

## CS2040S: Data Structures and Algorithms

### Tutorial Problems for Week 4: Sorting

*For: 30 Aug 2021, Tutorial 2*

#### **IMPORTANT: How to “Describe an algorithm”?**

You will be required to describe algorithms and write pseudocode in the tutorials, midterm exam and final exam. Having the right algorithm in your mind is not sufficient — you need to present it in a way that is clear and can be easily understood by others. Below are some of the things we expect from you when you are asked to describe / give / design an algorithm.

**Declare all important data structures and variables at the start.** Start your algorithm by listing all the data structures and important (usually global) variables, and what they should contain. If necessary, provide a short description of their purpose.

**Be concise.** There is no need to describe your algorithm as a single paragraph or chunk of text. You can write your algorithm in point form, as a sequence of steps, or in short sentences.

**No need to describe details of data structures / algorithms discussed in lecture or tutorial.** If you intend to use any data structure / algorithm covered in lecture or discussed in tutorial **without modification**, do not spend time describing them again. You may simply quote the name of the data structure or algorithm. For example, if you intend to sort the array  $A$ , you can simply say “Sort array  $A$ ”, instead of writing out the entire mergesort or quicksort algorithm. **However, if you intend to modify a data structure / algorithm, you must describe the modification in full detail (see next point).**

**No “black boxes”.** Any data structure / data structure operation / algorithm not discussed in lecture / tutorial must be described in full detail. For example, if you intend to truncate a linked list, you should describe exactly how you do so — even if it may seem obvious to you, “truncate” is not a linked list operation!

**Use the correct terminology.** Know the correct names and terms of the various data structures and algorithms, and use them correctly. For example, do not describe a modification of the mergesort algorithm, and say that it is a modification of the insertion sort algorithm.

**Be clear.** Ultimately, the goal is for another person to be able to understand your algorithm easily. You can draw diagrams or use an example to illustrate your algorithm. Note that however, your diagram / example **cannot** replace the description!

### Problem 1. Choice of Sorting Algorithm

In this question, consider only the following 4 sorting algorithms: **Insertion Sort**, **Quick Sort**, **Merge Sort**, and **Radix Sort**. Choose the most optimal sorting algorithm that is suitable for each of the following scenarios, and justify your choice along with any assumptions you make.

**Problem 1.a.** You are compiling a list of students (ID, weight) in Singapore, for your CCA. However, due to budget constraints, 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 algorithm should be used to sort all students based on weight (no fixed precision)? Why?**

**Problem 1.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. **Which sorting algorithm should be used to sort the data first by name, then by age, given that the data is already sorted by name? Why?**

**Problem 1.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. As there are many CA components, the marks have no fixed precision. **Which sorting algorithm should you use to sort the student by marks? Why?**

**Problem 1.d.** Before you used the sorting method in Problem 1c, you realize the marks are already in 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. **Which sorting algorithm should you use? Why?**

### Problem 2. $k$ -th smallest element

Given an **unsorted** array of  $n$  non-repeating (i.e unique) integers  $A[1 \dots n]$ , we wish to find the  $k$ -th smallest element in the array.

**Problem 2.a.** Design an algorithm that solves the above problem in  $O(n \log n)$  time.

**Problem 2.b.** Design an algorithm that solves the above problem in expected  $O(n)$  time. Briefly explain why your algorithm is correct. *Hint: Modify the quicksort algorithm.*

### Problem 3. Waiting for the Doctor

There are  $n$  patients currently waiting to see the doctor. The  $i$ th patient requires an estimated consultation time of  $t_i$  minutes, as determined by the triage.

There is only one doctor, and the doctor can only serve a single patient at any point in time, and all other patients must wait. For example, if the first patient that the doctor serves has  $t_i = 5$ , the remaining  $n - 1$  patients must all wait for an additional 5 minutes. You may assume that once the doctor finishes serving a patient, he will immediately serve the next patient — any time in between serving two patients is negligible.

The doctor can serve the patients in any order. The doctor must serve all patients. The doctor wishes to minimize the total waiting time of all patients. **Describe the most efficient algorithm you can think of to find the minimum total waiting time required to serve all patients. What is the running time of your algorithm?**

### Problem 4. Missing Family Members (AY18/19 Sem 4 Midterm)

The Addams family have just gone on a fishing trip and taken a photo. The tradition of the Addams family is to line up for their family photos, where each member of the family will wear a shirt having a number  $x$  where  $1 \leq x \leq N$  ( $N$  being the number of members in the family). The oldest will wear shirt 1, the second oldest will wear shirt 2, and so on and the youngest will wear shirt  $N$ .

Now if there are 6 members in the family, they could line up as such: 1,2,4,5,6,3.

After coming back from the trip, the Evve family who has been at odds with the Addams family has cast a spell on the photo and remove some of the members from the lineup in the photo. So if the original sequence is 1,2,4,5,6,3, after removing 2 and 5 from the photo, it will result in the remaining subsequence 1,4,6,3.

In order to fix the photo, Michael Addams, the patriarch of the family, needs to know the exact sequence of the lineup in the photo. However, no one can quite remember the exact lineup except that if one were to order all permutations of 1 to  $N$  ascending order of the permutation sequences (e.g 1,2,3 will have the permutations in ascending order as 1,2,3, 1,3,2, 2,1,3, 2,3,1, 3,1,2, 3,2,1), the original sequence in the photo will be the first such permutation that contains the remaining subsequence (sequence of the members remaining in the photo).

You are now tasked with finding the original lineup sequence given  $N$  and the remaining subsequence  $S$ . **Describe the most efficient algorithm that you can think of to solve this problem. What is the running time of your algorithm?**