**说明文档**

**① Parents**

**分析：**

1. 用一个ArrayList<Integer>[] lists将树构建成 一个邻接表类型的图。lists[index]是一个

ArrayList，储存所有和index节点相连的节点。

1. 使用DFS从index 1开始搜索，每搜索到一个节点，把lists[index] 除父节点外的所有节点

（即子节点）全部在parents数组中赋值为index，并递归继续搜索。

**时间复杂度：**

**Remark.** 对于N个节点，E条边的邻接表形式的图，DFS时间复杂度为O(N+E)

该算法建树时间复杂度O(N)，N个节点，N-1条边的邻接表DFS时间复杂度为O(2N-1)即O(N)，因此总时间复杂度O(N)

**② Path**

**分析：**

1. 用长度为n+1的HashMap<Integer, Integer>[] maps和ArrayList<Integer>[] lists将树构建成

一个邻接表类型的图。maps[index]是一个HashMap，储存和index节点连接的所有节点的<index2, weight>对。lists[index]是一个ArrayList，储存所有和index节点相连的节点。

1. 若lists[ i ].size() == 1，说明该节点是叶子节点。使用DSF从index 1开始搜索，int类型变

量sum记录当前路径上的weight和，搜索到叶子节点时，若sum == num为true，则pathNumber++，DFS结束后返回pathNumber

**时间复杂度：**

该算法建树时间复杂度O(N)，N个节点，N-1条边的邻接表DFS时间复杂度为O(2N-1)即O(N)，因此总时间复杂度O(N)