Sorting Assignment

Problem Statement 1:
Given an Integer N and a list arr. Sort the array using bubble sort algorithm.
Example 1:

Input:
N = 5
arr[] = {4, 1, 3, 9, 7}
Output:
1 3 4 7 9
Example 2:

Input:
N = 10
arr[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
Output:
1 2 3 4 5 6 7 8 9 10

Expected Time Complexity: O(N^2).
Expected Auxiliary Space: O(1).

Constraints:
1 <= N <= 10^3
1 <= arr[i] <= 10^3

Problem Statement 2:
Given an array of size N containing only 0s, 1s, and 2s; sort the array in ascending order.

Example 1:

Input:
N = 5
arr[]= {0 2 1 2 0}

Output:
0 0 1 2 2
Explanation:
0s 1s and 2s are segregated
into ascending order.
Example 2:

Input:
N = 3
arr[] = {0 1 0}

Output:
0 0 1

Explanation:
0s 1s and 2s are segregated
into ascending order.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
1 <= N <= 10^6
0 <= A[i] <= 2

Problem Statement 3:
Given two sorted arrays arr1[] and arr2[] of sizes n and m in non-decreasing order. Merge them in sorted order without using any extra space. Modify arr1 so that it contains the first N elements and modify arr2 so that it contains the last M elements.

Example 1:

Input:
n = 4, arr1[] = [1 3 5 7]

m = 5, arr2[] = [0 2 6 8 9]
Output:
arr1[] = [0 1 2 3]
arr2[] = [5 6 7 8 9]
Explanation:
After merging the two
non-decreasing arrays, we get,
0 1 2 3 5 6 7 8 9.
Example 2:

Input:
n = 2, arr1[] = [10 12]
m = 3, arr2[] = [5 18 20]
Output:
arr1[] = [5 10]
arr2[] = [12 18 20]
Explanation:
After merging two sorted arrays
we get 5 10 12 18 20.

Expected Time Complexity: O((n+m) log(n+m))
Expected Auxiliary Space: O(1)

Constraints:
1 <= n, m <= 10^5
0 <= arr1i, arr2i <= 10^7

Problem Statement 4:
Given an array arr[] of size N, check if it is sorted in non-decreasing order or not.

Example 1:

Input:
N = 5
arr[] = {10, 20, 30, 40, 50}
Output: 1

Explanation: The given array is sorted.
Example 2:

Input:
N = 6
arr[] = {90, 80, 100, 70, 40, 30}
Output: 0
Explanation: The given array is not sorted.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)
Constraints:
$1 \leq N \leq 10^5$
$1 \leq Arr[i] \leq 10^6$

Problem Statement 5:
Given an array arr[], its starting position l and its ending position r. Sort the array using the Merge Sort algorithm.
Example 1:

Input:
N = 5
arr[] = {4 1 3 9 7}
Output:
1 3 4 7 9
Example 2:

Input:
N = 10
arr[] = {10 9 8 7 6 5 4 3 2 1}
Output:
1 2 3 4 5 6 7 8 9 10

Expected Time Complexity: O(nlogn)
Expected Auxiliary Space: O(n)

Constraints:
1 <= N <= 10^5
1 <= arr[i] <= 10^5

Example 1:

Input:
N = 5
arr[] = { 4, 1, 3, 9, 7}
Output:
1 3 4 7 9
Example 2:

Input:
N = 10
arr[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
Output:
1 2 3 4 5 6 7 8 9 10

Expected Time Complexity: O(N*N).
Expected Auxiliary Space: O(1).

Constraints:
1 <= N <= 1000
1 <= arr[i] <= 1000

Problem Statement 6:
Given a random set of numbers, Print them in sorted order.

Example 1:

Input:
N = 4

arr[] = {1, 5, 3, 2}
Output: {1, 2, 3, 5}
Explanation: After sorting array will
be like {1, 2, 3, 5}.
Example 2:

Input:
N = 2
arr[] = {3, 1}
Output: {1, 3}
Explanation: After sorting array will
be like {1, 3}.
Expected Time Complexity: O(N * log N)
Expected Auxiliary Space: O(1)

Constraints:
$1 \leq N, A[i] \leq 105$

Problem Statement 7:
You are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n, representing the number of elements in nums1 and nums2 respectively.

Merge nums1 and nums2 into a single array sorted in non-decreasing order.

The final sorted array should not be returned by the function, but instead be stored inside the array nums1. To accommodate this, nums1 has a length of m + n, where the first m elements denote the elements that should be merged, and the last n elements are set to 0 and should be ignored. nums2 has a length of n.

Example 1:

Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3

Output: [1,2,2,3,5,6]
Explanation: The arrays we are merging are [1,2,3] and [2,5,6].
The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.
Example 2:

Input: nums1 = [1], m = 1, nums2 = [], n = 0
Output: [1]
Explanation: The arrays we are merging are [1] and [].
The result of the merge is [1].
Example 3:

Input: nums1 = [0], m = 0, nums2 = [1], n = 1
Output: [1]
Explanation: The arrays we are merging are [] and [1].
The result of the merge is [1].
Note that because m = 0, there are no elements in nums1. The 0 is only there to ensure the merge result can fit in nums1.

Constraints:

nums1.length == m + n
nums2.length == n
0 <= m, n <= 200
1 <= m + n <= 200
-109 <= nums1[i], nums2[j] <= 10^9

Problem Statement 8:

Given an array of integers nums, sort the array in ascending order and return it.

You must solve the problem without using any built-in functions in O(nlog(n)) time complexity and with the smallest space complexity possible.

Example 1:

Input: nums = [5,2,3,1]
Output: [1,2,3,5]
Explanation: After sorting the array, the positions of some numbers are not changed (for example, 2 and 3), while the positions of other numbers are changed (for example, 1 and 5).
Example 2:

Input: nums = [5,1,1,2,0,0]
Output: [0,0,1,1,2,5]
Explanation: Note that the values of nums are not necessarily unique.

Constraints:

1 <= nums.length <= 5 * 10^4
-5 * 104 <= nums[i] <= 5 * 10^4

Problem Statement 9:
You are given a 0-indexed integer array nums and a target element target.
A target index is an index i such that nums[i] == target.
Return a list of the target indices of nums after sorting nums in non-decreasing order. If there are no target indices, return an empty list. The returned list must be sorted in increasing order.

Example 1:

Input: nums = [1,2,5,2,3], target = 2
Output: [1,2]
Explanation: After sorting, nums is [1,2,2,3,5].
The indices where nums[i] == 2 are 1 and 2.

Example 2:

Input: nums = [1,2,5,2,3], target = 3
Output: [3]
Explanation: After sorting, nums is [1,2,2,3,5].
The index where nums[i] == 3 is 3.
Example 3:

Input: nums = [1,2,5,2,3], target = 5
Output: [4]
Explanation: After sorting, nums is [1,2,2,3,5].
The index where nums[i] == 5 is 4.

Constraints:

1 <= nums.length <= 100
1 <= nums[i], target <= 100

Problem Statement 10:
Given an array nums, return true if the array was originally sorted in non-decreasing order, then rotated some number of positions (including zero). Otherwise, return false.

There may be duplicates in the original array.

Note: An array A rotated by x positions results in an array B of the same length such that A[i] == B[(i+x) % A.length], where % is the modulo operation.

Example 1:

Input: nums = [3,4,5,1,2]
Output: true
Explanation: [1,2,3,4,5] is the original sorted array.

You can rotate the array by x = 3 positions to begin on the element of value 3: [3,4,5,1,2].
Example 2:

Input: nums = [2,1,3,4]
Output: false
Explanation: There is no sorted array once rotated that can make nums.
Example 3:

Input: nums = [1,2,3]
Output: true
Explanation: [1,2,3] is the original sorted array.
You can rotate the array by x = 0 positions (i.e. no rotation) to make nums.

Constraints:

1 <= nums.length <= 100
1 <= nums[i] <= 100

Problem statement 11:
Given an array arr[], its starting position l and its ending position r. Sort the array using the Merge Sort algorithm and also solve using the Quick Sort algorithm.
Example 1:

Input:
N = 5
arr[] = {4 1 3 9 7}
Output:
1 3 4 7 9
Example 2:

Input:
N = 10
arr[] = {10 9 8 7 6 5 4 3 2 1}
Output:

1 2 3 4 5 6 7 8 9 10

Expected Time Complexity: O(nlogn)
Expected Auxiliary Space: O(n)

Constraints:
1 <= N <= 10^5
1 <= arr[i] <= 10^5

Example 1:

Input:
N = 5
arr[] = { 4, 1, 3, 9, 7}
Output:
1 3 4 7 9
Example 2:

Input:
N = 10
arr[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
Output:
1 2 3 4 5 6 7 8 9 10

Expected Time Complexity: O(N*N).
Expected Auxiliary Space: O(1).

Constraints:
1 <= N <= 1000
1 <= arr[i] <= 1000

Problem Statement 12:
Given an array arr[] of size N, check if it is sorted in non-decreasing order or not.
Use quick sort
Example 1:

Input:
N = 5
arr[] = {10, 20, 30, 40, 50}
Output: 1
Explanation: The given array is sorted.
Example 2:

Input:
N = 6
arr[] = {90, 80, 100, 70, 40, 30}
Output: 0
Explanation: The given array is not sorted.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)
Constraints:
$1 \leq N \leq 10^5$
$1 \leq Arr[i] \leq 10^6$

Problem Statement 13:
Given an array of integers nums, sort the array in ascending order and return it.

You must solve the problem without using any built-in functions in O(nlog(n)) time complexity and with the smallest space complexity possible.
(Solve using merge sort)

Example 1:

Input: nums = [5,2,3,1]
Output: [1,2,3,5]
Explanation: After sorting the array, the positions of some numbers are not changed (for example, 2 and 3), while the positions of other numbers are changed (for example, 1 and 5).
Example 2:

Input: nums = [5,1,1,2,0,0]
Output: [0,0,1,1,2,5]
Explanation: Note that the values of nums are not necessarily unique.

Constraints:

1 <= nums.length <= 5 * 10^4
-5 * 104 <= nums[i] <= 5 * 10^4

Problem Statement 14:
Given an array of size N containing only 0s, 1s, and 2s; sort the array in ascending order. (sort using : Quicksort algorithm)

Example 1:

Input:
N = 5
arr[]= {0 2 1 2 0}
Output:
0 0 1 2 2
Explanation:
0s 1s and 2s are segregated
into ascending order.
Example 2:

Input:
N = 3
arr[] = {0 1 0}

Output:
0 0 1

Explanation:
0s 1s and 2s are segregated

into ascending order.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
1 <= N <= 10^6
0 <= A[i] <= 2

Problem Statement 15:
Given a random set of numbers, Print them in sorted order.
Use : merge and quicksort

Example 1:

Input:
N = 4
arr[] = {1, 5, 3, 2}
Output: {1, 2, 3, 5}
Explanation: After sorting array will
be like {1, 2, 3, 5}.
Example 2:

Input:
N = 2
arr[] = {3, 1}
Output: {1, 3}
Explanation: After sorting array will
be like {1, 3}.
Expected Time Complexity: O(N * log N)
Expected Auxiliary Space: O(1)

Constraints:
1 ≤ N, A[i] ≤ 105