在二叉树类 BinaryTree 中增加一个功能：判断二叉树是否是完全二叉树。

【解】如果一棵二叉树是完全二叉树，那么按照二叉树的性质 5 可知按层编号的最后一个结点的编号等于树上的结点总数。于是，我们可以层次遍历这棵树。在层次遍历时，对树的每个结点标记一个数值。标记方法如下：根结点为 1。对于每个值为 n 的结点，它的两个儿子的值分别为 2n 和 2n+1。在遍历时，顺便统计树上的结点个数。当结点数等于最后一个结点的编号时，这棵树是完全二叉树，反之为非完全二叉树。与层次遍历不同的是：在判别完全二叉树时，队列中不仅要保存指向结点的指针，还需要保存结点的编号。为此定义了一个结构体类型作为队列中的元素类型。它的定义如下。判断函数的实现见代码清单 5-14

```
struct elem {
    Node *p;
    int num;
};
```

代码清单 5-14 判断二叉树是否为完全二叉树

```
1.  bool isCompleteTree() {
2.      linkQueue<elem> que;
3.      elem cur, child;        //cur 当前处理的结点，child：cur 的儿子
4.      int count = 1;          //count：访问到的结点数
5.    int last = 1;           //last：最后一个结点的编号
6.
7.    if(root == NULL) return true;
8.      cur.p = root;
9.      cur.num = 1;
10.     que.enQueue(cur);                       // 根结点入队
11.     while ( !que.isEmpty() ){
12.         cur = que.deQueue();
13.         if ( cur.p->left != NULL ) {    //处理当前结点的左孩子
14.             ++count;                        // 结点数加 1
15.             child.p = cur.p->left;
16.             last = child.num = cur.num * 2; // 设置左孩子的编号
17.             que.enQueue(child);             // 左孩子入队
18.         }
19.         if ( cur.p->right != NULL ){    // 处理当前结点的右孩子
20.             ++count;                        // 结点数加 1
21.             child.p = cur.p->right;
22.             last = child.num = cur.num * 2 + 1; // 设置右孩子的编号
23.             que.enQueue(child);             // 右孩子入队
24.         }
25.     }
26.     return count == last;  // 返回结点数是否等于最后一个结点的编号
27. }
```