Morning Assembly

Problem Description: In a local school, N students have assembled for the morning assembly. All the students are standing in a straight line. Due to some technical issues, the mike is not working. The technician is trying to solve the issue. Students have to stand in the line, all this while. After a while, students start looking at each other. But, since they are not standing in any particular order of heights, not one cannot see everyone else.

Two students X and Y can see each other if they are standing next to each other or if no student standing between them has height greater than the height of either X or Y.

You will be given the height of students standing in the assembly line. You have to print the number of pairs of students that can see each other.

For example:
Input array: [2, 4, 1, 2, 2, 5, 1]
Output: 10

Explanation: (2, 4), (4, 1), (4, 2), (4, 2), (4, 5), (1, 2), (2, 2), (2, 5), (2, 5), (5, 1) form the valid pairs.

Best Solution: Time: O(n) Extra Space: O(n)

Let us take in example to make few observations. Let us suppose we are given an array of numbers *1, 20, 2, 8, 4, 6, 10.* So how many valid pairs can have 6 in it? 6 can make a valid pair with 4, 8, but not with 2, 20 or 1, it can also make a valid pair with 10. So,

1. It seems like we can only make a valid pair till we encounter element greater than 6.

Now let us look at 10, it can make a valid pair with 6, 8, 2, 20 but not with 4, 2, 1.

2. Let's consider 6 as a1, 4 as a2, 8 as a3. If a2<a1, we cant make a valid pair.

We can use a stack to keep a note of all the elements lesser than the required number and once a greater element comes we can pop out the numbers and make pairs with them.

Pseudocode:

```
ans =0
declare empty stack S
for item in array:
   if (s.empty()):
      s.push(item)

   else if(s.top() <= item){ //can make a pair
      ans++;
      s.pop();
   }

   if(!s.empty() and s.top()){
      ans++;
      s.push(item);
   }
```