2020 CCPC 威海 C Rencontre

树形DP， 到树上确定三点的距离之和最小的点性质，期望的性质

返回

C Rencontre

Located at the easternmost tip of Shandon Peninsula, Weihai is one of the most famous tourist destinations all over China. There are beautiful hills, seas, bays, springs, islands, and beautiful beaches in Weihai. It is also a coastal city abundant in seafood, including prawn, sea cucumber, abalone, shellfish, and algae.

Attracted by the distinctive scenery and pleasant environment, three theoretical computer scientists plan to have a trip to Weihai. However, they cannot reach a consensus on accommodation, since some people prefer some hotels while other people like others. They decide to stay in separate hotels at night and meet in one hotel the next day. The hotel they meet may not necessarily be one of the hotels they stay in.

There are some roads connecting the hotels in Weihai. The roads are specially designed such that there is a unique path between every pair of hotels. Every theoretical computer scientist has prepared a list of candidate hotels before their trip starts. When they arrive in Weihai, each of them will uniformly and independently choose one hotel from the candidate hotel list. Also, they will meet in a hotel such that the total length of their routes is minimized. As a member of the theoretical computer science group, can you tell the expected total length of their routes?

Input Specification:

The first line of the input contains a single integer n ($1 \le n \le 200\,000$), denoting the number of hotels in Weihai. Then follow $n-1$ lines, describing the roads connecting the hotels. Each of the $n-1$ lines contains three integers u, v, w ($1 \le u, v \le n, u \ne v, 1 \le w \le 1000$), denoting a road of length w connecting the hotels numbered u and v. It is guaranteed that there is a unique path between every pair of hotels.

The last three lines of the input specify the candidate hotel lists, one for each theoretical computer scientist. Each line begins with a single integer m ($1 \le m \le n$) and m distinct integers $a_1, a_2, \cdots, a_m$ ($1 \le a_i \le n$), meaning that the candidate hotel list contains the hotels numbered $a_1, a_2, \cdots, a_m$.

Output Specification:

Print the expected total length of their routes within an absolute or relative error of no more than 10^{-6}.

找到一个点 t，使其到树上三点 u_1, u_2, u_3 距离之和最小，求最小距离，即

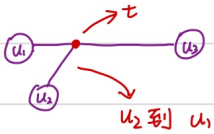
$$dis = \min_{te\ Node} [\ dis(u_1,t) + dis(u_2,t) + dis(u_3,t)\ ]$$

分情况讨论：

① 当 u_2 在 (u_1, u_3) 路径上时：

则易发现 $dis = \frac{1}{2}[\ dis(u_1,u_2) + dis(u_2,u_3) + dis(u_1,u_3)\ ]$

② 当 u_2 不在 (u_1, u_3) 路径上时：



u_2 到 $u_1 \leftrightarrow u_3$ 路径上的最短路径.

则 $dis = dis(u_1,t) + dis(u_2,t) + dis(u_3,t)$

$= \frac{1}{2}[\ dis(u_1,u_2) + dis(u_2,u_3) + dis(u_1,u_3)\ ]$

因此，无论 u_1, u_2, u_3 三点如何放置，均有：

$$dis = \frac{1}{2}[\ dis(u_1,u_2) + dis(u_2,u_3) + dis(u_1,u_3)\ ]$$

因为 $u_1\ u_2, u_3$ 为三个集合中任选的一点，因此原式转换为期望，并根据期望公式 $(E(x+y) = E(x) + E(y))$：

$$dis = E\left\{\frac{1}{2}[\ dis(u_1,u_2) + dis(u_2,u_3) + dis(u_1,u_3)\ ]\right\}$$

$= \frac{1}{2}E[dis(u_1,u_2)] + \frac{1}{2}E[dis(u_2,u_3)] + \frac{1}{2}E[dis(u_1,u_3)]$

考虑 $E[dis(u_1,u_2)]$ 的求法：

$E[dis(u_1,u_2)]$

$= \frac{\sum_{i\in u_1}\sum_{j\in u_2} dis(i,j)}{|u_1|\cdot|u_2|}$，即求树上两个集合间任意两点间的距离和，此问题可转化为另一经典树形DP问题：求树上任意两点间距离和问题

考虑此经典问题，从每条边的贡献出发：

（假设共有n个节点）

预先处理出以任意一点为根的子树大小，则此边贡献为 $son[v]\cdot(n-son[v])\cdot w$

通过考虑每条边贡献，求得任意两点间距离和

回到本题，要求的是两个集合间任意2个点间距离和，因此需将son数组多加一维变为 $son[3][maxn]$ 数组，加 $son[i][u]$ 代表以u为根的子树中类型为i的节点数，则在求一条边 

对集合i,j的贡献：

$(cnt[i]-son[i][v])\cdot son[j][v]\cdot w$

↓

集合i中点的数量

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 2e5 + 10;
struct node{
    int v, w;
};
vector<node> e[maxn];
int son[3][maxn], cnt[3]; //son[i][j]代表第i组中，节点j的子树的大小(包含j自身)
double ans = 0;

void add_edge(int u, int v, int w){
    e[u].push_back({v, w});
    e[v].push_back({u, w});
}
//获取子树大小
void get_son(int u, int fa){
    for(auto i: e[u]){
```

```
        int v = i.v;
        if(v == fa) continue;
        get_son(v, u);
        for(int i = 0; i < 3; i++)
            son[i][u] += son[i][v];
    }
}
//统计答案
void get_ans(int u, int fa){
    for(auto a: e[u]){
        int v = a.v, w = a.w;
        if(v == fa) continue;
        get_ans(v, u);
        for(int i = 0; i < 3; i++) {
            for(int j = 0; j < 3; j++) {
                if(i == j) continue; //勿忘
                ans += 1.0 * (cnt[i] - son[i][v]) * son[j][v] * w / cnt[i] /
cnt[j] / 2;
            }
        }
    }
}

int main(){
    int n, u, v, w, candi;
    cin >> n;
    for(int i = 1; i < n; i++){
        cin >> u >> v >> w;
        add_edge(u, v, w);
    }
    for(int i = 0; i < 3; i++) {
        cin >> cnt[i];
        for(int j = 0; j < cnt[i]; j++){
            cin >> candi;
            son[i][candi]++;
        }
    }
    get_son(1, 0);
    get_ans(1, 0);
    printf("%.9lf\n", ans);
}
```