**点分治 – 统计树上路径权值个数 O(nlog^2n)**

#define hh(u,i) for(int i=h[u];i;i=e[i].nt)

int h[N],sz[N],mx[N],vis[N],a[N];//数组a存路径权值

int cnt,tot,sum,rt;//边数,当前路径数,树的大小,重心

ll ans;

struct edge{

int to,nt,w;

}e[N<<1];

inline void add(int u,int v,int w){

e[++cnt]={v,h[u],w},h[u]=cnt;e[++cnt]={u,h[v],w},h[v]=cnt;

}

void grt(int u,int fa){ //求重心

mx[u]=0,sz[u]=1;hh(u,i){int v=e[i].to;if(v==fa||vis[v]) continue;

grt(v,u),mx[u]=max(mx[u],sz[v]),sz[u]+=sz[v];

}mx[u]=max(mx[u],sum-sz[u]);if(mx[u]<mx[rt]) rt=u;

}

void dfs(int u,int fa,int d){

a[++tot]=d;

hh(u,i){int v=e[i].to;if(v==fa||vis[v]) continue;

dfs(v,u,d+e[i].w);

}

}

////思路1,容斥计算

int cal(int u,int d){ //计算<=k的路径贡献

tot=0,dfs(u,0,d);int s=0;

sort(a+1,a+tot+1);

for(int l=1,r=tot;l<r;l++){

while(l<r&&a[l]+a[r]>k) r--;

s+=r-l;//这里没有+1 是因为要去掉(l->l) 这条路径

}

return s;

}

void solve(int u){

vis[u]=1,ans+=cal(u,0);hh(u,i){int v=e[i].to;if(vis[v]) continue;

ans-=cal(v,e[i].w),mx[rt=0]=sum=sz[v],grt(v,u),solve(rt);

}

}

///思路2　维护两个数组,求权值为m倍数的路径个数 最后答案ans=ans\*2+n; 有序+孤点,具体题目具体对待

void dfs(int u,int fa,int d){

++tmp[d];

hh(u,i){int v=e[i].to;

if(v==fa||vis[v]) continue;

dfs(v,u,(d+e[i].w)%m);

}

}

void cal(int u){

for(int i=0;i<m;i++) now[i]=0;now[0]=1;hh(u,i){ int v=e[i].to;if(vis[v]) continue;

for(int j=0;j<m;j++) tmp[j]=0;

dfs(v,u,e[i].w%m);

for(int j=0;j<m;j++) ans+=(tmp[j]\*now[(m-j)%m]);

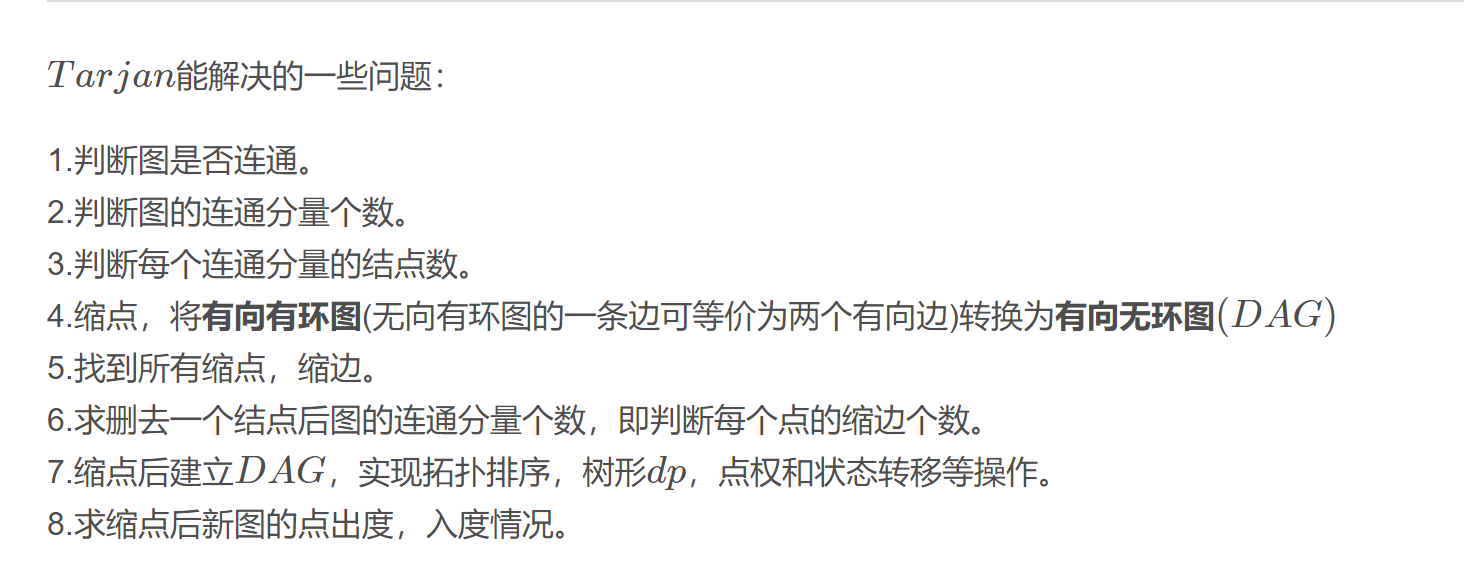
for(int j=0;j<m;j++) now[j]+=tmp[j];

}

}

TarJan





int n,m,dfn[N],low[N],id,vis[N],ans,col[N],num[N],sum=0;

vector<int>e[N]; //dfs[i]记录结点i遍历顺序,low[i]记录结点i及其子结点最小遍历顺序,vis[i]标记是否在栈中。

stack<int>s; //col[i]记录结点i属于那个连通块(本题没用),num[i]记录第i个连通块的结点数.

void dfs(int u){

dfn[u]=low[u]=++id;//记录dfs顺序

s.push(u);//入栈

vis[u]=1;//标记入栈.

for(auto v:e[u]){

if(!dfn[v]){

dfs(v); //没有遍历的点进行遍历然后更新low

low[u]=min(low[u],low[v]);

}

else if(vis[v]) low[u]=min(low[u],dfn[v]);//如果已经遍历过而且在栈中 则取low较小值(这里是潜在的连通块)

}

if(dfn[u]==low[u]){ //该点是连通块的"根" 注意孤立点也是连通块.

vis[u]=0; //回溯标记，并染色.

col[u]=++sum;

num[sum]++;

while(s.top()!=u){ //依次出栈.

col[s.top()]=sum;

vis[s.top()]=0;

s.pop();

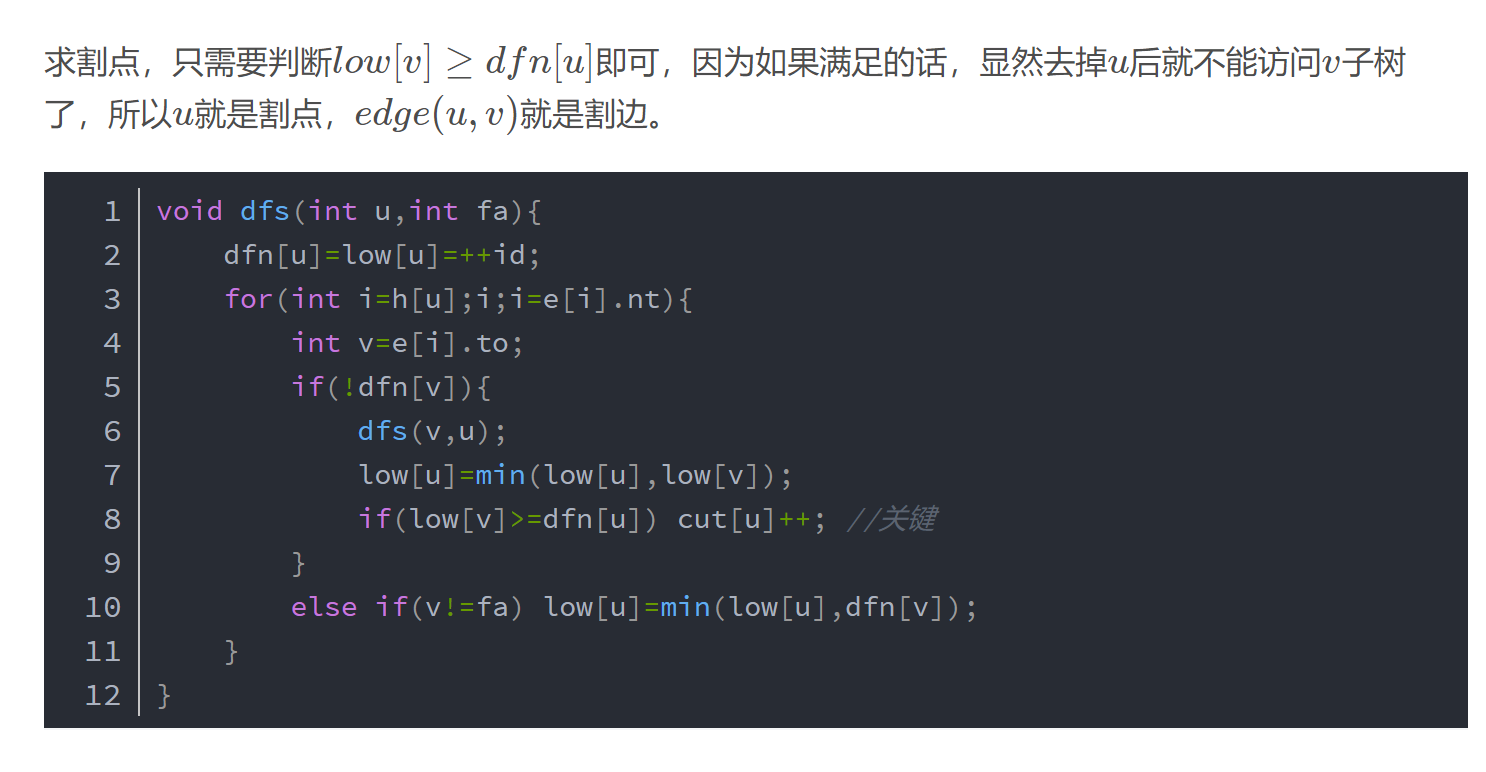
num[sum]++;

}

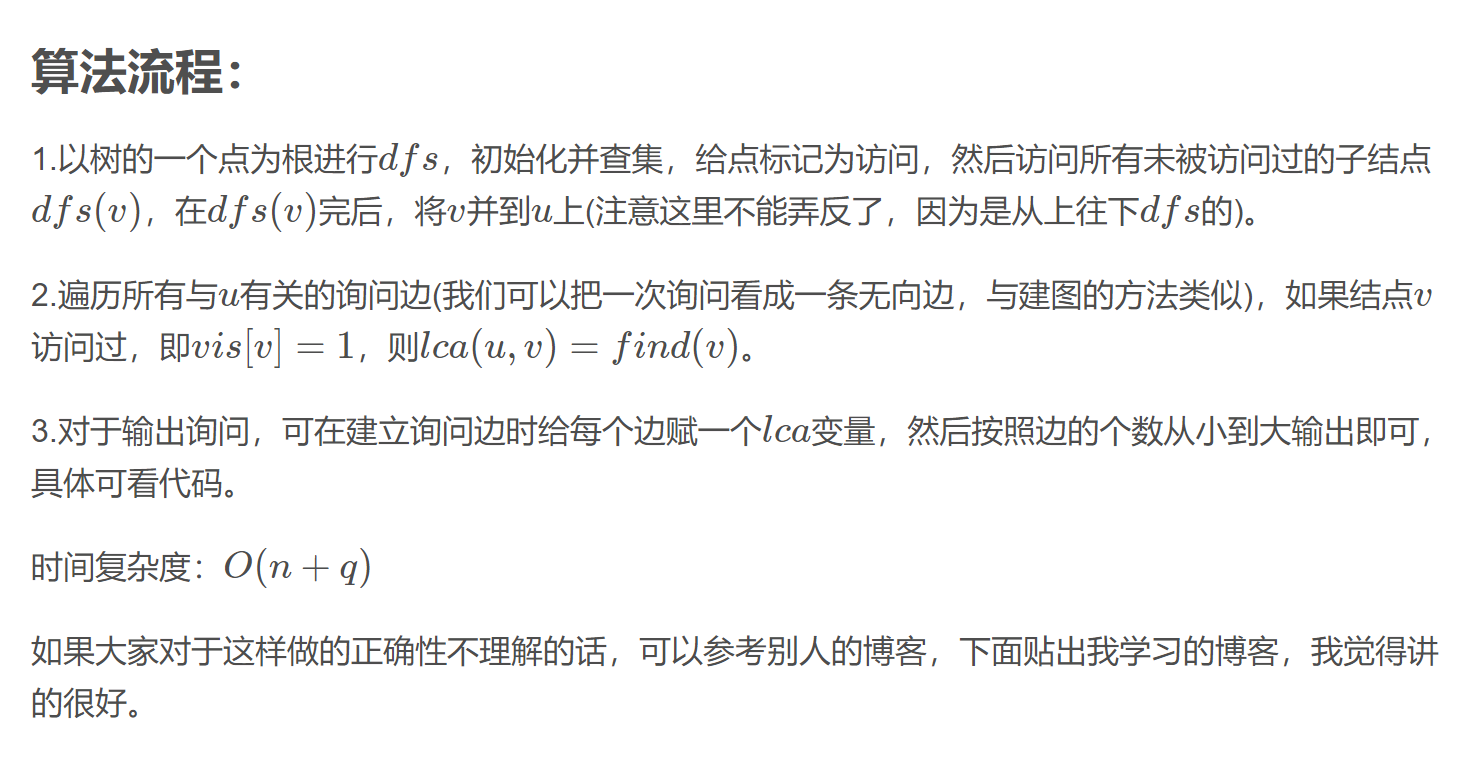
s.pop();

}

}



Tarjan LCA



#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N=5e5+5,M=2e4+5,inf=0x3f3f3f3f,mod=1e9+7;

#define mst(a,b) memset(a,b,sizeof a)

#define PII pair<int,int>

#define fi first

#define se second

#define pb push\_back

int h[N],h1[N],cnt1,cnt2;

int n,m,st,q;

int s[N],vis[N];

int find(int x){ //查找

return s[x]==x?x:s[x]=find(s[x]);

}

inline void merge(int x,int y){ //合并

x=find(x),y=find(y);

if(x!=y) s[y]=x;

}

struct node{

int to,nt,lca;

}e[N<<1],e1[N<<1];

inline void add(int u,int v){ //建图

e[++cnt1]={v,h[u]},h[u]=cnt1;

e[++cnt1]={u,h[v]},h[v]=cnt1;

}

inline void add1(int u,int v){ //询问边建图

e1[++cnt2]={v,h1[u]},h1[u]=cnt2;

e1[++cnt2]={u,h1[v]},h1[v]=cnt2;

}

void dfs(int u){ //dfs

s[u]=u,vis[u]=1;

for(int i=h[u];i;i=e[i].nt){

int v=e[i].to;

if(vis[v]) continue;

dfs(v),merge(u,v);

}

for(int i=h1[u];i;i=e1[i].nt){

int v=e1[i].to;

if(vis[v]) e1[i].lca=find(v);

}

}

int main(){

scanf("%d%d%d",&n,&m,&st);

for(int i=1,u,v;i<n;i++){

scanf("%d%d",&u,&v);add(u,v);

}

for(int i=1,u,v;i<=m;i++){

scanf("%d%d",&u,&v),add1(u,v);

}

dfs(st);

for(int i=1;i<=cnt2;i+=2){ //输出

printf("%d\n",!e1[i].lca?e1[i+1].lca:e1[i].lca); //这里值得注意，可能只有正向边或者反向边lca被赋值了

}

return 0;

}