

2020牛客暑期多校训练营（第二场）

[比赛网址](#)

训练结果

- 时间:2020-7-13 12:00~17:00
- rank:145/1159
- 完成情况:4/8/11

题解

B.Boundary

题意

给了n个点，画一个过原点的圆，问最多能有多少个点在圆上。

题解

我们固定原点和一个点，枚举剩下的点，求出这三个点的圆心，答案即为重合最多的圆心数加一。题目有点卡常，用分数计算会超时，无奈只能用double记录点玄学通过。

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<LL,LL> PII;
typedef pair<double,double> PDD;
#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 N=2000055;
const int maxn=2010;
PII A[maxn];
LL aa[N];
```

```
int n,ans,sum;
map<PDD,int> mp;
LL gcd(LL a,LL b){return b==0 ? a : gcd(b,a%b);}
int main()
{
    n=read();
    for(int i=1;i<=n;i++)scanf("%lld%lld",&A[i].X,&A[i].Y);
    for(int i=1;i<n;i++)
    {
        mp.clear();
        for(int j=i+1;j<=n;j++)
        {
            LL a=A[i].X,b=A[i].Y,c=A[j].X,d=A[j].Y;
            if(b*c-a*d==0) continue;
            LL p=a*a+b*b,q=c*c+d*d,r=b*c-a*d;
            LL x1=q*b-p*d,x2=r;
            LL y1=p*c-q*a,y2=r;
            // LL xx=gcd(abs(x1),abs(x2)),yy=gcd(abs(y1),abs(y2));
            double xx=1.*x1/x2;
            double yy=1.*y1/y2;
            /*      if(x1<0)      分数做法
            {
                x1*=-1;x2*=-1;
            }
            else if(x1==0)
            {
                x2=1;xx=1;
            }
            if(y1<0)
            {
                y1*=-1;y2*=-1;
            }
            else if(y1==0)
            {
                y2=1;yy=1;
            }
            x1/=xx;x2/=xx;y1/=yy;y2/=yy;
            LL hs2=0;
            hs2=hs2*998244353+x1;hs2=hs2*998244353+x2;
            hs2=hs2*998244353+y1;hs2=hs2*998244353+y2;
            // printf("%d %d %lld %lld %lld %lld\n",i,j,x1,x2,y1,y2);
            // aa[++sum]=hs2;*/
            ans=max(ans,++mp[PDD(xx,yy)]);
        }
    }
    /* sort(aa+1,aa+1+sum);
    for(int i=1;i<=sum;i++)
    {
        int tot=1;
```

```

        for(int j=i+1;j<=sum;j++)
        {
            if(aa[j]==aa[i]) tot++;
            else break;
        }
        ans=max(ans,tot);
        i+=tot-1;
    }*/
    printf("%d\n",++ans);
    return 0;
}

```

C.Cover the Tree

题意

用尽量少的路径覆盖一棵树的所有边

题解

solved by hxm

路径显然最优时从一个叶子出发，到另一个叶子结束。记叶子节点个数为 m ，那么答案应该就是 $\lceil \frac{m}{2} \rceil$

问题是如何找到一组解

选一个点作为根节点，如果能将不同子树里的叶子匹配，就能做到完全覆盖。那么只需要每次选剩余叶子最多的两个子树匹配。

容易发现，这样操作，只需要最大的子树里叶子节点个数不超过总个数的一半。

只需要随意选一个点，如果不满足条件，就往那个叶子最多的子树走，最后一定能走到一个合法的点。

```

#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 = 200005,maxm = 400005,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,h[maxn],ne,de[maxn];
struct EDGE{
    int to,nxt;
}ed[maxm];
void build(int u,int v){
    ed[++ne] = (EDGE){v,h[u]}; h[u] = ne;
    ed[++ne] = (EDGE){u,h[v]}; h[v] = ne;
    de[u]++; de[v]++;
}
int siz[maxn],rt,tot;
void dfs1(int u,int ff){
    if (de[u] == 1) siz[u] = 1;
    else {
        Redge(u) if ((to = ed[k].to) != ff){
            dfs1(to,u);
            siz[u] += siz[to];
        }
    }
}
void dfs2(int u,int ff){
    int flag = 1;
    Redge(u) if ((to = ed[k].to) != ff){
        if (siz[to] * 2 > tot){
            flag = 0;
            dfs2(to,u);
        }
        if (!flag) break;
    }
    if (flag) rt = u;
}
int m,pos[maxn],len[maxn];
vector<int> leaf[maxn];
struct node{
    int i;
};
inline bool operator <(const node& a,const node& b){
    return len[a.i] - pos[a.i] < len[b.i] - pos[b.i];
}
```

```

priority_queue<node> q;
void dfs3(int u,int ff){
    if (de[u] == 1) leaf[m].push_back(u),len[m]++;
    else {
        Redge(u) if ((to = ed[k].to) != ff){
            dfs3(to,u);
        }
    }
}
}
int main(){
    n = read();
    if (n == 1){printf("1\n1 1\n"); return 0;}
    if (n == 2){printf("1\n1 2\n"); return 0;}
    for (int i = 1; i < n; i++) build(read(),read());
    for (int i = 1; i <= n; i++){
        if (de[i] == 1) tot++;
        else rt = i;
    }
    dfs1(rt,0);
    dfs2(rt,0);
    //cout << rt << endl;
    Redge(rt){
        m++;
        dfs3(to = ed[k].to,rt);
    }
    //REP(i,m) cout << len[i] << endl;
    REP(i,m) q.push((node){i});
    printf("%d\n", (tot + 1) / 2);
    int cnt = 0,a,b;
    while (tot - cnt > 1){
        a = q.top().i; q.pop();
        b = q.top().i; q.pop();
        printf("%d %d\n",leaf[a][pos[a]++],leaf[b][pos[b]++]);
        if (pos[a] < len[a]) q.push((node){a});
        if (pos[b] < len[b]) q.push((node){b});
        cnt += 2;
    }
    if (tot & 1){
        a = q.top().i;
        printf("%d %d\n",leaf[a][pos[a]++],rt);
    }
    return 0;
}

```

D.Duration

solved by hxm 水题

E.Exclusive OR

题意

给了 n 个数，让你求 $1 \leq i \leq n$,选 i 个数（可以选相同的数）异或和的最大值。

题解

发现 i 在19之后答案和 $i-2$ 一致，原理参考线性基。所以我们只需求前19个的答案，用FWT即可完成。

```
#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 M=1050000;
int n,N,a[M],b[M],c[M],ans[M];
void FWT(int *a,int opt)
{
    for(int i=1;i<N;i<=1)
        for(int p=i<1,j=0;j<N;j+=p)
            for(int k=0;k<i;++k)
            {
                int X=a[j+k],Y=a[i+j+k];
                a[j+k]=(X+Y);a[i+j+k]=(X-Y);
                if(opt==-1)a[j+k]=1ll*a[j+k]/2,a[i+j+k]=1ll*a[i+j+k]/2;
            }
}
int main()
{
    n=read();int mx=0,x;
    for(int i=1;i<=n;i++)
    {
        x=read();
        a[x]=1;b[x]=1;mx=max(mx,x);
    }
    for(N=1;N<=mx;N<=1);
    ans[1]=mx;
    FWT(b,1);
```

```

for(int i=2;i<=min(25,n);i++)
{
    for(int j=0;j<N;j++)
        if(a[j]) a[j]=1;
    FWT(a,1);
    for(int j=0;j<N;j++)
        a[j]=a[j]*b[j];
    FWT(a,-1);
    for(int j=N-1;j>=0;j--)
    {
        if(a[j]) {ans[i]=j;break;}
    }
    // for(int j=0;j<N;j++)
    //     cout<<a[j]<<" ";
    // cout<<endl;
}
for(int i=1;i<=n;i++)
{
    if(i>25)
    {
        if(i&1) printf("%d ",ans[25]);
        else printf("%d ",ans[24]);
    }
    else printf("%d ",ans[i]);
}
puts("");
return 0;
}

```

F.Fake Maxpooling

题意

一个 $n \times m$ 的网格，每个格子 (x,y) 里写着 $lcm(x,y)$

现在用一个 $k \times k$ 的框去选中一些格子，得到其中最大值。

求所有选取方法最大值的总和。

题解

solved by hxm

暴力 $O(nm \log n)$ 求出 lcm 然后竖着用 m 个单调队列维护最大值，然后再用一个单调队列维护单调队列的最大值，即可求出当前区域的最大值。

```
#include<algorithm>
```

```
#include<iostream>
#include<cstdlib>
#include&ltcstring>
#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 = 5005,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 gcd(int a,int b) {
    return !b ? a : gcd(b,a % b);
}
int A[maxn][maxn],n,m,K;
int Q[maxn][maxn],H[maxn],T[maxn];
int q[maxn],pos[maxn],head,tail;
int main(){
    n = read(); m = read(); K = read();
    REP(i,n) REP(j,m) A[i][j] = i * j / gcd(i,j);
    REP(i,m) H[i] = 1;
    LL ans = 0;
    for (int i = 1; i <= n; i++){
        for (int j = 1; j <= m; j++){
            while (H[j] <= T[j] && i - Q[j][H[j]] >= K) H[j]++;
            while (T[j] >= H[j] && A[i][j] >= A[Q[j][T[j]]][j]) T[j]--;
            Q[j][++T[j]] = i;
        }
        if (i >= K){
            head = 1; tail = 0;
            for (int j = 1; j <= m; j++){
                while (head <= tail && j - pos[head] >= K) head++;
                while (head <= tail && A[Q[j][H[j]]][j] >= q[tail]) tail--;
                q[++tail] = A[Q[j][H[j]]][j];
                pos[tail] = j;
                if (j >= K) ans += q[head];
            }
        }
    }
}
```



```

    }
}
}
printf("%lld\n",ans);
return 0;
}

```

H.Happy Triangle

题意

一个multiset支持如下操作：

- 插入一个数 x
- 从中删除一个数 x （如果有重复的，只删除一个）
- 给定 x 问集合中是否存在两个数 a,b 使得 a,b,x 组成一个非退化三角形。

题解

补题 by fyh

询问即问是否存在两个数 a,b 使得 $|a-b|<x<a+b$ 这个很明显是一个区间覆盖问题，但是因为集合中两两组成的区间是 n^2 级别的，考虑减少区间 $a>b>c$ ，其中 (a,b) 与 (a,c) 组成的开区间分别是 $(a-b,a+b)$ ， $(a-c,a+c)$ 后者是完全被前者包含的，所以对于每一个数，只需要找他的前驱，用线段树维护一下这 n 个区间的并即可。

```

#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=200010;
int n,q,tp[maxn],a[maxn],b[maxn],tag[maxn<<2],w[maxn<<2];
multiset<int> S;
multiset<int>::iterator it,it2;
void pushdown(int L,int R,int o)
{
    int mid=L+R>>1,lo=o<<1,ro=lo|1;

```

```
    tag[lo]+=tag[o];tag[ro]+=tag[o];
    w[lo]+=tag[o];w[ro]+=tag[o];
    tag[o]=0;
}
void update(int L,int R,int o,int ql,int qr,int v)
{
    if(L==ql && R==qr)
    {
        w[o]+=v;
        tag[o]+=v;
        return;
    }
    pushdown(L,R,o);
    int mid=L+R>>1,lo=o<<1,ro=lo|1;
    if(qr<=mid)update(L,mid,lo,ql,qr,v);
    else if(ql>mid)update(mid+1,R,ro,ql,qr,v);
    else update(L,mid,lo,ql,mid,v),update(mid+1,R,ro,mid+1,qr,v);
}
int query(int L,int R,int o,int qx)
{
    if(L==R)return w[o];
    pushdown(L,R,o);
    int mid=L+R>>1,lo=o<<1,ro=lo|1;
    if(qx<=mid)return query(L,mid,lo,qx);
    else return query(mid+1,R,ro,qx);
}
void insert(int A)
{
    int pos=lower_bound(b+1,b+n+1,A)-b,l,r;
    it=S.lower_bound(A);
    if(it!=S.end() && it!=S.begin())
    {
        it--;it2=it;it++;
        l=lower_bound(b+1,b+n+1,(*it)-*(it2))-b;
        r=lower_bound(b+1,b+n+1,(*it)+*(it2))-b-1;
        if(*it-*it2==b[l])l++;
        update(1,n,1,l,r,-1);
    }
    if(it!=S.end())
    {
        l=lower_bound(b+1,b+n+1,*it-A)-b;
        r=lower_bound(b+1,b+n+1,*it+A)-b-1;
        if(*it-A==b[l])l++;
        update(1,n,1,l,r,1);
    }
    if(it!=S.begin())
    {
        it--;it2=it;it++;
        l=lower_bound(b+1,b+n+1,A-*it2)-b;
```

```

        r=lower_bound(b+1,b+n+1,*it2+A)-b-1;
        if(A-*it2==b[l])l++;
        update(1,n,1,l,r,1);
    }
    S.insert(A);
}
void delT(int A)
{
    int pos=lower_bound(b+1,b+n+1,A)-b,l,r;
    it=S.find(A);
    S.erase(it);
    it=S.lower_bound(A);
    if(it!=S.end() && it!=S.begin())
    {
        it--;it2=it;it++;
        l=lower_bound(b+1,b+n+1,(*it)-*(it2))-b;
        r=lower_bound(b+1,b+n+1,(*it)+*(it2))-b-1;
        if(*it-*it2==b[l])l++;
        update(1,n,1,l,r,1);
    }
    if(it!=S.end())
    {
        l=lower_bound(b+1,b+n+1,*it-A)-b;
        r=lower_bound(b+1,b+n+1,*it+A)-b-1;
        if(*it-A==b[l])l++;
        update(1,n,1,l,r,-1);
    }
    if(it!=S.begin())
    {
        it--;it2=it;it++;
        l=lower_bound(b+1,b+n+1,A-*it2)-b;
        r=lower_bound(b+1,b+n+1,*it2+A)-b-1;
        if(A-*it2==b[l])l++;
        update(1,n,1,l,r,-1);
    }
}
void ask(int x)
{
    int pos=lower_bound(b+1,b+n+1,x)-b;
    if(query(1,n,1,pos))puts("Yes");
    else puts("No");
}
int main()
{
    q=read();
    for(int i=1;i<=q;i++)tp[i]=read(),a[i]=b[i]=read();
    sort(b+1,b+q+1);
    n=unique(b+1,b+q+1)-b-1;
    b[n+1]=1e9;
    for(int i=1;i<=q;i++)
    {

```

```
        if(tp[i]==1)insert(a[i]);
        else if(tp[i]==2)delt(a[i]);
        else ask(a[i]);
    }
    return 0;
}
```

J.Just Shuffle

题意

一个排列初始时是 $1,2,\dots,n$ 存在某种置换 p ,使得排列在置换 k 次后的排列为 $A_1,\dots,A_n$ 求置换 p

题解

solved by fyh

本题的一大特性是 k 是质数,也就是 k 模任何数都非0。

置换其实就是若干个循环。每个循环都是独立的,现考虑某个长度为 $size$ 的环,置换 k 次的结果是等效于置换为 $k \% size$ 的结果。设 $k \% size = m$ 则我们当前得到的环其实是走 m 步的结果,我们要得到走1步的结果,便可以考虑在这个环上每步都好几倍,最后等效为走一步,即解 $m * k \% size = 1$ 的模方程的 x 至此,我们成功构造出了原置换。

```
#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;
int n,k,to[maxn],col[maxn],cnt,size[maxn],ans[maxn];
bool vis[maxn];
vector<int> V[maxn];
int main()
{

```

```

n=read();k=read();
for(int i=1;i<=n;i++)to[i]=read();
for(int i=1;i<=n;i++)
    if(!vis[i])
    {
        col[i]=++cnt;
        V[cnt].push_back(i);
        int now=i;
        while(!vis[to[now]])
            V[cnt].push_back(to[now]),
            vis[to[now]]=1,col[to[now]]=cnt,
            now=to[now],size[cnt]++;
    }
for(int i=1;i<=cnt;i++)
{
    if(size[i]==1)ans[V[i][0]]=V[i][0];
    else
    {
        int m=k%size[i],x=0;
        while(x*m%size[i]!=1)x++;
        for(int j=0;j<V[i].size();j++)
        {
            int now=V[i][j];
            ans[V[i][j]]=V[i][(j+x)%size[i]];
        }
    }
}
for(int i=1;i<n;i++)printf("%d ",ans[i]);
printf("%d\n",ans[n]);
return 0;
}

```

I.Interval

题意

对一个区间 $[l,r]$ 可进行两种操作：

- 1、将 $[l,r]$ 变为 $[l-1,r]$ 或 $[l+1,r]$
- 2、将 $[l,r]$ 变为 $[l,r-1]$ 或 $[l,r+1]$

现在有 m 个限制，限制区间 $[l_i,r_i]$ 不能进行操作 c （ $c=L$ 或 $c=R$ ）但是开启这个限制需要 w_i 的费用

问最少的费用花费，使得区间 $[1,n]$ 不能转移到任意一个区间 $[l,r]$ 使 $l=r$

题解

补题 solved by hxm

将每个区间看做二维平面上的点，我们的目标就是阻止从 $(1,n)$ 走到任意一个 (x,x)

显然相邻的点可以连边，我们把 $(1,n)$ 看做源点的话，新建一个汇点将所有 (x,x) 连向汇点，那么这就是一个最小割问题

但是会TJ

发现这是一个平面图，转化为对偶图的最短路即可

```
#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 = 250005,maxm = 100005;
const LL INF = 1000000000000000000ll;
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 h[maxn],ne;
struct EDGE{
    int to,w,nxt;
}ed[maxn * 2];
void build(int u,int v,int w){
    ed[++ne] = (EDGE){v,w,h[u]}; h[u] = ne;
    ed[++ne] = (EDGE){u,w,h[v]}; h[v] = ne;
    //printf("build %d to %d costs %d\n",u,v,w);
}
int C[505][505],R[505][505];
```

```

int n,m,S,T;
int id(int x,int y){
    return x * (x - 1) / 2 + y;
}
LL d[maxn],vis[maxn];
struct node{int u; LL d;};
inline bool operator <(const node& a,const node& b){return a.d > b.d;}
priority_queue<node> q;
void dijkstra(){
    for (int i = 1; i <= T; i++) d[i] = INF,vis[i] = false;
    d[S] = 0;
    node u;
    q.push((node){S,d[S]});
    while (!q.empty()){
        u = q.top(); q.pop();
        if (vis[u.u]) continue;
        vis[u.u] = true;
        Redge(u.u) if (!vis[to = ed[k].to] && d[to] > d[u.u] + ed[k].w){
            d[to] = d[u.u] + ed[k].w;
            q.push((node){to,d[to]});
        }
    }
}
int main(){
    n = read(); m = read();
    int l,r,w; char c;
    for (int i = 1; i <= m; i++){
        l = read(); r = read(); scanf("%c",&c); w = read();
        if (c == 'L') R[l][r] = w;
        else C[l][r] = w;
    }
    for (int i = 1; i <= n - 1; i++)
        for (int j = 1; j <= i; j++){
            //puts("LXT");
            //printf("[%d,%d] [%d,%d]\n",i + 1,j + 1,i + 1,j);
            //cout << C[i + 1][j + 1] << ' ' << R[i + 1][j] << endl;
            if (j < i && C[j + 1][i + 1]) build(id(i,j),id(i,j + 1),C[j + 1][i + 1]);
            if (i < n - 1 && R[j][i + 1]) build(id(i,j),id(i + 1,j),R[j][i + 1]);
        }
    S = 0; T = n * (n - 1) / 2 + 1;
    for (int i = 1; i <= n - 1; i++) if (C[1][i + 1])
        build(S,id(i,1),C[1][i + 1]);
    for (int i = 1; i <= n - 1; i++) if (R[i][n]) build(id(n - 1,i),T,R[i][n]);
    dijkstra();
    if (d[T] != INF) printf("%lld\n",d[T]);
    else puts("-1");
    return 0;
}

```

}

训练实况

开场 发现**D**很简单

12:08 hxm 过**D** wxg想出**B**题做法 fyh开写**B**

12:08~12:50 fyh狂wa**B** wxg hxm想出**C** hxm开写**C**

13:33 hxm过**C** 在想**B**的错误和**F**，尝试用分数处理精度问题 改为wxg写**B**

13:33~14:30 **B**卡常 又wa又T hxm想出**F** 开写

15:04 hxm过**F** wxg继续尝试**B**，放弃 wxg开想**G**

15:04~16:00

16:00 看**J**题过得人多 fyh想**J**

16:36 fyh 过**J** wxg写**G**,未知原因段错误，结束。

训练总结

因为很少人过，没有读**I**

K题因为过的人少没有深想

H没有仔细想

B陷入无解，纠结太久

总结：计算几何最后再写 对榜的利用价值??

From:

<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:

https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:die_java:front_page_summertrain2



Last update: 2020/10/17 23:02