2020.08.06codeforces加训

比赛情况

题号	A	B	C	D	E	F	G	H	I	J
状态	O	-	O	-	-	-	O	Ø	O	O

O 在比赛中通过 Ø 赛后通过 ! 尝试了但是失败了 - 没有尝试

比赛时间

2020-08-06 12:00-17:00

题解

A - Hacker Cups and Balls

题意：

给定一个序列，给q次操作，每次操作升序或降序排列某区间中的数，问q次操作后序列中间位置的数是什么。

题解：

我们可以考虑二分答案，然后把所有大于当前二分出答案的数字赋值为1，其余的赋值为0，这样一次排序操作其实就是查询一个区间有多少个0多少个1，然后把前后两部分分别连续赋值为0和1，这样一个操作通过线段树就能完成。

然后查询中间位置的数字是0还是1，如果是1那么中位数就应该是更大，如果为0那么中位数就可能是当前答案或更小。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5+5;
int tr[N<<2],lazy[N<<2],a[N];
struct Node {
    int l,r;
}q[N];
int midans,n,m;
void Push_Down(int p,int l,int r) {
    if (lazy[p]==-1||l==r)return;
    int mid = (l+r)>>1;
    tr[p<<1] = (mid-l+1)*lazy[p];
    tr[p<<1|1] = (r-mid)*lazy[p];
    lazy[p<<1] = lazy[p<<1|1] = lazy[p];
    lazy[p] = -1;
}
void Build(int p,int l,int r) {
```

```
    lazy[p] = -1;
    if (l==r) {
        tr[p] = a[l] > midans;
        return;
    }
    int mid = (l+r)>>1;
    Build(p<<1,l,mid);
    Build(p<<1|1,mid+1,r);
    tr[p] = tr[p<<1]+tr[p<<1|1];
}
void Update(int p,int l,int r,int a,int b,int c) {
    if (a > b) return;
    Push_Down(p,l,r);
    if (l >= a && r <= b) {
        tr[p] = (r-l+1)*c;
        lazy[p] = c;
        return;
    }
    int mid = (l+r)>>1;
    if (a <= mid) Update(p<<1,l,mid,a,b,c);
    if (b > mid) Update(p<<1|1,mid+1,r,a,b,c);
    tr[p] = tr[p<<1] + tr[p<<1|1];
}
int Getans(int p,int l,int r,int a,int b) {
    Push_Down(p,l,r);
    if (l >= a && r <= b) return tr[p];
    int ans = 0;
    int mid = (l+r)>>1;
    if (a <= mid) ans+= Getans(p<<1,l,mid,a,b);
    if (b > mid) ans+= Getans(p<<1|1,mid+1,r,a,b);
    return ans;
}
bool check(int x) {
    midans = x;
    Build(1,1,n);
    for (int i = 1;i<= m;i++) {
        int L = min(q[i].l,q[i].r);
        int R = max(q[i].l,q[i].r);
        int tmp = Getans(1,1,n,L,R);
        if (q[i].l < q[i].r) {
            Update(1,1,n,L,R-tmp,0);
            Update(1,1,n,R-tmp+1,R,1);
        } else {
            Update(1,1,n,L,L+tmp-1,1);
            Update(1,1,n,L+tmp,R,0);
        }
    }
    int midpos = (n>>1)+1;
    int t = Getans(1,1,n,midpos,midpos);
    return t == 1;
```

```
}
int GetFinalans() {
    int l = 1,r = n+1;
    while (l < r) {
        int mid = (l+r)>>1;
        if (check(mid))l = mid+1;
        else r = mid;
    }
    return l;
}
int main()
{
    scanf("%d%d",&n,&m);
    for (int i = 1;i<= n;i++)scanf("%d",&a[i]);
    for (int i = 1;i<= m;i++)
        scanf("%d%d",&q[i].l,&q[i].r);
    printf("%d\n",GetFinalans());
    return 0;
}
```

C. Crazy Dreamoon

给出若干线段问内部有线段的网格有多少个。

扫描线比较暴力地搞一下就好。

```
#include<bits/stdc++.h>
#define ll long long
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pii pair<int,int>
#define pdd pair<double,double>
using namespace std;
const int N=2010;
const double eps=1e-10;
double sqr(double x){return x*x;}
int dcmp(double x){return (x<-eps)?-1:(x>eps);}
int vis[N][N];
struct Point
{
    double x,y;
    Point(double x=0,double y=0):x(x),y(y){};
    Point operator - (Point a){return Point(x-a.x,y-a.y);}
    double operator * (Point a){return x*a.x+y*a.y;}
    void init(){scanf("%lf%lf",&x,&y);}
```

```
}p1,p2;
int _ceil(double x){int y=x-1;while(dcmp(y-x)<0)y++;return y;}
int _floor(double x){int y=x+1;while(dcmp(y-x)>0)y--;return y;}
int main()
{
    int n,res=0;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        p1.init(),p2.init();
        if(p1.x>p2.x)swap(p1,p2);
        if(dcmp(p2.x-p1.x)==0)continue;
        for(int j=_floor(p1.x);j<=_ceil(p2.x);j++)
        {
            if(dcmp(j-p1.x)<0)continue;
            if(dcmp(j+1-p2.x)>0)continue;
            double y1=(p2.y-p1.y)*(j-p1.x)/(p2.x-p1.x)+p1.y,y2=(p2.y-
p1.y)*(j+1-p1.x)/(p2.x-p1.x)+p1.y;
            if(y1>y2)swap(y1,y2);
            int yy1=_floor(y1),yy2=_ceil(y2);
            for(int k=yy1;k<yy2;k++)vis[j][k]=1;
        }
    }
    for(int i=0;i<N;i++)
    for(int j=0;j<N;j++)res+=vis[i][j];
    printf("%d\n",res);
    return 0;
}
```

G. Dreamoon and NightMarket

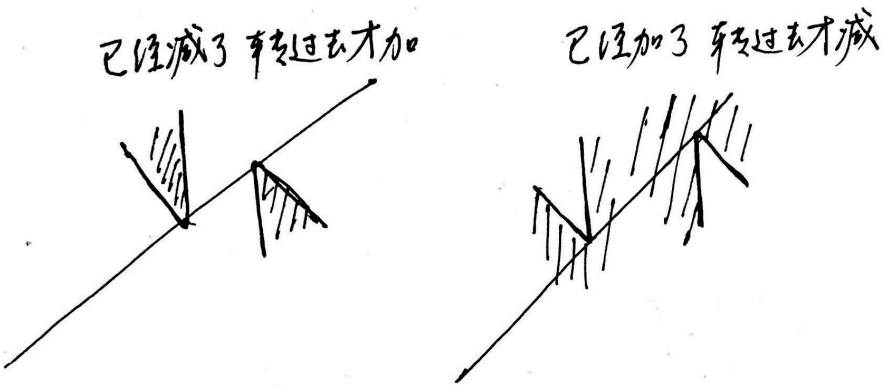
问和第 k 小的集合。

维护一个优先队列，排序后先把最小的数加入，存pair记录加和与当前所用到最大的数是第几个，每次取出时加上下一个数放入，或去掉当前数再加上下一个放入。

H. Split Game

过原点一直线最多能将多边形分成多少块。

因为是过原点可以想到极角排序，然后扫描线，只有在经过某些顶点时分块数才会改变。然后比较重要的是要区分转到顶点时的改变和转过之后才改变的改变。改变主要分为以下情况（特殊的是边与直线重合，但撕烤一下可以被处理方式兼容，详见代码）



代码里记录的是与直线相交的点数，块数 $\frac{\mathrm{cnt}}2+1$

```
#include<bits/stdc++.h>
#define ll long long
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define pii pair<int,int>
#define pll pair<ll,ll>
using namespace std;
const int N=2e5+10;
const double eps=1e-10;
double sqr(double x){return x*x;}
int sig(ll x){return x<0?-1:x>0;}
int n,a[N];
struct Point
{
    ll x,y;
    Point(int x=0,int y=0):x(x),y(y){};
    void init(){scanf("%lld%lld",&x,&y);}
}p[N],o=Point(0,0);
ll Cross(Point p0,Point p1,Point p2){return (p1.x-p0.x)*(p2.y-p0.y)-(p1.y-
p0.y)*(p2.x-p0.x);}
bool cmp2(Point o1,Point o2){return Cross(o,o1,o2)==0;}
bool cmp1(Point p1,Point p2){return p1.y*1.0/p1.x<p2.y*1.0/p2.x;}
int main()
{
    scanf("%d",&n);
    for(int i=0;i<n;i++)p[i].init(),a[i]=i;
    sort(a,a+n,[&](int x,int y){return cmp1(p[x],p[y]);});
    int cnt=0,res=1;
    for(int i=0;i<n;i++)
    {
        int t=0;
        Point p0=p[a[i]],p1=p[(a[i]-1+n)%n],p2=p[(a[i]+1)%n];
        int u=sig(Cross(o,p0,p1)),v=sig(Cross(o,p0,p2));
        int w=sig(Cross(p0,p1,p2));
```

```
        if(w==-1)if(u+v>0)t+=u+v;else cnt+=u+v;
        else if(w==1)if(u+v>0)cnt+=u+v;else t+=u+v;
        while(i+1<n&&cmp2(p[a[i]],p[a[i+1]]))
        {
            i++;
            Point p0=p[a[i]],p1=p[(a[i]-1+n)%n],p2=p[(a[i]+1)%n];
            int u=sig(Cross(o,p0,p1)),v=sig(Cross(o,p0,p2));
            int w=sig(Cross(p0,p1,p2));
            if(w==-1)if(u+v>0)t+=u+v;else cnt+=u+v;
            else if(w==1)if(u+v>0)cnt+=u+v;else t+=u+v;
        }
        res=max(cnt/2+1,res);
        cnt+=t,res=max(cnt/2+1,res);
    }
    printf("%d\n",res);
    return 0;
}
```

J - Zero Game

题意：

给定一个01序列，定义一次操作是把任意一个字符换到任意一个位置，有q次询问，第i次询问问k_i次操作后这个序列最长可以有多长的连续0

题解:

考虑只对于一个区间内的字符进行操作的答案，因为有两种可能，一种是把1换走使得两个连续的0序列合并，另一种是换进来一些0，那么我们可以发现对于一个区间(l,r)如果$sum_r-sum_{l-1}\leq k_i$,那么这个区间的答案应该是$l+r-1-(sum_r-sum_{l-1})+(k_i-(sum_r-sum_{l-1}))$$sum_i$为从位置$1$到$i$之间有多少个1），那么整理一下，这个式子变成了$r-2\times sum_r + 2\times sum_{l-1}-(l-1)+k_i$对于同一个$r$我们只需要找一个满足$sum_r-sum_{l-1}$的l中$2\times sum_{l-1}-(l-1)$最大的即可，一开始觉得是数据结构，后来发现单调队列维护就行了。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6+5;
char s[N];
int sum[N];
int lsum[N];
int k[N],id[N],ans[N],que[N],head,tail;
int main()
{
    int n,q;
    scanf("%s",s+1);
    n = strlen(s+1);
    scanf("%d",&q);
```

```
    for (int i = 1;i<= q;i++)
    {
        scanf("%d",&k[i]);
        id[i] = k[i];
    }
    sort(id+1,id+q+1);
    int cnt = 0;
    for (int i = 1;i<= q;i++)
        if (id[i]!=id[cnt])
            id[++cnt] = id[i];
    for (int i = 1;i<= n;i++)
    {
        sum[i] = sum[i-1]+(s[i]=='1');
        lsum[i] = 2*sum[i]-i;
    }
    for (int i = 1;i<= cnt;i++)
    {
        int K = id[i];
        head = tail = 1;
        que[1] = 0;
        ans[K] = ans[id[i-1]];
        for (int j = 1;j<= n;j++) {
            while (head <= tail && sum[j]-sum[que[head]] > K)head++;
            if (head<=tail)ans[K] =
max(ans[K],lsum[que[head]]+j-2*sum[j]+K);
            while (head <= tail && lsum[j]>=lsum[que[tail]])tail--;
            que[++tail] = j;
        }
        ans[K] = min(ans[K],n-sum[n]);
    }
    for (int i = 1;i<= q;i++)
        printf("%d\n",ans[k[i]]);
    return 0;
}
```

比赛总结与反思