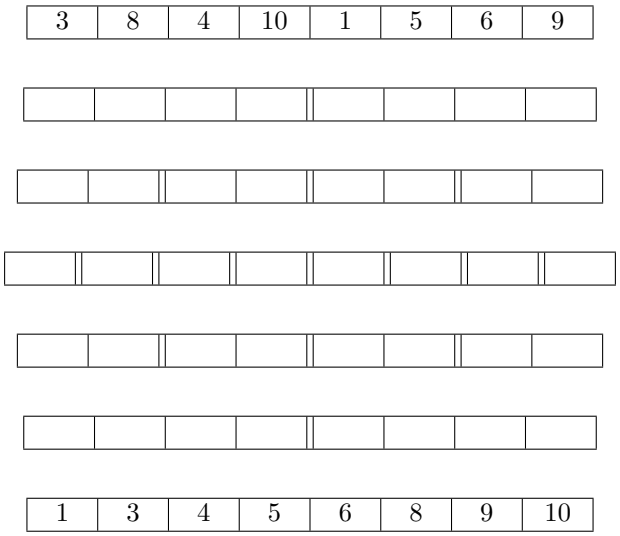
**SWE2016-44 Algorithms**

# Homework #1 (Due: September 26, 2019)

*Professor: Joon Hee Choi* *TA: Junsung Cho*

**Problem 1** Merge Sort

(a) Given the following array as input, illustrate how the Merge Sort algorithm performs. To illustrate the Merge Sort's behavior, start with the dividing of the array until the end condition of the recursive function is met and then show how the merge is performed. (5 pts)



(b) Given the following mergesort function, write down a merge function (only Pseudo Code). (15 pts)

**mergesort(arr, l, r)**

```
if (l<r) {
        m = l+(r-l)/2
}
mergesort(arr, l, m)
mergesort(arr, m+1, r)
merge(arr, l, m, r)
```

**merge(arr, l, m, r)**

```
var L as array = arr[l], ... arr[m]
var R as array = arr[m+1], ... arr[r]
```

**Problem 2** Asymptotic Notations

(a) What is the big O notation? Use the Tree method. (10 pts)

$$\begin{aligned} T(n) &= T(n/4) + T(n/2) + cn^2 \\ T(1) &= c \\ T(0) &= 0 \end{aligned}$$

(b) What is the big $\Theta$ notation? Use the Master method. (10 pts)

$$\begin{aligned} T(n) &= 5T(n/5) + \sqrt{n} \\ T(1) &= 1 \\ T(0) &= 0 \end{aligned}$$

**Problem 3** Quick Sort

(a) Given the following array [10, 5, 3, 9, 22, 24, 28, 27, ?] and assuming that Quicksort will be used to sort this array in ascending order, select values for the last element of the array (indicated by "?") such that the partitioning performed by Quicksort is most balanced. Explain why this makes Quicksort perform efficiently. (10 pts)

(b) The pseudo code shown below represents the implementation of a function that partitions an array around a pivot element. Briefly explain what the purpose of the two while loops in line 6 and 7 is. (10 pts)

**partition(arr, leftBound, rightBound)**

```
1:        pivot = arr[rightBound]
2:        left  = leftBound
3:        right = rightBound - 1
4:
5:        while(true) {
6:                while(arr[left]<pivot)                        {left++ }
7:                while(right>leftBound && arr[right]>=pivot) {right--}
8:                if (left >= right) {
9:                        break
10:                }
11:                else {
12:                        swap arr[left] and arr[right]
13:                }
14:        }
15:        swap arr[left] and arr[rightBound]
16:        return left
```

(c) Given the partition function shown in problem 3b, complete the pseudo code shown below for the QuickSort function (10 pts).

**quicksort(arr, left, right)**

```
if(right-left<=0) {
        ----------------------------------
} else {
        pi = ------------------------------
        ----------------------------------
        ----------------------------------
}
```

**Problem 4** Coding Assignment

- Submit your source code for solving the following question via **icampus**. Use C, C++, Java or Python. Do **NOT** copy other codes.
- For this problem, your goal is to sort an array of 0, 1 and 2's but you must do this in place, in linear time and without any extra space (such as creating an extra array). This is called the **Dutch national flag** sorting problem. For example, if the input array is $[2, 0, 0, 1, 2, 1]$ then your program should output $[0, 0, 1, 1, 2, 2]$ and the algorithm should run in O(n) time. (30 pts)