# 省选级别试题 第五组

# 数字游戏

### 题解

假设n个数中正数有x个，负数有(n-x)个，则积为正数的对数有x*(x-1)/2+(n-x)*(n-x-1)/2对，会发现当x最大或最小时上式将取到最大值。

因此二分求出x的最大值和最小值即可

### 标准代码

#### C++

```
#include<bits/stdc++.h>
using namespace std;
int n;
long long k;
long long cal(int x)
{
        int y=n-x;
        long long a=0,b=0;
        if(x)a=1ll*x*(x-1)/2;
        if(y)b=1ll*y*(y-1)/2;
        return a+b;
}
int main()
{
        scanf("%d%lld",&n,&k);
        if ( k > 1ll*n*(n-1)/2 ) {
                printf ( "-1\n" );return 0;
        }
        int l=0,r=n,ret1=-1;
        while(l<=r)
        {
                int mid=(l+r)>>1;
                if(1ll*mid*(mid-1)/2<=k)l=mid+1,ret1=mid;
                else r=mid-1;
        }
        l=0,r=n;int ret2=-1;
        while(l<=r)
        {
                int mid=(l+r)>>1;
```

```
            if(1ll*(2*n-mid-1)*mid/2>=k)r=mid-1,ret2=mid;
            else l=mid+1;
        }
//      printf("%d %d\n",ret1,ret2);
        long long ans=-1;
        if(ret1!=-1)ans=max(ans,cal(ret1));
        if(ret2!=-1)ans=max(ans,cal(ret2));
        printf("%lld\n",ans);
}
```

# 拼图王

### 题解

依次考虑每个字符串的所属，假设当前考虑到第 $i$ 个字符串，那么前 $i-1$ 个字符串已经被分成了两个子序列，显然 $a_{i-1}$ 是某个子序列的末尾，或者说每次考虑的最后一个字符串必然为一个子序列的末尾，假设另一个序列末尾为 $a_j$ ，若 $a_i$ 跟在 $a_{i-1}$ 后面，那么 $a_j$ 依旧是另一个子序列的末尾，若 $a_i$ 跟在 $a_j$ 后面，那么 $a_{i-1}$ 变成另一个序列末尾，故不需要考虑另一个序列具体是哪一个字符串，只需要知道另一个序列末尾的状态即可

以 $dp[i][j][k]$ 表示前 $i$ 个字符串分好后，不以 $a_i$ 为结尾的子序列后 $j$ 位状态为 $k$ 时 $S$ 的最小值，那么根据 $a_i$ 的所属有转移：

1.若 $a_i$ 跟在 $a_{i-1}$ 后面，那么另一个序列后 $j$ 位的状态都不会改变，此时对 $S$ 值的影响为 $a_{i-1}$ 接上 $a_i$ 所增加的串长，也即 $dp[i][j][k]+=len-deal(a_{i-1},a_i)$ ，其中 $len$ 为每个字符串的串长， $deal(x,y)$ 表示求出 $x$ 的后缀与 $y$ 的前缀最长公共部分

2.若 $a_i$ 跟在另一个序列后面，那么此时的所谓的另一个序列变成以 $a_{i-1}$ 结尾，那么其末尾 $j$ 位状态即为 $a_{i-1}$ 后 $j$ 位的状态，记为 $suf(a_{i-1},j)$ ，而此时对 $S$ 值的影响是将 $a_i$ 接在了另一个子序列上，枚举另一个子序列末尾与 $a_i$ 的公共部分长度 $k$ 有转移：

$$dp[i][j][suf(a_{i-1},j)]=\min_k(dp[i-1][k][pre(a_i,k)]+len-k)$$

其中 $pre(a_i,j)$ 为 $a_i$ 前 $j$ 位状态

此时转移的复杂度为 $O(n\cdot len\cdot 2^{len})$ ，但注意到，记 $temp=len-deal(a_{i-1},a_i)$ ，则第二种转移等价于

$$dp[i][j][suf(a_{i-1},j)]=\min_k(dp[i-1][k][pre(a_i,k)]+len-k-temp)+temp$$

如此每次从 $dp[i-1]$ 转移到 $dp[i]$ 都会加上 $temp$ 这一定值，若在转移中不考虑该定值，而是累加该值在转移结束后直接加到答案中，那么就不用进行第一种转移，此时复杂度为 $O(n\cdot len)$

## 标准代码

### C++ 11

```
#include<cstdio>
#include<iostream>
#include<cstring>
#include<algorithm>
#include<cmath>
#include<vector>
#include<queue>
#include<map>
#include<set>
#include<ctime>
using namespace std;
typedef long long ll;
typedef pair<int,int>P;
const int INF=0x3f3f3f3f,maxn=200005;
int n,len,a[maxn],dp[21][1<<20];
char s[21];
int pre(int S,int i)
{
    return S>>(len-i);
}
int suf(int S,int i)
{
    return S&((1<<i)-1);
}
int deal(int S,int T)
{
    for(int i=len;i;i--)
        if(suf(S,i)==pre(T,i))return i;
    return 0;
}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%s",s);
        len=strlen(s);
        for(int j=0;j<len;j++)a[i]=2*a[i]+s[j]-'0';
    }
    int res=0;
    memset(dp,INF,sizeof(dp));
    dp[0][0]=len;
    for(int i=2;i<=n;i++)
    {
        int l=len-deal(a[i-1],a[i]);
        res+=l;
```

```
        int mn=INF;
        for(int j=0;j<=len;j++)mn=min(mn,dp[j][pre(a[i],j)]+len-j-1);
        //另一个序列的末尾接上a[i],a[i-1]变成另一个序列末尾
        for(int j=0;j<=len;j++)dp[j][suf(a[i-1],j)]=min(dp[j][suf(a[i-1],j)],mn);
    }
    printf("%d\n",dp[0][0]+res);
    return 0;
}
```

# 中位数之中位数

### 题解

答案的单调性是显然的，所以可以二分答案，把最值问题转化为判定性问题。

现在要求的就是：满足区间的中位数不超过 $x$ 的区间数量。（ $x$ 为我们二分的值）

定义一个 $p$ 数组，满足 $p_i = p_{i-1} + [a_i > x]$

说白了就是求出：前 $i$ 个数中有多少个超过了 $x$

那么如果一个序列满足条件，就可以转化为满足这个式子：

$$r - l > 2 \times (p_r - p_l)$$

注意这里必须是严格大于。

这个式子的意思就是：这个区间中，大于x的数不会达到一半。

但这个式子跟两个位置的值有关，所以需要转化一下：

$$r - 2 \times p_r > l - 2 \times p_l$$

这样两边都只跟一个位置的值有关了。

所以可以用一个平衡树，存储在每个位置的 $i - 2 \times p_i$ ，在它前面找小于它的值的个数，就是以i为右端点，且满足条件的区间数。

### 标准代码

### C++ 11

```
#include<cstdio>
#include<cstring>
```

```
#include<cmath>
#include<algorithm>
#include<vector>
#include<set>
#define SF scanf
#define PF printf
#define MAXN 100010
#define INF 0x3FFFFFFF
using namespace std;
typedef long long ll;
int a[MAXN],p[MAXN];
ll n;
struct node *NIL;
struct node{
    node *ch[2],*fa;
    int val,sum,num;
    bool Dir(){
        return this==fa->ch[1];
    }
    void setchild(node *x,int d){
        ch[d]=x;
        if(x!=NIL)
            x->fa=this;
    }
    void pushup(){
        //pushdown();
        //if(x->ch[0]!=NIL)x->ch[0]->pushdown();
        //if(x->ch[1]!=NIL)x->ch[1]->pushdown();
        sum=ch[0]->sum+ch[1]->sum+num;
    }
}tree[MAXN],*root,*ncnt;
node * Newnode(node *x,int val){
    x->ch[0]=x->ch[1]=x->fa=NIL;
    x->val=val;
    x->sum=x->num=1;
    return x;
}
void Rotate(node *x){
    node *y=x->fa;
    //y->pushdown(),x->pushdown();
    int d=x->Dir();
    if(y==root)
        root=x,x->fa=NIL;
    else
        y->fa->setchild(x,y->Dir());
    y->setchild(x->ch[!d],d);
    x->setchild(y,!d);
    y->pushup();
}
void Splay(node *x,node *rt){
    //x->pushdown();
```

```
    while(x->fa!=rt){
        node *y=x->fa;
        if(y->fa==rt){
            Rotate(x);
            break;
        }
        if(x->Dir()==y->Dir())
            Rotate(y);
        else
            Rotate(x);
        Rotate(x);
    }
    x->pushup();
}
void Ins(node *&root,int val){
    if(root==NIL){
        root=Newnode(++ncnt,val);
        return ;
    }
    node *y=root;
    int d;
    while(1){
        y->sum++;
        if(y->val==val){
            y->num++;
            Splay(y,NIL);
            return ;
        }
        d=(y->val)<val;
        if(y->ch[d]==NIL)
            break;
        y=y->ch[d];
    }
    y->setchild(Newnode(++ncnt,val),d);
    Splay(y->ch[d],NIL);
}
node *find(node *x,int val){
    if(x->val==val)
        return x;
    return find(x->ch[(x->val)<val],val);
}
ll check(int x){
    ncnt=root=NIL;
    for(int i=1;i<=n;i++)
        p[i]=0;
    for(int i=1;i<=n;i++)
        p[i]=p[i-1]+(a[i]>x)*2;
    ll res=0;
    Ins(root,0);
    for(int i=1;i<=n;i++){
        Ins(root,i-p[i]);
```

```
        node *x=find(root,i-p[i]);
        Splay(x,NIL);
        res+=x->ch[0]->sum;
    }
    return res;
}
int main(){
    NIL=&tree[0];
    NIL->ch[0]=NIL->ch[1]=NIL->fa=NIL;
    root=ncnt=NIL;
    SF("%lld",&n);
    for(int i=1;i<=n;i++)
        SF("%d",&a[i]);
    int l=1,r=INF,ans=INF;
    while(l<=r){
        int mid=(l+r)>>1;
        if(check(mid)<(n*(n+1ll)/4ll+1ll))
            l=mid+1;
        else{
            ans=mid;
            r=mid-1;
        }
    }
    PF("%d",ans);
}
```