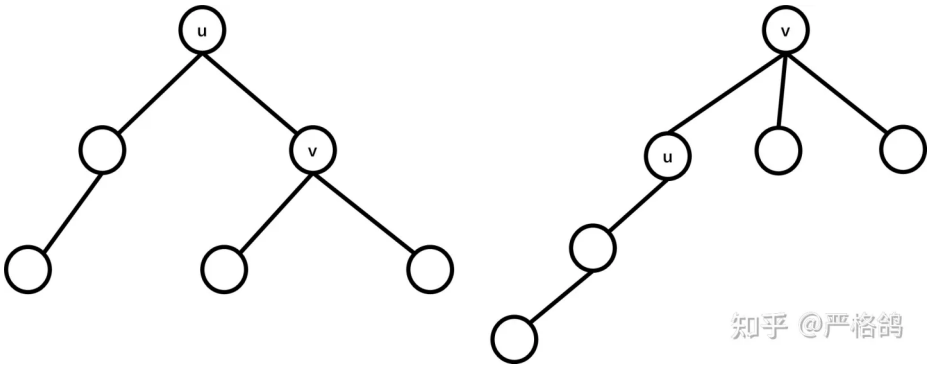
用记忆化搜索解决换根 DP 问题 —— 2022 CCPC 桂林 G

严格鸽
柚子厨 萝莉控 半退役型破铜烂铁 ACMer

换根问题是常见的一种树上 dp，之前我们都是讨论。

考虑将根由 u 变成 v 的变换



但是，我们不考虑换根的情况，考虑基本的 DFS 做法，我们枚举每个 u 作为根节点，进行 $O(n^2)$ 的 DFS。

来回想下 dfs 的写法

```
void dfs(int u, int fa) {
    for (int v : g[u]) {
        if (v == fa)continue;
        dfs(v, u);
    }
}
```

我们仔细的思考下

$[u, fa]$ 的状态只有 $O(n)$ 个。

所以只要我们把 dfs 写成递归的形式，然后套上记忆化搜索，理论上就可以 $O(n)$ 做出题目了。（不考虑记忆化的复杂度。

来看个经典的题目

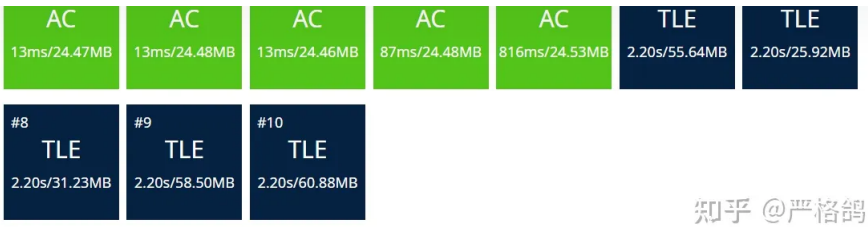
题目链接

题意：

> 给定一个 n 个点的树，请求出一个结点，使得以这个结点为根时，所有结点的深度之和最大。
>
> 一个结点的深度之定义为该节点到根的简单路径上边的数量。
>
> **输入格式**
>
> 第一行有一个整数，表示树的结点个数 n。
> 接下来 $(n-1)$ 行，每行两个整数 u, v，表示存在一条连接 u, v 的边。

我们可以先写出 dfs 的版本，注意，记忆化的前提是，没有用到全局变量。

```
vector<int>g[N];
int siz(int u, int fa) {
    int sz = 1;
    for (int v : g[u]) {
        if (v == fa)continue;
        sz += siz(v, u);
    }
    return sz;
}
int dfs(int u, int fa) {
    int res = 0;
    for (int v : g[u]) {
        if (v == fa)continue;
        res += dfs(v, u) + siz(v, u);
    }
    return res;
}
void slove() {
    cin >> n;
    for (int i = 1; i <= n - 1; i++) {
        int u, v; cin >> u >> v;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    int ans = 0, mxsum = 0;
    for (int u = 1; u <= n; u++) {
        int sum = dfs(u, 0);
        if (sum > mxsum) {
            ans = u;
            mxsum = sum;
        }
    }
    cout << ans << endl;
}
```



这样显然是个 $O(n^2)$ 的分数，我们加上记忆化。

```
unordered_map<int, int>dp1[N];
unordered_map<int, int>dp2[N];
int siz(int u, int fa) {
    if (dp1[u][fa])return dp1[u][fa];
    int sz = 1;
    for (int v : g[u]) {
        if (v == fa)continue;
        sz += siz(v, u);
    }
    return dp1[u][fa] = sz;
}
int dfs(int u, int fa) {
    if (dp2[u][fa])return dp2[u][fa];
    int res = 0;
    for (int v : g[u]) {
        if (v == fa)continue;
        res += dfs(v, u) + siz(v, u);
    }
    return dp2[u][fa] = res;
}
void slove() {
    cin >> n;
    for (int i = 1; i <= n - 1; i++) {
        int u, v; cin >> u >> v;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    int ans = 0, mxsum = 0;
    for (int u = 1; u <= n; u++) {
        int sum = dfs(u, 0);
        if (sum > mxsum) {
            ans = u;
            mxsum = sum;
        }
    }
    cout << ans << endl;
}
```

暴内存了。。。。这个题的数据是 1e6，没有给我们记忆化留下空间。

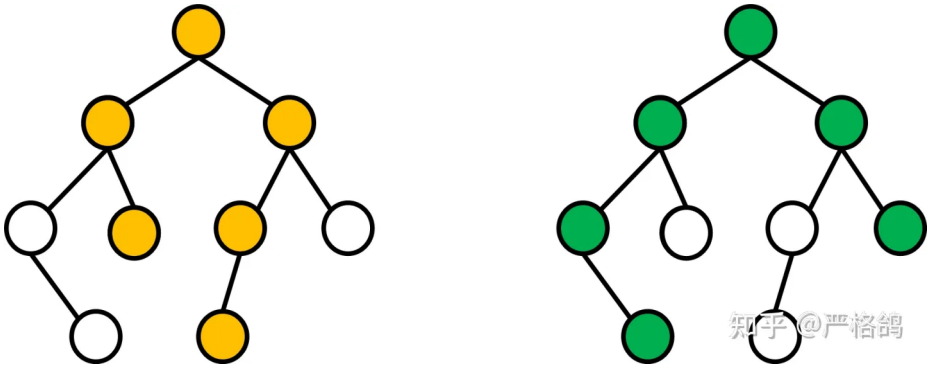
但是前一段时间的桂林也有个换根 DP。

题意：

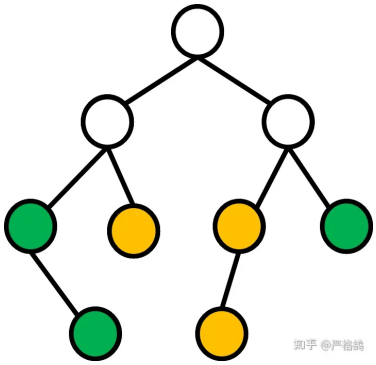
给定一颗 $n(n \leq 2 * 10^5)$ 个节点的树，每个节点都有点权 $a[i](a[i] \leq 10^4)$ 。

你需要选择两条路径，最后的权值为，路径经过的点，去掉两个路径重合的点的点权。

例如我们选择这两条路径



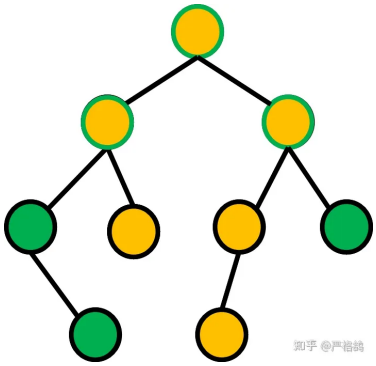
选择的部分就是这些点



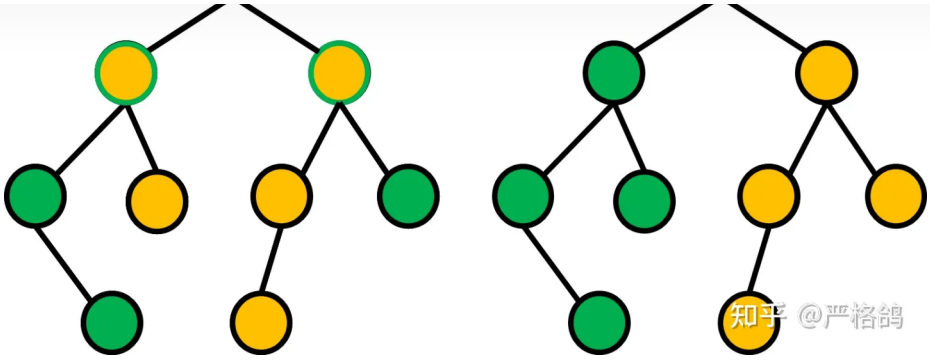
做法：

首先是个结论，两条路径最多只有一个交点。

例如上面的例子



我们可以选择这两条路径。



使得重合的点只有 1 个。

所以我们就讨论，只有一个交点，和没有交点两种情况。

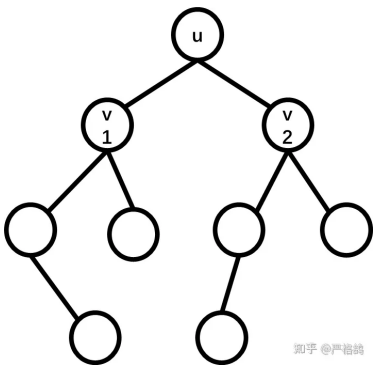
只有一个交点，可以变成，枚举每个点为根，计算选择最长的 4 条链。

```
map<int, int>dp1[N];
int dfs1(int u, int fa) {
    if (dp1[u][fa])return dp1[u][fa];
    int res = a[u];
    for (int v : g[u]) {
        if (v == fa)continue;
        res = max(res, dfs1(v, u) + a[u]);
    }
    return dp1[u][fa] = res;
}
//====main函数里面的
for (int u = 1; u <= n; u++) {
    vector<int>line = { 0,0,0,0 };
    for (int v : g[u]) {
        line.push_back(dfs1(v, u));
    }
    sort(line.begin(), line.end()); reverse(line.begin(), line.end());
    ans = max(ans, line[0] + line[1] + line[2] + line[3]);//选择4条长链
}
```

考虑没有交点的情况，我们可以枚举每一条边 $u \leftrightarrow v$，计算 v 的那一块包含的最长的路径，u 的那一块包含的最长的路径。

这个也是可以记忆化的。

例如下图



u 包含的最长路径，来自 v1,v2 包含的最长路径，和 u 的两条最长的链。

```
map<int, int>dp2[N];
int dfs2(int u, int fa) {
    if (dp2[u][fa])return dp2[u][fa];
```

```
    for (int v : g[u]) {
        if (v == fa)continue;
        res = max(res, dfs2(v, u));
        int line = dfs1(v, u);
        if (mxline[0] < line) {
            mxline[1] = mxline[0];
            mxline[0] = line;
        }
        else if (mxline[1] < line) {
            mxline[1] = line;
        }
    }
    res = max(res, mxline[0] + mxline[1] + a[u]);
    return dp2[u][fa] = res;
}
//====main中的=================
for (int u = 1; u <= n; u++) {
    for (int v : g[u]) {
        ans = max(ans, dfs2(u, v) + dfs2(v, u));
    }
}
```

然后呢，然后就没了。。。。

code

```
map<int, int>dp1[N];
int dfs1(int u, int fa) {
    if (dp1[u][fa])return dp1[u][fa];
    int res = a[u];
    for (int v : g[u]) {
        if (v == fa)continue;
        res = max(res, dfs1(v, u) + a[u]);
    }
    return dp1[u][fa] = res;
}

map<int, int>dp2[N];
int dfs2(int u, int fa) {
    if (dp2[u][fa])return dp2[u][fa];
    int res = 0;
    int  mxline[2] = {0};
    for (int v : g[u]) {
        if (v == fa)continue;
        res = max(res, dfs2(v, u));
        int line = dfs1(v, u);
        if (mxline[0] < line) {
            mxline[1] = mxline[0];
            mxline[0] = line;
        }
        else if (mxline[1] < line) {
            mxline[1] = line;
        }
    }
    res = max(res, mxline[0] + mxline[1] + a[u]);
    return dp2[u][fa] = res;
}

void slove() {
    cin >> n;
    for (int i = 1; i <= n; i++)cin >> a[i];
    for (int i = 1; i <= n - 1; i++) {
        int u, v; cin >> u >> v;
        g[u].push_back(v);
```

```
    int ans = 0;
    for (int u = 1; u <= n; u++) {
        vector<int>line = { 0,0,0,0 };
        for (int v : g[u]) {
            line.push_back(dfs1(v, u));
        }
        sort(line.begin(), line.end()); reverse(line.begin(), line.end());
        ans = max(ans, line[0] + line[1] + line[2] + line[3]);
    }
    for (int u = 1; u <= n; u++) {
        for (int v : g[u]) {
            ans = max(ans, dfs2(u, v) + dfs2(v, u));
        }
    }
    cout << ans << endl;
}
```

使用记忆化搜索就不用考虑换根 dp 的那么多的情况了。

发布于 2022-11-03 11:44 · IP 属地四川

CCPC ACM 竞赛 ACM-ICPC

推荐阅读