CS 456 : Advanced Algorithms

Programming Assignment #03

Total Points: 150

Assigned Date : Wednesday, April 05, 2017
Due Date : Wednesday, April 19, 2017 @ 02:59:59 p.m.

Overview

Your third **programming assignment** focuses on randomized algorithms and compares them to their non-randomized counterparts. In particular, you will focus on a **Las Vegas Algorithm** implementation of quicksort, the **Randomized Quicksort**, compared to a fixed pivot **Quicksort**. Also, you will explore a **Monte Carlo Algorithm** implementation of square matrix multiplication verification known as **Freivald's Algorithm** and use **Strassens Matrix Multiplication Algorithm** we discussed in class to verify your results.
Recognize early that probabilistic analysis has an added layer of difficulty that you have not utilized in previous projects, and the **report is important**, so start early. Las Vegas Algorithms are characterized by providing a correct solution with a fluctuating run time while utilizing a randomizing element. For the quicksort, you should design your project to compare the running time of the two versions and display those results on varying input sizes and distributions. Monte Carlo Algorithms are different because they can't guarantee a correct solution but do have a fixed runtime while also using a randomized element. To demonstrate this principle, use the output of the Strassen algorithm and verify the result with Freivald's algorithm. In addition to finding when the algorithm gives false negatives, alter the output of Strassen's algorithm to also test the probability of giving a false positive. An excellent lecture on this topic can be found at https://www.youtube.com/watch?v=cNB2lADK3_s. Specifically, you are expected to think about and address the following questions:

a. For quicksort, what are the input conditions that most affect the runtimes of the different algorithms?
b. For matrix multiplication verification, how does the size of the input affect runtime and its probability of correctness?
c. What is the asymptotic bound of the various algorithms?
d. How does the runtime of the matrix multiplication compare to its probabilistic verification?
e. How does the measured runtime correspond to the theoretical complexity analysis?

Instructions

- This is an individual assignment. **Do your own work**.
- **Start early!!**
- **Take backups of your code often!!**.

- 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. You **MUST** submit a makefile.
- The report part of your solution must be produced using a word processor. I highly recommend **Latex**. Any figures, graphs, plots, etc., should also be produced using appropriate computer applications. Graphs/plots should be properly labeled. Your final report should be in **PDF** format. No exceptions.
- Follow a good coding standard. Use Google coding standard for your language found here `https://google.github.io/styleguide/`, if you don't already follow one.
- For input, give user the option to choose between a filename or random input.
- If a filename is chosen, the user must be prompted to name two files. If random is chosen, the user is prompted to enter first, the length of one side of the matrix, and second, the size of the input to be sorted by the quicksort algorithm.
- One file must have the length of one side of the matrix on the first line, followed by the numbers comprising the matrix in a left to right top to bottom order with only one element per line. Immediately after the last element of the first matrix is read, the first element of the second matrix must begin. Remember that these are square matrices containing the same number of elements. It should be named **matrix[length of side].txt**.
- The second file should be called **sort[#of elements].txt** and should only contain the numbers to be sorted, one per line. See below.
- For output, generate a file named **[#of sorted elements]Out.txt**. The top four lines should include the times of the four algorithms followed by whether the verification algorithm produced the correct response followed by the solution to the matrix multiplication in n X n matrix form. Last should be the sorted list from quicksort.
- Be sure to test enough different size files to accurately graph behavior.
- Total points: [**150 points**]

Deliverables

The due date of this assignment is **Wednesday, April 19, 2017 @ 02:59:59 p.m.** A dropbox will be opened for submission on Moodle before the due date. A complete solution comprises of:

- A report that includes the followings:
  - Motivation and background of experiment. A detailed explanation of Las Vegas and Monte Carlo algorithms is required. [**5 points**]
  - Pseudocode of your algorithms appropriately annotated with the theoretical runtime analysis. It is advised to add a code walkthrough of the algorithms that explains why they have the time complexity that they have.[**12 points**]
  - Testing Plan and Test Results including explanations of the sample input files used to test your algorithms. In addition, some discussion of how you designed your code for easy manipulation to minimize the testing time is required. [**12 points**]
  - Graphs displaying the timing results of your algorithms. Be sure to discuss the complexity of the algorithms and how closely your program follows the theoretical complexities. A minimum of the following graphs are required:
    - A theoretical-actual comparison graph for the timing results of the **QuickSort Algorithm**. [**2 points**]

    - A theoretical-actual comparison graph for the timing results of the **Randomized Quick-Sort Algorithm**. [**2 points**]
    - A theoretical-actual comparison graph of the timing results of the **Strassen Algorithm**. [**2 points**]
    - A theoretical-actual comparison graph for the timing results of the **Freival Algorithm**. [**2 points**]
    - A comparison graph between the two quicksort runtimes[**4 points**]
    - A comparison graph of the two matrix multiplication algorithm runtimes[**4 points**]
    - A graph representing the number of runs versus the accuracy of the matrix multiplication verification method.[**4 points**]
  - Justification of your observations. You must be able to justify and/or argue the empirical asymptotic behavior you are observing. Be sure to discuss what graph sizes and properties begin to display asymptotic behavior, for which algorithms do the different graphs run best, which input properties cause worst case, and any graphs that produced irregular results. You must also discuss the probabilities and proofs concerning the validity of the randomized algorithms. This is the main focus of the assignment[**10 points**]
  - Conclusion and problems encountered/key insights. Explain in your own words what you learned from this experiment.[**5 points**]
- [**86 points**] A compressed tarball of the directory containing your source code and **(Makefile. Important:** Be sure to test your code on a linux machine. No exceptions! 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_pr3.tar.gz source/`. Obviously, change the name to your last name.

Tentative Grading Rubric

- Styling: Easy to read, properly indented, etc..., [8 points]
- Input specifications [8 points]
- Output Specifications [10 points]
- Algorithmic Correctness[60 points]
  - QuickSort Algorithm [10 points]
  - Randomized Quicksort Algorithm[10 points]
  - Strassen's Algorithm [20 points]
  - Freival's Algorithm [20 points]

- Report Specifications[64 points]

Sample Set Up

First matrix:
1 3
6 2
Second matrix:
3 1

5 2

Sample input file

matrix2.txt

2
1
3
6
2
3
1
5
3

sort5.txt

3
6
2
10
1

Sample output file

5Out.txt

QuickSort: .002s
Randomized QuickSort: .003s
Freival's Algorithm: .021s
Strassen's Algorithm: .031s

Matrix Multiplied

yes
18 7
28 10

Sorted List 1

2
3
6
10