信奥算法

collection

部分贪心思想

当S很大时,49会使用很多次!

猜想

49会恰好使用$\left\lfloor \frac{S}{49} \right\rfloor$次

发现猜想
错误反例

可能剩余部分

$$S - 49 \times \left\lfloor \frac{S}{49} \right\rfloor$$

需要较多小数字拼凑

S=52
=49+1+1+1
=36+16

修改思路

1的使用次数不会大于等于4次

4的使用次数不会大于等于9次

… …

36的使用次数不会大于等于49次

```
 9      ll ans=S;
10      for(ll c1=0;c1<4;++c1)
11          for(ll c4=0;c4<9;++c4)
12              for(ll c9=0;c9<16;++c9)
13                  for(ll c16=0;c16<25;++c16)
14                      for(ll c25=0;c25<36;++c25)
15                          for(ll c36=0;c36<49;++c36){
16                              ll t=c1+c4*4+c9*9+c16*16+c25*25+c36*36;
17                              if
18                                  ans=min(ans,c1+c4+c9+c16+c25+c36+(S-t)/49);
19                          }
```

building

```
77      input();
78      if(OK())
79          solveFactorial();
80      else if(n<=10)
81          暴力解法
82      else
83          满分解法
```

```
13 bool OK(){
14     sort(w+1,w+n+1);
15     return 
16 }
17 void solveFactorial(){
18     ll ans=1;
19     for(ll i=2;i<=n;++i)
20         (ans*=i)%=MOD;
21     printf("%lld\n",ans);
22 }
```

枚举全排列

```
36 void solveBF2(){
37     for(ll i=1;i<=n;++i) p[i]=i;
38     ll ans=0;
39     do{
40         ll i=2;
41         for(;i<=n;++i)
42             if(w[p[i]]>w[p[i-1]]+d)
43                 break;
44         if(i<=n){
45             sort(p+i+1,p+n+1,greater<ll>());
46             continue;
47         }
48         if((++ans)==MOD) ans=0;
49     }while(next_permutation(p+1,p+1+n));
50     printf("%lld\n",ans);
51 }
```

加速剪枝

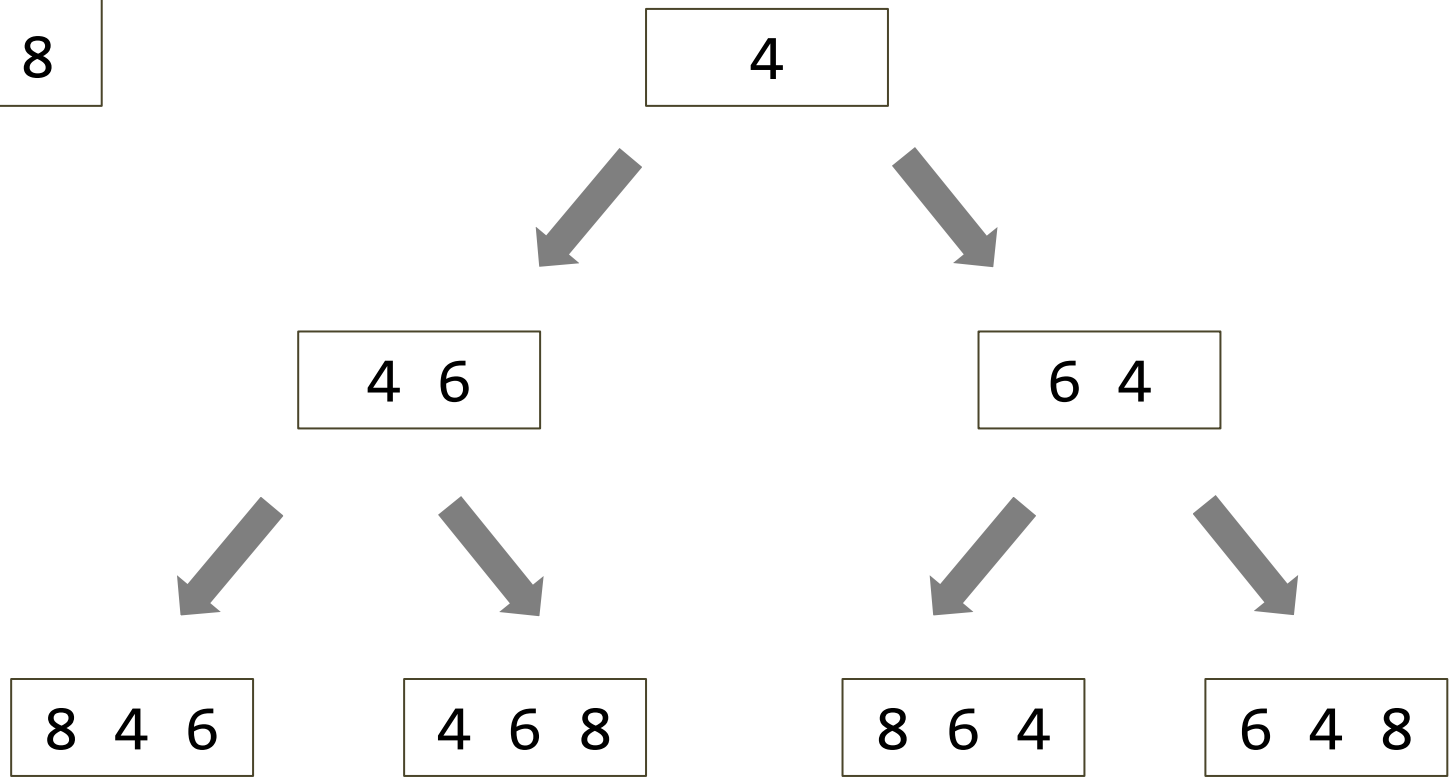
正解

贪心步骤计数

每次选最小的数安排位置

分步计数用乘法原理

输入
3 2
4 6 8



双游标定位最大的j

满足：前j-1个元素不可以贴着i号下方

```
62 void solveTwoPointers(){
63     sort(w+1,w+n+1);
64     ll ans=1;
65     ll j=1;
66     for(int i=2;i<=n;i++){
67         while(
68             ++j;
69         ll cnt=(i-j+1);
70         (ans*=cnt)%=MOD;
71     }
72     printf("%lld\n",ans);
73 }
```

friends

预处理

离散化

```
33      for(ll i=1;i<=n;++i)xs[i]=x[i];
34      sort(xs+1,xs+1+n);
35      for(ll i=1;i<=n;++i)
36          x[i]=
```

预处理

pre[i]表示i号左侧最近的数值为x[i]的编号
L[i]表示i号往左最远到几号 可以保持[L[i],i]是合法区间

x[i]=

i=0	i=1	i=2	i=3	i=4	i=5
	2	2	1	2	1

pre[i]=

i=0	i=1	i=2	i=3	i=4	i=5
0	0	1	0	2	3

L[i]=

i=0	i=1	i=2	i=3	i=4	i=5
0					

预处理

pre[i]表示i号左侧最近的数值为x[i]的编号
L[i]表示i号往左最远到几号 可以保持[L[i],i]是合法区间
pos[v]表示当前发现数值为v的最新编号

```
37  for(ll i=1;i<=n;++i){
38      pre[i]=
39      pos[x[i]]=
40      L[i]=
41  }
```

L[]满足单调不降

ST表

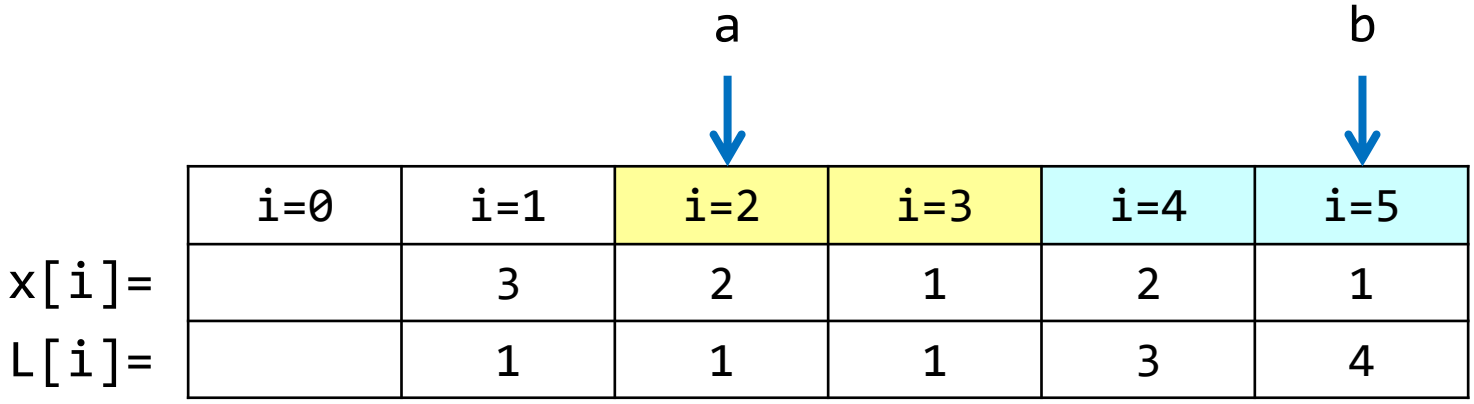
st[p][i]表示以[i,i+2^p-1]范围内的编号
为区间右端点的合法区间最长长度

```
st[0][i]=i-L[i]+1;
```

rmq(l,r)返回值表示
右端点在[l,r]范围内的合法区间最长长度

问询处理

最优区间的右端点在[a,b]范围内



两种情况分类讨论

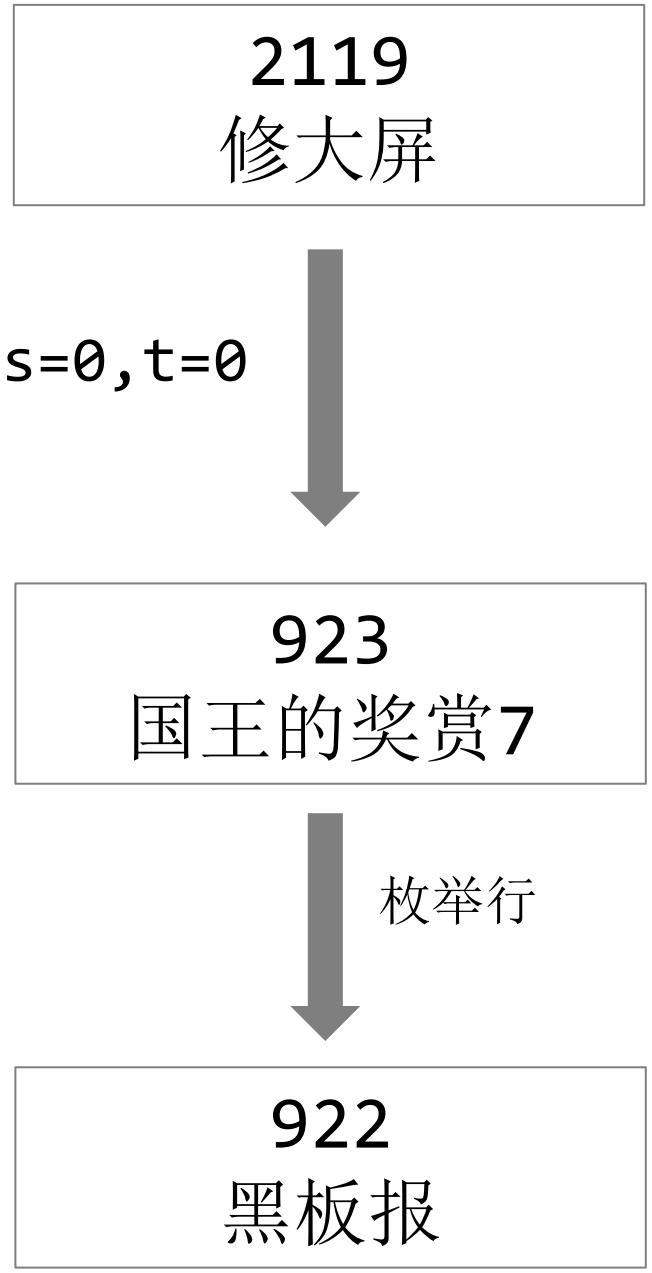
无限制最优解左端点在区间外

无限制最优解左端点在区间内

由于L[]单调不降
可以二分定位最小的id 使得 a<L[id]

```
44  for(ll i=1;i<=m;++i){
45      scanf("%lld %lld",&a,&b);
46
47      ll id=
48      ll ans=
49      if(id<=b)
50          ans=max(ans,rmq(id,b));
51      printf("%lld\n",ans);
52  }
```

led



现场挑战
922

请同学写出题目大意
已知什么求什么

枚举+优化

L[i]代表i号左侧比h[i]低的最大编号

R[i]代表i号右侧比h[i]低的最小编号

```
25      ll ans=0;
26      for(ll i=1;i<=n;i++)
27          ans=max(ans,(R[i]-L[i]-1)*h[i]);
```

枚举高度，求两端

L[i]代表i号左侧比h[i]低的最大编号

单调栈

栈内储存可能成为
之后某L[i]答案的编号

单调栈计算L[]

cnt表示单调栈当前元素个数

s[cnt]表示栈顶元素在原数组的编号

```
11      ll cnt=0;
12      for(ll i=1;i<=n;i++){
13          while(cnt&&h[i]<=h[s[cnt]])cnt--;
14          L[i]=(cnt?s[cnt]:0);
15          s[++cnt]=i;
16      }
```

为什么单调

L[i]代表i号左侧比h[i]低的最大编号

单调栈计算R[]

```
17      cnt=0;
18      for(ll i=n;i>=1;i--){
19          while(                              )cnt--;
20          R[i]=(                          );
21          s[++cnt]=i;
22      }
```

现场挑战
923

请同学写出题目大意
已知什么求什么

923破题路径

建模元素

二维矩阵

枚举降维

枚举降维

```
23      cin>>n;
24      for(int i=1;i<=n;i++)
25          for(int j=1;j<=n;j++){
26              char ch; cin>>ch;
27              if(ch=='o')f[i][j]=f[i-1][j]+1;
28          }
29      for(int r=1;r<=n;r++)solve(f[r]);
```

f[i][j]表示从第i行第j列往上共有几个连续空格

f[r]对应f第r行数组

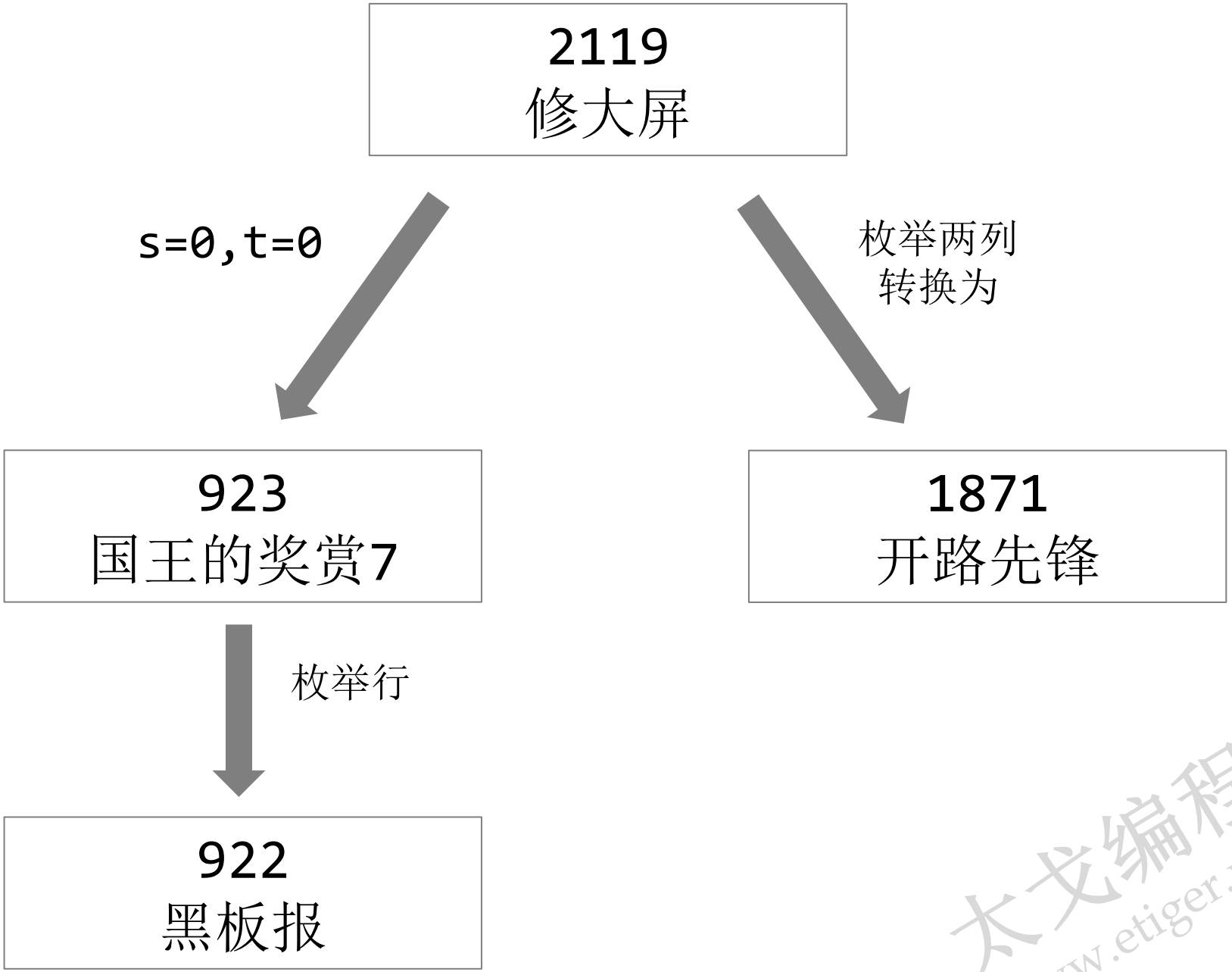
枚举降维

枚举矩形下边界的行号r

请描述对应子问题
solve(f[r])
对应的含义

恰好对应922题

```
5  void solve(int *h){
6      int cnt=0;
7      for(int i=1;i<=n;i++){
8          while(cnt&&h[i]<=h[s[cnt]])cnt--;
9          L[i]=(cnt?s[cnt]:0);
10         s[++cnt]=i;
11     }
12     cnt=0;
13     for(int i=n;i>=1;i--){
14
15
16
17     }
18     for(int i=1;i<=n;i++)
19         ans=max(ans,(R[i]-L[i]-1)*h[i]);
20 }
```



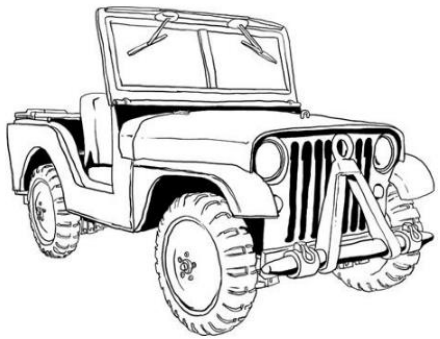
```
59      scanf("%d %d %d %d",&nR,&nC,&s,&t);
60      for(int i=1;i<=nR;++i)
61          for(int j=1;j<=nC;++j){
62              int x;
63              scanf("%d",&x);
64              sRowBad[i][j]=sRowBad[i][j-1]+1-x;
65          }
66      int ans=
67      for(int l=1;l<=nC;++l)
68          for(int r=l;r<=nC;++r){
69
70              int len=solve(l,r);
71              ans=max(ans,len*(r-l+1));
72
73          }
```

朴素贪心解

最优性剪枝1

最优性剪枝2

1871



30分

暴力枚举决策

枚举最终要清零的连续段在哪里

枚举左端点a
枚举右端点b

判断编号[a,b]区间内能否全部清零

其中最大的s个数字直接清零
剩余数字总和能否控制在t以内

30分

暴力

```
16      cin>>n>>s>>t;
17      for(int i=1;i<=n;++i)cin>>h[i];
18      int ans=0;
19      for(int i=1;i<=n;++i)
20          for(int j=i;j<=n;++j){
21              if(OK(i,j))
22                  ans=max(ans,j-i+1);
23              else
24                  break;
25          }
26      cout<<ans<<endl;
```

30分
暴力

```
5  bool OK(int a,int b){
6      int m=b-a+1;
7      for(int i=a;i<=b;++i)g[i-a]=h[i];
8      sort(g,g+m);
9      int sum=0;
10     for(int i=0;      ;++i)sum+=g[i];
11     return
12 }
```

OK()复杂度	整体复杂度
O(nlogn)	$O(n^3 logn)$

40分

二分枚举答案

二分枚举最终要清零的连续段长度x

OK(x)判断能否找到长度x的区间 使其都清零
需要再枚举左端点a

OK(x)复杂度	整体复杂度
$O(n^2 \log n)$	$O(n^2 \log^2 n)$

40分

二分

```
5  bool OK(int a,int b){
6      int m=b-a+1;
7      for(int i=a;i<=b;++i)g[i-a]=h[i];
8      sort(g,g+m);
9      int sum=0;
10     for(int i=0;i<m-s;++i)sum+=g[i];
11     return sum<=t;
12 }
13 bool OK(int len){
14     for(int i=1;i<=n-len+1;++i)
15         if(OK(i,i+len-1))return 1;
16     return 0;
17 }
```

100分

蠕动区间

右端不断右移
直到区间内无法全部清零

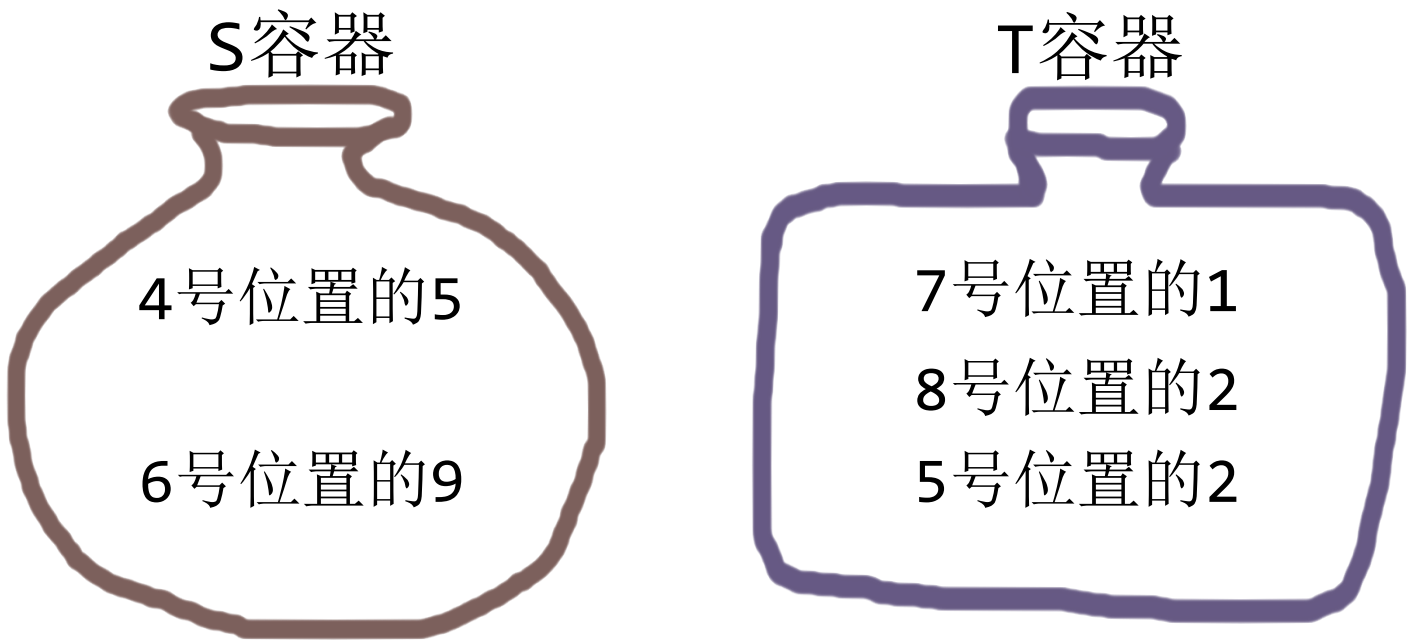
左端不断右移
直到区间内可以全部清零

使用2个容器互相导数据
维护能直接清零和减1的数

蠕动区间

i=	1	2	3	4	5	6	7	8
h[i]=	1	4	3	5	2	9	1	2

s=2,t=5



```
5  struct info{
6      int id,h;
7      bool operator<(const info&x)const{
8          return h<x.h||h==x.h&&id<x.id;
9      }
10 };
```

```
14     cin>>n>>s>>t;
15     for(int i=1;i<=n;++i)cin>>h[i];
16     set<info> S,T;
17     set<info>::iterator it;
```

容器S里存放计划直接清零的元素

容器T里存放计划多次减1的元素

```
18      int ans=0;
19      int l=1,r=1;
20      int cT=0;
21      while(l<=n){
22
33          if(cT<=t)ans=max(ans,r-l);
34          else ans=max(ans,r-l-1);
35          if(r>n)break;
36
54      }
```

cT记录容器T内数值总和

右端不断右移直到区间不能清零

左端不断右移直到区间能清零

```
22          while(r<=n&&cT<=t){
23              info x=(info){r,h[r]};
24              S.insert(x);
25              if(S.size()>s){
26                  x=*S.begin();
27                  cT+=x.h;
28                  T.insert(x);
29                  S.erase(S.begin());
30              }
31              r++;
32          }
```

右端不断右移
直到区间不能清零

跟着老师翻译理解每一行

```
36          do{
37              info x=(info){l,h[l]};
38              it=T.find(x);
39              if(it!=T.end()){
40                  cT-=h[l];
41                  T.erase(it);
42              }else{
43                  it=S.find(x);
44                  S.erase(it);
45                  if(!T.empty()){
46                      it=T.end(); it--;
47                      S.insert(*it);
48                      cT-=it->h;
49                      T.erase(it);
50                  }
51              }
52              l++;
53          }while(l<r&&cT>t);
```

左端不断右移
直到区间内
全部能清零

跟着老师翻译理解每一行