基环树

算法简介

基环树由普通树额外增加一条边组成，是图论经典问题之一。

算法例题

例题一

CF711D

题意

给定一个有向图(不保证连通)，每个点有一条出边。问有多少种方案能将边重定向，使得图中不存在有向环。

题解

首先不难得出结论每个点一条出边的图正好构成基环树森林。

单独考虑每棵基环树，发现只要基环树上的环上所有边不同向即可，其余边方向任意。设第 i 个基环树的环长度为 w_i则答案为

$$ 2^{n-\sum w_i}\prod (2^{w_i}-2) $$

```
const int MAXN=2e5+5,mod=1e9+7;
struct Edge{
    int to,id,next;
}edge[MAXN<<1];
bool vis[MAXN<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v,int id){
    edge[++edge_cnt]=Edge{v,id,head[u]};
    head[u]=edge_cnt;
}
int d[MAXN],cyc_len;
void dfs(int u,int dep){
    if(!d[u])
    d[u]=dep;
    else{
        cyc_len=dep-d[u];
        return;
    }
```

```
    for(int i=head[u];i;i=edge[i].next){
        if(vis[edge[i].id])continue;
        vis[edge[i].id]=true;
        int v=edge[i].to;
        dfs(v,dep+1);
    }
}
int pw[MAXN];
int main()
{
    int n=read_int();
    pw[0]=1;
    _rep(i,1,n){
        pw[i]=(pw[i-1]<<1)%mod;
        int u=i,v=read_int();
        Insert(u,v,i);
        Insert(v,u,i);
    }
    int ans=1,s=0;
    _rep(i,1,n){
        if(!d[i]){
            dfs(i,1);
            ans=1LL*ans*(pw[cyc_len]+mod-2)%mod;
            s+=cyc_len;
        }
    }
    ans=1LL*ans*pw[n-s]%mod;
    enter(ans);
    return 0;
}
```

例题二

洛谷p2607

题意

给定 n 个物品，每个物品有一个权值 w_i同时第 i 个物品不能和第 p_i 个物品同时选择。问能得到的最大权值。

题解

问题等价于基环树森林的最大权独立集。

考虑每个基环树，任取环上的一条边，记为 (u,v)删除这条边，然后跑强制不取 u 情况下的最大权独立集和强制不取 v 情况下的最大权独立集。

于是每个基环树的贡献为上述两种情况下的最大者，可以通过树形 dp 计算，最后答案为所有基环树贡献相加。

关于找 (u,v) 边可以通过并查集，而强制不取可以将点权设为 $-\text{inf}$ 或者将不取的点为根节点进行 dp

```
const int MAXN=1e6+5;
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 p[MAXN],w[MAXN];
int Find(int x){
    return x==p[x]?x:p[x]=Find(p[x]);
}
LL dp[MAXN][2];
void dfs(int u,int fa){
    dp[u][0]=0;
    dp[u][1]=w[u];
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        dfs(v,u);
        dp[u][0]+=max(dp[v][0],dp[v][1]);
        dp[u][1]+=dp[v][0];
    }
}
vector<pair<int,int> >del;
int main()
{
    int n=read_int();
    _rep(i,1,n)p[i]=i;
    _rep(u,1,n){
        w[u]=read_int();
        int v=read_int();
        int x=Find(u),y=Find(v);
        if(x!=y){
            Insert(u,v);
            Insert(v,u);
            p[x]=y;
        }
        else
        del.push_back(make_pair(u,v));
    }
    LL ans=0;
    _for(i,0,del.size()){
```

```
        int u=del[i].first,v=del[i].second;
        dfs(u,0);
        LL t=dp[u][0];
        dfs(v,0);
        t=max(t,dp[v][0]);
        ans+=t;
    }
    enter(ans);
    return 0;
}
```