树上分治找准重心然后暴力分治 解决点对距离不超过 k 的个数问题

因为树上分治的递归树，只有 $\log(n)$ 层，并且，每一层的总和只有 n 个计算量，所以时间复杂度是 $n\log(n)$

也因为只有 n 层，你甚至能暴力遍历完子树的所有节点，并排序，时间复杂度也最多只有 $n\log^2(n)$

而启发式合并，他是选择保留重儿子的信息，那么我们可以遍历重儿子留下的信息吗？

是不可以的，如果遍历了重儿子留下的信息，时间复杂度就会退化成 n^2，也正因为保留了重儿子的信息，才能使得时间复杂度优化成 $n\log(n)$

能用启发式做这道题吗？是可以的，但是合并信息时，就需要借助搜索树或者权值线段树，插入信息，查询时也一样

总的时间复杂度也仍然是 $n\log^2(n)$

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

const long long N = 2e5;
const long long inf = 1e9;
long long n, m;
long long g[N], e[N], w[N], ne[N], ant;
long long ans;

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 rt;
long long mx;
long long tree_size;
long long size[N];
long long vis[N];

void get_root(long long u, long long v)
{
    size[u] = 1;
    long long t = 0;
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v || vis[e[i]]) continue;
        get_root(e[i], u);
```

```
        size[u] += size[e[i]];
        t = std::max(t, size[e[i]]);
    }
    t = std::max(t, tree_size - size[u]);
    if (t < mx)
    {
        mx = t; rt = u;
    }
}

std::vector<long long> pa, pb, pt;

void calc()
{
    for (auto x : pb) if (x <= m) ans ++;
    long long i = 0;
    long long j = pb.size() - 1;
    while (i < pa.size() && j >= 0)
    {
        if (pa[i] + pb[j] > m) j --;
        else
        {
            ans += j + 1; i ++;
        }
    }
}

void merge()
{
    pt.clear();
    long long i, j;
    for (i = 0, j = 0; i < pa.size() && j < pb.size();)
    {
        if (pa[i] <= pb[j]) pt.push_back(pa[i ++]);
        else pt.push_back(pb[j ++]);
    }
    while (i < pa.size()) pt.push_back(pa[i ++]);
    while (j < pb.size()) pt.push_back(pb[j ++]);
    pa.swap(pt);
    pb.clear();
}

void get_path(long long u, long long v, long long d)
{
    if (d > m) return;
    pb.push_back(d);
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v || vis[e[i]]) continue;
        get_path(e[i], u, d + w[i]);
    }
}

void dv_solve(long long u, long long v)
```

```
{
    vis[u] = 1; // 重心处分割，分割出子树
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v || vis[e[i]]) continue;
        rt = 0; mx = inf; tree_size = size[e[i]];
        get_root(e[i], u);
        dv_solve(rt, u);
    }

    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v || vis[e[i]]) continue;
        get_path(e[i], u, w[i]);
        std::sort(pb.begin(), pb.end());
        calc();
        merge();
    }

    pa.clear();
    vis[u] = 0;
}

void solve()
{
    std::cin >> n;
    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);
    }
    std::cin >> m;
    rt = 0; mx = inf; tree_size = n;
    get_root(1, 0);
    dv_solve(rt, 0);
    std::cout << ans << '\n';
}

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

    solve();
    return 0;
}
```