CS1020E: Data Structures and Algorithms I

Tutorial 9 – Sorting

(Week 11, starting 24 October 2016)

1. *Divide and Conquer!*

Explore the differences between Quick Sort and Merge Sort. While there are a number of different implementations available, use the version of Quick Sort, partition, and Merge Sort discussed in lectures. You are given this array as sample input. Do note that there are two different elements with key value 8.

5	9	12	7	**8**	51	31	8

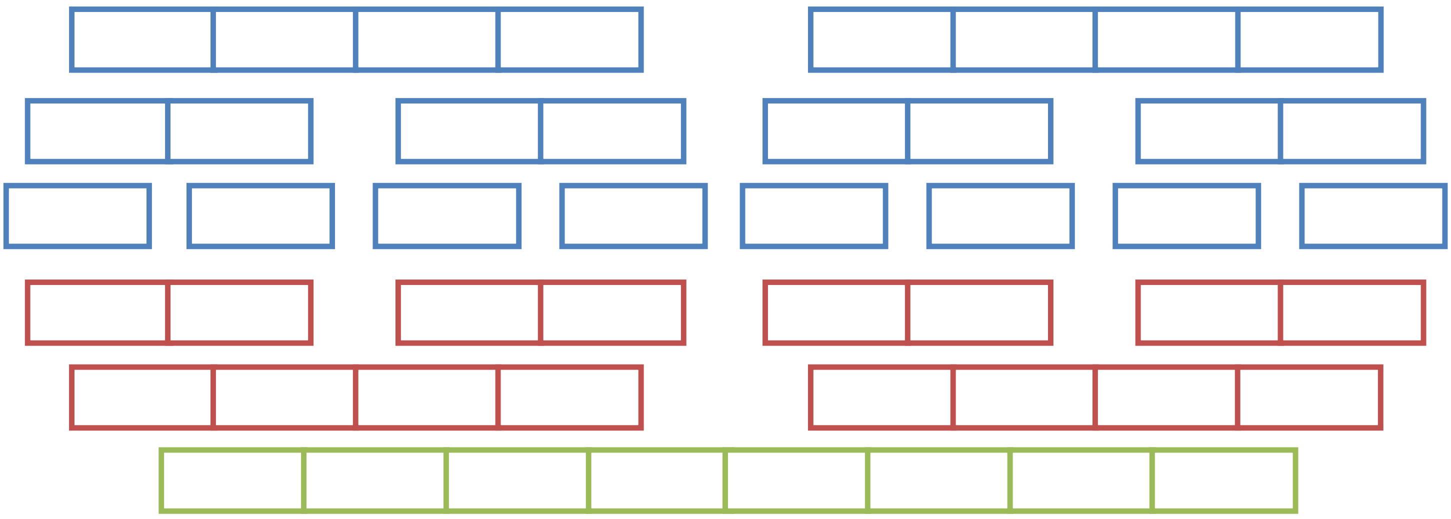
(a) One advantageous property of quick sort is that it is ___________, while merge sort is ___________.

(b) Display the contents of the array after each `partition()` call completes. Mark the pivot. What do you notice about the two elements with the same key value, and what causes this to happen?

1st Step								
2nd Step								
3rd Step								
4th Step								
5th Step								
6th Step								

(c) Using the same array, trace the execution of Merge Sort. Which boxes represent the temporary arrays, and which boxes are just logical (contents of the original array)?

5	9	12	7	**8**	51	31	8



2. Choice of Sorting Algorithm

In this question, consider only 4 sorting algorithms: Insertion Sort, Quick Sort, Merge Sort, and Radix Sort. Choose the fastest sorting method that is suitable for each scenario.

(a) You are compiling a list of students (ID, weight) in Singapore, for your CCA. However, due to budget cut, you are facing a problem in the amount of memory available for your computer. After loading all students in memory, the extra memory available can only hold up to 20% of the total students you have! Which sorting method should be used to sort all students based on weight (no fixed precision)?

(b) After your success in creating the list for your CCA, you are hired as an intern in NUS to manage a student database. There are student records, already sorted by name. However, we want a list of students first ordered by age. For all students with the same age, we want them to be ordered by name. In other words, we need to preserve the ordering by name as we sort the data by age.

(c) After finishing internship in NUS, you are invited to be an instructor for CS1010E. You have just finished marking the final exam papers randomly. You want to determine your students' grades, so you need to sort the students in order of marks *(yes, that Bell Curve system).* As there are many CA components, the marks have no fixed precision.

(d) Before you used the sorting method in (c), you realize the marks are already in *near* sorted order. However, just to be very sure that you did not cut and paste a student record in the wrong order, you still want to sort the result.

3. Comparison vs Non-Comparison Sort

`arr` is an **unsorted** integer array of length N (very long), containing only non-negative values.

(a) Print all integers that appear at least k times in the array. **Requirement:** Time complexity of O(N log N)

(b) Now, the array contains only integers within the range [0, 1000]. Print the integer that appears the most number of times. If more than one integer appears the same number of times, print the first to reach that number of occurrences. The array has to be sorted in ascending order afterwards. **Requirement:** Time complexity of **O(N)**…

Yikes! =O [Hint: Count…]

O(1)

more weeks to go!!!