「学习总结」久月

Jiayi Su (ShuYuMo)

2020-09-06 20:05:35

久月也是个可爱的月份呢 QAQ~

点分治

点分治三连：- Grt(o): 找到结点 o 所在子树的。- solve(o): 在以结点 o 为根的子树中，处理经过 的。- divide(o): 点分治主函数，调用 `solve(o)` 处理以 o 为根 经过点 o 的信息，然后删除点 o ，递归处理其儿子（`divide(ex)`）

注意

- 应保证 `solve(o)` 在执行过程中严格 $O(\text{size}_o)$ 否则会导致总时间复杂度不正确。
- 应保证 两次执行 `Grt()` 第二次调用只是为了确定以重心为根节点的子树大小。
- `solve(o)` 中往往是在考虑所有之前访问过的子树结点 到 当前刚刚访问的结点 之间统计答案。

例题 1

给定一棵有 n 个点的树，询问树上距离为 k 的点对是否存在。

```
int sum;
int si[_N];
bool vis[_N];
bool judge[int(1e7 + 100)];
int qer[_N];
int MX[_N];
int ans[int(1e7 + 100)];
int rt;

void grt(int o, int f)
{
    si[o] = 1;
    MX[o] = 0;
    int MAX = 0;
    for (int i = head[o]; i; i = edge[i].nxt)
    {
        int ex = edge[i].node;
        if (ex == f || vis[ex])
            continue;
        grt(ex, o);
        si[o] += si[ex];
        MAX = max(si[ex], MAX);
    }
```

```
    MAX = max(MAX, sum - si[o]);
    MX[o] = MAX;
    if (MAX < MX[rt])
        rt = o;
}

int cnt = 0;
int tmp[_N];
void GetAllDis(int o, int f, int dis)
{
    if (dis > 1e7)
        return;
    tmp[++cnt] = dis;
    for (int i = head[o]; i; i = edge[i].nxt)
    {
        int ex = edge[i].node;
        if (ex == f || vis[ex])
            continue;
        GetAllDis(ex, o, dis + edge[i].w);
    }
}

queue<int> Q;
void solve(int o)
{
    judge[0] = true;
    for (int i = head[o]; i; i = edge[i].nxt)
    {
        int ex = edge[i].node;
        if (vis[ex])
            continue;
        cnt = 0;
        GetAllDis(ex, 0, edge[i].w);
        for (int j = 1; j <= cnt; j++)
        {
            for (int k = 1; k <= m; k++)
            {
                if (qer[k] >= tmp[j])
                    ans[k] |= judge[qer[k] - tmp[j]];
            }
        }
        for (int j = 1; j <= cnt; j++)
        {
            judge[tmp[j]] = 1;
            Q.push(tmp[j]);
        }
    }
```

```
    while (!Q.empty())
    {
        judge[Q.front()] = 0;
        Q.pop();
    }
}

void divide(int o)
{
    vis[o] = 1;
    solve(o);
    for (int i = head[o]; i; i = edge[i].nxt)
    {
        int ex = edge[i].node;
        if (vis[ex])
            continue;
        sum = si[ex];
        MX[rt = 0] = INT_MAX;
        grt(ex, 0);
        grt(rt, 0);
        divide(rt);
    }
}
int main()
{
    n = read(), m = read();
    for (int i = 1; i < n; i++)
    {
        int u = read(), v = read(), w = read();
        add(u, v, w);
        add(v, u, w);
    }
    for (int i = 1; i <= m; i++)
        qer[i] = read();
    sum = n;
    MX[rt = 0] = INT_MAX;
    grt(1, 0);
    grt(rt, 0);
    divide(rt);
    for (int i = 1; i <= m; i++)
    {
        if (ans[i])
            puts("AYE");
        else
            puts("NAY");
    }
    return 0;
```

```
}
```

例题 **2**

「**luogu-P2634**」「国家集训队」聪聪可可　给出一个带权树，求出所有满足长度为 3 的倍数的简单路径数量。

统计时，统计每个子树内，分别统计 根到子树中结点的路径长度 mod 3 后，不同值的数量。在根节点拼接即可。

```
int n;
bool vis[_N];
namespace GetCenter
{
    int sum;
    int si[_N];
    int MX[_N];
    int rt;
    void dfs(int o, int f)
    {
        si[o] = 1;
        MX[o] = 0;
        int MAX = 0;
        for (int i = head[o]; i; i = edge[i].nxt)
        {
            int ex = edge[i].node;
            if (ex == f || vis[ex])
                continue;
            dfs(ex, o);
            si[o] += si[ex];
            MAX = max(MAX, si[ex]);
        }
        MAX = max(MAX, sum - si[o]);
        MX[o] = MAX;
        if (MX[rt] > MX[o])
            rt = o;
    }
    int grt(int o, int _sum)
    {
        MX[rt = 0] = INT_MAX;
        sum = _sum;
        dfs(o, 0);
        dfs(rt, 0);
        return rt;
    }
} // namespace GetCenter
using GetCenter::grt;
using GetCenter::si;

int ans = 0;
int cnt[3];
```

```
void GetDis(int o, int f, int dis)
{
    cnt[dis % 3]++;
    for (int i = head[o]; i; i = edge[i].nxt)
    {
        int ex = edge[i].node;
        if (ex == f || vis[ex])
            continue;
        GetDis(ex, o, dis + edge[i].w);
    }
}
int Bef[3];
void solve(int o)
{
    Bef[0] = 1;
    for (int i = head[o]; i; i = edge[i].nxt)
    {
        int ex = edge[i].node;
        if (vis[ex])
            continue;
        memset(cnt, 0, sizeof(cnt));
        GetDis(ex, 0, edge[i].w);
        for (int i = 0; i < 3; i++)
            ans += Bef[i] * cnt[(3 - i) % 3] * 2;

        for (int i = 0; i < 3; i++)
            Bef[i] += cnt[i];
    }
    memset(Bef, 0, sizeof(Bef));
}

void divide(int o)
{
    vis[o] = true;
    solve(o);
    for (int i = head[o]; i; i = edge[i].nxt)
    {
        int ex = edge[i].node;
        if (vis[ex])
            continue;
        int rt = grt(ex, si[ex]);
        divide(rt);
    }
}
int main()
{
    n = read();
```

```
    for (int i = 1; i < n; i++)
    {
        int u = read(), v = read(), w = read();
        add(u, v, w);
        add(v, u, w);
    }
    int rt = grt(1, n);
    divide(rt);
    ans += n;
    int A = ans, B = n * n;
    int g = gcd(A, B);
    printf("%d/%d\n", A / g, B / g);
    return 0;
}
```

栗题 3

「luogu-P4178」聪聪可可　给定一棵 n 个节点的树，每条边有边权，求出树上两点距离小于等于 k 的点对数量

对每个儿子统计所有可能的长度，乘法原理以此匹配即可。

```
int vis[_N];
namespace GetCenter
{
    int si[_N];
    int MX[_N];
    int rt;
    int sum;
    void dfs(int o, int f)
    {
        si[o] = 1;
        int MAX = INT_MIN;
        for (int i = head[o]; i; i = edge[i].nxt)
        {
            int ex = edge[i].node;
            if (ex == f || vis[ex])
                continue;
            dfs(ex, o);
            si[o] += si[ex];
            MAX = max(si[ex], MAX);
        }
        MAX = max(MAX, sum - si[o]);
        MX[o] = MAX;
        if (MAX < MX[rt])
            rt = o;
    }
    int grt(int o, int _sum)
    {
```

```
        MX[rt = 0] = INT_MAX;
        sum = _sum;
        dfs(o, 0);
        dfs(rt, 0);
        return rt;
    }
} // namespace GetCenter
using GetCenter::grt;
using GetCenter::si;

int judge[_N], tjt = 0;
int tmp[_N], ttt = 0;
void GetDis(int o, int f, int w)
{
    if (w > k)
        return;
    tmp[++ttt] = w;
    for (int i = head[o]; i; i = edge[i].nxt)
    {
        int ex = edge[i].node;
        if (ex == f || vis[ex])
            continue;
        GetDis(ex, o, w + edge[i].w);
    }
}
int ans = 0;
void solve(int o)
{
    for (int i = head[o]; i; i = edge[i].nxt)
    {
        int ex = edge[i].node;
        if (vis[ex])
            continue;

        ttt = 0;
        GetDis(ex, 0, edge[i].w);

        for (int i = 1; i <= ttt; i++)
        {
            int now = k - tmp[i];
            int Q = (upper_bound(judge + 1, judge + 1 + tjt, now) - judge - 1);
            ans += Q;
        }
        for (int i = 1; i <= ttt; i++)
            judge[++tjt] = tmp[i];
        sort(judge + 1, judge + 1 + tjt);
    }
```

```
    ans += (upper_bound(judge + 1, judge + 1 + tjt, k) - judge - 1);
    tjt = 0;
}

void divide(int o)
{
    vis[o] = true;
    solve(o);
    for (int i = head[o]; i; i = edge[i].nxt)
    {
        int ex = edge[i].node;
        if (vis[ex])
            continue;
        int rt = grt(ex, si[ex]);
        divide(rt);
    }
}

int main()
{
    n = read();
    for (int i = 1; i <= n - 1; i++)
    {
        int u = read(), v = read(), w = read();
        add(u, v, w);
        add(v, u, w);
    }
    k = read();
    int rt = grt(1, n);
    divide(rt);
    printf("%d", ans);
    return 0;
}
```

线段树合并

栗题

「luogu-P4556」「Vani 有约会」雨天的尾巴　首先村落里的一共有 n 座房屋，并形成一个树状结构。然后救济粮分 m 次发放，每次选择两个房屋 (x, y)，然后对于 x 到 y 的路径上 (含 x 和 y) 每座房子里发放一袋 z 类型的救济粮。

然后深绘里想知道，当所有的救济粮发放完毕后，每座房子里存放的最多的是哪种救济粮。

重点是维护树上差分，然后在每一个叶子结点，建立一棵线段树，到达每一个非叶子结点，合并其子节点的线段树。

线段树应动态开点，保证空间复杂度正确。

marge　分情况讨论：

```
int marge(int x, int y, int nowl, int nowr){
    if(!x) return y;
```

```
    if(!y) return x;
    if(nowl == nowr) return (Val[x].Max += Val[y].Max, x);
    int mid = (nowl + nowr) >> 1;
    ls(x) = marge(ls(x), ls(y), nowl, mid);
    rs(x) = marge(rs(x), rs(y), mid + 1, nowr);
    maintain(x);
    return x;
}
```