

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];
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;
}
```

例题三

[洛谷p4381](#)

题意

题目大意就省略了，大概就是给定基环树森林，求所有基环树的直径之和。

题解

考虑每棵基环树的情况，首先找到环，先不考虑环上的边，求出环上每个点的子树的直径的最大值，这是可能的答案之一。

另外求出环上每个点到它子树的叶子节点的最远距离 d 设环长为 s 任取环上的点对 (u,v) 于是另一种答案为

$$\max(d(u)+d(v)+\text{dis}(u,v),d(u)+d(v)+s-\text{dis}(u,v))$$

考虑规定一个环上的起始点，于是 $\text{dis}(u,v)=\text{pre}(u)-\text{pre}(v)$ 维护两个前缀 \max 即可 $O(n)$ 统计所有点对贡献。

```
const int MAXN=1e6+5;
const LL inf=1e16;
struct Edge{
    int to,w,id,next;
}edge[MAXN<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v,int w,int id){
    edge[++edge_cnt]=Edge{v,w,id,head[u]};
    head[u]=edge_cnt;
}
bool edge_vis[MAXN],node_vis[MAXN],node_cyc[MAXN];
LL d[MAXN],cyc_len;
vector<int> cyc;
bool dfs1(int u,LL dep){
```

```

    if(node_vis[u]){
        cyc_len=dep-d[u];
        cyc.push_back(u);
        return node_cyc[u]=true;
    }
    d[u]=dep;
    node_vis[u]=true;
    bool flag=false;
    for(int i=head[u];i;i=edge[i].next){
        if(edge_vis[edge[i].id])continue;
        edge_vis[edge[i].id]=true;
        int v=edge[i].to;
        if(dfs1(v,dep+edge[i].w)&&!node_cyc[u]){
            cyc.push_back(u);
            node_cyc[u]=true;
            flag=true;
        }
    }
    return flag;
}
LL dfs2(int u,int fa){
    LL ans=0;
    d[u]=0;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa||node_cyc[v])continue;
        ans=max(ans,dfs2(v,u));
        ans=max(ans,d[u]+d[v]+edge[i].w);
        d[u]=max(d[u],d[v]+edge[i].w);
    }
    return ans;
}
LL pre[MAXN];
LL solve(int u){
    cyc.clear();
    dfs1(u,0);
    for(int i=1,j=cyc.size()-1;i<j;i++,j--)swap(cyc[i],cyc[j]);
    _for(i,1,cyc.size())
    pre[cyc[i]]=d[cyc[i]]-d[cyc[0]];
    LL ans=0,max1=-inf,max2=-inf;
    _for(i,0,cyc.size()){
        ans=max(ans,dfs2(cyc[i],0));
        ans=max(ans,d[cyc[i]]+pre[cyc[i]]+max1);
        ans=max(ans,d[cyc[i]]+cyc_len-pre[cyc[i]]+max2);
        max1=max(max1,d[cyc[i]]-pre[cyc[i]]);
        max2=max(max2,d[cyc[i]]+pre[cyc[i]]);
    }
    return ans;
}
int main()
{

```

```
int n=read_int();
_rep(u,1,n){
    int v=read_int(),w=read_int();
    Insert(u,v,w,u);
    Insert(v,u,w,u);
}
LL ans=0;
_rep(u,1,n){
    if(!node_vis[u])
        ans+=solve(u);
}
enter(ans);
return 0;
}
```

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

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E5%9F%BA%E7%8E%AF%E6%A0%91&rev=1626434302

Last update: 2021/07/16 19:18