查找树中元素

知识点树 BFS 搜索广搜
时间限制：1s 空间限制：256MB 限定语言：不限

题目描述：

已知树形结构的所有节点信息，现要求根据输入坐标（x,y）找到该节点保存的内容值；其中：

x表示节点所在的层数，根节点位于第0层，根节点的子节点位于第1层，依次类推；

y表示节点在该层内的相对偏移，从左至右，第一个节点偏移0，第二个节点偏移1，依次类推；

举例：上图中，假定圆圈内的数字表示节点保存的内容值，则根据坐标(1,1)查到的内容值是23

输入描述:

每个节点以一维数组（int[]）表示，所有节点信息构成二维数组（int[][]），二维数组的0位置存放根节点；
表示单节点的一维数组中，0位置保存内容值，后续位置保存子节点在二维数组中的索引位置；
对于上图中，根节点的可以表示为{10,1,2}，树的整体表示为
{{10,1,2},{-21,3,4},{23,5},{14},{35},{66}}
查询条件以长度为2的一维数组表示，上图查询坐标为(1,1)时表示为
{1,1}
使用Java标准IO键盘输入进行录入时，先录入节点数量，然后逐行录入节点，最后录入查询的位置，对于上述示例为：

```
6
10  1  2
-21  3  4
23  5
14
35
66
1  1
```

输出描述:

查询到内容值时，输出{内容值}，查询不到时输出{}
上图中根据坐标(1,1)查询输出{23}，根据坐标(1,2}查询输出{}
补充说明：
考试者不需要自己编写解析输入文本的代码，请直接使用上述代码中的Parser类解析输入文本；

示例1

输入：

```
6
10  1  2
-21  3  4
23  5
14
35
66
1  1
```

输出：

```
{23}
```

示例2

输入：

```
14
0    1   2  3   4
-11  5   6   7   8
113  9   10  11
24   12
35
66   13
77
88
99
101
102
103
25
104
2    5
```

输出：

```
{102}
```

示例3

输入：

```
14
0     1    2   3    4
-11   5   6    7   8
113   9   10   11
24    12
35
66    13
77
88
99
101
102
103
25
104
3     2
```

输出：

```
{}
```

示例4

输入：

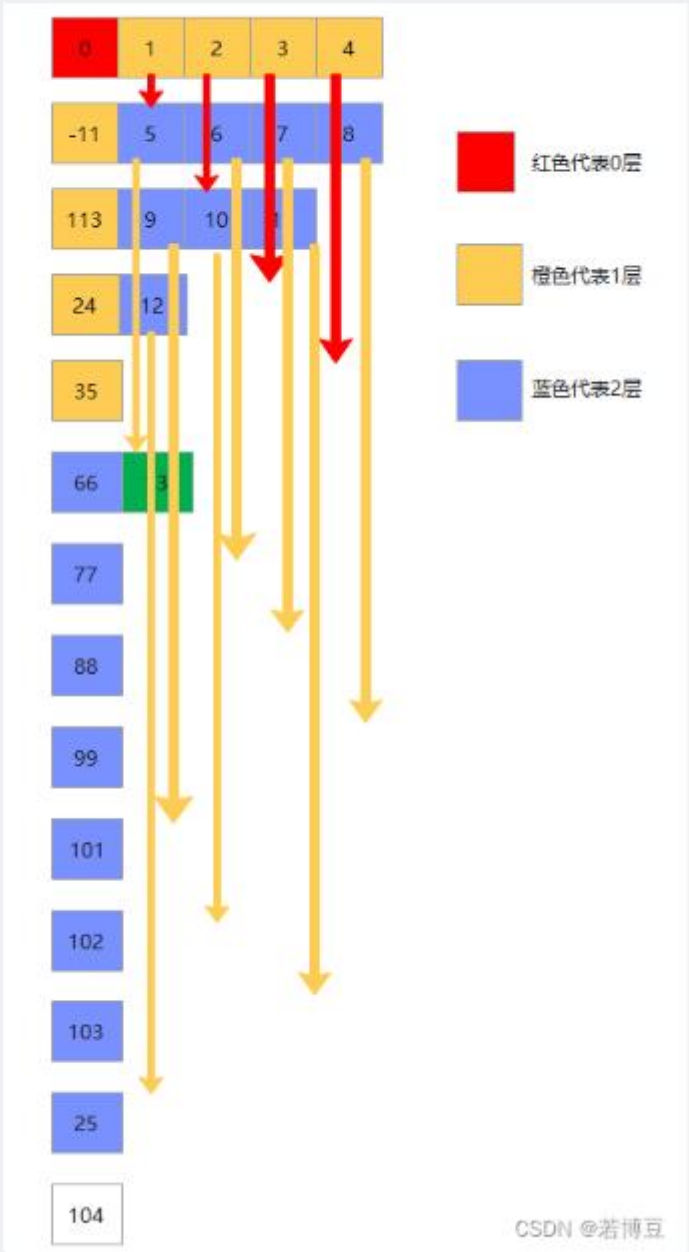
```
1
1000
0     0
```

输出：

```
{1000}
```

解题思路：

通过回溯法求出x层中所有数据加入集合中，再求出集合中索引为y的值。
如示例2：



根据上图可以得出二层元素集合{66,77,88,99,101,102,103,25}，index=5也就是第6个元素为102。

```
public class Main{

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int size = Integer.parseInt(in.nextLine());
        int[][] nodes = new int[size][];
        for (int i = 0; i < size; i++) {
            nodes[i] = parseOneLine(in.nextLine());
        }
        int[] xy = parseOneLine(in.nextLine());
        String result = doQuery(nodes, xy[0], xy[1]);
        System.out.println(result);
    }

    private static int[] parseOneLine(String text) {
        ByteArrayInputStream stream = new
ByteArrayInputStream(text.getBytes(StandardCharsets.UTF_8));
        Scanner in = new Scanner(stream);
        List<Integer> list = new ArrayList<>();
        while (in.hasNext()) {
            list.add(in.nextInt());
        }
        return list.stream().mapToInt(it -> it).toArray();
    }

    private static String doQuery(int[][] nodes, int x, int y) {
        if (x < 0 || y < 0) {
            return "{}";
        }

        List<Integer> list = new ArrayList<>();
        handle(nodes, 0, x, list);

        if (y >= list.size()) {        //所在层数子节点不足 y+1 个
            return "{}";
        }

        return "{" + list.get(y) + "}";
    }

    /**
     *   求出 n 层所有数据并加入 list 集合中
     * @param nodes        节点信息二维数组
```

```
     * @param index        子节点索引
     * @param n            树的层数递减（遍历几次代表几层）
     * @param list         所求层的所有数据集合
     */
    private static void handle(int[][] nodes, int index, int n, List<Integer> list) {

        int[] node = nodes[index];
        if (n == 0) {     //表示已经到了所求层数
            list.add(node[0]);
            return;
        }

        if (node.length == 1) {        //说明没有子节点
            return;
        }

        for (int i = 1; i < node.length; i++) {        //各数组中的 0 元素代表其值，后面的代表其子节点，所有从 1 开始遍历
            handle(nodes, node[i], n - 1, list);
        }
    }
}
```