暴力启发式合并 解决距离为 k 的点对是否存在问题

`unordered_map` 比 `map` 还要慢上不少!

看数据量而定。

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

const long long N = 2e4 + 100;
long long n, m;
long long g[N], e[N], ne[N], w[N], ant;

inline 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;
}

inline long long read()
{
    register long long x = 0, f = 1;
    register char c = getchar();
    while (!(c >= '0' && c <= '9'))
    {
        if (c == '-') f = -1;
        c = getchar();
    }
    while (c >= '0' && c <= '9')
    {
        x = x * 10 + c - '0';
        c = getchar();
    }
    return x * f;
}


long long hson[N], hsonw[N], size[N];

void dfs_1(long long u, long long v)
{
    size[u] = 1;
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v) continue;
        dfs_1(e[i], u);
        size[u] += size[e[i]];
        if (hson[u] == 0 || size[hson[u]] < size[e[i]])
```

```
        {
            hson[u] = e[i];
            hsonw[u] = w[i];
        }
    }
}

long long k;
std::map<long long, long long> mp;
long long sum;
long long stop;

void find_ans(long long u, long long v, long long s)
{
    if (stop) return;
    if (s > k) return;
    if (s == k)
    {
        stop = 1;
        return;
    }
    if (mp.count(k - s - sum))
    {
        stop = 1;
        return;
    }
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v) continue;
        find_ans(e[i], u, s + w[i]);
    }
}

void add_path(long long u, long long v, long long s)
{
    if (s > k) return;
    if (stop) return;
    register long long t = s - sum;
    mp[t] = 1;
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v) continue;
        add_path(e[i], u, s + w[i]);
    }
}

void dfs_2(long long u, long long v, long long keep)
{
    if (stop) return;
    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v || e[i] == hson[u]) continue;
        dfs_2(e[i], u, 0);
    }
```

```
    if (hson[u])
    {
        dfs_2(hson[u], u, 1);
        if (mp.count(k - sum))
        {
            stop = 1;
            return;
        }
        sum += hsonw[u];
        mp[hsonw[u] - sum] = 1;
        if (mp.count(k - sum))
        {
            stop = 1;
            return;
        }
    }

    for (long long i = g[u]; i; i = ne[i])
    {
        if (e[i] == v || e[i] == hson[u]) continue;
        find_ans(e[i], u, w[i]);
        add_path(e[i], u, w[i]);
    }

    if (keep == 0)
    {
        sum = 0;
        mp.clear();
    }
}

void solve()
{
    n = read(); m = read();
    srand(time(0));
    register long long rt = rand() % n + 1;
    // rt = 1;
    for (long long i = 1; i < n; i ++)
    {
        long long x, y, z; x = read(); y = read(); z = read();
        add(x, y, z); add(y, x, z);
    }
    dfs_1(rt, 0);
    while (m --)
    {
        k = read();
        sum = 0;
        mp.clear();
        stop = 0;
        dfs_2(rt, 0, 0);
        if (stop) printf ("AYE\n");
        else printf ("NAY\n");
    }
```

```
}

int main()
{
    solve();
    return 0;
}
```