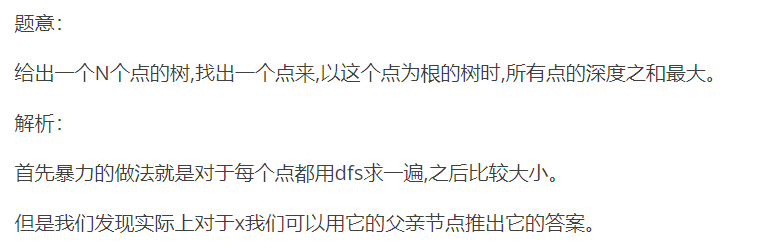
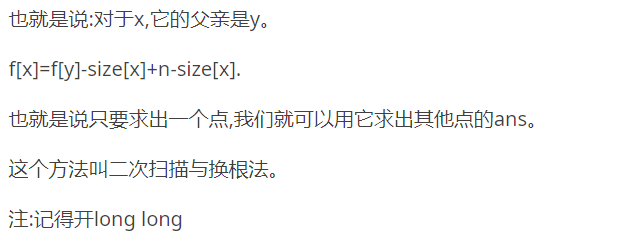
二次扫描法也叫换根dp，一般第一次扫描考虑子树内的贡献，第二次扫描考虑子树外/全局的贡献，一般这种题的特征是，你会发现，如果只用一次dfs，则因为要考虑父亲那边的贡献而无法作答，这个时候就需要使用两次dfs，第二次的时候将父亲那边的part相当于一个子树才可以计算出答案

例题：P3478 [POI2008] STA-Station



#include<bits/stdc++.h>

#define ll long long

#define inf 1e18

#define mn 1000005

using namespace std;

ll f[mn],h[mn],sz[mn],d[mn],

ans=0,cnt=0,n,m,p;

struct w{ll next,to;}a[mn\*2];

void add(ll x,ll y)

{

a[++cnt]=(w){h[x],y};

h[x]=cnt;

}

void dfs(ll x,ll prt)//只管子树

{ll i,y,j;

sz[x]=1;

for(i=h[x];i;i=a[i].next)

{

y=a[i].to;

if(y==prt)continue;

d[y]=d[x]+1;

dfs(y,x);

sz[x]+=sz[y];

}

}

void dfs2(ll x,ll prt)//再更新父亲

{ll i,y,j;

for(i=h[x];i;i=a[i].next)

{

y=a[i].to;

if(y==prt)continue;

f[y]=f[x]-sz[y]+n-sz[y];

dfs2(y,x);

}

}

int main()

{

ll te,x,y,i,j,k,z;

char ch;

cin>>n;

for(i=1;i<n;i++)

{

scanf("%lld%lld",&x,&y);

add(x,y);

add(y,x);

}

dfs(1,0);

for(i=1;i<=n;i++)f[1]+=d[i];

dfs2(1,0);

ll maxx=0;

for(i=1;i<=n;i++)

if(f[i]>maxx){maxx=f[i];ans=i;}

cout<<ans;

return 0;

}