牛客练习赛81

比赛链接

C-小Q与构造

题意

求满足如下条件的集合 S 个数：

1. $x\in S\to 1\le x\le n$
2. $x\in S,y\in S,x\le y\to y \neq kx,y\neq k^px$

题解

把 $1\sim n$ 拆分成若干条链 $a,ak,ak^2,ak^3\cdots$ 易知每条链之间的选择互不影响，答案即为每条边的答案之积。

对每条链，直接状压 dp对位置 i如果位置 $i-1,i-p$ 已选则一定不能选，否则任意。

不难发现每条链的答案只有链的长度有关，与首项 a 的具体值无关。另外所有 $k\not\mid i$ 均可以作为链的首项 a

考虑暴力枚举链的长度，显然链的最大长度不超过 $\log_k n$对每个长度，计算出 a 的上下界然后删去 k 的倍数统计贡献即可。

时间复杂度 $O((2^p+\log n)\log n)$

```
const int Mod=10086001;
int quick_pow(int a,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*a%Mod;
        a=1LL*a*a%Mod;
        k>>=1;
    }
    return ans;
}
LL my_pow(int a,int k){
    LL ans=1;
    while(k--)ans*=a;
    return ans;
}
const int MAXL=64,MAXP=10;
int dp[MAXL][1<<MAXP],s[MAXL];
int main()
```

```
{
    LL n=read_LL();
    int k=read_int(),p=read_int(),S=(1<<p)-1;
    if(k==1){
        enter(1);
        return 0;
    }
    dp[1][0]=dp[1][1]=1;
    s[1]=2;
    _for(i,2,MAXL){
        _for(j,0,1<<p){
            dp[i][(j<<1)&S]=(dp[i][(j<<1)&S]+dp[i-1][j])%Mod;
            if((j&1)==0&&(j&(1<<(p-1)))==0)
            dp[i][(j<<1|1)&S]=(dp[i][(j<<1|1)&S]+dp[i-1][j])%Mod;
        }
        _for(j,0,1<<p)
        s[i]=(s[i]+dp[i][j])%Mod;
    }
    int ans=1;
    for(int i=1;;i++){
        LL lef=n/my_pow(k,i)+1,rig=n/my_pow(k,i-1);
        LL cnt=rig-lef+1-((rig/k)-(lef+k-1)/k+1);
        ans=1LL*ans*quick_pow(s[i],cnt%(Mod-1))%Mod;
        if(lef==1)break;
    }
    enter(ans);
    return 0;
}
```

D-小Q与树

题意

给定一棵点权树，求 $\sum_{u=1}^n\sum_{v=1}^n\min (a_u,a_v)\text{dis}(u,v)$

题解

点分治，然后统计路径贡献。定义偏序关系 $P(u,v)$ 表示 $a_u\gt a_v$ 或 $a_u = a_v,u\gt v$

每次点分治时，对每个点 u统计
$\sum_{P(u,v)}a_v\text{dis}(a_v,rt)+\text{dis}(a_u,rt)\sum_{P(u,v)}a_v$ 即可。

时间复杂度 $O(n\log^2 n)$

```
const int MAXN=2e5+5,Mod=998244353;
struct Edge{
```

```
    int to,next;
}edge[MAXN<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
struct Node{
    int s,dis;
    Node(int s=0,int dis=0):s(s),dis(dis){}
    void operator += (const Node &b){
        s=s+b.s;
        if(s>=Mod)s-=Mod;
        dis=dis+b.dis;
        if(dis>=Mod)dis-=Mod;
    }
    bool operator < (const Node &b)const{
        return false;
    }
};
int a[MAXN],sz[MAXN],mson[MAXN],tot_sz,root,root_sz,ans;
bool vis[MAXN];
vector<pair<int,Node> >sd;
typedef vector<pair<int,Node> >::iterator iter;
void dfs1(int u,int fa,int dis){
    sd.push_back(make_pair(a[u],Node(1LL*a[u]*dis%Mod,dis)));
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v]||v==fa)
        continue;
        dfs1(v,u,dis+1);
    }
}
void dfs2(int u,int fa,int n,int f){
    iter it=lower_bound(sd.begin(),sd.end(),make_pair(a[u],Node()));
    Node cur=(it==sd.begin())?Node(0,0):(--it)->second;
    ans=(ans+(cur.s+1LL*a[u]*(n-cur.dis))*f)%Mod;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v]||v==fa)
        continue;
        dfs2(v,u,n,f);
    }
}
void cal(int u,int dis,int f){
    sd.clear();
    dfs1(u,0,dis);
    sort(sd.begin(),sd.end());
    iter it=sd.begin();
    ++it;
    for(;it!=sd.end();it++){
```

```
        iter p=--it;
        ++it;
        it->second+=p->second;
    }
    dfs2(u,0,(--it)->second.dis,f);
}
void query(int u){
    cal(u,0,1);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(!vis[v])
        cal(v,1,-1);
    }
}
void find_root(int u,int fa){
    sz[u]=1;mson[u]=0;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v]||v==fa)
        continue;
        find_root(v,u);
        sz[u]+=sz[v];
        mson[u]=max(mson[u],sz[v]);
    }
    mson[u]=max(mson[u],tot_sz-sz[u]);
    if(mson[u]<root_sz){
        root=u;
        root_sz=mson[u];
    }
}
void solve(int u){
    int cur_sz=tot_sz;
    vis[u]=true;query(u);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v])
        continue;
        tot_sz=sz[v]>sz[u]?cur_sz-sz[u]:sz[v];root_sz=MAXN;
        find_root(v,u);
        solve(root);
    }
}
int main()
{
    int n=read_int();
    _rep(i,1,n)a[i]=read_int();
    _for(i,1,n){
        int u=read_int(),v=read_int();
        Insert(u,v);
        Insert(v,u);
```

```
    }
    tot_sz=n,root_sz=MAXN;
    find_root(1,0);
    solve(root);
    enter((ans+Mod)%Mod*2%Mod);
    return 0;
}
```