错题集 5

1□Furukawa Nagisa's Tree

链接

题意

给定一个点权树，点 i 点权为 a_i□

树上有向路径 $s\to t$ 被认为是好的当且仅当 $a_s+a_{v_1}x^1+a_{v_2}x^2+\cdots a_tx^k\equiv y(\bmod p)$□其中 $v_1,v_2\cdots$ 为 $s\to t$ 依次经过的点。

三元组 (u,v,w) 被认为是好的当且仅当 $u\to v,u\to w,v\to w$ 三条路径都是好的或都是坏的。(不要求 u,v,w 互异)

问有多少个好的三元组。

题解

构建图 G□如果树上有向路径 $u\to v$ 是好的，则 $u\to v$ 连一条黑边，否则 $u\to v$ 连一条白边。

于是图 G 是完全图，所以坏的三元组个数为异色角个数除以 2。

假设点 i 黑边入度为 in_1□黑边出度为 out_1□白边入度为 in_0□白边出度为 out_0□

考虑点 i 在三元组 (u,v,w) 中不同位置的异色角贡献。

当 i 作为点 u 时，假设 $u\to v_1$ 是白边□$u\to v_2$ 是黑边□$(u,v_1,v_2),(u,v_2,v_1)$ 是不同的坏的三元组。

于是 i 的异色角贡献为 $2\times \text{out}_0\text{out}_1$□类似的，当 i 作为点 w 时，i 的异色角贡献为 $2\times \text{in}_0\text{in}_1$□

当 i 作为点 v 时，易知 i 的贡献为 $\text{out}_0\text{in}_1+\text{in}_0\text{out}_1$□

于是点 i 的异色角总贡献为 $2\times \text{out}_0\text{out}_1+2\times \text{in}_0\text{in}_1+\text{out}_0\text{in}_1+\text{in}_0\text{out}_1$□

接下来问题转化为怎么计算 $\text{in}_1,\text{out}_1,\text{in}_0,\text{out}_0$□不妨只计算 in_1,out_1□然后有 $\text{in}_0=n-\text{in}_1,\text{out}_0=n-\text{out}_1$□

考虑点分治，设当前重心为 rt□于是对点对 (s,t)□需要判定 $a_s+a_{v_1}x^1+a_{v_2}x^2+\cdots a_tx^k\equiv y(\bmod p)$□

移项，得 $a_{v_k'+1}x^{k'+1}+\cdots +a_tx^k\equiv y-(a_s+a_{v_1}x^1+a_{v_2}x^2+\cdots a_{\text{rt}}x^{k'})(\bmod p)$

设 $\text{pre}(s)=a_s+a_{v_1}x^1+a_{v_2}x^2+\cdots a_{\text{rt}}x^k,\text{suf}(s)=a_{rt}+a_{v_1}x^1+\cdots +a_sx^k$于是上式化简为

$$ \text{suf}(s)-a_{rt}\equiv (y-\text{pre}(s))x^{-k'}(\bmod p) $$

于是 dfs 过程中维护 $\text{pre}(s),\text{suf}(s)$ 同时记录 $\text{suf}(s)-a_{rt}$ 和 $(y-\text{pre}(s))x^{-k'}$最后双指针统计答案。

另外关于两个结点都在同一个子树的情况，直接容斥即可。总时间复杂度为 $O(n\log^2 n)$

```
const int MAXN=1e5+5;
int x,px[MAXN],invx[MAXN],y,Mod;
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;
}
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;
}
int val[MAXN],in0[MAXN],in1[MAXN],out0[MAXN],out1[MAXN];
int sz[MAXN],mson[MAXN],tot_sz,root,root_sz;
bool vis[MAXN];
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];
    }
}
vector<pair<int,int> >vec1,vec2;
```

```
void update(int u,int dep,int &pre,int &suf){
    pre=(1LL*pre*x+val[u])%Mod;
    suf=(suf+1LL*px[dep]*val[u])%Mod;
    vec1.push_back(make_pair(1LL*(y-pre+Mod)*invx[dep]%Mod,u));
    vec2.push_back(make_pair((suf-val[root]+Mod)%Mod,u));
}
void dfs(int u,int fa,int dep,int pre,int suf){
    update(u,dep,pre,suf);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v]||v==fa)continue;
        dfs(v,u,dep+1,pre,suf);
    }
}
void cal(int u,int sign,int dep,int pre,int suf){
    vec1.clear();
    vec2.clear();
    update(u,dep,pre,suf);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v])continue;
        dfs(v,u,dep+1,pre,suf);
    }
    sort(vec1.begin(),vec1.end());
    sort(vec2.begin(),vec2.end());
    for(int pos1=0,pos2=0,cnt;pos1<vec1.size();pos1++){
        if(pos1==0||vec1[pos1].first!=vec1[pos1-1].first)cnt=0;
        while(pos2<vec2.size()&&vec2[pos2].first<vec1[pos1].first)pos2++;
while(pos2<vec2.size()&&vec2[pos2].first==vec1[pos1].first)pos2++,cnt++;
        out1[vec1[pos1].second]+=cnt*sign;
    }
    for(int pos1=0,pos2=0,cnt;pos2<vec2.size();pos2++){
        if(pos2==0||vec2[pos2].first!=vec2[pos2-1].first)cnt=0;
        while(pos1<vec1.size()&&vec1[pos1].first<vec2[pos2].first)pos1++;
while(pos1<vec1.size()&&vec1[pos1].first==vec2[pos2].first)pos1++,cnt++;
        in1[vec2[pos2].second]+=cnt*sign;
    }
}
void query(int u){
    cal(u,1,0,0,0);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(vis[v])continue;
        cal(v,-1,1,val[u],val[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();
    Mod=read_int(),x=read_int(),y=read_int();
    _rep(i,1,n)val[i]=read_int();
    _for(i,1,n){
        int u=read_int(),v=read_int();
        Insert(u,v);
        Insert(v,u);
    }
    px[0]=1;
    _for(i,1,MAXN)px[i]=1LL*px[i-1]*x%Mod;
    invx[MAXN-1]=quick_pow(px[MAXN-1],Mod-2);
    for(int i=MAXN-1;i;i--)invx[i-1]=1LL*invx[i]*x%Mod;
    tot_sz=n,root_sz=MAXN;
    find_root(1,0);
    solve(root);
    LL ans=0;
    _rep(i,1,n){
        in0[i]=n-in1[i];
        out0[i]=n-out1[i];
        ans+=2LL*in0[i]*in1[i];
        ans+=2LL*out0[i]*out1[i];
        ans+=1LL*in0[i]*out1[i];
        ans+=1LL*in1[i]*out0[i];
    }
    enter(1LL*n*n*n-ans/2);
    return 0;
}
```

2□Trips

链接

题意

给定一个图，初始时图没有边，接下来每次一个加边操作，然后询问加边后满足所有点度数不小于 k 的最大子图。

题解

考虑先加入所有边，然后从最后的操作开始向前删边。

对每个点，当该点度数小于 k 时直接删去该点和对应边。每个点仅删去一次，每条边最多访问三次，于是总时间复杂度 $O(n+m)$。

```
const int MAXN=2e5+5;
struct Edge{
    int u,v;
    bool vis;
}edge[MAXN];
vector<pair<int,int> > g[MAXN];
int deg[MAXN],ans,k;
bool live[MAXN];
void update(int u);
void del(int u){
    ans--;
    live[u]=false;
    _for(i,0,g[u].size()){
        int v=g[u][i].first,edge_id=g[u][i].second;
        if(!edge[edge_id].vis){
            edge[edge_id].vis=true;
            update(v);
        }
    }
}
void update(int u){
    if(live[u]){
        deg[u]--;
        if(deg[u]<k)
        del(u);
    }
}
int main()
{
    int n=read_int(),m=read_int();
    k=read_int();
    ans=n;
    _rep(i,1,n)live[i]=true;
    _for(i,0,m){
        int u=read_int(),v=read_int();
        edge[i]=Edge{u,v};
        g[u].push_back(make_pair(v,i));
        g[v].push_back(make_pair(u,i));
        deg[u]++;
        deg[v]++;
    }
    _rep(i,1,n){
        if(live[i]&&deg[i]<k)
```

```
            del(i);
        }
        stack<int>s;
        for(int i=m-1;i>=0;i--){
            s.push(ans);
            if(edge[i].vis)continue;
            edge[i].vis=true;
            update(edge[i].u);
            update(edge[i].v);
        }
        while(!s.empty()){
            enter(s.top());
            s.pop();
        }
        return 0;
}
```

3□CDN流量调度问题

2021CCPC华为云挑战赛 1003

题意

给定 n 个网络和 m 个额外结点。第 i 个网络初始有一个结点，且最多被分配 b_i 个结点，当被分配 k 个结点时，费用为 $\lceil\frac {a_i}k\rceil$□

最小化所有网络的费用和。

题解

易知使得每个网络的费用发生变化的 k 只有 $O(\sqrt {a_i})$ 个，可以 $O\left(\sum_{i=1}^n a_i\right)$ 暴力预处理。

然后设 $\text{dp}(i,j)$ 表示只考虑前 i 个网络花费 j 个额外结点的最小费用，然后状态转移只枚举有效的 k□

总时间复杂度 $O(nm\sqrt a)$□个人感觉时间复杂度过大，但题目测试数据貌似比较水，所以可以过。

ps. 本人一开始考虑优先队列，贪心取变化量最大的网络加结点，但实际受取整函数影响，变化量不是单调的，因此假了。