南川算法笔记之题解
Acwing 3215. 网络延时

（高性能）一种基于拓扑的求解无向连通图最长路径方案

南川笔记，2021年02月20日

题目详情

3215. 网络延时

题目 | 提交记录 | 讨论 | 题解 | 视频讲解

给定一个公司的网络，由 n 台交换机和 m 台终端电脑组成，交换机与交换机、交换机与电脑之间使用网络连接。

交换机按层级设置，编号为 1 的交换机为根交换机，层级为 1。

其他的交换机都连接到一台比自己上一层的交换机上，其层级为对应交换机的层级加 1。

所有的终端电脑都直接连接到交换机上。

当信息在电脑、交换机之间传递时，每一步只能通过自己传递到自己所连接的另一台电脑或交换机。

请问，电脑与电脑之间传递消息、或者电脑与交换机之间传递消息、或者交换机与交换机之间传递消息最多需要多少步。

输入格式

输入的第一行包含两个整数 n, m，分别表示交换机的台数和终端电脑的台数。

第二行包含 $n-1$ 个整数，分别表示第 2、3、 …… 、n 台交换机所连接的比自己上一层的交换机的编号。第 i 台交换机所连接的上一层的交换机编号一定比自己的编号小。

第三行包含 m 个整数，分别表示第 1、2、 …… 、m 台终端电脑所连接的交换机的编号。

输出格式

输出一个整数，表示消息传递最多需要的步数。

数据范围

前 30% 的评测用例满足：$n \le 5, m \le 5$。
前 50% 的评测用例满足：$n \le 20, m \le 20$。
前 70% 的评测用例满足：$n \le 100, m \le 100$。
所有评测用例都满足：$1 \le n \le 10000$, $1 \le m \le 10000$。

输入样例1：

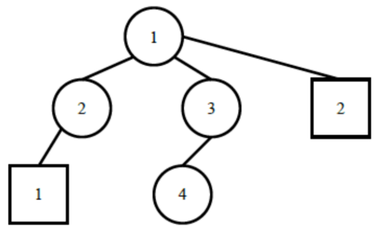
```
4 2
1 1 3
2 1
```

输出样例1：

```
4
```

样例1解释

样例的网络连接模式如下，其中圆圈表示交换机，方框表示电脑：



其中电脑 1 与交换机 4 之间的消息传递花费的时间最长，为 4 个单位时间。

输入样例2：

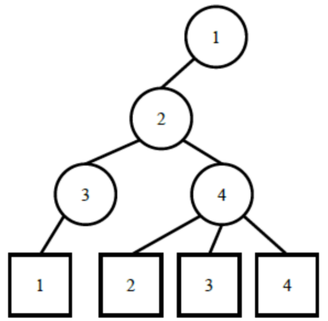
```
4 4
1 2 2
3 4 4 4
```

输出样例2：

```
4
```

样例2解释

样例的网络连接模式如下：

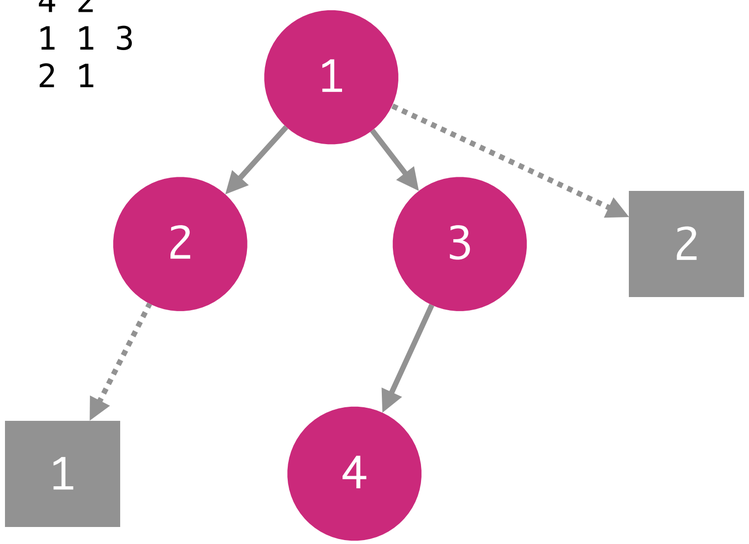


其中电脑 1 与电脑 4 之间的消息传递花费的时间最长，为 4 个单位时间。

拓扑思路的可行性

测例：

```
4 2
1 1 3
2 1
```



思路分析：

1. 虽然这道题名义上是一棵树，但由于最终求的是**这棵树中的最长路径**，并且可以从孩子往父亲方向，所以本质上是一个**无向、无环、无权图。**

2. 而在这样的一个无向无环无权图中，其最长路径就等于某个结点的两条不同的最长路径的总和，比如在图中，经过结点1的两条路径 1->2->1 和 1->3->4 的长度都是2，它们汇总的结果是4，为正确答案。并且在这个例子里，这两条路径的组合是唯一的，虽然这并不重要。

3. 这给我们的启示是，如果我们记录每个结点的所有不同路径的长度，然后从中挑选**两条最长子路径的和**，不就是目标答案吗？没错，这是完全可行的。

4. 但我们怎么求出每个结点的子路径长度呢？可以基于**动态规划/拓扑**去求。

拓扑求每个结点最长子路径办法：

1. 由于每个PC（图中的方形结点）只能与交换机直接连接，即PC不能直接与PC连接，所以每个PC的度数都是1，而与它们直接连接的那台交换机最长子路径最短为1，否则，若它无孩子结点，则为0。
2. 基于此，**我们在读取PC数据时，根本无需存储，直接将它的信息“压缩”进其连接的交换机里去**。之所以说是“压缩”，是因为可能有多台PC连接同一个交换机，但不管多少台，这个交换机的最长子路经始终至少为1，这是易知的。
3. 于是，我们首先就删除了所有PC机的结点，那么对于剩下的交换机结点，我们可以从叶子结点出发，逐个消除。
4. 具体地，我们每次选一个叶子结点，假设它当前的最长子路径长度为 La，则将它消除，并将它的父亲度数减一（当度数为0时，也就成了新的叶子结点），然后将 La 信息上传，它的父结点的最长子路径就等于 La+1 和原来长度的最大值。
5. 但程序如何退出？由于我们自底向上逐步更新，所以**每个父结点在访问之前存储的就是它在其他路径上的最长子路径，我们维护一个最长子路径之和，用当前子路径长与该父亲子路径长再加一不断更新即可。**

代码实现

```
#include "cstdio"
using namespace std;
#define Max(a, b) (a) = (b) > (a) ? (b) : (a)

const int N = 10000;
int degOut[N+1], father[N+1], height[N+1];
int n, m, v, k, ans;

bool topo(int i)
{
    --degOut[i];                    // lock
    --degOut[father[i]];            // topologic
    Max(ans, height[i] + height[father[i]] + 1);
    Max(height[father[i]], height[i]+1);

    if(--k == 1) return printf("%d", ans) && 0;
    if(!degOut[father[i]]) return topo(father[i]);
}

int main()
{
    scanf("%d %d", &n, &m);
    for(int i=2; i<=n; ++i) scanf("%d", &father[i]), ++degOut[father[i]];
    // ++degOut[v], // the deg of PCs should be excluded
    for(int i=1; i<=m; ++i) scanf("%d", &v), height[v] = 1;

    k = n;
    for(int i=1; i<=n; ++i)
        if(!degOut[i])
            topo(i);
}
```

3215. 网络延时

题目 | 提交记录 | 讨论 | 题解 | 视频讲解

提交时间	状态	运行时间	语言	模式
41分钟前	Accepted	17 ms	C++	普通
45分钟前	Accepted	15 ms	C++	普通
48分钟前	Accepted	39 ms	C++	普通
48分钟前	Accepted	41 ms	C++	普通
49分钟前	Accepted	270 ms	C++	普通

性能评估：

1. 我们无需存储PC机结点，只需要度数（用于拓扑）、父亲（用于拓扑的度数更新）、高度（用于维护每个结点的最长子路径）三个数组即可，空间上复杂度为$\Theta(3n)$。
2. 拓扑时，每次会消去一个结点，并更新它的父亲结点，更新时是常数级别操作，见`topo`函数，所以时间复杂度为 $\Theta(n)$。
3. 具体表现为 15-17 ms，比题解区使用 dfs/bfs 的传统方法快上一倍。
4. 还能更快吗?
    1. 基于拓扑的度数操作不能再快
    2. 但每次更新高度时，也许还有一定的提升空间
    3. 是否必须等到结点只剩最后一个时才能退出呢？直觉上应该是的，因为一步步消除，最后一个结点的长度应该是最长的，但也许这个规则可能被打破，留给您思考~