题目描述

助教团队发现了一张藏宝图，令人惊喜的是，地图中的所有宝藏点和道路构成了一棵“树”。地图显示第 `i` 个宝藏点的宝藏价值为 `value[i]`。助教们从1号根节点出发，为了在有限时间内高效寻宝，三位助教提出了各自的寻宝策略：

1. 王助教的策略是：仅访问价值 `value > p` 的宝藏点，这意味着如果某个宝藏点的价值小于或等于 `p`，那么该宝藏点及其子树中的宝藏点都不会被访问。如果根节点宝藏价值小于等于`p`，王助教就不寻宝了。
2. 胡助教的策略是：**除根节点外**，仅访问满足 `value > q` 且 `value + 父节点value > k` 的宝藏点。换句话说，如果某个宝藏点不满足 `value > q` 或 `value + 父节点value > k` 的条件，则该宝藏点及其子树中的宝藏点将不会被访问。
3. 谢助教的策略是：**除根节点外**，仅访问价值为偶数的宝藏点，这意味着 如果某个宝藏点的价值为奇数，则该宝藏点及其子树中的宝藏点都不会被访问。

现在，助教团队希望根据每位助教的策略，分别计算他们能够获得的宝藏总价值。

请你帮助助教团队解决这个问题，并编写具有 **可扩展性** 和 **易维护性** 的代码。也就是说，如果未来有新的助教加入并提出新的寻宝策略，你的代码应当能够在最小的改动下支持新的策略。

请使用 **函数指针或函数对象** 来实现你的程序。

输入格式

第一行包含四个整数 `n`，`p`，`q`，`k`，分别表示树中宝藏点的个数及策略中涉及的三个参数。

接下来 `n - 1` 行，每行包含两个整数 `x` 和 `y`，表示 `x` 号节点和 `y` 号节点之间有一条边，保证 `x` 是 `y` 的父节点，节点编号的范围为`1`到`n`。

接下来一行包含 `n` 个整数，表示每个宝藏点的价值 `value[i]`。

输出格式

输出一行，包含三个整数，分别表示按照王助教、胡助教和谢助教的策略可以获得的宝藏总价值。

输入样例

```
8 4 2 7
1 2
1 3
2 4
2 5
3 6
3 7
3 8
10 4 5 2 8 4 1 9
```

输出样例

```
24 40 24
```

提示

对于树的存储方式没有限制：

- 你可以将树视为无向图，使用邻接表或链式前向星来存储；
- 或使用以下结构存储每个 `TreeNode`：

```
struct TreeNode {
    int val;
    vector<TreeNode*> children;
    TreeNode(int x) : val(x) {}
};
```

数据规模与约定

对于100%的数据，`1 <= n, p, q, k <= 1000`，`1 <= value[i] <= 2000`