树上问题

基本概念

树的深度：从树根往下数，叶子节点所在的最大层数称为树的深度。（有教材定义不一样，看题意吧。要区分高度）

树的直径：树中两节点距离的最大值称为树的直径。

子树大小：对每一个节点的大小都等于左子树大小+右子树大小+1（自己）

节点重量：删去该节点后，所有树大小的最大值。即该节点所有子树大小的最大值称为节点重量。

求树的直径

做法1：两遍dfs

```
const int N = 10000 + 10;

int n, c, d[N];
vector<int> E[N];

void dfs(int u, int fa) {
  for (int v : E[u]) {
    if (v == fa) continue;
    d[v] = d[u] + 1;
    if (d[v] > d[c]) c = v; //找到最深的点
    dfs(v, u);
  }
}

int main() {
  scanf("%d", &n);
  for (int i = 1; i < n; i++) {
    int u, v;
    scanf("%d %d", &u, &v);
    E[u].push_back(v); //邻接表存图
```

```
    E[v].push_back(u);
  }
  dfs(1, 0); //任意选1号点作为根节点出发
  d[c] = 0;
  dfs(c, 0); //以里1号点最远的点作为根，再次找最远的点。
  printf("%d\n", d[c]);
  return 0;
}
```

做法2：树形dp

```
const int N = 10000 + 10;

int n, d = 0;
int d1[N], d2[N];
vector<int> E[N];

void dfs(int u, int fa) {
  d1[u] = d2[u] = 0; //对于每一个点u，而言，d1和d2是从不同边出发的
  for (int v : E[u]) {
    if (v == fa) continue;
    dfs(v, u); //先搜索到每一个叶子节点
    int t = d1[v] + 1; //再向上更新高度
    if (t > d1[u])
      d2[u] = d1[u], d1[u] = t; //记录最大深度和次大深度
    else if (t > d2[u])
      d2[u] = t;
  }
  d = max(d, d1[u] + d2[u]); //更新答案
}

int main() {
  scanf("%d", &n);
  for (int i = 1; i < n; i++) {
    int u, v;
    scanf("%d %d", &u, &v);
    E[u].push_back(v); //邻接表存图
    E[v].push_back(u);
  }
  dfs(1, 0); //以1作为根节点向下遍历
  printf("%d\n", d);
  return 0;
}
```

最近公共祖先（LCA）

倍增算法 O(mlogn)

```
void dfs(int u,int fa){  //dfs求深度
    dep[u]=dep[fa]+1;
    p[u][0]=fa;
    for(int i=1;(1<<i)<=dep[u];i++)
        p[u][i]=p[p[u][i-1]][i-1];
```

```
    for(int i=head[u];i;i=edge[i].next){
        int to=edge[i].to;
        if(to!=fa) dfs(to,u);
    }
}

int lca(int x,int y){
    if(dep[x]>dep[y]) swap(x,y);
    for(int i=20;i>=0;i--){
        if(dep[x]<=dep[y]-(1<<i)) y=p[y][i];
    }//统一深度
    if(x==y) return x;
    for(int i=20;i>=0;i--){
        if(p[x][i]==p[y][i]) continue;
        else x=p[x][i],y=p[y][i];
    }
    return p[x][0];
}
```

Tarjan算法 O(n+(2*m))

将所有询问存储起来，再dfs中离线处理。

```
void tarjan(int u) {
  fa[u] = u;
  vis[u] = 1;
  for(int i=head[u];i!=-1;i=e[i].n){
    Edge& e=e[i];
    if(!vis[e.v]){
      tarjan(e.v);
      fa[e.v]=u;
    }
  }
  for (int i=queryHead[u];i!=-1;i=query[i].n){
    Edge& e=query[i];
    if(vis[e.v]){
      query[i^1].LCA=e.LCA=find(e.v);
    }
  }
}
```

树链剖分

```
int sz[N],fa[N],son[N],val[N],dep[N],dfn[N],idx,idfn[N],bel[N];
void dfs1(int x){
    sz[x]=1;
    for(int i=head[x];i;i=e[i].n){
        int to=e[i].v;
        if(to==fa[x]) continue;
        fa[to]=x;
        val[to]=e[i].w;
        dep[to]=dep[x]+1;
        dfs1(to);
```

```
        sz[x]+=sz[to];
        if(sz[son[x]]<sz[to]) son[x]=to;
    }
}

void dfs2(int x,int tp){
    dfn[x]=++idx;
    idfn[idx]=x;
    bel[x]=tp;
    if(son[x]) dfs2(son[x],tp);
    for(int i=head[x];i;i=e[i].n){
        int to=e[i].v;
        if(to==son[x]||to==fa[x]) continue;
        dfs2(to,to);
    }
}
int LCA(int x,int y){
    while(bel[x]!=bel[y]){
        if(dep[bel[x]]<dep[bel[y]]) swap(x,y);
        x=fa[bel[x]];
    }
    return dep[x]<dep[y]?x:y;
}
```

RMQ算法

用st表实现，去倍增算法类似，可在线。思想是找区间dfs序最小的那个点。

```
void ST(int n){
    for(int i=1;i<=n;i++) dp[i][0]=i;
    for(int j=1;(1<<j)<=n;j++){
        for(int i=1;i+(1<<j)-1<=n;i++){
            int a=dp[i][j-1],b=dp[i+(1<<(j-1))][j-1];
            dp[i][j]=R[a]<R[b]?a:b;
        }
    }
}

int RMQ(int l,int r){
    int k=0;
    while((1<<(k+1))<=r-l+1) k++;
    int a=dp[l][k],b=dp[r-(1<<k)+1][k];
    return R[a]<R[b]?a:b;
}

int LCA(int u,int v){
    if(dfn[u]>dfn[v]) swap(u,v);
    return val[RMQ(u,v)];
}
```

求树的重心

对于树上的每一个点，计算其所有子树中最大的子树节点数，这个值最小的点就是这棵树的重心。

以树的重心为根时，所有子树的大小都不超过整棵树大小的一半。

在 DFS 中计算每个子树的大小，记录“向下”的子树的最大大小，利用总点数 - 当前子树（这里的子树指有根树的子树）的大小得到“向上”的子树的大小，然后就可以依据定义找到重心了。

```
// 这份代码默认节点编号从 1 开始，即 i ∈ [1,n]
int size[MAXN],  // 这个节点的“大小”（所有子树上节点数 + 该节点）
    weight[MAXN],  // 这个节点的“重量”
    centroid[2];   // 用于记录树的重心（存的是节点编号）
void GetCentroid(int cur, int fa) {  // cur 表示当前节点 (current)
  size[cur] = 1;
  weight[cur] = 0;
  for (int i = head[cur]; i != -1; i = e[i].nxt) {
    if (e[i].to != fa) {  // e[i].to 表示这条有向边所通向的节点。
      GetCentroid(e[i].to, cur);
      size[cur] += size[e[i].to];
      weight[cur] = max(weight[cur], size[e[i].to]);
    }
  }
  weight[cur] = max(weight[cur], n - size[cur]);
  if (weight[cur] <= n / 2) {  // 依照树的重心的定义统计
    centroid[centroid[0] != 0] = cur;
  }
}
```

带权树的重心

参考luoguP1364医院设置

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
using namespace std;
const int maxn=10010;
struct E{
    int next,to;
}edge[maxn<<1];

int head[maxn],cnt,w[maxn],n,size[maxn];
long long ans=inf,f[maxn],a,b;

inline void addedge(int from,int to){
    edge[++cnt].next=head[from];
    edge[cnt].to=to;
    head[from]=cnt;
}

void dfs(int node,int fa,int dep){
    size[node]=w[node];
    for(int i=head[node];i;i=edge[i].next)
        if(edge[i].to!=fa)
            dfs(edge[i].to,node,dep+1),size[node]+=size[edge[i].to];
    f[1]+=w[node]*dep;
}
```

```
void dp(int node,int fa){
    for(int i=head[node];i;i=edge[i].next)
        if(edge[i].to!=fa)
            f[edge[i].to]=f[node]+size[1]-
size[edge[i].to]*2,dp(edge[i].to,node);
    ans=min(ans,f[node]);
}

int main(){
    ans*=ans;
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>w[i]>>a>>b;
        if(a){
            addedge(i,a);
            addedge(a,i);
        }
        if(b){
            addedge(i,b);
            addedge(b,i);
        }
    }
    dfs(1,0,0);
    dp(1,0);
    cout<<ans<<endl;
    return 0;
}
```

例题

[AHOI2008]紧急集合

给出一个有 n 个结点的树，有 q 次询问，每次询问给三个点 A, B, C，要求找出一个集合地点 O，使得三个点到 O 的距离和最小。n, q <= 500000

对于一个询问两两求*lca*，可以证明最优点在三者之中。

```
#include<bits/stdc++.h>
using namespace std;
const int N=5e5+7;

int n,m;
vector<int>G[N];

int sz[N],dep[N],son[N],fa[N],bel[N],dfn[N],idfn[N],idx=0;
void dfs(int x,int f){
    sz[x]=1;
    for(auto to:G[x]){
        if(to==f) continue;
        fa[to]=x;
        dep[to]=dep[x]+1;
        dfs(to,x);
        sz[x]+=sz[to];
        if(sz[to]>sz[son[x]]) son[x]=to;
```

```
    }
}

void dfs2(int x,int tp){
    dfn[x]=++idx;idfn[idx]=x;
    bel[x]=tp;
    if(son[x]) dfs2(son[x],tp);
    for(auto to:G[x]){
        if(to==fa[x]||to==son[x]) continue;
        dfs2(to,to);
    }
}

int lca(int x,int y){
    while(bel[x]!=bel[y]){
        if(dep[bel[x]]<dep[bel[y]]) swap(x,y);
        x=fa[bel[x]];
    }
    return dep[x]>dep[y]?y:x;
}

int X,Y,Z;
int check(int m){
    int s1=dep[m]+dep[X]-2*dep[lca(m,X)];
    int s2=dep[m]+dep[Y]-2*dep[lca(m,Y)];
    int s3=dep[m]+dep[Z]-2*dep[lca(m,Z)];
    return s1+s2+s3;
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>m;
    for(int i=1,u,v;i<n;++i){
        cin>>u>>v;
        G[u].emplace_back(v);
        G[v].emplace_back(u);
    }
    dep[1]=1;fa[1]=0;dfs(1,0);dfs2(1,1);
    for(int i=1;i<=m;++i){
        cin>>X>>Y>>Z;
        int la=lca(X,Y),lb=lca(X,Z),lc=lca(Y,Z);
        int ca=check(la),cb=check(lb),cc=check(lc);
        if(ca<=cb&&ca<=cc){
            cout<<la<<" "<<ca<<"\n";
        }else if(cb<=ca&&cb<=cc){
            cout<<lb<<" "<<cb<<"\n";
        }else cout<<lc<<" "<<cc<<"\n";
    }
    return 0;
}
```

2022牛客多校3 A - Ancestor

给定两颗根节点均为 1 的树 A, B以及每个点的权值，对于一个包含 k 个点的序列，求在序列中选取恰好一个点，除去该点剩下的点在 A 上的 lca 的权值严格大于下的点在 B 上的 lca 的权值。

注意到lca是可以合并的，对于两棵树各维护一个lca的前缀和和后缀和就做完了。

```
//#pragma GCC optimize("Ofast", "inline", "-ffast-math")
//#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
using namespace std;
const int N=1e5+7;
const int mod=1e9+7;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}
void write(int x){if(x>9) write(x/10);putchar(x%10+'0');}
int n,k,x[N],is[N];

struct Tree{
    int val[N],dep[N],fa[N],sz[N],siz[N],son[N],top[N],suf[N],pre[N];
    vector<int>G[N];
    inline void dfs1(int x){
        if(is[x]) sz[x]=1;
        siz[x]=1;
        for(auto to:G[x]){
            if(to==fa[x]) continue;
            fa[to]=x;
            dep[to]=dep[x]+1;
            dfs1(to);
            sz[x]+=sz[to];
            siz[x]+=siz[to];
            if(sz[son[x]]<sz[to]) son[x]=to;
        }
    }
    inline void dfs2(int x,int tp){
        top[x]=tp;
        if(son[x]) dfs2(son[x],tp);
        for(auto to:G[x]){
            if(to!=fa[x]&&to!=son[x]) dfs2(to,to);
        }
    }
    inline int LCA(int x,int y){
        while(top[x]!=top[y]){
            if(dep[top[x]]>dep[top[y]])x=fa[top[x]];
            else y=fa[top[y]];
        }
        return dep[x]<dep[y]?x:y;
    }
    inline void init(){
        for(int i=1;i<=n;++i) sz[i]=fa[i]=dep[i]=suf[i]=pre[i]=0;
        dfs1(1);
        dfs2(1,1);
        pre[1]=x[1],suf[k]=x[k];
        for(int i=2;i<=k;++i){
            pre[i]=LCA(pre[i-1],x[i]);
```

```
        }
        for(int i=k-1;i>=1;--i){
            suf[i]=LCA(suf[i+1],x[i]);
        }
    }
}A,B;


void Solve(){
    n=read();
    k=read();
    for(int i=1;i<=k;++i){
        x[i]=read();
        is[x[i]]=1;
    }
    for(int i=1;i<=n;++i) A.val[i]=read();
    for(int i=2,y;i<=n;++i){
        y=read();
        A.G[i].emplace_back(y);
        A.G[y].emplace_back(i);
    }
    for(int i=1;i<=n;++i) B.val[i]=read();
    for(int i=2,y;i<=n;++i){
        y=read();
        B.G[i].emplace_back(y);
        B.G[y].emplace_back(i);
    }
    A.init();B.init();
    int ans=0;
    if(A.val[A.suf[2]]>B.val[B.suf[2]]) ++ans;
    if(A.val[A.pre[k-1]]>B.val[B.pre[k-1]]) ++ans;
    for(int i=2;i<k;++i){
        if(A.val[A.LCA(A.pre[i-1],A.suf[i+1])]>B.val[B.LCA(B.pre[i-
1],B.suf[i+1])]){
            ++ans;
        }
    }
    write(ans);putchar('\n');
}

signed main(){
    int T=1;
//  cin>>T;
//  clock_t start,finish;
//  start=clock();
    while(T--){
        Solve();
    }
//  finish=clock();
//  cerr<<((double)finish-start)/CLOCKS_PER_SEC<<endl;
    return 0;
}
```

参考资料

https://oi-wiki.org/graph/tree-centroid/

https://blog.csdn.net/liangzhaoyang1/article/details/52549822