2020牛客暑期多校训练营（第十场）

比赛网址

训练结果

- 时间:2020-8-10 12:00~17:00
- rank:146/?
- 完成情况：3/4/10

题解

A.Permutation

题意

给了一个质数 p ,让你构造一个 $1到p-1$ 的数列，满足 $x_{i+1}≡2x_i(modp) \ or \ x_{i+1}≡3x_i(modp)$

题解

直接从开始一直乘二，不能乘就乘三，就过了

```
#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;
}
int T,d[2000005],ans[2000055],vis[10005];
int ksm(int x,int k,int mod)
{
    int ans=1;
    for(;k;k>>=1,x=1ll*x*x%mod)
        if(k&1)
            ans=1ll*ans*x%mod;
    return ans;
```

```
}
int main()
{
    for(T=read();T;T--)
    {
        int p=read();
        if(p==2) {puts("1");continue;}
        int now=1,tot=0,sum,cnt=1,len;
        for(;;)
        {
            if(d[now]) break;
            tot++;
            d[now]=1;
            ans[tot]=now;
            now=now*2%p;
        }// cout<<tot<<" ";
        sum=(p-1)/tot;len=tot;
        while(tot!=p-1)
        {
            cnt++;
            for(int i=1;i<p;i++)
            if(!d[i]) {now=i;break;}
            for(;;)
            {
                if(d[now]) break;
                tot++;
                d[now]=cnt;
                ans[tot]=now;
                now=now*2%p;
            }
        }
        int ans=-1;int pw=ksm(2,len-1,p);
        for(int i=1;i<p;i++)
        {
            now=i;int fl=0;
            for(int j=1;j<=sum;j++)
            {
                if(vis[d[now]]) {fl=1;break;}
                vis[d[now]]=1;
                now=1ll*now*pw%p;
                now=now*3%p;
            }
            for(int j=1;j<=sum;j++)
                vis[j]=0;
            if(!fl) {ans=i;break;}
        }
        for(int i=1;i<p;i++)
            d[i]=0;
        if(ans==-1) puts("-1");
```

```
        else
        {
            for(int i=1;i<=sum;i++)
            {
                printf("%d ",ans);
                for(int j=1;j<len;j++)
                    ans=ans*2%p,printf("%d ",ans);
                ans=ans*3%p;
            }
            puts("");
        }
    }
    return 0;
}
```

C.Decrement on the Tree

题意

有一棵边权树，每次你可以选择一条路径（路径上边权都大于0）把所有的边权减一，问最小的操作次数将所有边权都改为 0。每次询问会修改一条边。

题解

发现不带修就是一个树型dp□f_i 表示改 i 的子树要的最小步数减去父亲到他的边权□$f_i= max(mx_i-v_i,(sum_i-v_i)/2)$,mx_i 指到子树的最大边权,sum_i 指到子树的所有边权和,v_i 指父亲到他的边权，这样发现 f_i 只和 i 连出去的边有关，所以每次修改一条边，只有两个点的 f_i 发生变化，暴力改就行了

```
 #include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<set>
typedef long long ll;
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,tot,fa[N],to[N],head[N],nextt[N],dep[N];
int a[N],b[N],c[N];
ll res,sum[N],mx[N],mx2[N],val[N],f[N],w[N];
multiset<int> st[N];
```

```
void add(int a,int b,int c)
{
    to[++tot]=b;
    nextt[tot]=head[a];
    head[a]=tot;
    w[tot]=c;
}
void dfs(int u,int F,int d)
{
    dep[u]=d;fa[u]=F;
    for(int i=head[u];i;i=nextt[i])
    if(to[i]!=F)
    {
        val[to[i]]=w[i];
        dfs(to[i],u,d+1);
        sum[u]+=w[i];
        if(w[i]>mx[u]) mx[u]=w[i];
        st[u].insert(w[i]);
    }
    if(sum[u]<=val[u]) f[u]=0;
    else res+=(f[u]=max(mx[u]-val[u],(sum[u]+1-val[u])/2));
}
int query(int x)
{
    if(!st[x].size()) return 0;
    multiset<int>::iterator it;
    it=st[x].end();it--;
    return *it;
}
int main()
{
    n=read();m=read();
    for(int i=1;i<n;i++)
    {
        a[i]=read();b[i]=read();c[i]=read();
        add(a[i],b[i],c[i]);add(b[i],a[i],c[i]);
    }
    dfs(1,0,1);
    printf("%lld\n",res);
    for(int i=1;i<=m;i++)
    {
        ll x=read(),y=read();
        int u=a[x],v=b[x];
        if(dep[u]<dep[v]) swap(u,v);
        st[v].erase(st[v].lower_bound(c[x]));
        st[v].insert(y);
        res-=f[u];f[u]=0;
        if(sum[u]>y)
            f[u]=max(query(u)-y,(sum[u]+1-y)/2);
```

```
        res+=f[u];
        res-=f[v];f[v]=0;
        ll s=sum[v]-c[x]+y,mxx=query(v);
        if(s>val[v])
            f[v]=max(mxx-val[v],(s+1-val[v])/2);
        res+=f[v];
        val[u]=y;sum[v]+=y-c[x];c[x]=y;
        printf("%lld\n",res);
    }
    return 0;
}
```

D.Hearthstone Battlegrounds

题意

题解

```

```

E.Game

题意

题解

```

```

G.Math Test

题意

二分签到题

J.Identical Trees

题意

有两棵有根树，标号不同，但保证同构，问最少修改一棵树上多少点编号，使得这两棵树完全相同
$n\leq500$

题解

设$dp[x][y]$ 表示第一棵树以x为根，第二棵树以y为根，最少需要改多少个。这个状态合法的前提是$x和y$同构，然后把他们的儿子之前如果同构的话就可以匹配，匹配的代价是$dp[v1][v2]$□$v1$$,v2$是x,y的儿子），此处需要一个最小代价最大匹配，根据最后的代价进行转移□$dp[x][y]=mincost+(x!=y)$ 我感觉这玩意复杂度是$O(n^5)$的。。。但是跑的飞快。

注意！！！判同构的时候既要判哈希值也要判size!!!本题有两个点卡哈希！

```
#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
typedef unsigned long long ULL;
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=600;
int n,rt1,rt2,a,size1[maxn],size2[maxn],dp[maxn][maxn];
vector <int> G1[maxn],G2[maxn];
int prime[maxn],len;
ULL H1[maxn],H2[maxn];
struct ZKW {
    int n,m,s,t,cost,ans;
    int
d[maxn<<1],vis[maxn<<1],first[maxn<<1],inq[maxn<<1],next[510*510*2];
    struct Edge {int from,to,flow,cost;}edges[510*510*2];
    void init(int n) {
        this->n=n;m=0;
        memset(first,-1,sizeof(first));
        memset(inq,0,sizeof(inq));
    }
    void AddEdge(int from,int to,int cap,int cost) {
edges[m]=(Edge){from,to,cap,cost};next[m]=first[from];first[from]=m++;
        edges[m]=(Edge){to,from,0,-cost};next[m]=first[to];first[to]=m++;
    }
    int BFS() {
        for(int i=1;i<=n;i++) d[i]=maxn;
        queue<int> Q;Q.push(t);d[t]=0;
        while(!Q.empty()) {
            int x=Q.front();Q.pop();inq[x]=0;
```

```
            for(int i=first[x];i!=-1;i=next[i]) {
                Edge& e=edges[i^1];
                if(e.flow&&d[e.from]>d[x]+e.cost) {
                    d[e.from]=d[x]+e.cost;
                    if(!inq[e.from]) inq[e.from]=1,Q.push(e.from);
                }
            }
        }
        for(int i=0;i<=m;i++) edges[i].cost+=d[edges[i].to]-
d[edges[i].from];
        cost+=d[s];return d[s]!=maxn;
    }
    int DFS(int x,int a) {
        if(x==t||!a) {ans+=cost*a;return a;}
        int flow=0,f;vis[x]=1;
        for(int i=first[x];i!=-1;i=next[i]) {
            Edge& e=edges[i];
            if(e.flow&&!e.cost&&!vis[e.to]&&(f=DFS(e.to,min(e.flow,a)))) {
                e.flow-=f;edges[i^1].flow+=f;
                a-=f;flow+=f;if(!a) break;
            }
        }
        return flow;
    }
    int solve(int s,int t) {
        this->s=s;this->t=t;ans=cost=0;
        while(BFS()) do memset(vis,0,sizeof(vis));while(DFS(s,maxn));
        return ans;
    }
}sol;
bool vis[10010];
void init(int x)
{
    for(int i=2;i<=x;i++)
    {
        if(!vis[i])prime[++len]=i;
        for(int j=1;j<=len && i*prime[j]<=x;j++)
        {
            vis[i*prime[j]]=1;
            if(i%prime[j]==0)break;
        }
    }
}
void dfs1(int now)
{
    H1[now]=size1[now]=1;
    for(int v:G1[now])
    {
        dfs1(v);
        H1[now]+=H1[v]*(ULL)prime[size1[v]];
        size1[now]+=size1[v];
```

```
    }
}
void dfs2(int now)
{
    H2[now]=size2[now]=1;
    for(int v:G2[now])
    {
        dfs2(v);
        H2[now]+=H2[v]*(ULL)prime[size2[v]];
        size2[now]+=size2[v];
    }
}
void DP(int x,int y)
{
    dp[x][y]=maxn;
    if(H1[x]!=H2[y] || size1[x]!=size2[y])return;
    if(size1[x]==1 && size2[y]==1)
    {
        dp[x][y]=(x!=y);
        return;
    }
    for(int v1:G1[x])
        for(int v2:G2[y])DP(v1,v2);
    int N=2+G1[x].size()+G2[y].size(),s=1,t=N;
    sol.init(N);
    for(int i=0;i<G1[x].size();i++)sol.AddEdge(s,i+2,1,0);
    for(int i=0;i<G1[x].size();i++)
        for(int j=0;j<G2[y].size();j++)
        {
            int v1=G1[x][i],v2=G2[y][j];
            if(H1[v1]==H2[v2] &&
size1[v1]==size2[v2])sol.AddEdge(i+2,j+2+G1[x].size(),1,dp[v1][v2]);
        }
    for(int i=0;i<G2[y].size();i++)sol.AddEdge(i+2+G1[x].size(),t,1,0);
    int ans=sol.solve(s,t);
    dp[x][y]=ans+(x!=y);
}
int main()
{
    init(4000);
    n=read();
    for(int i=1;i<=n;i++)
    {
        a=read();
        if(a==0)rt1=i;
        else G1[a].push_back(i);
    }
    dfs1(rt1);
    for(int i=1;i<=n;i++)
```

```
    {
        a=read();
        if(a==0)rt2=i;
        else G2[a].push_back(i);
    }
    dfs2(rt2);
    DP(rt1,rt2);
    printf("%d\n",dp[rt1][rt2]);
    return 0;
}
```

训练实况

训练总结

wxg: 菜是原罪

hxm:菜啊

fyh:对网络流复杂度不熟，导致不敢写J