# Tester’s Report

目录

[Tester’s Report 1](#_Toc433463918)

[Chapter 3: Testing Result(Current Status: pass ) 1](#_Toc433463919)

[The first test case : 1](#_Toc433463920)

[The second test case : 2](#_Toc433463921)

[The third test case : 2](#_Toc433463922)

[The fourth test case : 3](#_Toc433463923)

[The fifth test case : 4](#_Toc433463924)

[The sixth test case : 4](#_Toc433463925)

[The seventh test case : 5](#_Toc433463926)

[The eighth test case : 5](#_Toc433463927)

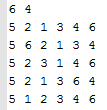
[Chapter 4: Analysis and Comments 6](#_Toc433463928)

## Chapter 3: Testing Result(Current Status: pass )

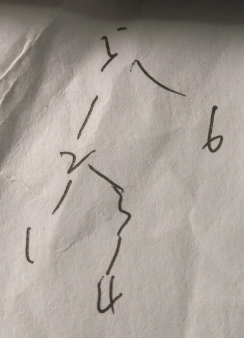
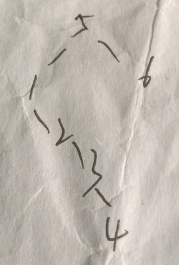
As a tester, I use a set of test cases to test the program. The data is in “p2.in” and the result is in “p2.out”.

Now, we can analyse the input and the output:

### The first test case :



As we can see, the first 4 lines generate the same tree, while the fifth line generates a different tree.

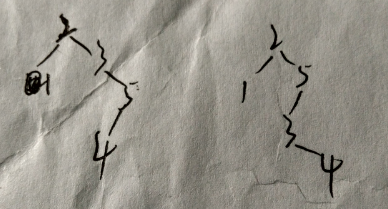
 

So the answer is .

### The second test case :

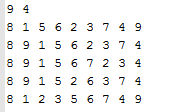


The first 2 lines generate the same tree, while the third line generates a different tree.

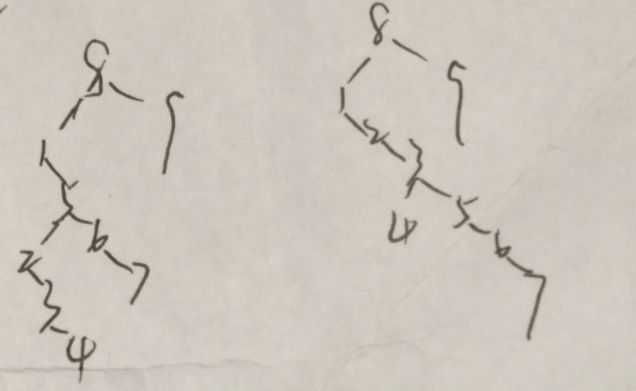


So the answer is .

### The third test case :

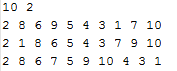


The first 4 lines generate the same tree, while the fifth line generates a different tree.

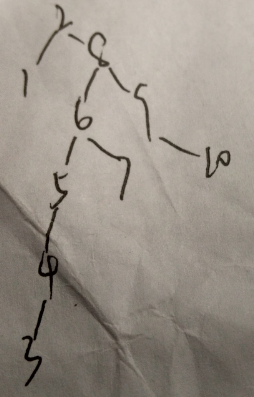


So the answer is .

### The fourth test case :



The 3 lines generate the same tree.



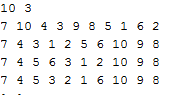
So the answer is .

### The fifth test case :

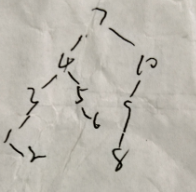
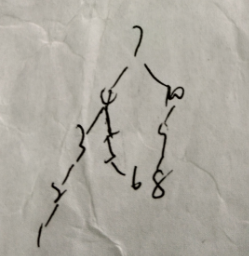


Obviously, the first 2 lines generate the same tree, while the third line generates a different tree. So the answer is 

### The sixth test case :



The first 3 lines generate the same tree, while the fourth line generates a different tree.

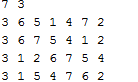
So the answer is .

### The seventh test case :

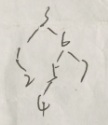
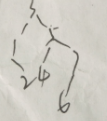


Obviously, the answer is .

### The eighth test case :



The first 3 lines generate the same tree, while the fourth line generates a different tree.

So the answer is .

## Chapter 4: Analysis and Comments

The time complexity of the function Insert is O(NlogN). In constant time we descend a level in the tree, thus operating on a tree is now roughly half as large. Indeed, one insert operation is O(d), where d is the depth of the node containing in the accessed key. We can prove that the average depth over all nodes in a is O(logN) on the assumption that all trees are equally likely.

The time complexity of the function Isequal is O(N) because all the node is supposed to be compared once in the worst case.

So the time complexity of the whole program is O(L\*N\*(logN+1))=O(L\*NlogN) for each test case.

The space complexity of the program is O(L\*N) for each test case as one node need constant space to store.

As the scale of the data is small, so I think the program performs well enough. My only comment is that we can free the space after the operation of one tree so the space complexity of the program can reduce to O(N).