795. Number of Subarrays with Bounded Maximum

Given an integer array `nums` and two integers `left` and `right`, return *the number of contiguous non-empty* ***subarrays*** *such that the value of the maximum array element in that subarray is in the range* `[left, right]`.

The test cases are generated so that the answer will fit in a **32-bit** integer.

Example 1:

```
Input: nums = [2,1,4,3], left = 2, right = 3
Output: 3
Explanation: There are three subarrays that meet the requirements: [2], [2, 1], [3].
```

Example 2:

```
Input: nums = [2,9,2,5,6], left = 2, right = 8
Output: 7
```

Constraints:

- `1 <= nums.length <= 10<sup>5</sup>`
- `0 <= nums[i] <= 10<sup>9</sup>`
- `0 <= left <= right <= 10<sup>9</sup>`

-
  ```
  import sys
  class Solution:
      def numSubarrayBoundedMax(self, nums: List[int], left: int, right: int) -> int:
          ans = 0
          prev = 0
          i = 0
          j = 0
          n = len(nums)
          while i<n:
              if nums[i] in range(left,right+1):
                  prev = i-j+1
                  ans+=prev
              elif nums[i]<left:
  ```

```
                ans+=prev
            else:
                j = i+1
                prev = 0
            i = i+1
        return ans
```