CS 456 : Advanced Algorithms

Programming Assignment #01

Total Points: 150

Assigned Date : Monday, February 01, 2016
Due Date : Monday, February 15, 2016 @ 02:59:59 p.m.

Overview

For your first programmming assignment, you will implement two versions of **Quicksort** – the regular quicksort and the randomized quicksort (see chapter 7 of the main textbook) – and empirically validate their asympototic 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:

a. At what size n_0 does your implementation start to exhibit asymptotic complexity?
b. 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?
c. How does the measured run time correspond to the abstract complexity analysis using operation counting (as discussed in class)?
d. How to create your test driver so that it exercises your sort programs.
e. How to create the sorting class so that it will be extensible and reusable for future projects.

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. Absolutely **DO NOT** include executables with your submissions. A Makefile is mandatory.
- For input, the program must allow a user the choice between choosing random numbers or reading from a file. If random is selected, the user should be asked how many points. If a file is selected, the user should be asked to supply a filename to be read.
- For output, the program should print to a file the two times for the sort as well as the sorted list. The times should be on the first two lines followed by the sorted list with only one number per line. See example below.
- The output file should be named sorted[number of points]. See example 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.
- 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.

Deliverables

The due date of this assignment is **Monday, February 15, 2016 @ 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 your algorithm appropriately annotated with an invariant proof [**20 points**].
  - Testing Plan (for best/average/worse cases) and Test Results [**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 [**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-111-pr1.tar.gz source/`. Obviously, change the name to your last name and 111 to the last three digits of your SIUE ID.
  - 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**]

Sample input file

sample10.txt

5
7
2
15
1
12
6
3
14
8

Sample output file

sorted10.txt

QuickSort: .0125s
Random Quicksort: .0212s
1
2
3
5
6
7
8
12
14
15