

## CS 456 : Advanced Algorithms

### Programming Assignment #01

Total Points: 150

**Assigned Date** : Wednesday, January 25, 2017  
**Due Date** : Wednesday, February 08, 2017 @ 02:59:59 p.m.

### Overview

For your first programming assignment, you will implement a **Bucket sort** similar to the one found in Sec 8.4 of your CLRS book. However, instead of using the insertion sort used in the pseudocode on p. 201, you will implement three different versions using a Quicksort, Bubblesort, and Radixsort – and then empirically validate the asymptotic runtime behavior for *best case*, *average case*, and *worst case* using **computer generated results**. More specifically, you are expected to think about and address the following questions:

- At what size  $n_0$  does your implementation start to exhibit asymptotic complexity?
- What is the characteristic of your input required to generate *average* complexity. How about best case and worst case scenarios? How do you plan to generate the appropriate input?
- How does the measured run time correspond to the abstract complexity analysis using operation counting (as discussed in class)?
- How to create your test driver so that it exercises your sort programs.
- How to create the sorting class so that it will be extensible and reusable for future projects.
- How does initially sorting the input into buckets affect the runtime of the different sort algorithms?

### Instructions

- This is an individual assignment. **Do your own work.**
- Start early!!**
- Take backups of your code often!!.**
- Make sure to test your program properly before your final submission.
- You may use any programming language of your choice. However, you **must** make sure that your code compiles and runs on a typical Linux machine. Failure to compile and run successfully on a Linux machine or not providing a makefile will result in a loss of points. Absolutely **DO NOT** include executables or more than the few sample input files requested below with your submissions. A Makefile is mandatory.
- As shown below, input for your program should come from a file. Your program must ask the user to input a file name.
- As shown below, the file must contain a list with one integer per line.
- The output format and information must be consistent with the example shown below.

- The report part of your solution must be produced using a word processor. Any figures, graphs, plots, etc., should also be produced using appropriate computer applications. Be professional with your reports; properly label and title your graphs; properly caption and cross-reference your figures; make sure to include all sections/subsection mentioned below. Please recognize the importance of the report to your overall project grade and be sure to give enough time to writing a thorough report.
- Your final report should be in **PDF** format. No exceptions.
- Follow a good coding standard. Use the Google C++ coding standard found here <http://goo.gl/1rC1o>, if you don't already follow one.
- Total points: [150 points]

## Deliverables

The due date of this assignment is **Wednesday, February 08, 2017 @ 02:59:59 p.m.** A dropbox will be opened for submission on Moodle before the due date. A complete solution comprises of:

- [120 points] A report that includes the followings:
  - Motivation and background of the experiment [10 points].
  - Pseudocode of the different algorithms appropriately annotated with an invariant proof [20 points].
  - Testing Plan (for best/average/worse cases) and Test Results. This section should include graphs of the results. In particular, you should have several graphs displaying the time vs. number of buckets with consistent input for each different sorting method as well as graphs comparing the different runtimes of the sorting methods with a consistent number of buckets and changing input size. Be sure to test very large data sets. [20 points].
  - A correctness proof of your programs [20 points].
  - Problems Encountered/Key insights [10 points].
  - Justification of your observations. You must be able to justify and/or argue the empirical asymptotic behavior you are observing [20 points].
  - Conclusion and performance comparisons. This includes comparing the different sorts against each other as well as how the number of buckets chosen effects each individual sorting method. This section should also explain the graphs that were displayed in the test results section [20 points].
- [30 points] A compressed tarball of the directory containing your source codes, Makefile, and sample test files for best/average/worse case behaviors. Do not include executables in this tarball; we will do a fresh compile of your code using your Makefile. To create a compressed tarball of the directory `source`, use the following command: `tar -zcvf name-pr1.tar.gz source/`. Obviously, change the name to your last name.
  - Correct implementation of the algorithms [15 points]
  - Correct input procedures [5 points]
  - Correct output procedures [5 points]
  - Good coding practices e.g. naming conventions, readable code, commenting, etc. [5 points]

## User input screen

Please enter a text file to be searched: test.txt

How many buckets: 2

## Sample input file

**test.txt**

17  
63  
31  
95  
51  
38  
30  
22  
33  
83  
32  
64

## Sample output file

**testOut.txt**

Sort Size: 12

Number of Buckets: 2

Bubble Sort: .0073s

Quick Sort: .0071s

Radix Sort: .0083s

17  
22  
30  
31  
32  
33  
38  
51  
63  
64  
83  
95