结论

1、树上最远距离

树上到每个点距离最远的距离一定为树上一条直径的两个端点之一。

分别从两个端点开始 dfs 即可 $O(n)$ 求取每个点的树上最远点。

证明见 链接

或者可以考虑树形 dp第一次 dfs 维护每个节点子树方向上的最远距离和次远距离。

第二次 dfs 维护每个节点祖先方向上的最远距离，答案即为子树方向上的最远距离和祖先方向上的最远距离的较大者。

```
int dp[MAXN][3],hson[MAXN],dis[MAXN];
void dfs1(int u,int fa){
    dp[u][0]=dp[u][1]=dp[u][2];
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        dfs1(v,u);
        if(dp[v][0]+edge[i].w>dp[u][0]){
            hson[u]=v;
            dp[u][1]=dp[u][0];
            dp[u][0]=dp[v][0]+edge[i].w;
        }
        else if(dp[v][0]+edge[i].w>dp[u][1])
        dp[u][1]=dp[v][0]+edge[i].w;
    }
}
void dfs2(int u,int fa){
    dis[u]=max(dp[u][0],dp[u][2]);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        if(v==hson[u])
        dp[v][2]=edge[i].w+max(dp[u][1],dp[u][2]);
        else
        dp[v][2]=edge[i].w+max(dp[u][0],dp[u][2]);
        dfs2(v,u);
    }
}
```

2、树的遍历代价

给定一棵树，树上有一些关键点，问任意从树上选一点出发，遍历所有关键点后回到起点的最短路径长度

为多少。

建立虚树。先考虑虚树上的叶子结点 v□v 必为关键节点，设 $u=fa(v)$□

发现不管 v 是否为起点，都需要遍历 $edge(u,v)$ 两次(出发与返回)。

而要达到 v 必须达到 u□于是考虑把 v 删去，把 u 作为关键节点，答案加上 $2\times edge(u,v)$□继续重复上述过程。

最终发现答案为虚树上边权和的两倍。

设关键点按原树上的 dfs 序排序后为 $u_1,u_2,u_3\cdots u_k$□

通过画图观察发现虚树上边权和的两倍恰好等于 $\text{dis}(u_1,u_2)+\text{dis}(u_2,u_3)+\cdots\text{dis}(u_{k-1},u_k)+\text{dis}(u_k,u_1)$□

而加入一个点 v 作为新关键点时只需要查询 dfs 序与 v 相邻的两个点，记为 u_i,u_{i+1}□

则答案增量为 $\text{dis}(u_i,v)+\text{dis}(v,u_{i+1})-\text{dis}(u_i,u_{i+1})$□

删除一个关键点操作类似。

3、 连通性

无向图的点双连通分量可以通过给所有边定向转化为强连通分量。