CS1020E Tutorial + Lab 09

Mark NG

a0116298@u.nus.edu
http://mollymr305.github.io

October 27, 2016

Tutorial Solutions

“Tutorial 9 – Sorting”

Question 1: Divide and Conquer!

First... see `https://visualgo.net/sorting`

Question 1: Divide and Conquer!

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

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

Question 1: Divide and Conquer!

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

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

Answer

Quick Sort: In place; Merge Sort: Stable

Question 1: Divide and Conquer!

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

(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								

Question 1: Divide and Conquer!

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

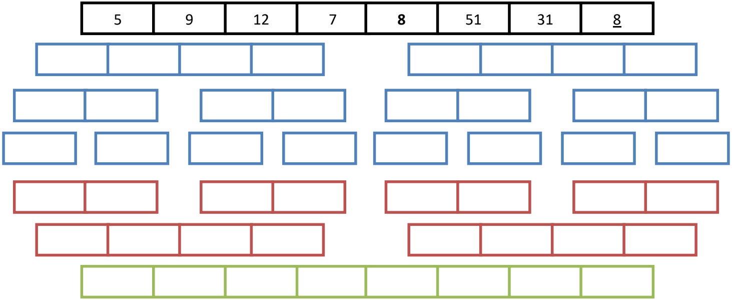
Answer

1st Step	5	9	12	7	**8**	51	31	8
2nd Step	5	8	7	**8**	9	51	31	12
3rd Step	5	7	8	**8**	9	51	31	12
4th Step	5	7	8	**8**	9	12	31	51
5th Step	5	7	8	**8**	9	12	31	51
6th Step								

Question 1: Divide and Conquer!

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

(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)?

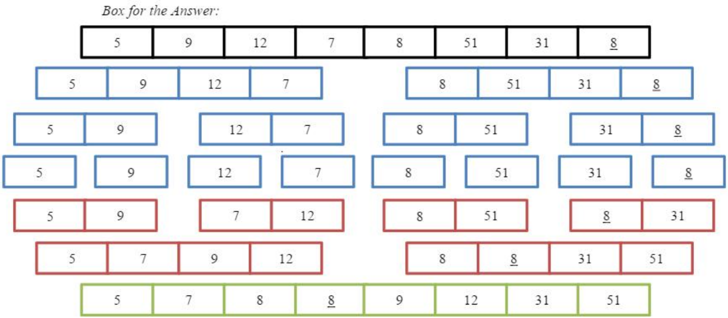


5	9	12	7	**8**	51	31	8

Question 1: Divide and Conquer!

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

Answer



Box for the Answer:

5 9 12 7 8 51 31 8

5 9 12 7   8 51 31 8

5 9   12 7   8 51   31 8

5   9   12   7   8   51   31   8

5 9   7 12   8 51   8 31

5 7 9 12   8 8 31 51

5 7 8 8 9 12 31 51

Question 2: Choice of Sorting Algorithm

(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)?

Question 2: Choice of Sorting Algorithm

(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)?

(a) Quick Sort. Due to **memory** constraint, you will need an **in-place** sorting algorithm. Hence, a sorting algorithm that is both in-place and **works for floating point** is Quick Sort. Do note that: The system requires some extra space on the call stack, due to the recursive implementation of Quick Sort (and similarly for Merge Sort), although we say that Quick Sort is in-place.

Question 2: Choice of Sorting Algorithm

(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.

Question 2: Choice of Sorting Algorithm

(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.

(b) Radix Sort. The requirements call for a **stable** sorting algorithm, so that the **ordering by name is not lost**. Since memory is not an issue, Radix Sort can be used. Radix Sort has a lower time complexity than comparison based sorts here, O(dn) where d = 2, vs O(n log n) for Merge Sort.

Question 2: Choice of Sorting Algorithm

(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.

Question 2: Choice of Sorting Algorithm

(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.

(c) Quick Sort. Being a **comparison-based** sort, Quick Sort is able to sort **floating point** numbers, unlike Radix Sort. Quick Sort is also a good choice because the grades are **randomly distributed**, resulting in O(n log n) average-case time.

Question 2: Choice of Sorting Algorithm

(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.

Question 2: Choice of Sorting Algorithm

(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.

(d) Insertion sort. Insertion sort has an O(n) best-case time, which occurs when elements are already in **almost sorted order**. Suppose there are 10 students that have swapped place with the next student. We will then only make 20 extra key comparisons and 10 shifts. Hence, we get O(n) time.

Question 2: Choice of Sorting Algorithm

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)

Question 2: Choice of Sorting Algorithm

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)

Answer: $\mathcal{O}(n \log n)$ **sort, followed by** $\mathcal{O}(n)$ **scan.**

Question 2: Choice of Sorting Algorithm

(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)**...

Question 2: Choice of Sorting Algorithm

(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)**...

Answer: $\mathcal{O}(n)$ **counting sort, followed by** $\mathcal{O}(n)$ **scan.**

End of Tutorial Discussion

Note: Detailed solutions (i.e. the file `T9_ans.pdf`) will be released soon at

`http://www.comp.nus.edu.sg/~stevenha/cs1020e.html`

Take Home Lab

Some notes...

Take Home Lab

Some notes...

- Quite easy compared to older labs.

Take Home Lab

Some notes...

- Quite easy compared to older labs.
- You might find the `cmath` library useful, or not...

Let’s take a short break!

Sorting Problems

- https://open.kattis.com/problems/sortofsorting
- https://open.kattis.com/problems/sidewayssorting

Any Questions?

See you next week!