CSE221_LabAssignment02_Summer2022

Submission Guidelines :

1. You can code all of them either in Python or Java. But you should choose a specific language for all tasks.
2. For each task write separate python files like task1.py, task2.py and so on.
3. For each problem, take input from files called "inputX.txt", and output at "outputX.txt", where X is the task number. So, for problem 2, the input file is basically this, "input2.txt". Same for output.
4. For each task include the input files (if any) in the submission folder.
5. Finally zip all the files and rename this zip file as per this format : **LabSectionNo_ID_CSE221LabAssignmentNo_Summer2022.zip**. [Example : **LabSection01_21101XXX_CSE221LabAssignment02_Summer2022.zip**]
6. Don't copy from your friends
7. You **MUST** follow all the guidelines, naming/file/zipping convention stated above. ***Failure to follow instructions will result in straight 50% mark deduction.***

Problem Descriptions

The following tasks need to be solved using Sorting Algorithms. Sorting algorithm is an algorithm that is used to arrange data in certain order, i,e- either ascending or descending order. We use sorting algorithms on numerical and lexicographical data to optimize the efficiency of other algorithms. You have learned several sorting algorithms such as bubble sort, insertion sort, selection sort, quick sort,merge sort etc. The problems below are related or similar to some of the sorting algorithms you learned in class. Read the task descriptions carefully and implement the algorithm using either Java or Python. The output format **MUST** match exactly as shown in each task.

Task : 1 [5 Marks]

Here is code of bubble sort. Its run time complexity is θ(n^2). Change the code in a way so that its time complexity is θ(n) for the **best case** scenario. Find the best case and change the code a little bit. And **explain how you did it in a comment block of your code.**

```
def bubbleSort(arr):
    for i in range(len(arr)-1):
        for j in range(len(arr)-i-1):
            if arr[j] > arr[j+1]:
                swap( arr[j+1], arr[j] )
```

The first line of the input will contain N, which is the size of the array. Next line will contain the N number of elements. Output will contain the sorted elements.

P.S: sample input and output may not be the preferred answer choice.

Input 1: 5 3 2 1 4 5	Input 2: 6 10 20 5 15 25 30
Output 1: 1 2 3 4 5	Output 2: 5 10 15 20 25 30

Task : 2 [5 Marks]

You have a list of elements and their prices. Select your preferred lists from the item based on lowest price. So to complete the task you have a tool called **selection sort.**
In selection sort:

- Select the minimum element from the unsorted part of the given array.
- Move the selected element to a sorted part of the array.
- Repeat this process to make the unsorted array sorted.

Here is pseudo code:

```
$for\ i \leftarrow 0\ to\ n - 1$
  $m = argmin_j\,(A[i],\ A[i + 1],\ .......\ A[n - 1])$
  $swap\ (A[i],\ A[m])$
$end\ for$
```

Use the above pseudo code to complete the selection sort.

First line of the input will contain **N** items and **M** preferred choices (where **M ≤ N**). The next line will contain the price of each element. Output will contain the price of **M** number of preferred elements.

Input 1: 5 3 5 10 2 1 4	Input 2: 7 4 10 2 3 4 1 100 1
Output 1: 1 2 4	Output 2: 1 1 2 3

Task : 3 [5 Marks]

Suppose you are given a task to rank the students. You have gotten the marks and id of the students. Now your task is to rank the students based on their marks using only **insertion sort.**

Here is the pseudocode for insertion sort for **ascending order**. You need to change it for **descending order**.

```
$for\ i \leftarrow 0\ to\ n - 1$
  $temp \leftarrow A[i + 1]$
  $j = i$
  $while\ j >= 0$
     $if(A[j] > temp)$
        $A[j + 1] \leftarrow A[j]$
     $else$
        $break$
    $j = j - 1$
  $end\ for$
$A[j + 1] \leftarrow temp$
$end\ for$
```

Implement this pseudocode to complete your task.

First line of the input file will contain an integer **N**. The next line will contain N number of id of the students.The next line will contain the N number of the marks of corresponding students. Output will be ranking the id based on their marks (descending order).

Input 1: 5 11 45 34 22 12 40 50 20 10 10	Input 2: 6 1 2 3 4 5 6 50 60 80 20 10 30
Output 1: 45 11 34 22 12	Output 2: 3 2 1 6 4 5

Task : 4 [5 Marks]

Here is the problem, just simply sorting an array. Now, to sort the array you should use efficient sorting techniques. It will have worst-case time complexity better than the above sorting algorithms. The sorting algorithm pseudocode is given below:

```
$MERGE\ (A,\ p,\ q,\ r\ )$
$n1 \leftarrow q - p + 1$
$n2 \leftarrow r - q$
$Create\ arrays\ L[1\ ..\ n1 + 1]\ and\ R[1\ ..\ n2 + 1]$
$FOR\ i \leftarrow 1\ to\ n1$
    $DO\ L[i] \leftarrow A[p + i - 1]$
$FOR\ j \leftarrow 1\ to\ n2$
    $DO\ R[j] \leftarrow A[q + j\ ]$
$L[n1 + 1] \leftarrow \infty$
$R[n2 + 1] \leftarrow \infty$
$i \leftarrow 1$
$j \leftarrow 1$
$FOR\ k \leftarrow p\ TO\ r$
    $DO\ IF\ L[i\ ] \leq R[\ j]$
        $THEN\ A[k] \leftarrow L[i]$
            $i \leftarrow i + 1$
        $ELSE\ A[k] \leftarrow R[j]$
            $j \leftarrow j + 1$


$MERGE - SORT\ (A,\ p,\ r)$

$if\ p < r$                       $//\ Check\ for\ base\ case$

  $q = (p + r)/2$

  $MERGE - SORT\ \ (A,\ p,\ q)$

  $MERGE\ - SORT\ (A,\ q + 1,\ r)$

  $MERGE\ (A,\ p,\ q,\ r)$
```

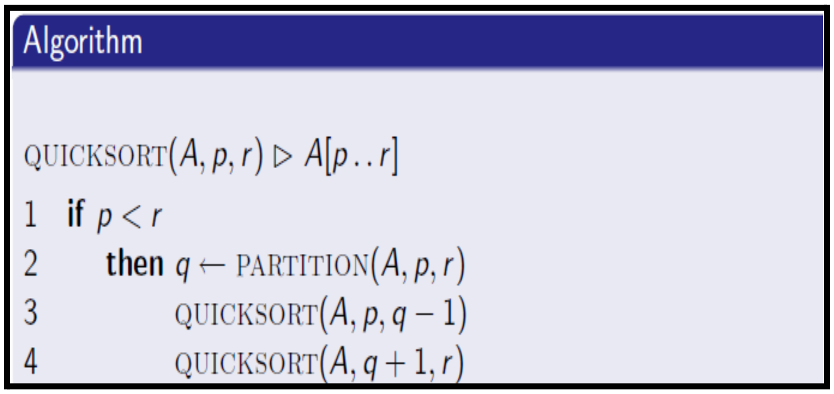
Take help from pseudocode or use your way to complete the task.

First line will contain N . The next line will contain N number of elements. The output will contain a sorted N number of elements (ascending order).

Input 1: 5 5 -1 10 2 8	Input 2: 6 10 20 3 40 5 6
Output 1: -1 2 5 8 10	Output 2: 3 5 6 10 20 40

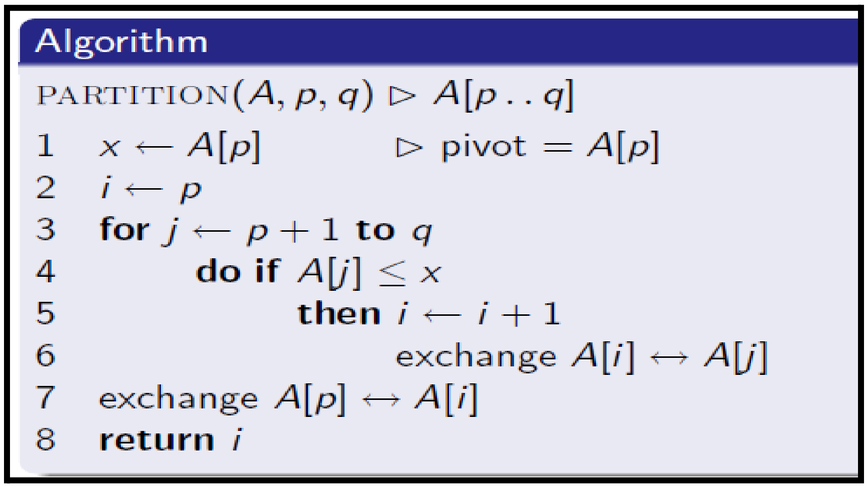
Task : 5 [5 Marks]

a. Study the algorithm below and implement the quickSort method . Additionally you will also need to implement the "partition" method. After sorting, print both the unsorted array and sorted array and also the time it takes to complete sorting.



Algorithm

```
QUICKSORT($A, p, r$) ▷ $A[p..r]$
if $p < r$
    then $q \leftarrow$ PARTITION($A, p, r$)
        QUICKSORT($A, p, q-1$)
        QUICKSORT($A, q+1, r$)
```



Algorithm

```
PARTITION($A, p, q$) ▷ $A[p..q]$
$x \leftarrow A[p]$        ▷ pivot $= A[p]$
$i \leftarrow p$
for $j \leftarrow p+1$ to $q$
    do if $A[j] \leq x$
        then $i \leftarrow i+1$
            exchange $A[i] \leftrightarrow A[j]$
exchange $A[p] \leftrightarrow A[i]$
return $i$
```

b. Implement an algorithm “findK” that uses the “partition” algorithm to find the kth (smallest) element from an array without sorting. E.g. for the array in our example, the 5^{th} element will be “9”

Input:
The array: 1 3 4 5 9 10 10
K=5
K=7
K= 2

Output:
 9
10
3