**CSE221: Algorithms**

**Summer 2024 Lab 03**

**Task 01:**

In this problem, you will be given a list of numbers. You have to sort the list using the Merge Sort algorithm.

**Pseudocode of Merge Sort Algorithm:**

| def merge(a, b):  # write your code here  # Here a and b are two sorted list  # merge function will return a sorted list after merging a and b   def mergeSort(arr):  **if** len(arr) <= 1:  return arr  else:  mid = len(arr)//2  a1 = mergeSort(............) # write the parameter   a2 = mergeSort(............) # write the parameter  return merge(a1, a2) # complete the merge function above |
| --- |

**Note: You already have coded the merge function. Do you know which task?**

**Input**

The first line contains an integer N (1 <= N <= 105), denoting the length of Alice’s sorted list. In the next line, there will be N integers separated by space.

**Output:**

You have to sort the number using the Merge Sort algorithm and show the sorted list.

**Sample Input/Output:**

| Sample Input 1 | Sample Output 1 |
| --- | --- |
| 8  9 5 4 6 1 3 2 9 | 1 2 3 4 5 6 9 9 |
| Sample Input 2 | Sample Output 2 |
| 1  10 | 10 |
| Sample Input 3 | Sample Output 3 |
| 6  8 1 4 2 1 3 | 1 1 2 3 4 8 |
| Sample Input 4 | Sample Output 4 |
| 7  7 6 5 4 3 2 1 | 1 2 3 4 5 6 7 |

**Task 02**

Alice and you are playing with a list of **N** non negative integers. Today you will try to find out the maximum number of the list. Alice writes the following code to find the maximum number.

| maxValue = arr[0] for i in range(1,N):  if maxValue < arr[i]:  maxValue = arr[i] |
| --- |

Recently you have learned merge sort. Now, you are thinking if you can use the divide and conquer approach to find out the maximum from the given list.

**Please note, you are not allowed to sort the given list.** The motive for this task is not sorting but to use the concepts of divide and conquer approach.

**Input**

The first line contains an integer N (1 <= N <= 105), denoting the length of Alice’s list. In the next line, there will be N integers separated by space.

**Output:**

You have to find out the maximum value from the list using the divide and conquer approach.

**Sample Input/Output:**

| Sample Input 1 | Sample Output 1 |
| --- | --- |
| 8  1 7 13 4 5 7 13 12 | 13 |
| Sample Input 2 | Sample Output 2 |
| 7  5 15 2 3 10 1 9 | 15 |
| Sample Input 3 | Sample Output 3 |
| 1  9 | 9 |
| Sample Input 4 | Sample Output 4 |
| 6  5 2 3 10 1 9 | 10 |

What is the time complexity of your code?

**Task 03**

Somewhere in the universe, the Biannual Regional Alien Competition is taking place.

There are **N** aliens standing in a line. You will be given a permutation of N, which denotes the height of each alien. A sequence of N numbers is called a permutation if it contains all integers from 1 to N exactly once. For example, the sequences [3,1,4,2], [1] and [2,1] are permutations, but [1,2,1], [0,1] and [1,3,4] — are not.

In the competition, for each alien, the judge wants to count how many aliens are standing on its right side with a strictly smaller height. Then the judge wants to add up all the counts. To do this, the judge writes the following piece of code.







| **count** = 0 **for** i **in** **range**(**n**):  **for** j **in** **range**(i+1,**n**):  **if** H[i] > H[j]:  **count**+=1 |
| --- |

However, their algorithm wasn’t efficient at all. Hence, the alien calls you to write a better solution for the program.

More formally, you have to count how many pairs of aliens are standing in the line such that H[i] > H[j] and i < j. Here, A is a permutation of the aliens’ heights. And i,j denote the Aliens’ positions.

**Input**

The first line contains a single integer 1 <= N <= 106 - the number of total aliens.

The next line contains N integers H1,H2,…………,Hn(1 ≤ Hi ≤ N)- the height of the i-th alien. It is guaranteed that the given heights will be a permutation of N.

**Output**

Print a single integer, which denotes the total number of pairs (i, j) such that i < j and Hi >Hj.

**Sample Input/Output:**

| Sample Input 1 | Sample Output 1 |
| --- | --- |
| 5  1 2 3 4 5 | 0 |
| Sample Input 2 | Sample Output 2 |
| 5  5 4 3 2 1 | 10 |
| Sample Input 3 | Sample Output 3 |
| c | 11 |

**Sample Input 3 Explanation:**

In the sample input 3, the following pairs on alien’s heights satisfy the condition: (2,1), (7,4), (7,1), (7,5), (7,6), (7,3), (4,1), (4,3), (5,3), (6,3), (8,3)

**Task 04**

You are given a list A of N integers. You have to choose two indices i and j such that 1 <= i < j <= N and A[i] + A[j]2 is maximum possible. Here, we are considering 1-based indexing.

Write a code which will find the maximum value of A[i] + A[j]2 in O(N log N).

**Input**

The first line contains a single integer 1 <= N <= 106 - the length of the list.

The next line contains N integers A1,A2,…………,An (-108 ≤ Ai ≤ 108) separated by a space.

**Output**

Print a single integer - which denotes the maximum possible value of A[i] + A[j]2.

**Sample Input/Output:**

| Sample Input 1 | Sample Output 1 |
| --- | --- |
| 5  9 6 5 8 2 | 73 |
| Sample Input 2 | Sample Output 2 |
| 8  5 10 4 -3 1 6 -10 2 | 110 |
| Sample Input 3 | Sample Output 3 |
| 7  -5 -2 -6 -7 -1 8 2 | 63 |

**Task 05**

In this problem, you will be given a list of numbers. You have to sort the list using the Quick Sort algorithm in ascending order.

**Pseudocode of Quick Sort Algorithm:**

****

**[The code snippet has been taken from the book: Introduction to Algorithms]**

**Input**

The first line contains an integer N (1 <= N <= 105), denoting the length of Alice’s list. In the next line, there will be N integers separated by space.

**Output:**

You have to sort the number using the Quick Sort algorithm in ascending order and show the sorted list.

**Sample Input/Output:**

| Sample Input 1 | Sample Output 1 |
| --- | --- |
| 8  9 5 4 6 1 3 2 9 | 1 2 3 4 5 6 9 9 |
| Sample Input 2 | Sample Output 2 |
| 1  10 | 10 |
| Sample Input 3 | Sample Output 3 |
| 6  8 1 4 2 1 3 | 1 1 2 3 4 8 |
| Sample Input 4 | Sample Output 4 |
| 7  7 6 5 4 3 2 1 | 1 2 3 4 5 6 7 |

**Task 06**

In this problem, you will be given a list of numbers. You have to find the **k**-th smallest value from the list without sorting using the Partition function of Quick sort.

We will consider the 1-based indexing of the list.

**Input**

The first line contains an integer N (1 <= N <= 106), denoting the length of the list.

The next line contains N integers A1,A2,…………,An( 1 ≤ Ai ≤ 106) separated by a space.

The third line contains a single integer Q (1 <= Q <= 100) - which denotes the number of queries you have to answer.

Each of the next Q lines will contain a single integer K (1 ≤ K≤ N).

**Output:**

For each query, you have to find the K-th smallest number from the given list.

**Sample Input/Output:**

| Sample Input 1 | Sample Output 1 |
| --- | --- |
| 9 // Total Elements  10 11 10 6 7 9 8 15 2  4 // Total queries  5  3  2  7 | 9  7  6  10 |