Height and Total Height of a BST

This problem will deal with trees. A few definitions follow:

- *internal node*: any node of a tree with child elements (*children*)
- *external node* or *leaf*: any node of a tree without children
- *root node*: topmost node of a tree
- *node height*: distance of a node from its most distant leaf
- *tree height*: synonymous with *root height*, node height of the root node

A Binary Search Tree (*BST*), also called ordered binary tree, is a type of binary tree where the nodes are arranged in order. A BST has the following properties:

1. Each node has a unique value*.
2. A total order is defined on these values.
3. The left subtree of a node contains only values less than the node's value.
4. The right subtree of a node contains only values greater than the node's value.

* No duplicates. Be careful.

You will be given an array of integers to insert into a BST that you create. Complete the `heightOfBST` function below to return an integer array. The first element should be the tree height of the BST and the second element should be the sum of the heights of all of the BST nodes, the *total height*.

Input Format

In the first line you will be given an integer n which represents the number of space-separated integers in the following line. In the following line, there are n space-separated integers $a_0, a_1, \ldots, a_{n-1}$, denoting the values to be inserted into the BST in this exact order.

Constraints

- $1 \leq n \leq 2000$
- $0 \leq a_i \leq 2000$

Output Format

Return an integer array of two elements: $[tree\_height, total\_height]$ to be printed by the code stub.

Sample Input 0

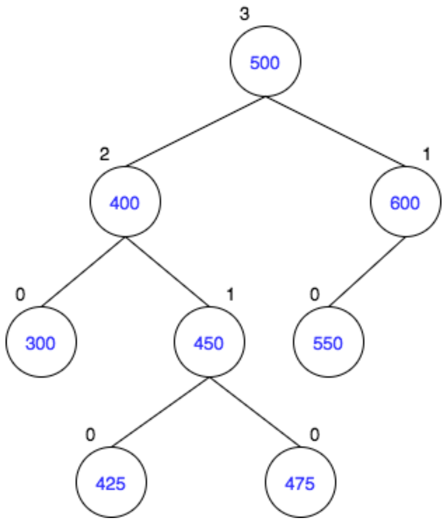
```
8
500 400 300 450 425 475 600 550
```

Sample Output 0

```
3
7
```

Explanation 0

The image below shows the completed BST and the height of each of the nodes:


3
500
2
400
1
600
0
300
1
450
0
550
0
425
0
475