贪心算法

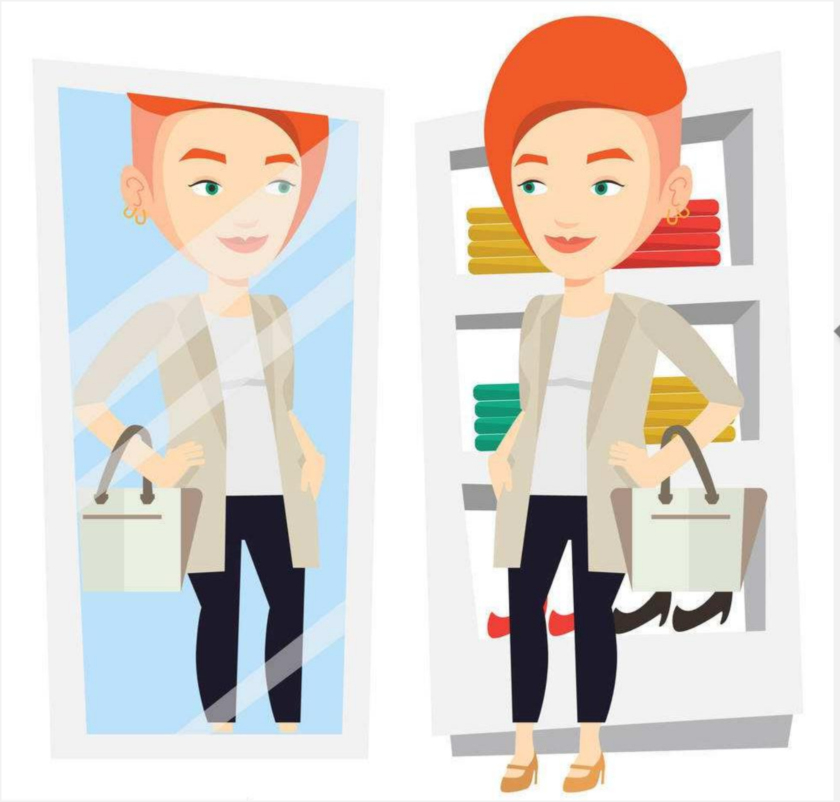
Greedy

1.什么是贪心？

什么是贪心算法？

所谓贪心算法是指，在对问题求解时，总是做出在**当前**看来最好的选择。

也就是说，不从整体最优解出发来考虑，它所做出的仅是在某种意义上的局部最优解。



其它店可能还有更好的

引例A：找零

某便利店店员需要向客户找零97元**人民币**，问最少需要多少张钱？

贪心原则：尽量用面值大的纸币。

1.讨论面值100，100>97， 不行；
2.讨论面值50， 50<=97， 可行，张数+=97/50， 余钱=97%50=47；
3.讨论面值20， 20<=47， 可行，张数+=47/20， 余钱=47%20=7;
4.讨论面值10， 10>7， 不行；
5.讨论面值5， 5<=7， 可行，张数+=7/5， 余钱=7%5=2；
6.讨论面值1， 1<=2， 可行，张数+=2/1， 余钱=2%1=0，结束；

为什么上述方法是正确的呢？

我们考虑如果任意两种面值交换一下使用顺序 ， 这样所需张数会不会更少 ？

不会！

这是一种常用的证明贪心正确性的方法，称为“**邻项交换法**”。

引例B：最大整数

设有n个正整数(n<=10000,long long范围内),将它们联接成一排组成一个最大的多位整数。

例如:n=3时,3个整数13, 312, 343联接成的最大整数为:34331213。

又如:n=4时,4个整数7, 13, 4, 246联接成的最大整数为:7424613。

贪心策略：“字典序”大的放左边

1.将n个数按“字典序”由大到小排序 return x+y>y+x

2.将排序后的数字依次输出即可

证明：邻项交换法

任意交换相邻两个数，都不可能得到更优的结果。

2. 贪心思想的应用
第一节

例1：排队打水 nkoj5215

有n个人排队到r个水龙头去打水，他们装满水桶的时间T1、T2..........Tn为整数且各不相等，应如何安排他们的打水顺序才能使每个人花费的时间总和最少？

输入格式:

第一行n，r (n<=1000,r<=100)

第二行为n个人打水所用的时间Ti (1<=Ti<=100)

输出格式: 一个整数，最少花费的总时间

样例输入:

3 2

1 2 3

样例输出:

7

样例说明：

第1，2两人在1号龙头接水，2排1后。

第1个人接水，耗时1。 第2个人排队耗时1，接水耗时2，总耗时3。 第2个人在2号龙头接水，耗时3。

总耗时1+3+3=7

例1：排队打水 解题分析

贪心原则：一遇到空闲的水龙头，就让**接水耗时少的人优先打水**。

证明：

设有1个水龙头，4个人打水，接水时间 T1 < T2 < T3 < T4

我们**按耗时小的优先打水**的贪心原则处理，总耗时Sum为：

Sum=T1+(T1+T2)+(T1+T2+T3)+(T1+T2+T3+T4)=4*T1+3*T2+2*T3+T4

越早打水的人，被累加的次数越多。

根据"邻项交换法"：任意交换两个人的的打水顺序，得到的结果一定**不比**上面的小

所以，贪心策略是正确的。

例1：排队打水 解题分析

贪心原则：一遇到空闲的水龙头，就让**接水耗时少的人优先打水。**

```
sort(T+1,T+n+1);            //按接水时间由小到大排序
Ans=0;
for (int i=1,j=0; i<=n; i++)
{
      j++;
      if(j==r+1) j=1;       //前r个人为一组，第r+1个人回到第一个水龙
      Sum[j]+=T[i];         //第i个人的耗时，即第j号水龙头接下来排队需要的耗时
      Ans+=Sum[j];
  }
```

例2：数列极差 nkoj5225

有一个由n个正整数组成的数列，要求进行如下操作：每次可擦去其中任意的两个数a和b，然后在数列中加入一个数a*b+1，如此下去直至黑板上剩下一个数为止，在所有按这种操作方式最后得到的数中，最大的为Max，最小的为Min， 则该数列的极差定义为Max-Min。

对于给定的数列，请你计算出相应的极差 。

2<=n<=10000

例2：数列极差 解题分析

问题的关键是如何求Max和Min：

设三个数x,y,z且$x<y<z$，根据题意，由如下三种方案可选：

1. Ans1=(x*y+1)*z+1=x*y*z+z+1
2. Ans2=(x*z+1)*y+1=x*y*z+y+1
3. Ans3=(y*z+1)*x+1=x*y*z+x+1

显然：Ans1>Ans2>Ans3

于是我们可以得出贪心策略：

优先选最大的两个数出来计算可以使得最终结果小

优先选最小的两个数出来计算可以使得最终结果大

例2：数列极差 参考代码

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
priority_queue<ll>Mi;
priority_queue<ll,vector<ll>,greater<ll> >Ma;
int main()
{
    ll x,y,z,n;
    scanf("%lld",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%lld",&x);
        Mi.push(x);
        Ma.push(x);
    }
    for(int i=1;i<n;i++)
    {
        x=Mi.top();  Mi.pop();
        y=Mi.top();  Mi.pop();
        z=x*y+1;     Mi.push(z);

        x=Ma.top();  Ma.pop();
        y=Ma.top();  Ma.pop();
        z=x*y+1;     Ma.push(z);
    }
    printf("%lld",Ma.top()-Mi.top());
}
```

例3：葡萄酒交易 nkoj3569

一条笔直公路上分布着n个村庄，从左往右编号1到n,每个村庄要么需要买酒，要么需要卖酒。

设第i个村庄对葡萄酒的需求为Ai，其中Ai>0表示该村需要买酒，Ai<0表示该村需要卖酒。所有村庄供需平衡,即所有Ai之和等于0

把k升葡萄酒从一个村庄运到相邻村庄需要k块钱的运费，请你计算最少需要多少运费就可以满足所有村庄对酒的需求。

$2<=n<=300000$

例3：葡萄酒交易 解题分析

先从最特殊的位置考虑：左右两端的村庄。

左端的只能与其右边的村庄交易，右端的只能与其左边的村庄交易

1. 考虑1号村庄，该村庄的需求为A_1。

1号村只能与右侧2号村直接交易，为达到需求，一定有A_1升酒在1、2号村间运输，产生运费$|A_1|$。交易结束时，1号村的需求为0，2号村的需求$A_2=A_2+A_1$

2. 考虑2号村庄，该村庄当前的需求为A_2。

左侧的1号村已经达到需求平衡，不用再考虑，可以忽略。此时可以把2号村看作位于最左侧。

那么，2号村只能与右侧3号村直接交易，为达到需求，一定有A_2升酒在2、3号村间运输，产生运费$|A_2|$。交易结束时，2号村的需求为0，3号村的需求$A_3=A_3+A_2$

3. 考虑3号村庄，……

根据上述分析，可得贪心策略：

对于任意一个村庄x，我们只需考虑与其右边村庄x+1交易，通过与右边村庄交易达到需求平衡即可，不用考虑与其他村庄间接交易的情况。

例3：葡萄酒交易 解题分析

```
for(i=1;i<=n;i++)
{
    Ans+=abs(A[i]);
    A[i+1]+=A[i];
}
```

3. 应用贪心的步骤

贪心问题解题步骤：

- 1.观察分析规律，**发现贪心策略**；
- 2.验证贪心策略是否正确：

  常用方法：邻项交换法、列举反例、范围缩放法、数学归纳和反证法等
- 3.程序实现

接下来，我们重点讨论：**发现贪心策略**和**验证贪心策略正确性**这两个核心问题

4. 贪心思想的应用
第二节 线段覆盖

例4：摄像头1 nkoj5220

在一条长度为L的笔直的公路上安装若干个摄像头，用于监控交通状况。我们可以把这条公路看作数轴[0,L]。

有n个要求需要满足，每个要求形如[x,y]，表示在[x,y]这段区间至少要安置一个摄像头。

最少需要安装多少个摄像头？

1<=n<=100000

0<=x<=y<=L<=1000000000

例4：摄像头1 解题分析

贪心策略：

1.将n个要求按**右边界**由小到大排序；

2.靠右安装摄像头：若一段区间[x,y]中必须要有一个摄像头，我们将它安装在y位置；

3.从左往右扫描每一段区间，若该区间没有摄像头，则在区间右端点处安装一个；

```
sort(D+1,D + n + 1, cmp);      //按区间右端点由小到大排序,区间i的左右界分别是D[i].Left   D[i].Right
int cnt = 1;
int Now = D[1].Right;          //Now表示当前安装摄像头的位置
for(i = 2; i <= n; i++)
      if( Now < D[i].Left )    //i号区间没有被摄像头覆盖
      {
            cnt++;
            Now = D[i].Right;  //新安装摄像头的位置
      }
```

例5：摄像头2 nkoj5221

在一条长度为L的笔直的公路，我们可以把这条公路看作数轴[0,L]。

在这条路上安装有n个摄像头，每个摄像头都有一定的拍摄区间，第i个摄像头覆盖的区间为[Xi,Yi]

最少开启几个摄像头就可以将整个这条路都置于视频监控中？

1<=n<=100000

1<=L<=1000000000

0<=x<=y<=1000000000

最小线段覆盖问题：最少几条线段就能覆盖整个指定区间

例5：摄像头2 解题分析

贪心策略：

已知：$x_1 < x_2 < x_3 < x_4 < x_5 < x_6$

考虑1：有如下四条线段，优先选哪一条？

$[x_1,x_2]$　$[x_1,x_4]$　$[x_2,x_3]$　$[x_2,x_4]$

显然，选 $[x_1,x_4]$，即左端点越小越好，若相同，右端点越大越好。

考虑2：若已选 $[x_1,x_4]$，如下三条线段，优先选哪一条？

$[x_2,x_5]$　$[x_3,x_5]$　$[x_3,x_6]$

显然，选 $[x_3,x_6]$，即选左端点在区间 $[x_1,x_4]$ 中且右端点尽可能大的线段

贪心算法：

1.将线段按左端点有小到大排序，若左端点相同，按右端点由大到小排序（这是根据**考虑1**的结论）；

2.从左往右讨论每条线段。优先选择右端点大的线段：

记录目前已选线段中，往右最远能覆盖到的位置NowFar

讨论所有 左端点<=NowFar且未被讨论过的线段，记录其中右端点的最大值MaxRight（这是根据**考虑2**的结论）

若MaxRight>NowFar，则将NowFar=MaxRight 否则 无解。

例5：摄像头2 解题分析

```
int solve()                        //数组D[]记录每条线段，已排序，D[i].L和D[i].R记录i号线段的左右端点。
{
      if(D[1].L>0)return -1;       //本题覆盖的起点从0开始
      int cnt=1;
      int NowFar=D[1].R;            //NowFar记录当前覆盖到的位置，即[0,NowFar]都被覆盖了。
      int MaxRight=D[1].R;
      if ( NowFar>=Len ) return cnt;
      int i=2;
      while(i<=n)
      {    //下面for循环找出所有左端点<=Now且未被讨论过的线段中，右端点的最大值
           for(;i<=n&&D[i].L<=NowFar; i++) MaxRight=max(MaxRight,D[i].R);
           if(MaxRight<=NowFar) return -1;                //说明区间断开了，无法连续覆盖
           cnt++;
           NowFar=MaxRight;
           if(NowFar>=Len) return cnt;
      }
      return -1;
}
//时间复杂度？
//每条线段只有一次被讨论的机会，时间复杂度O(n)
```

例6：遛狗 nkoj5227

公园里有一条长度为L的笔直的道路，我们可以把这条路看作数轴[0,L]。
有n个人想要在该路上遛狗，第i个人的狗喜欢在区间[Xi,Yi]上活动。
但是一旦两条狗相遇，它们就会打架。
公园规定不准狗打架。问，最多能安排多少人遛狗？

1<=n<=100000

1<=L<=1000000000

0<=x<=y<=1000000000

即最多选出多少条不相交的线段？

例5：遛狗 解题分析

贪心策略：优先选择长度短的线段

算法：

1.将线段按右端点有小到大排序；

2.从左往右讨论每条线段。优先选择与前面线段不相交且右端点小的线段：

```
sort(a+1,a+1+n,cmp);
int End=a[1].R;              //End记录前一条选中的狗活动的右边界
for(int i=2;i<=n;i++)
    if(a[i].L>End){End=a[i].R; ans++; }
```

5. 贪心思想的应用
第三节

例7：打怪 nkoj5233

在一款电脑游戏中，你需要打败n只怪物（从1到n编号）。

为了打败第i只怪物，你需要消耗d[i]点生命值，但怪物死后会掉落血药，使你恢复a[i]点生命值。任何时候你的生命值都不能降到0（或0以下）。

问是否存在一种打怪顺序，使得你可以打完这n只怪物而不死掉

1<=n,z<=100000

例7：打怪 解题分析

贪心策略：

1.先打会加血(D[i]<=A[i])的怪，再打会减血(D[i]>A[i])的怪；

2.对于会加血的怪，按D[i]值由小到大的顺序去打；

3.对于会减血的怪，按A[i]值由大到小的顺序去打；

加血的情况显然，下面证明会减血的情况：

设有两个怪，对应得参数分别为D1,D2,A1,A2，且A1>A2, D1>A1 ,D2>A2

若先打1号再打2号，失血分别为D1 和 D1-A1+D2

若先打2号再打1号，失血分别为D2 和 D2-A2+D1

因为D1>A1且D2>A2

有D2+D1-A1<D2-A2+D1

例8：数列+1 nkoj5234

给出n个整数，每次操作可以将一个数+1，要使这n个数都不相同， 求至少要加多少次？

n<=10000

例8：数列+1 解题分析

贪心策略：

1.将n个数由小到大排序，然后从左往右讨论每一个数；

2.若Num[i]<=Num[i-1]，则在第i个数上不断+1，直到Num[i]==Num[i-1]+1

6. 小结与习题

贪心的基本思路

建立数学模型来描述问题；

把求解的问题分成若干个子问题；

对每一子问题求解，得到子问题的局部最优解；

把子问题的解局部最优解合成原来解问题的一个解。

作业列表：贪心习题

奋斗吧
少年

巨大的成功需要付出巨大的代价

no sacrifice, no success