树链剖分+线段树 求解路径边全最小值

将边权下放到点权，以点权代替边权。

```
#include <bits/stdc++.h>

const long long N = 2e6 + 100;
long long n, m;
long long g[N], e[N], ne[N], w[N], ant;

void add(long long x, long long y, long long z)
{
    ant ++;
    e[ant] = y;
    w[ant] = z;
    ne[ant] = g[x];
    g[x] = ant;
}

long long fa[N];    // 节点的父亲
long long hfa[N];   // 重链头
long long hson[N];  // 重儿子
long long deep[N];  // 深度
long long size[N];  // 子树节点个数
long long st[N];    // 子树的起始 id
long long ed[N];    // 子树的末尾 id
long long id[N];    // 节点的 id 编号
long long rid[N];   // id 对应的节点值
long long val[N];   // 节点对应的权值
long long cur;

void dfs(long long u, long long v)
{
    std::cout << u << " ";
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v) continue;
        dfs(e[i], u);
    }
}

void dfs_1(long long u, long long v)
{
    fa[u] = v;
    deep[u] = deep[v] + 1;
    size[u] = 1;
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v) continue;
        val[e[i]] = w[i];
```

```
        dfs_1(e[i], u);
        long long t = size[e[i]];
        size[u] += t;
        if (size[hson[u]] < t)
        {
            hson[u] = e[i];
        }
    }
}

void dfs_2(long long u, long long v)
{
    if (u == 0 || v == 0) return;
    hfa[u] = v;
    st[u] = ++ cur;
    id[u] = cur;
    rid[cur] = u;
    dfs_2(hson[u], v);
    for (long long i = g[u]; i; i = ne[i])
    {
        if (hfa[e[i]]) continue;
        dfs_2(e[i], e[i]);
    }
    ed[u] = cur;
}

// ------------------------------------


long long tr[N], ls[N], rs[N], cnt;
long long root;
long long lf, rf;

void push_up(long long u, long long l, long long r)
{
    if (l == r) return;
    tr[u] = std::min(tr[ls[u]], tr[rs[u]]);
}

void build(long long& u, long long l, long long r)
{
    u = ++ cnt;
    if (l == r)
    {
        tr[u] = val[rid[l]];
        return;
    }

    long long mid = (l + r) >> 1;
    build(ls[u], l, mid);
    build(rs[u], mid + 1, r);

    push_up(u, l, r);
}
```

```
long long lca(long long u, long long v)
{
    if (hfa[u] == hfa[v])
    {
        if (deep[u] > deep[v]) std::swap(u, v);
        return u;
    }

    if (deep[hfa[u]] > deep[hfa[v]]) std::swap(u, v);

    return lca(u, fa[hfa[v]]);
}

long long query(long long u, long long l, long long r, long long ll, long long rr)
{
    if (ll <= l && r <= rr) return tr[u];
    long long mid = (l + r) >> 1;

    long long ans = 1e9;
    if (ll <= mid) ans = std::min(ans, query(ls[u], l, mid, ll, rr));
    if (rr > mid) ans = std::min(ans, query(rs[u], mid + 1, r, ll, rr));

    return ans;
}

long long query(long long u, long long v)
{
    if (hfa[u] == hfa[v])
    {
        if (u == v) return 1e9;
        return query(root, lf, rf, id[v] + 1, id[u]);
    }

    long long ans = query(root, lf, rf, id[hfa[u]], id[u]);
    return std::min(ans, query(fa[hfa[u]], v));
}

void solve()
{

    std::cin >> n >> m;
    for (long long i = 1; i < n; i ++)
    {
        long long x, y, z; std::cin >> x >> y >> z;
        add(x, y, z); add(y, x, z);
    }

    // dfs(1, 0);std::cout << "debug\n"; return;
    dfs_1(1, 0);
    dfs_2(1, 1);

    lf = 1; rf = cur;
    build(root, lf, rf);
```

```
    while (m --)
    {
        long long u, v; std::cin >> u >> v;
        if (deep[u] > deep[v]) std::swap(u, v);
        long long f = lca(u, v);
        if (f == u) std::cout << query(v, u) << '\n';
        else
        {
            long long x = query(u, f);
            long long y = query(v, f);
            std::cout << std::min(x, y) << '\n';
        }
    }

}

int main()
{
    std::ios::sync_with_stdio(0);
    std::cin.tie(0); std::cout.tie(0);


    solve();
    return 0;
}
```