Leetcode problem no. 2289

Link- (https://leetcode.com/problems/steps-to-make-array-non-decreasing/description/)

Problem Description:

You are given a **0-indexed** integer array nums. In one step, **remove** all elements nums[i] where nums[i - 1] > nums[i] for all 0 < i < nums.length.

Return *the number of steps performed until* nums *becomes a* ***non-decreasing*** *array*.

Sample Input:

nums = [5,3,4,4,7,3,6,11,8,5,11]

Sample Output: 3
Explanation: The following are the steps performed:
- Step 1: [5,**3**,4,4,7,**3**,6,11,**8**,**5**,11] becomes [5,4,4,7,6,11,11]
- Step 2: [5,**4**,4,7,**6**,11,11] becomes [5,4,7,11,11]
- Step 3: [5,**4**,7,11,11] becomes [5,7,11,11]
[5,7,11,11] is a non-decreasing array. Therefore, we return 3.

Intuition:

Since we have to remove the elements which are lesser than its previous element in each operation we can consider it like each level has one operation in the que or the set. In each level we are going to remove the elements that are lesser than the previous

element and we have to modify this array. For modifying the array it takes o(n) complexity which will leads to the TLE for that we can use the linked list which can easily modify this array in the o(1) time and can also add the elements which are need to be removed in the next operation it is like,

Operation 1 - > 5,3,4,4,7,3,6,11,8,5,11 -> 5,4,4,7,6,11,11 -> set={4,6};

Operation 2 - > 5,4,4,7,6,11,11 -> 5,4,7,11,11 -> set = {4};

Operation 3 -> 5,4,7,11,11 -> 5,7,11,11 -> set = {} so stop the loop and return the ans

Code:

```
class Solution {
public int totalSteps(int[] nums) {
ListNode head = new ListNode(nums[0]);
ListNode prevv = head;
int n = nums.length;
Set<ListNode> set = new HashSet<>();

for(int i=1;i<n;i++){
ListNode newNode = new ListNode(nums[i]);
prevv.next = newNode;
newNode.prev = prevv;
if(nums[i]<nums[i-1]) set.add(newNode);
prevv = prevv.next;
}
int ans = 0;
while(!set.isEmpty()){
Set<ListNode> nset = new HashSet<>();
for(ListNode curr : set){
// System.out.print(curr.val+" ");
ListNode next = curr.next;
ListNode c = curr.prev;
if(next!=null){
next.prev = c;
c.next = next;
if(next.val<c.val && !set.contains(next)) nset.add(next);
}else{
c.next = null;
}
```

```
}
// System.out.println();
// ListNode curr = head;
// while(curr!=null){
// System.out.print(curr.val+" ");
// curr = curr.next;
// }
// System.out.println();
set = new HashSet<>(nset);
ans++;
}
return ans;
}
public class ListNode{
ListNode next;
int val;
ListNode prev;
public ListNode(int val){
this.val = val;
next = prev = null;
}
}
}
```

Time Complexity:

O(n)

Space Complexity:

O(n)

Best Time Complexity:

O(n)

Best space complexity:

O(n)