2020-2021 BUAA ICPC Team Supplementary Training 02

比赛网址

训练结果

- 时间:‘ ’2020/8/6’ ’
- rank:‘’5/18’’
- 完成情况：‘ ’6/8/10’ ’

题解

A. Hacker Cups and Balls

题意

给了一个数列,每次操作一个区间,把里面的数顺序或逆序排序.求最后位置在最中间的数的值

题解

我们可以二分答案,小于他的数设为零,大于等于它的数设为一.区间操作就变成了零一的排序,这个用线段树就可以实现.答案确实可以保证二分的性质.

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#define ll long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
const int N=100055;
int n,m,a[N],b[N],c[N];
int sum[N<<2],la[N<<2];
#define lson k<<1,l,mid
#define rson k<<1|1,mid+1,r
void pu(int k)
```

```
{
    sum[k]=sum[k<<1]+sum[k<<1|1];
}
void pd(int k,int l,int r)
{
    if(!la[k])
    {
        la[k<<1]=la[k<<1|1]=sum[k<<1]=sum[k<<1|1]=0;
        la[k]=-1;
    }
    else if(la[k]==1)
    {
        la[k<<1]=la[k<<1|1]=1;
        la[k]=-1;
        int mid=l+r>>1;
        sum[k<<1]=mid-l+1;
        sum[k<<1|1]=r-mid;
    }
}
void build(int k,int l,int r,int x)
{
    la[k]=-1;
    if(l==r) {sum[k]=(a[l]>=x);return;}
    int mid=l+r>>1;
    build(lson,x);build(rson,x);
    pu(k);
}
void ch(int k,int l,int r,int a,int b,int v)
{
    if(a>b) return;
    if(a<=l&&b>=r)
    {
        if(v==0) sum[k]=la[k]=0;
        else sum[k]=r-l+1,la[k]=1;
        return;
    }
    int mid=l+r>>1;pd(k,l,r);
    if(a<=mid) ch(lson,a,b,v);
    if(b>mid) ch(rson,a,b,v);
    pu(k);
}
int query(int k,int l,int r,int a,int b)
{
    if(a<=l&&b>=r) return sum[k];
    int mid=l+r>>1,res=0;pd(k,l,r);
    if(a<=mid) res=query(lson,a,b);
    if(b>mid) res+=query(rson,a,b);
    return res;
}
int chk(int x)
```

```
{
    build(1,1,n,x);
    for(int i=1;i<=m;i++)
    {
        if(b[i]<c[i])
        {
            int s=query(1,1,n,b[i],c[i]);
            ch(1,1,n,c[i]-s+1,c[i],1);
            ch(1,1,n,b[i],c[i]-s,0);
        }
        else if(b[i]>c[i])
        {
            int s=query(1,1,n,c[i],b[i]);
            ch(1,1,n,c[i],c[i]+s-1,1);
            ch(1,1,n,c[i]+s,b[i],0);
        }
    }
//  cout<<x<<endl;
//  for(int i=1;i<=n;i++)
//      cout<<query(1,1,n,i,i)<<endl;
    return query(1,1,n,(n+1)/2,(n+1)/2);
}
int main()
{
    n=read();m=read();
    for(int i=1;i<=n;i++)
        a[i]=read();
    for(int i=1;i<=m;i++)
        b[i]=read(),c[i]=read();
    int l=1,r=n,mid,res=n;
    while(r>=l)
    {
        mid=l+r>>1;
        if(chk(mid)) res=mid,l=mid+1;
        else r=mid-1;
    }
    cout<<res<<endl;
    return 0;
}
```

C. Crazy Dreamoon

题意

一个2000*2000矩形上有n条线段，问这n条线段覆盖了多少格点。

题解

直接走一遍每个线段，依次打标记即可（注意整点附近的精度处理）

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=2010;
const double eps=1e-6;
int n,X1,Y1,X2,Y2,ans;
bool have[maxn][maxn];
double f(double x)
{
    return (double)((Y2-Y1)*x+X2*Y1-X1*Y2)/(double)(X2-X1);
}
int main()
{
    n=read();
    for(int i=1;i<=n;i++)
    {
        X1=read();Y1=read();X2=read();Y2=read();
        if(X1==X2)continue;
        if(Y1==Y2)continue;
        if(X1>X2)swap(X1,X2),swap(Y1,Y2);
        for(int x=X1;x<X2;x++)
        {
            double f1=f((double)x),f2=f((double)(x+1.0));
            if(f1>f2)swap(f1,f2);
            if(fabs(f1-ceil(f1))<eps)
            {
                f1=(double)ceil(f1);
            }
            if(fabs(f2-ceil(f2))<eps)
            {
                for(int y=floor(f1);y<floor(f2);y++)have[x][y]=1;
                continue;
            }
            for(int y=floor(f1);y<=floor(f2);y++)have[x][y]=1;
```

```
        }
    }
    for(int i=0;i<=2000;i++)for(int j=0;j<=2000;j++)if(have[i][j])ans++ ;
    printf("%d\n",ans);
    return 0;
}
```

D.Forest Game

题意

现在有一棵n个节点的树，每次从中删去一个点，得到这个点所在树的大小的代价。问给定一棵树随机删除的所有情况代价和

题解

每一种删除方案对应一种排列 我们考虑每个点对的贡献，每个点对中一个点对另一个点产生贡献当且仅当另一个点是这两点之间所有点中第一个删除的，那么距离为$m-1$的点对(即路径上有m个点)产生的代价为$2 \times C_{n}^{m}(m-1)!(n-m)!$

那么我们只需要计算每种距离的点对有几个

使用点分治+fft统计

```
#include<algorithm>
#include<iostream>
#include<cstring>
#include<cstdio>
#include<cmath>
#include<map>
#define Redge(u) for (int k = h[u],to; k; k = ed[k].nxt)
#define REP(i,n) for (int i = 1; i <= (n); i++)
#define mp(a,b) make_pair<int,int>(a,b)
#define cls(s) memset(s,0,sizeof(s))
#define cp pair<int,int>
#define LL long long int
using namespace std;
const int maxn = 400005,maxm = 100005,INF = 1000000000;
inline int read(){
    int out = 0,flag = 1; char c = getchar();
    while (c < 48 || c > 57){if (c == '-') flag = -1; c = getchar();}
    while (c >= 48 && c <= 57){out = (out << 3) + (out << 1) + c - 48; c =
getchar();}
    return out * flag;
}
int n,h[maxn],ne = 1;
struct EDGE{int to,nxt;}ed[maxn];
```

```
inline void build(int u,int v){
    ed[++ne] = (EDGE){v,h[u]}; h[u] = ne;
    ed[++ne] = (EDGE){u,h[v]}; h[v] = ne;
}
struct E{
    double a,b;
    E(){}
    E(double x,double y):a(x),b(y) {}
    E(int x,int y):a(x),b(y) {}
    inline E operator =(const int& b){
        this->a = b; this->b = 0;
        return *this;
    }
    inline E operator =(const double& b){
        this->a = b; this->b = 0;
        return *this;
    }
    inline E operator /=(const double& b){
        this->a /= b; this->b /= b;
        return *this;
    }
};
inline E operator *(const E& a,const E& b){
    return E(a.a * b.a - a.b * b.b,a.a * b.b + a.b * b.a);
}
inline E operator *=(E& a,const E& b){
    return a = E(a.a * b.a - a.b * b.b,a.a * b.b + a.b * b.a);
}
inline E operator +(const E& a,const E& b){
    return E(a.a + b.a,a.b + b.b);
}
inline E operator -(const E& a,const E& b){
    return E(a.a - b.a,a.b - b.b);
}
const double pi = acos(-1);
int R[maxn];
void fft(E* a,int n,int f){
    for (int i = 0; i < n; i++) if (i < R[i]) swap(a[i],a[R[i]]);
    for (int i = 1; i < n; i <<= 1){
        E wn(cos(pi / i),f * sin(pi / i));
        for (int j = 0; j < n; j+= (i << 1)){
            E w(1,0);
            for (int k = 0; k < i; k++,w *= wn){
                E x = a[j + k],y = w * a[j + k + i];
                a[j + k] = x + y; a[j + k + i] = x - y;
            }
        }
    }
    if (f == -1) for (int i = 0; i < n; i++) a[i] /= n;
}
```

```
LL ans[maxn];
int F[maxn],fa[maxn],siz[maxn],vis[maxn],N,rt;
void getrt(int u){
    siz[u] = 1; F[u] = 0;
    Redge(u) if (!vis[to = ed[k].to] && to != fa[u]){
        fa[to] = u; getrt(to);
        siz[u] += siz[to];
        F[u] = max(F[u],siz[to]);
    }
    F[u] = max(F[u],N - siz[u]);
    if (F[u] < F[rt]) rt = u;
}
int dep[maxn],md;
E A[maxn],B[maxn];
void dfs(int u){
    A[dep[u]].a+=1; siz[u] = 1; md = max(md,dep[u]);
    Redge(u) if (!vis[to = ed[k].to] && to != fa[u]){
        fa[to] = u; dep[to] = dep[u] + 1; dfs(to);
        siz[u] += siz[to];
    }
}
void dfs1(int u){
    B[dep[u]].a+=1; md = max(md,dep[u]);
    Redge(u) if (!vis[to = ed[k].to] && to != fa[u])
        dfs1(to);
}
void solve(int u){
    vis[u] = true; siz[u] = N; fa[u] = 0;
    for (int i = 0; i <= N; i++) A[i] = B[i] = 0;
    dep[u] = 0; A[0] = 1; md = 0;
    Redge(u) if (!vis[to = ed[k].to]){
        fa[to] = u; dep[to] = 1; dfs(to);
    }
    int m = (md << 1),L = 0,n = 1;
    while (n <= m) n <<= 1,L++;
    for (int i = 1; i < n; i++) R[i] = (R[i >> 1] >> 1) | ((i & 1) << (L -
1));
    for (int i = md + 1; i < n; i++) A[i] = 0;
    fft(A,n,1);
    for (int i = 0; i < n; i++) A[i] *= A[i];
    fft(A,n,-1);
    for (int i = 0; i < n; i++) ans[i + 1] += (LL)(A[i].a + 0.1);
    Redge(u) if (!vis[to = ed[k].to]){
        md = 1; dfs1(to);
        m = (md << 1),L = 0,n = 1;
        while (n <= m) n <<= 1,L++;
        for (int i = 1; i < n; i++) R[i] = (R[i >> 1] >> 1) | ((i & 1) <<
(L - 1));
        fft(B,n,1);
        for (int i = 0; i < n; i++) B[i] *= B[i];
        fft(B,n,-1);
```

```
        for (int i = 0; i < n; i++) ans[i + 1] -= (LL)(B[i].a + 0.1);
        for (int i = 0; i < n; i++) B[i] = 0;
    }
    Redge(u) if (!vis[to = ed[k].to]){
        N = siz[to]; F[rt = 0] = INF; getrt(to);
        solve(rt);
    }
}
const int M = 1000000007;
int Qpow(int a,int b){
    int ans = 1;
    for (; b; b >>= 1,a = 1ll * a * a % M)
        if (b & 1) ans = 1ll * ans * a % M;
    return ans;
}
int f[maxn],fv[maxn];
int C(int x,int y){
    return 1ll * f[x] * fv[y] % M * fv[x - y] % M;
}
void calwork(){
    f[0] = 1;
    for (int i = 1; i <= n; i++) f[i] = 1ll * f[i - 1] * i % M;
    fv[0] = 1;
    for (int i = 1; i <= n; i++) fv[i] = Qpow(f[i],M - 2);
    int Ans = 1ll * f[n] * n % M;
    for (int i = 1; i <= n; i++) ans[i] = (ans[i] / 2) % M;
    //REP(i,n) printf("dis %d  cnt %lld\n",i,ans[i]);
    //cout << C(3,2) << endl;
    for (int i = 2; i <= n; i++){
        if (!ans[i]) continue;
        Ans = (Ans + 2ll * C(n,i) % M * f[i - 1] % M * f[n - i] % M *
ans[i] % M) % M;
    }
    printf("%d\n",Ans);
}
int main(){
    n = read();
    for (int i = 1; i < n; i++) build(read(),read());
    F[rt = 0] = INF; N = n; getrt(1);
    solve(rt);
    calwork();
    return 0;
}
```

E. Lines Game

题意

有 n 条线段,端点为 $(0,i)$,$(1,p_i)$ 每次可以花 v_i 的价值选线段 i ,把 i 和 与 i 相交的线段全部删了, 问删完所有线段的最小代价.

题解

如果存在 $i<j$,$p_i>pj$ 那么选 j 就一定不会选 i ,因为 i 能删的线段 j 一定能删.

所以答案的 p_i 一定是单调的.我们可以推出一个dp 方程 $dp_j= dp_i+v_j$ 其中的 i 必须满足所有的 $ i < k < j , p_k>p_j || p_k<p_i$

发现这个dp可以用cdq分治优化,把区间分成两段,我们左右两边按 p_i 排序,左边维护编号单调递减的单调栈, 因为如果出现 $i<j$,$p_i<p_j$ i 就不能用了更新答案了,因为选 i 就删不了 j

右边维护编号单调递增的单调栈, 因为如果出现 $i>j$,$p_i<p_j$ 选 j 就一定会删掉 i 这样右边对于 i , 他能选到左边的线段 p 的范围就是右边栈顶的位置加一到 p_i ,用线段树维护左边的dp值并区间查询答案就行.

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#define ll long long
#define int long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
const int N=100055,inf=1ll<<50;
int n,m,a[N],b[N],c[N],v[N],f[N];
int s[N],s2[N];
int mn[N<<2];
bool cmp(int x,int y)
{
    return a[x]<a[y];
}
#define lson k<<1,l,mid
#define rson k<<1|1,mid+1,r
void pu(int k)
{
    mn[k]=min(mn[k<<1],mn[k<<1|1]);
}
void build(int k,int l,int r)
{
    if(l==r) {mn[k]=inf;return;}
```

```
    int mid=l+r>>1;
    build(lson);build(rson);
    pu(k);
}
void ch(int k,int l,int r,int a,int b)
{
    if(l==r) {mn[k]=b;return;}
    int mid=l+r>>1;
    if(a<=mid) ch(lson,a,b);
    else ch(rson,a,b);
    pu(k);
}
int query(int k,int l,int r,int a,int b)
{
    if(a<=l&&b>=r) return mn[k];
    int mid=l+r>>1,res=inf;
    if(a<=mid) res=query(lson,a,b);
    if(b>mid) res=min(res,query(rson,a,b));
    pu(k);
    return res;
}
void solve(int l,int r)
{
    if(l==r) return;
    int mid=l+r>>1,tp=0,tp2=0,l1=mid-l+1,l2=r-mid;
    solve(l,mid);
    for(int i=1;i<=l1;i++)
        b[i]=l+i-1;
    for(int i=1;i<=l2;i++)
        c[i]=mid+i;
    sort(b+1,b+1+l1,cmp);sort(c+1,c+1+l2,cmp);
    int now=1;
    for(int i=1;i<=l2;i++)
    {
        while(now<=l1&&a[b[now]]<a[c[i]])
        {
            while(tp&&s[tp]<b[now])
                ch(1,0,n,a[s[tp]],inf),tp--;
            ch(1,0,n,a[s[++tp]=b[now]],f[b[now]]);
            now++;
        }
        while(tp2&&s2[tp2]>c[i]) tp2--;
        int pos=0;
        if(tp2) pos=a[s2[tp2]]+1;
        f[c[i]]=min(f[c[i]],v[c[i]]+query(1,0,n,pos,a[c[i]]));
        s2[++tp2]=c[i];
    }
    for(int i=1;i<=l1;i++)
        ch(1,0,n,a[b[i]],inf);
    solve(mid+1,r);
```

```
}
main()
{
    n=read()+1;a[n]=n;
    for(int i=1;i<=n;i++)
        f[i]=inf;
    for(int i=1;i<n;i++)
        a[i]=read();
    for(int i=1;i<n;i++)
        v[i]=read();
    build(1,0,n);
    solve(0,n);
//  for(int i=1;i<=n;i++)
//      cout<<f[i]<<endl;
    cout<<f[n]<<endl;
    return 0;
}
```

G. Dreamoon and NightMarket

题意

给了 n 个有价值的物品,求价值第 k 小的集合.

题解

我们用优先队列存已有的集合,每次我们取出最小的集合,设该集合价值和为 v ,最大价值的物品为 a_i ,我们就加入 $v-a_i+a_{i+1}$ 和 $v+a_{i+1}$,可以保证队列里面的集合一定有最小的.

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<queue>
#define ll long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
const int N=200055;
int n,m,a[N];
typedef pair<ll,int> P;
priority_queue<P, vector<P>, greater<P> > q;
```

```
int main()
{
    n=read();m=read();
    for(int i=1;i<=n;i++)
        a[i]=read();
    sort(a+1,a+1+n);
    q.push(P(a[1],1));
    for(int i=1;i<m;i++)
    {
        P k=q.top();q.pop();
        if(k.second<n)
        {
            q.push(P(k.first+a[k.second+1],k.second+1));
            q.push(P(k.first-a[k.second]+a[k.second+1],k.second+1));
        }
    }
    cout<<(q.top()).first<<endl;
    return 0;
}
```

F. Lonely Dreamoon 2

题意

调整一个序列的顺序，使得$min\{|a_i-a_{i-1}|\}$最大。

题解

分奇偶讨论，偶数是a_i和$a_{i+\frac{n}{2}}$相邻，奇数就找一个$|a_i-a_{i+\frac{n}{2}}|$最小的位置拿出来。

H. Split Game

题意

给一个多边形，全在第一象限，有一条过原点的直线，问最多能把这个多边形划分成多少区域

题解

先考虑给定一条分界线怎么数区域，我的做法是先算出所有交点然后看这条线两侧有多少个山峰，便是多少个区域，我们便可以从这个思路继续拓展，继续想直线在旋转的过程中答案的增量，十分善良的是数据已经是按照逆时针转好的，注意讨论这个点前驱后继组成的形状。

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=100010;
struct Point{
    LL x,y;
    Point() {}
    Point(double _1,double _2):x(_1),y(_2) {}
    Point operator - (const Point&s) const {return Point(x-s.x,y-s.y);}
    LL operator * (const Point&s) const {return x*s.y-y*s.x;}
    LL len() {return x*x+y*y;}
}a[maxn];
int n,N,ans,now_ans,R[maxn];
bool del[maxn];
bool cmp(int xx,int yy)
{
    return (double)a[xx].y/(double)a[xx].x<(double)a[yy].y/(double)a[yy].x;
}
int main()
{
    n=read();
    for(int i=1;i<=n;i++)scanf("%lld%lld",&a[i].x,&a[i].y);
    a[0]=a[n];a[n+1]=a[1];
    for(int i=1;i<=n;i++)if((a[i]-a[i+1])*(a[i]-a[i-1])==0)del[i]=1;
    for(int i=1;i<=n;i++)if(!del[i])a[++N]=a[i];
    a[0]=a[n];a[N+1]=a[1];
    n=N;
    for(int i=1;i<=n;i++)R[i]=i;
    sort(R+1,R+N+1,cmp);
    int i=1;
    while(i<=n)
    {
        int tmp=0,j;
        for(j=i;j<=n && a[R[i]]*a[R[j]]==0;j++)
        {
            int pos=R[j];
            if(a[pos]*a[pos-1]>=0 && a[pos]*a[pos+1]>0)
            {
                if((a[pos-1]-a[pos])*(a[pos+1]-a[pos])>0)now_ans++;
```

```
                else tmp++;
            }
            if(a[pos]*a[pos-1]<0 && a[pos]*a[pos+1]<=0)
            {
                if((a[pos-1]-a[pos])*(a[pos+1]-a[pos])<0)now_ans--;
                else tmp--;
            }
        }
        if(tmp>0)ans=max(ans,now_ans+tmp);
        else ans=max(ans,now_ans);
        now_ans+=tmp;
        i=j;
    }
    printf("%d\n",ans+1);
    return 0;
}
```

J. Zero Game

题意

有一个01串，允许进行至多K次操作，每次移动一个数的位置，问能构造出最长的连续0的长度

题解

把原串中连续0和1统计一下长度，那么序列就变成了一个0和1交错的序列，我们无视开头的1，每一串0和下一串1视作放在一个位置，那么记$A[i]$为0的前缀和$B[i]$为1的前缀和，
设$f[i]$为i位置的0串结尾的最长长度

那么对于给定的K有以下转移方程：

$f[i]=max\{A[i]-A[j]+K-(B[i-1]-B[j])\}$

用单调队列转移即可

复杂度$O(NQ)$

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<cstdio>
#include<vector>
#include<queue>
#include<cmath>
#include<map>
```

```
#include<set>
#define LL long long int
#define REP(i,n) for (int i = 1; i <= (n); i++)
#define Redge(u) for (int k = h[u],to; k; k = ed[k].nxt)
#define cls(s,v) memset(s,v,sizeof(s))
#define mp(a,b) make_pair<int,int>(a,b)
#define cp pair<int,int>
using namespace std;
const int maxn = 1000005,maxm = 100005,INF = 0x3f3f3f3f;
inline int read(){
    int out = 0,flag = 1; char c = getchar();
    while (c < 48 || c > 57){if (c == '-') flag = 0; c = getchar();}
    while (c >= 48 && c <= 57){out = (out << 1) + (out << 3) + c - 48; c =
getchar();}
    return flag ? out : -out;
}
int n,m,cnt,K;
char s[maxn];
int A[maxn],B[maxn];
void init(){
    scanf("%s",s + 1); n = strlen(s + 1);
    int i = 1;
    while (s[i] == '1') i++;
    while (i <= n){
        int j = i;
        while (j < n && s[j + 1] == s[i]) j++;
        A[++m] = j - i + 1;
        j = i = j + 1;
        while (j < n && s[j + 1] == s[i]) j++;
        if (i <= n) B[m] = j - i + 1;
        else B[m] = 0;
        i = j + 1;
    }
    for (int i = 1; i <= m; i++){
        A[i] += A[i - 1],B[i] += B[i - 1];
        //printf("[%d] ",B[i]);
    }
    //puts("");
    for (int i = 1; i <= n; i++) cnt += (s[i] == '0');

}
int q[maxn],head,tail;
void work(){
    q[head = tail = 1] = 0;
    int ans = 0;
    for (int i = 1; i <= m; i++){
        while (head <= tail && B[i - 1] - B[q[head]] > K) head++;
        if (head <= tail) ans = max(ans,A[i] - B[i - 1] + K + B[q[head]] -
A[q[head]]);
        else ans = max(ans,A[i] - A[i - 1] + K);
        //printf("[%d] %d %d\n",ans,A[q[head]],B[q[tail]]);
```

```
        while (head <= tail && B[i] - A[i] >= B[q[tail]] - A[q[tail]])
tail--;
        q[++tail] = i;
    }
    printf("%d\n",min(ans,cnt));
}
int main(){
    init();
    int Q = read();
    while (Q--){
        K = read();
        work();
    }
    return 0;
}
```

训练实况

开局 fyh看**E** hxm看**A** wxg中途加入看到**G**有人过就看**G**

wxg想出**G**开写**G**

0:35 wxg过**G**，fyh看**C** 和hxm讨论出**C**做法开写**C**

0:56 fyh过**C**，wxg想出**A**，开写**A**

1:24 wxg过**A** fyh看**H**，**J** 成功给hxm翻译错题，hxm想出错题做法，开写**J**

1，40 hxm没过**J**，重新读题，发现读错题，之后重新想**J**，wxg想**E** fyh想**I**和**H**

2，56hxm过**J**，推出**D**，讨论出做法后开写**D**并调，fyh写**H**并调

4:24 hxm过**D**，wxg想出**E**，开写**E**

4:58 wxg绝杀**E** fyh没过H**

训练总结

wxg: 罚时永远的痛

hxm:这一场发挥不错，但在罚时方面还需改进。

fyh:被一道不卡精度的题愣是生生卡到了精度，以后再也不用atan2进行极角排序了呜呜呜