Swinging Monkeys

Tarzan, the king of the jungle, is organizing a party and he is inviting all animals to attend this spectacular party. After sending out the invitations, he excitedly waited for the day of the party. He prepared the party by cooking the most delicious meal (vegetarian, of course) to all the attendees. He has also decorated his lovely house and eagerly waited for the day of the party.

After several weeks, it is finally time for the party. As we all know, there are a lot of trees in the jungle and monkeys like to swing from tree to tree. This time, the monkeys are going to Tarzan's party by swinging from trees to trees.

In the jungle, the height of each tree varies. Monkeys can directly swing from one tree to another as long as there are no trees in between that is taller than or have the same height as either one of the two trees. For example, if there are 5 (five) trees lined up with height 19m, 17m, 20m, 20m, and 20m respectively, then the monkeys will be able to swing from one tree to the other as shown below:

19 m Tree #01	17 m Tree #02	20 m Tree #03	20 m Tree #04	20 m Tree #05

1. Monkeys can swing from the first tree to the second tree.
2. Monkeys can swing from the first tree to the third tree.
3. Monkeys can swing from the second tree to the third tree.
4. Monkeys can swing from the third tree to the fourth tree.
5. Monkeys can swing from the fourth tree to the fifth tree.

Since the monkeys are very creative and playful, they want to know how many different pairings of trees that they can swing from and to. In the example above, the total number of pairings is 5 as explained above.

The problem here is that the monkeys cannot count very well. Therefore, they ask you, the best programmer in the jungle, to write a program that can tell them how many pair of trees that they can swing from and to. As a reward for accomplishing this task, the monkeys agree to take you to Tarzan's party so that you can enjoy the wonderful food provided by Tarzan in his party. Good luck!

Input

The first line contains a single integer **N** ($2 <= N <= 500000$), the number of trees in the path. The next line contains N integers a_0 … a_{n-1} where $0 < a_i < 2^{31}$

Output

Print the number of pair of trees that the monkeys can swing from and to. Your output should contain a newline character.

Sample Input 1
```
4
3 4 1 2
```

Sample Output 1
```
4
```

Sample Input 2
```
5
19 17 20 20 20
```

Sample Output 2
```
5
```

Sample Input 3
```
4
12 21 21 12
```

Sample Output 3
```
3
```

Explanation

Sample Input 1: **4 pairs of trees**

1. From the first tree (3), monkeys can only swing to the second tree (4) since the second tree "blocks" the path from the first tree. There is one pair, which is (3,4).
2. From the second tree (4), monkeys can swing to the third (1) and fourth tree (2), so there are two pairs here: ((4,1) and (4,2)).
3. From the third tree, monkeys can swing to the fourth tree: (1,2).

Sample Input 2: **5 pairs of trees**

1. From the first tree (19), monkeys can swing to the second (17) and third tree (20), but not the rest. So, there are two pairs here: (19,17) and (17,20).
2. From the second tree (17), monkeys can only swing to the third tree (20), therefore there is one pair here: (17,20).
3. From the third tree (20), monkeys can only swing to the fourth tree (20), therefore there is one pair here: (20,20).
4. From the fourth tree (20), monkeys can swing to the fifth tree (20), adding one more pair (20,20) to the total.

Sample Input 3: **3 pairs of trees**

1. From the first tree (12), monkeys can swing to the second tree (21), but not the rest. Hence, one pair is added to the total: (12,21).
2. From the second tree (21), monkeys can swing to the third tree (21) but not the fourth. Hence, one pair is added to the total: (21,21).
3. From the third tree (21), monkeys can swing to the fourth tree (12). Hence, one pair is added to the total: (21, 12).

Skeleton

You are given the skeleton file `Swing.java`.

Notes

1. You must either use **stack** or **queue** to solve this problem, whichever is suitable.
2. Your program might give out the correct answer, but killed on CodeCrunch. It means that your program is not efficient enough (i.e. it runs too slow). You should design a more efficient algorithm to solve this problem if this is the case. Consider the above note as a hint.
3. The expected solution has a time complexity of **O(N).**