

太戈编程  
etiger.vip

# 信奥算法

# 贪心法

## Greedy Algorithm

*A greedy algorithm follows heuristic of making the locally optimal choice at each stage with the intent of finding a global optimum.*

贪心法直观地做出  
局部性最优决策  
试图找到全局性最优解

# 贪心法思维框架

“每次找**最@#%的**...就把它...”

“每次发现有**\*&^%特点**的...就把它...”

依次找**最便宜**的饮料，尽量多喝

依次找**最性价比**的物品，尽量多放入背包

依次找**最早结束**的电影，前去观看

从边边角角入手最容易确定局部最优解

# 合并数字

有 $n$ 个正整数，每次删去2个数 $a$ 和 $b$ ，然后加入1个数 $a*b+1$ 。反复操作直到只有一个数，求最小剩下几？

输入样例

3

1 2 3

输出样例

8

输入样例

6

8 6 5 9 7 1

输出样例

15367

请写出算法

贪心算法： 每次找数值最大的两个数合并

# 合并果子

共 $n$ 堆果子，第 $i$ 堆重量 $w_i$ ，把两堆果子合并时，消耗的体力等于两堆果子的重量之和。经过 $n-1$ 次合并之后，就只剩下一堆了。求最少耗费多少体力

输入样例

3

1 2 3

输出样例

9

请写出算法

先按照重量排序

按照重量从小到大依次考虑：  
找重量最小的两堆合并

**每次找重量最小的两堆，合并**

现场挑战

**1744**

# 贪心法思维框架

每次挑形容词的名词，动词。

请填空造句



A  
每次挑 最重 的 人 , 搭配另1人尽量重  
若无人搭配就独自开船

B  
每次挑 最重 的 人 , 搭配随便1人  
若无人搭配就独自开船

C  
每次挑 最重 的 人 , 搭配另1人尽量轻  
若无人搭配就独自开船

X

每次挑 最轻 的 人 ,

搭配另1人 尽量重

若无人搭配就独自开船

Y

每次挑 最轻 的 人 ,

搭配 随便 1人


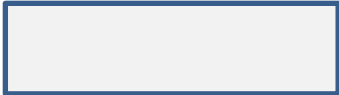

若无人搭配就独自开船

Z

每次挑 最轻 的 人 ,

搭配另1人 尽量轻

若无人搭配就独自开船

```
9      cin>>n>>c;
10     for(ll i=1;i<=n;i++)cin>>w[i];
11     sort(w+1,w+1+n);
12     ll ans=0;
13     ll i=1,j=n;
14     while(i<=j){
15         
16         ll r=c-w[j];
17         
18         if(w[i]<=r)
19             
20     }
21     cout<<ans<<endl;
```

r代表剩余载重量

```
9      cin>>n>>c;
10     for(ll i=1;i<=n;i++)cin>>w[i];
11     sort(w+1,w+1+n);
12     ll ans=0;
13     ll i=1,j=n;
14     while(i<j){
15         ans++;
16         if(w[i]+w[j]<=c)
17             i++,j--;
18         else
19             
20     }
21     if(i==j) 
22     cout<<ans<<endl;
```

# 现场挑战

## 655

# 贪心法思维框架

每次挑形容词的名词，动词。

请填空造句

# 贪心法思维框架

每次挑 最便宜 的 车 , 卖给最贫穷的人

每次挑 最昂贵 的 车 , 卖给最富有的人

每次挑 最贫穷 的 人 , 买到最便宜的车

每次挑 最富有 的 人 , 买到最昂贵的车

# 贪心法思维框架

每次挑 最便宜 的 车 , 卖给最贫穷的人

这两个算法思想  
非常相似

每次挑 最贫穷 的 人 , 买到最便宜的车

若买不起  
就找更富有的人



```
9      cin>>n>>m;
10     for(ll i=1;i<=n;i++)cin>>a[i];
11     for(ll i=1;i<=m;i++)cin>>b[i];
12     sort(a+1,a+1+n);
13     sort(b+1,b+1+m);
14     ll cnt=0;
15     ll i=1,j=1;
16     while(i<=n&& j<=m){
17         if() {
18             i++;
19             j++;
20             cnt++;
21         } else j++;
22     }
23     cout<<cnt<<endl;
```

推理路径

先理解第18-21行含义  
写下第21行执行条件  
思考第18-20行条件

补全  
程序

# 贪心法思维框架

这两个算法思想  
非常相似

每次挑 最昂贵 的 车 , 卖给最富有的人

若买不起  
就找更便宜的车

每次挑 最富有 的 人 , 买到最昂贵的车

```
9      cin>>n>>m;
10     for(ll i=1;i<=n;i++)cin>>a[i];
11     for(ll i=1;i<=m;i++)cin>>b[i];
12     sort(a+1,a+1+n);
13     sort(b+1,b+1+m);
14     ll cnt=0;
15     ll i=n,j=m;
16     while(i>=1&&j>=1){
17         if()
18             i--;
19         else{
20             
21         }
22     }
23 }
24
25 cout<<cnt<<endl;
```

推理路径  
先理解第18行含义  
写下第18行执行条件  
也就是17行条件  
第19行就是第17行的  
反面条件

补全  
程序

# 均分纸牌

有  $N$  堆纸牌,  $N \leq 100$ 。每堆上有若干张, 但纸牌总数必为  $N$  的倍数。可以在任一堆上取若干张纸牌, 然后移动。

移牌规则: 在编号为 1 的堆上取纸牌, 只能移到编号为 2 的堆上; 在编号为  $N$  的堆上取的纸牌, 只能移到编号为  $N-1$  的堆上; 其他堆上取的纸牌, 可以移到相邻左边或右边的堆上。

找出一种移动方法, 用最少的移动次数使每堆上纸牌数都一样多。

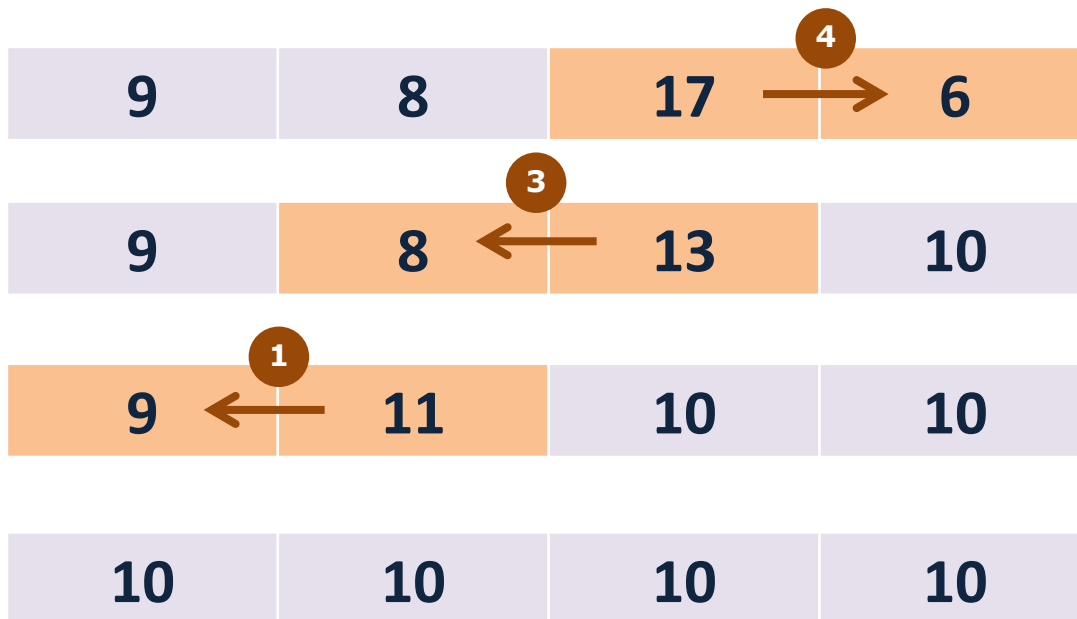
输入样例

4

9 8 17 6

输出样例

3



# 均分纸牌

先计算出平均数量 $m$

从左到右依次考虑每一个牌堆：

如果牌数不是 $m$ ，则将其和右侧交换  
使其牌数变为 $m$

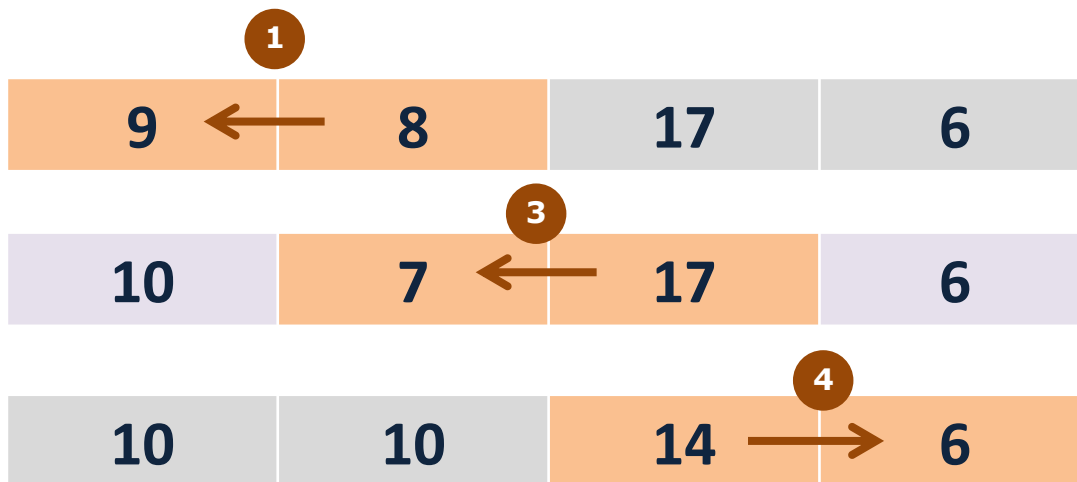
输入样例

4

9 8 17 6

输出样例

3



从边边角角入手最容易确定局部最优解

# 花匠

(题意) 花的整数高度  $h_1, h_2, \dots, h_n$  移走一部分花后，剩下花的高度依次为  $g_1, g_2, \dots, g_m$ ，以下两个条件中至少要满足一个：

条件  $A$ ：对于所有  $g_{2i} > g_{2i-1}, g_{2i} > g_{2i+1}$

条件  $B$ ：对于所有  $g_{2i} < g_{2i-1}, g_{2i} < g_{2i+1}$

请问，最多能将多少株花留在原地。m可以为1

$1 \leq n \leq 10^5, 0 \leq h_i \leq 10^6$

输入样例

5

5 3 2 1 2

输出样例

3

5

3

2

1

2

输入样例

6

1 4 2 8 5 7

输出样例

6

1

4

2

8

5

7

输入样例

5

2 2 1 1 2

输出样例

3

2

2

1

1

2

# 例题：花匠

除了两边端点以外

每次发现高度**拐点**，就保留下来

输入样例

5  
5 3 2 1 2

输出样例

3

5

3

2

1

2

输入样例

6  
1 4 2 8 5 7

输出样例

6

1

4

2

8

5

7

输入样例

5  
2 2 1 1 2

输出样例

3

2

2

1

1

2

# 花匠： 错误代码

```
1 #include<iostream>
2 using namespace std;
3 int n,cnt=0,x[100005];
4 int main(){
5     cin>>n;
6     for(int i=0;i<n;i++)cin>>x[i];
7     for(int i=1;i<n-1;i++){
8         int a=x[i]-x[i-1];
9         int b=x[i]-x[i+1];
10        if(a*b>0)cnt++;
11    }
12    cout<<cnt+2<<endl;
13    return 0;
14 }
```

错在哪里

如何修改

太戈网  
www.etiger.vip



# 花匠：构造数据

输入样例

5

5 3 1 1 2

输出样例

3

5

3

1

1

2

相邻数相同

输入样例

3

6 6 6

输出样例

1

6

6

6

全部数相同

输入样例

3

1 1000000 1

输出样例

3

1

1000000

1

数值差很大

# 贪心法总结

有时排序操作能启发算法

1

把解答拆成多个步骤，每步依次执行

2

每一步都找局部最优解

每次挑形容词的名词，动词。

从边边角角入手最容易确定局部最优解

# 后记

虽然对于有些最优化问题  
贪心法并不是正确解法

但贪心法的解可能是近似解  
也可以作为启发式搜索的参照

一般情况，贪心法复杂度都不高

# 太戈编程

1744

655

694

拓展题

113,413,538,539,500