### Chapter 3: Testing Results

Purpose: Read a tree from the standard input, then count the number of leaves and output them for every seniority level starting from the root.

Expected result: After the tree being legally input, print the number of leaves of each level from the root.

Possible errors: If the data of the tree is invalid (level>100), then the output may be wrong.

Table of test cases:

|  |  |  |
| --- | --- | --- |
| Input | Tree | Output |
| 2 1  01 1 02 |  | 0 1 |
| 7 4  01 2 02 03  06 1 07  02 2 04 05  03 1 06 |  | 0 0 2 1 |
| 8 3  01 3 02 03 04  02 4 05 06 07 08 |  | 0 2 4 |
| 99 98  01 1 02  ….  98 1 99 | … | 0 0 0 0…..0 1 |
| 1 0 |  | 1 |

### Chapter 4: Analysis and Comments

The time complexity of the core algorithm (Traverse all nodes and counting the number of leaves) is O(N).

We use link list to store the tree. So suppose the tree have *m* nodes, every node needs to store an int type data and a pointer. It will take more space than using array. But because the number of nodes is variable, using link list will be more convenient, and more space-saving when there are light nodes.