Lab 06 (Optional) Specification – A Hand-on Exercise to practice One More Sorting Algorithm.
50 points

Due by: 04/29/2021 2:00 PM (Individual Lab)

Lab Goals

- Sharpening your thinking skills to solve algorithmic challenges.
- Practice implementing algorithms in a programming language of your choice (Python or Java).

Learning Assignment

If not done previously, it is strongly recommended to read all of the relevant "GitHub Guides", available at the following website:
`https://guides.github.com/`
that explains how to use many of the features that GitHub provides. This reading assignment is useful to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, it is also recommended to do the reading assignment from the section of the course textbook outlined below:

- **Sedgewick chapter 02**

Assignment Details

Now that we have discussed a great deal of sorting algorithms and analyzed these algorithms asymptotically in the last few lectures, it is now time to do it practically. In this lab, students will practice developing a new sorting algorithm that was not discussed in our class discussions so far. This also includes developing one or more code files to implement a series of algorithms using a programming language such as Python or Java. At any duration during and/or after the lab, students are recommended to team up with the Professor and the TL(s) to clarify if there is any confusion related to the lab and/or class materials.

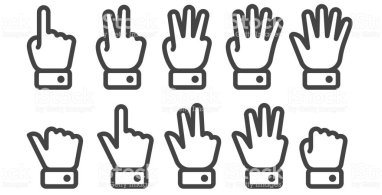
The Professor proof-read the document more than once, if there is an error in the document, it will be much appreciated if student(s) can communicate that to the Professor. The class will be then informed as soon as possible regarding the error in the document. Additionally, it is highly recommended that students will reach out to the Professor in advance of the lab submission with any questions. Waiting till the last minute will minimize the student chances to get proper assistance from the Professor and the Technical Leader(s).

Students are recommended to get started with this part in the laboratory session, by discussing ideas and clarifying with the Professor and the Technical Leader(s). It is acceptable to discuss high-level ideas with your peers, while all the work should be done individually. Late submission is accepted for the part(s) in this section, based on the late policy outlined in the course syllabus.

It is required for all students to follow the honor code. Some important points from the class honor code are outlined below for your reference:

1. Students are not allowed to share code files and/or other implementation details. It is acceptable to have a healthy discussion with your peers. However, this discussion should be limited to sharing ideas only.
2. Submitting a copy of the other's program(s) and algorithm(s) is strictly not allowed. Please note that all work done during lab sessions will be an opportunity for students to learn, practice, and master the materials taught in this course. By doing the work individually, students maximize the learning and increase the chances to do well in other assessments such as skill test, exams, etc · · ·

An implementation of the Counting Sort algorithm. (50 points)



Counting sort assumes that we have a known, reasonably small, range of values to be sorted. For simplicity, let's assume that all possible values in the array/list `a[0]`, `a[1]`, ..., `a[n-1]` belong to the range $0, 1, 2, \ldots, M-1$ for some "reasonable" constant M that does not depend upon the array/list size n. ("Reasonable" means, for instance, something that is much smaller than n, or that does not exceed the maximum array/list size allowed by a particular language or system.)

Create an `int` array/list named `count` of size M, initialized to all zeros. Make one pass through the array/list `a` and, for each element `a[i]`, add 1 to the counter for the value `a[i]`. Then run through the values in the `count` array/list and, for each i from 0 through $M-1$, place `count[i]` copies of i into the array/list `a`. Here's a simple example with $n = 15$ and $M = 5$:

BEFORE:

a =	4	3	0	1	2	4	0	2	1	3	4	1	0	2	2

, count =

0	0	0	0	0

AFTER PASS THROUGH `a`:

a =	4	3	0	1	2	4	0	2	1	3	4	1	0	2	2

, count =

3	3	4	2	3

AFTER PASS THROUGH `count`:

a =	0	0	0	1	1	1	2	2	2	2	3	3	4	4	4

, count =

3	3	4	2	3

The running time here is $O(n)$, since we must examine each element of `a` just once. (We must also examine each element of `count`, but since its size M is a constant independent of n, this just adds a constant amount of time.)

Obviously, this method is useless when we don't know in advance what the range of values will be, or when the range is not easy to enumerate (think about sorting real numbers like 3.1540023, -0.0000003021, and 38827233.3772), or when the range is very large (we wouldn't want to create a `count` array/list of size 2^{32}, for instance, but there are 2^{32} possible values in a Java `int` variable).

1. First, understand thoroughly the algorithmic solution provided above. The description provided above may seem confusing. Please read the description a few times and try to work on a few examples using the process outlined. Discuss with your peers, the TL's and the Professor if the solution is not fully understood.
2. Implement the algorithm proposed above using Python or Java programming language. For simplicity, a starter code-named `CS.java` and `cs.py`are provided. The input of the program is automatically generated within the starter-code using a randomized sequencing. The output of the program should be generated using the ascending order format. Make necessary code modifications to this starter-code file. Read through the comments in the code file for additional information on the structure of the code.
3. The implementation is left out as an open-ended problem in this specification. This is simply because to give the student an opportunity to think through and come up with their own implementation ideas.

Honor Code

Make sure to **Sign** the following statement in the `honor-code.txt` file in your repository. To sign your name, simply replace Student Name with your name. The lab work will not be graded unless the honor code file is signed by you.

This work is mine unless otherwise cited - Student Name

Submission Details

For this assignment, please submit the following to your GitHub repository by using the link shared to you by the Professor:

1. Modified source code in either Python "cs.py" or Java "CS.java"program.
2. A signed honor code file, named `honor-code.txt`.
3. To reiterate, it is highly important, for you to meet the honor code standards provided by the college. The honor code policy can be accessed through the course syllabus.

Grading Rubric

1. This lab is optional. If you complete this lab, the lab score will be used to swap with any of your lowest lab score. An opportunity to boost your lab scores.
2. There will be full points awarded for the lab if all the requirements in the lab specification are correctly implemented. Partial credits may be awarded if deemed appropriate.
3. Failure to upload the lab assignment code to your GitHub repository will lead to receiving no points given for the lab submission. In this case, there is no solid base to grade the work.
4. There will be no partial credit awarded if your code doesn't compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before the due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student's responsibility to let the professor know about it after the final submission in GitHub. In this way, an updated version of the student's submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not submitted any code, then, in this case, there are no points awarded to the student.
5. If a student needs any clarification on their lab grade, it is strongly recommended to talk to the Professor. The lab grade may be changed if deemed appropriate.