Microscopic View of Judge Algorithm for histogram

KNOWN FACTS: 1. Largest area swaped by a histogram has can be easily calculated by using Immediate_Smaller_In_Left (ISL) array & Immediate_Smaller_In_Right (ISR) array.

2. JUDGE_STACK : Always keeps elements in increasing order.

JUDGE - ALGO: For each element in histogram array

IF :
STACK_IS_EMPTY OR 'CURRENT_ELEMENT' ≥ stack.top element
then push the index of current element to stack.

ELSE :
Pop the stack until either the stack becomes empty or stack.top is greater than current element.

FOR-EACH : Popped element calculate the area of region swaped :

AREA = HEIGHT_OF_POPPED_BAR * (INDEX_OF_ISR_OF_POPPED_BAR − INDEX_OF_ISL_OF_POPPED_BAR − 1)

MAX_AREA = Math.max (old_max_area, AREA)

Note: 1. INDEX_OF_ISR_OF_POPPED_BAR = index of CURRENT_ELEMENT
2. INDEX_OF_ISL_OF_POPPED_BAR = index present at stack.top

6. JUDGE_STACK : helps to get ISL. ISL of popped element is stack.top.
Whereas ISR, comes from index of 'CURRENT_ELEMENT' in for loop

Classes of consecutive duplicate elements on JUDGE STACK:

Consecutive duplicate element on Judge stack can be because of following input types :

1. Consecutive duplicate element in input
eg. [2, 3, 4, 7, 7, 6]

2. Effectively consecutive duplicate in input
→ If between two duplicate elements, there exists no smaller element than they can be called Effectively Consecutive duplicates as in Judge_Stack they will appear one after the other at some point in the stack progression.

eg. [3, 4, 5, 3] ⇒ stack progression [3] → [3,4] → [3,4,5] → [3,4] → [3,3]

eg. [2, 4, 5, 3, 2] ⇒ [2] → [2,4] → [2,4,5] → [2,4] → [2,3] → [2,2]

Note: [2, 4, 5, 3, 2, 1, 2] → This is not an effective consecutive duplicate as '1' is smaller than duplicate elements '2'.
↳ [2] → [2,4] → [2,4,5] → [2,4] → [2] → [2,3] → [2] → [] → [1] → [1,2]
both the duplicate 2's cannot come together.

Q Why consecutive duplicate & effectively consecutive duplicate elements have the same ISL & the same ISR ?

Sol^n Effectively consecutive duplicate contains greater elements between the duplicate elements, means we can ignore them as they can't be smaller in left & smaller in right for duplicate element.

Q Why Judge - Algorithm fails to calculate the ISL & ISR if input contains consecutive duplicate & effectively consecutive duplicate ; but successfully calculates the maximum area swaped by histogram irrespective of the input content ?

Since in case of histogram current max area overlaps old max area so eventually bottom-most duplicate element's max area will be fixed by algo for consecutive duplicate & effective consecutive duplicate.

Sol^n : In case of consecutive duplicates & effectively consecutive duplicates JUDGE_STACK contains duplicate elements which mislead the calculation of their ISL except of the bottom-most duplicate. If we consider the ISL & ISR of the bottom-most duplicate on Judge-Stack, then Judge-algorithm can be applied flawlessly.
Means we can apply Judge-algo to calculate ISL & ISR if input doesn't contain consecutive duplicates & effectively consecutive duplicates.