CS2040 Tutorial 5

Week 7, starting 26 Sep 2022

Q1 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, but 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 just once, 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. As there are many CA components, the marks have no fixed precision

(d) For the same data in part (c), you realize the marks already seem to be 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

Q2 Missing Family Members

The Addams family had just gone on a fishing trip and taken a photo of all of their family members standing side by side in a long line. Each member of the family will wear a distinct shirt labelled 1..**N**, **N** being the number of people in the family and can be large. Unfortunately, the photo was edited with some family members being removed (but no one has their positions changed/swapped)

If the original sequence of shirts in the photograph is always the first permutation of 1..**N** that contains the sequence of all family members in the edited photo, find an **efficient** way to reproduce this original sequence

For example, suppose **N**=6 and [1, 4, 6, 3] was observed in the edited photo. The original sequence is [**1**, 2, **4**, 5, **6**, **3**] as that is the first permutation of {1,2,3,4,5,6} that contains [1, 4, 6, 3] in that order

```
int[] findOrigSeq(int N, ArrayList<Integer> editedSeq)
```

Q3 Waiting for Doctor

You work at a 'high-tech' clinic with only ONE doctor. Before the clinic opens for the day, **ALL N** patients will be waiting outside the clinic, and have made an appointment to visit the doctor

As the appointment system keeps track of a patient's history and medical condition, it is able to assign for each patient **i**, the time **t_i** required to serve patient **i**. As the doctor can only see 1 patient at any time, how do you minimize the total waiting time of all patients?

(a) Implement an **efficient algorithm**, returning the total waiting time of all patients for the day. The waiting time of a patient refers to the time elapsed from the start of the day to the point the patient starts to visit the doctor

(b) If each serving time **t_i** is an integer in [1, 3**N**), can we do **better** than O(**N** log **N**) time?

```
int computeMinWait(List<Integer> servingTimes)
```

Question 4 (Online Discussion) – Still Waiting for Doctor?

As in Q3, if now each serving time **t_i** is an integer in [1, $\mathbf{N}^3$), can we do **better** than O(**N** log **N**) time?

(See next page…)

Notes on mid-course feedback:

- Tutorials are designed to solidify your understanding of lecture material, identify any gaps you may have and give you some opportunity to apply your knowledge
- We want tutorials to bridge the gap from lecture to lab, not just spam application questions. There is a very large variance in existing skill level this semester with a large cohort, so it is all the more important that we spend the 40-45min focusing on the basics
- If you want more application questions, we can consider adding more for online discussion if there is discussion on the weekly Q4 online discussion question, but currently there isn't much/any discussion on it so…
- Problem solving and application is **expected** at this level… Tracing ability can be self-taught and assisted by lectures, visualgo and self-practice… Likewise expect such thinking and application in the written assessments
- Yes the application problems require some time to process and come up with ideas (just like lab), some unfortunately wrong, some inefficient, but this will help you improve and you WILL get better
- If you want the tutorial answer to be less wordy, just focus on the code / how it works depending on which you think is useless, though that compromises learning. We provide both because it is very important to see the idea behind the solution, yet low-level design and implementation is also very important at the same time
- Not everyone will take / have taken CS2030. If we include some minimal material that overlaps with CS2030, it is because there is practical use for it in labs/in real life (e.g. List<SomeType> ref pointing to ArrayList<SomeType> object). The answers to such questions will be simple enough to self-learn by googling (e.g. java API Comparable) and trying out in <10min
- Few public test cases are given in **labs** as you are expected to create your own test cases covering edge cases, large inputs (that may cause overflow / loss of precision / TLE) etc. For every problem there are some cases that an algorithm is at risk of not handling well, train yourself to identify such cases as you do the tutorials and labs

Continue to work hard and have fun! =) Yes, you may be busy, but this course DOES require lots of consistent hard work as you are learning a language, many DSes, many algos, how they can be applied to solve problems, how efficiency matters… So **strategize** and **make the hard work meaningful** 😊☻ …!!