下控制输出

AC代码:

```
#include<iostream>
using namespace std;
int n;
int main(){
    while(cin>>n&&n){
        double ans = 0;
        if(n<=4) ans = 10;
        else{
            int tt = n/8;//每8次一个循环 统计有几个8次
            int ss = n%8;//零头公里怎么处理
            if(ss<=4) ans = tt*18+ss*2.4;
            else ans = tt*18+10+(ss-4)*2;
        }
        if(ans-(int)ans==0) printf("%d\n",(int)ans);
        else printf("%.1lf\n",ans);
    }
```

```
    return 0;
}
```

F 迷瘴

题目:

> 小明正在玩游戏，他控制的角色正面临着幽谷的考验——
> 幽谷周围瘴气弥漫，静的可怕，隐约可见地上堆满了骷髅。由于此处长年不见天日，导致空气中布满了毒素，一旦吸入体内，便会全身溃烂而死。
> 幸好小明早有防备，提前备好了解药材料（各种浓度的万能药水）。现在只需按照配置成不同比例的浓度。
> 现已知小明随身携带有n种浓度的万能药水，体积V都相同，浓度则分别为Pi%。并且知道，针对当时幽谷的瘴气情况，只需选择部分或者全部的万能药水，然后配置出浓度不大于 W%的药水即可解毒。
> 现在的问题是：如何配置此药，能得到最大体积的当前可用的解药呢?
> 特别说明：由于幽谷内设备的限制,只允许把一种已有的药全部混入另一种之中（即：不能出现对一种药只取它的一部分这样的操作）。

思路:

> 贪心点 先拿小浓度的　把浓度从小到大排序 从开头开始计算 直到浓度超过退出

```
#include<iostream>
#include<algorithm>
using namespace std;
int const N = 111;
int n,v,w;
double g[N];
int main(){
    int t;cin>>t;
    while(t--){
        cin>>n>>v>>w;
        for(int i=1;i<=n;i++) cin>>g[i];
        int vv=0;double ww=0;
        sort(g+1,g+1+n);
        for(int i=1;i<=n;i++){
            if((vv*ww+v*g[i])/(vv+v)<=w){
                ww=(vv*ww+v*g[i])/(vv+v);
                vv+=v;
            }else break;
        }
        printf("%d %.2lf\n",vv,ww/100);
    }
    return 0;
}
```

G 找零钱

题目:

> 小智去超市买东西，买了不超过一百块的东西。收银员想尽量用少的纸币来找钱。纸币面额分为50 20 10 5 1 五种。请在知道要找多少钱n给小明的情况下，输出纸币数量最少的方案。 1<=n<=99;

思路:

> 贪心　常识性　我们肯定先换大的,再换小的 既然要纸币最少的方案 那我们优先对大额钞票就行兑换 然后依次就行判断即可

```
#include<iostream>
using namespace std;
int n;
int main(){
    while(cin>>n){
        if(n/50){
            printf("50*%d",n/50);
            if(n%50) printf("+");n%=50;
        }
        if(n/20){
            printf("20*%d",n/20);
            if(n%20) printf("+");n%=20;
        }
        if(n/10){
            printf("10*%d",n/10);
            if(n%10) printf("+");n%=10;
        }
        if(n/5){
            printf("5*%d",n/5);
            if(n%5) printf("+");n%=5;
        }
        if(n) printf("1*%d",n);
        printf("\n");
    }
    return 0;
}
```

洛谷基础题单 贪心题

P2240 部分背包问题:

阿里巴巴走进了装满宝藏的藏宝洞。藏宝洞里面有 $N(N \leq 100)$ 堆金币，第 i 堆金币的总重量和总价值分别是 $m_i, v_i (1 \leq m_i, v_i \leq 100)$。阿里巴巴有一个承重量为 $T(T \leq 1000)$ 的背包，但并不一定有办法将全部的金币都装进去。他想装走尽可能多价值的金币。所有金币都可以随意分割，分割完的金币重量价值比（也就是单位价格）不变。请问阿里巴巴最多可以拿走多少价值的金币？

思路:

> 因为题目说了金币可以以任意形式分割 所以说我们就可以知道每一个单位重量的金币的价值是多大 我们首先肯定带走价值大的金币 这就是这一道题贪心的点

> 我们对价值大的排序 对于价值来说 有可以使用`g[i].v*1.0/g[i].m`来获得他们的单价

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N = 111;
struct E{int m,v;double t;}g[N];
int n,t;
bool cmp(E x,E y){return x.t>y.t;}
int main(){
    cin>>n>>t;
    for(int i=1;i<=n;i++)
    cin>>g[i].m>>g[i].v,g[i].t=g[i].v*1.0/g[i].m;
    sort(g+1,g+1+n,cmp);
    double res=0;int w=0;
    for(int i=1;i<=n;i++){
        if(w+g[i].m<=t) w+=g[i].m,res+=g[i].v;
        else {
            res+=g[i].t*(t-w);w=t;break;
        }
    }
    printf("%.2lf",res);
    return 0;
}
```

P1223 排队接水

有 n 个人在一个水龙头前排队接水，假如每个人接水的时间为 T_i，请编程找出这 n 个人排队的一种顺序，使得 n 个人的平均等待时间最小。

思路:

> 有n个人在水龙头面前接水 大家每一个都有一个节水时间　按照常识 我们应该先让接水时间短的优先接水 这样就可以保证每一个人接水时 前面的人接水时间之和为最小
>
> 这个题跟短作业优先一个道理
>
> 所以我们需要存一下这是第几个人 以及接水需要的时间按照`接水时间升序排序`

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N = 1111;
int n;
struct E{int x,i;}g[N];
bool cmp(E a,E b){return a.x<b.x;}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++)
    cin>>g[i].x,g[i].i=i;
```

```
    sort(g+1,g+1+n,cmp);
    double res=0;int sums=0;
    for(int i=1;i<=n;i++){
        printf("%d ",g[i].i);
        res+=sums;sums+=g[i].x;
    }
    printf("\n%.2lf\n",res/n);
    return 0;
}
```

P1803 凌乱的yyy / 线段覆盖

题目描述

现在各大 oj 上有 n 个比赛，每个比赛的开始、结束的时间点是知道的。

yyy 认为，参加越多的比赛，noip 就能考的越好（假的）。

所以，他想知道他最多能参加几个比赛。

由于 yyy 是蒟蒻，如果要参加一个比赛必须善始善终，而且不能同时参加 2 个及以上的比赛。

思路:

> 如果认真看过codeup题解篇(上面A题)的应该清楚 这道题就是一个换壳 本质上都是一样的
>
> 人只有一个人 能够选择的区间也是只有这么多个
>
> 我们首先讲每一个区间的开始时间`l`和结束时间`r`读入进来,然后我们对每一个区间的结束时间排一个序 开始时间不用管 因为我们知道 结束得越早越好就行 排好序以后 从第二个时间段开始扫描 如果当前的时间已经大于了上次比赛的时间 那我们就直接把`pre`标记为本次看完的时间并且`ans++`即可
>
> AC代码与A 看电视一致 只不过数据范围得改一下

AC代码:

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N = 1e5+111;
int n;
struct E{int l,r;};
E g[N];
bool cmp(E a,E b){return a.r<b.r;}
int main(){
    while(cin>>n){
        for(int i=1;i<=n;i++)
        cin>>g[i].l>>g[i].r;
        sort(g+1,g+1+n,cmp);
        int pre = g[1].r,ans = 1;
```

```
        for(int i=2;i<=n;i++){
            if(pre<=g[i].l)
                pre=g[i].r,ans++;
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

P1090 [NOIP2004 提高组] 合并果子

题目

在一个果园里，多多已经将所有的果子打了下来，而且按果子的不同种类分成了不同的堆。多多决定把所有的果子合成一堆。

每一次合并，多多可以把两堆果子合并到一起，消耗的体力等于两堆果子的重量之和。可以看出，所有的果子经过 $n-1$ 次合并之后， 就只剩下一堆了。多多在合并果子时总共消耗的体力等于每次合并所耗体力之和。

因为还要花大力气把这些果子搬回家，所以多多在合并果子时要尽可能地节省体力。假定每个果子重量都为 1 ，并且已知果子的种类 数和每种果子的数目，你的任务是设计出合并的次序方案，使多多耗费的体力最少，并输出这个最小的体力耗费值。

例如有 3 种果子，数目依次为 1 ， 2 ， 9 。可以先将 1 、 2 堆合并，新堆数目为 3 ，耗费体力为 3 。接着，将新堆与原先的第三堆合并，又得到新的堆，数目为 12 ，耗费体力为 12 。所以多多总共耗费体力 $=3+12=15$ 。可以证明 15 为最小的体力耗费值。

思路

> 这道题的题意就是要使每次合并的花费最小化 那么显而易见的 我们必须使得先合并小的两堆,然后现在只有n-1堆了
>
> 然后我们再挑选`最小的两堆`注意 是最小的两堆 而并不是刚才那一堆和新的一堆 比如 1 1 1 1 1 合并一次以后 我们有 2 1 1 1 我们不应该选择 2 1
>
> 应该选择1 1 来合并 这样每次花费的力气是最小的
>
> 这里有一道类似的题(http://acm.suse.edu.cn/problem.php?id=1193)
>
> 我们代码采用的是优先队列来做 优先队列就是每次给你自动排序好的队列 保证每一次出队列都是最小的
>
> priority_queue<int,vector,greater > que; 是小根堆 升序排序
>
> priority_queue<int,vector,less > que; 是大根堆 降序排序 默认写成priority_queue就是大根堆

```
#include<iostream>
#include<queue>
using namespace std;
const int N = 1e5+111;
int n;
```

```cpp
priority_queue<int,vector<int>,greater<int> > que;
int main(){
    cin>>n;for(int i=1;i<=n;i++) {
        int x;cin>>x;que.push(x);
    }
    int res = 0;
    while(que.size()!=1){
        int _1 = que.top();que.pop();
        int _2 = que.top();que.pop();
        que.push(_1+_2);res+=_1+_2;
    }
    cout<<res<<endl;
    return 0;
}
```

P3817 小A的糖果

题目

小 A 有 n 个糖果盒，第 i 个盒中有 a_i 颗糖果。

小 A 每次可以从其中一盒糖果中吃掉一颗，他想知道，要让任意两个相邻的盒子中糖的个数之和都不大于 x，至少得吃掉几颗糖。

思路

> 已知一个序列 和 一个k 要求相邻两个元素之和不大于k
>
> 例如 `3 1 4 1 5` 和 `k=2` 我们肯定优先删除第i+1个位置上的元素 对于1~n-1 因为这样能够影响下一对 `i,i+1` 如果减成负数了置为0 再去减`i`

```cpp
#include<iostream>
#include<algorithm>
#define int long long
using namespace std;
const int N = 1e5+111;
int n,m;
int g[N];
signed main(){
    cin>>n>>m;for(int i=1;i<=n;i++) cin>>g[i];
    int res = 0;
    for(int i=1;i<n;i++){
        if(g[i]+g[i+1]>m){
            res+=(g[i]+g[i+1]-m);g[i+1]-=(g[i]+g[i+1]-m);
            if(g[i+1]<0) g[i]+=g[i+1],g[i+1]=0;//减去一个负数等于加上它
        }
    }
    cout<<res<<endl;
    return 0;
```

```
}
```

P1106 删数问题

题目

键盘输入一个高精度的正整数 N（不超过 250 位），去掉其中任意 k 个数字后剩下的数字按原左右次序将组成一个新的非负整数。编程对给定的 N 和 k，寻找一种方案使得剩下的数字组成的新数最小。

思路

> 很显然比较大的数我们不应该留着 同时仔细观察一下数的大小 同样一个数 是123留着还是132留着比较好呢 肯定是123
>
> 比如样例的 175438
>
> 每一次删除都选择这一项大于后一项的值 然后从头开始扫描
>
> 但是还有一个问题 万一是 123456789呢?
>
> 很显然 当我们删完数后
>
> 如果遍历扫描删除数的时候没有`这一项大于后一项的值`这种情况的时候 那么 这一个序列一定是升序的
>
> 一个升序的序列我们是不是把最后一项删除即可
>
> 最后 还有一个坑 就是如果是 00001 删除1个呢 字符串为 0001 很显然 我们应该去除前缀0

```cpp
#include<iostream>
#include<cstring>
using namespace std;
char s[333];int v[333],k;
int main(){
    cin>>s>>k;int l=strlen(s);
    while(k--){//k次扫描
        for(int i=0;i<l;i++)//遍历
            if(s[i]>s[i+1]){//删除数据
                for(int j=i;j<l;j++) s[j]=s[j+1];
                break;//找到了退出
            }
        l--;//没找到和找到了都减1的长度
    }
    int f=0;
    for(int i=0;i<l;i++) {
        if(!f&&s[i]=='0') continue;//去除前导0
        f=1;if(f) cout<<s[i];
    }
    if(!f) cout<<0;
    return 0;
```

```
}
```

P1478 陶陶摘苹果

题目

又是一年秋季时，陶陶家的苹果树结了 n 个果子。陶陶又跑去摘苹果，这次他有一个 a 公分的椅子。当他手够不着时，他会站到椅子上再试试。

这次与 NOIp2005 普及组第一题不同的是：陶陶之前搬凳子，力气只剩下 s 了。当然，每次摘苹果时都要用一定的力气。陶陶想知道在 $s < 0$ 之前最多能摘到多少个苹果。

现在已知 n 个苹果到达地上的高度 x_i，椅子的高度 a，陶陶手伸直的最大长度 b，陶陶所剩的力气 s，陶陶摘一个苹果需要的力气 y_i，求陶陶最多能摘到多少个苹果。

> 思路:
>
> 贪心 怎么贪? 力气花费小的苹果先拿 这就是贪心点
>
> 排个序　遍历一遍 当前苹果的花费气力<=剩余气力时才继续 否则就退出
>
> 对于每一个苹果来说 如果能摘得到 并且比他力气小的已经都遍历过了 那我们就选这个苹果 然后答案++ ,气力减少`g[i].y`

```cpp
#include<iostream>
#include<algorithm>
using namespace std;
const int N = 1e5+111;
struct E{int x,y;};
E g[N];int n,m,a,b;
bool cmp(E a,E b){return a.y<b.y;}
int main(){
    cin>>n>>m>>a>>b;
    for(int i=1;i<=n;i++) cin>>g[i].x>>g[i].y;
    sort(g+1,g+1+n,cmp);
    int res=0;
    for(int i=1;i<=n&&g[i].y<=m;i++){
        if(g[i].x<=a+b) res++,m-=g[i].y;
    }
    cout<<res<<endl;
    return 0;
}
```

P5019 [NOIP2018 提高组] 铺设道路

题目

春春是一名道路工程师，负责铺设一条长度为 n 的道路。

铺设道路的主要工作是填平下陷的地表。整段道路可以看作是 n 块首尾相连的区域，一开始，第 i 块区域下陷的深度为 d_i 。

春春每天可以选择一段连续区间 $[L, R]$ ，填充这段区间中的每块区域，让其下陷深度减少 1 。在选择区间时，需要保证，区间内的每块区域在填充前下陷深度均不为 0 。

春春希望你能帮他设计一种方案，可以在最短的时间内将整段道路的下陷深度都变为 0 。

思路

> 一个贪心问题 填坑 若$a[i] > a[i-1]$,计数器$ans+ = a[i] - a[i-1]$;
>
> 假设现在有两个坑 但是旁边是一个坑
>
> 我们可以选择 进行-1的操作 小的坑也会被大的坑填掉 大的坑会减少$a[i] - a[i-1]$的深度
>
> 我们可以看一下每一个坑和前面那个坑的情况 然后直接相加
>
> 因为小的坑会被大的坑带掉 所以我们只关心大的坑需要单独填几次 小的坑总会被大的坑带掉
>
> 这里也有一道类似的题目

```cpp
#include<iostream>
using namespace std;
const int N = 1e5+111;
int g[N],n,ans=0;
int main(){
    cin>>n;
    for(int i=1;i<=n;i++) {
        cin>>g[i];
        if(g[i]>g[i-1]) ans+=g[i]-g[i-1];
    }
    cout<<ans<<endl;
    return 0;
}
```

P1208 [USACO1.3]混合牛奶

由于乳制品产业利润很低，所以降低原材料（牛奶）价格就变得十分重要。帮助 Marry 乳业找到最优的牛奶采购方案。

Marry 乳业从一些奶农手中采购牛奶，并且每一位奶农为乳制品加工企业提供的价格是不同的。此外，就像每头奶牛每天只能挤出固定数量的奶，每位奶农每天能提供的牛奶数量是一定的。每天 Marry 乳业可以从奶农手中采购到小于或者等于奶农最大产量的整数数量的牛奶。

给出 Marry 乳业每天对牛奶的需求量，还有每位奶农提供的牛奶单价和产量。计算采购足够数量的牛奶所需的最小花费。

注：每天所有奶农的总产量大于 Marry 乳业的需求量。

> 思路 按单价排序 我们肯定优先买单价少的农民的牛奶 然后再去买多的的牛奶
>
> 如果当前能卖的部分小于等于n那么全买
>
> 否则就只买n份牛奶
>
> 这道题跟P2240 部分背包问题: 类似 大家可以结合起来理解

```cpp
#include<iostream>
#include<algorithm>
using namespace std;
const int N = 2*1e6+111;
int n,m,ans;
struct E{int x,t;}g[N];
bool cmp(E a,E b){return a.x<b.x;}
int main(){
    cin>>n>>m;
    for(int i=1;i<=m;i++) cin>>g[i].x>>g[i].t;
    sort(g+1,g+1+m,cmp);
    for(int i=1;i<=m&&n;i++){//当n不为0时继续
        if(n>=g[i].t) ans+=g[i].x*g[i].t,n-=g[i].t;
        else ans+=g[i].x*n,n=0;
    }
    cout<<ans<<endl;
    return 0;
}
```

P1094 [NOIP2007 普及组]

题目

元旦快到了，校学生会让乐乐负责新年晚会的纪念品发放工作。为使得参加晚会的同学所获得 的纪念品价值相对均衡，他要把购来的纪念品根据价格进行分组，但每组最多只能包括两件纪念品， 并且每组纪念品的价格之和不能超过一个给定的整数。为了保证在尽量短的时间内发完所有纪念品，乐乐希望分组的数目最少。

你的任务是写一个程序，找出所有分组方案中分组数最少的一种，输出最少的分组数目。

思路

> 这道题是的题意是叫我们最多选两件物品 要使每一组的物品之和小于他给定的值
>
> 一般我们的想法是什么呢 肯定是最小的和最大的先匹配 如果匹配失败 说明最大的只能单独作为一组 然后再去和第二大的做比较
>
> 直到匹配成功 然后第二小的和第k(假设第一个和第k给匹配成功)大的进行匹配 依次递推下去
>
> 这一道题我们可以设置两个指针来完成这一道问题

```cpp
#include<iostream>
#include<algorithm>
using namespace std;
const int N = 1e5+111;
int n,m,g[N],ans,tot;
int main(){
    cin>>n>>m;for(int i=1;i<=m;i++) cin>>g[i];
    sort(g+1,g+1+m);
    for(int i=1,j=m;i<=j;){
        if(g[i]+g[j]<=n) ans++,i++,j--;
        else j--,ans++;
    }
    cout<<ans<<endl;
    return 0;
}
```