CSE 2320 Experimentation Assessment Project

Design Submission: April 1, 2010
Final Submission: April 29, 2010

Goal:

Demonstration of "the ability to design and conduct experiments, analyze and interpret data" (ABET outcome b) based on the following narrative:

Quicksort partitioning may be implemented in several ways, including the two versions in Notes 8. In addition, section 7.4 of Sedgewick suggests that small "subfiles" be ignored during quicksort and then processed by insertion sort. Besides processing arrays of various key types, quicksort may be used as a ***suffix array*** construction algorithm (SACA) for an input string/sequence (array of characters). Conceptually, a suffix array orders the suffixes of a string based on the conventional string comparison function. So, given the input string "`abcdabcdabc`", the array representation and suffix array are:

```
        0  1  2  3  4  5  6  7  8  9 10 11
        a  b  c  d  a  b  c  d  a  b  c \0

sa  11  8  4  0  9  5  1 10  6  2  7  3
lcp -1  0  3  7  0  2  6  0  1  5  0  4
```

The *longest common prefix* (`lcp[i]`) is the number of prefix matches for `sa[i-1]` and `sa[i]`.

Your task is to ***evaluate*** the performance of quicksort on tables of random integers and for suffix array construction using both partitioning methods under various values for the subfile "cutoff".

Requirements:

The following requirements (with weights for the design + final submissions) are to be satisfied by submitting a preliminary report (parts 1, 2, and 3) by 10:45 a.m. on April 1 (graded by April 13) and a final report (parts 1, 2, 4, 5, 6) by 10:45 a.m. on April 29. Both submissions are to be sent as e-mail attachments to `fawaz.bokhari@mavs.uta.edu`.

1. Proposed solution and background information. (10% + 5%) You may assume your reader has a copy of Sedgewick, so this section should be short.

2. Testable hypotheses. (10 + 5%) These should be relevant to the task of comparing the methods. Some preliminary executions will be useful for formulating hypotheses and (1.).

3. Java code for collecting performance statistics. (30 + 0%) This will not be long, but should still follow the expectations in the course syllabus.

4. Description of the collected data. (0 + 10%) Be sure that someone reading your report gets a good overview.

5. Collected data as tables or graphs. (0 + 15%) Since you will need to work with fairly large tables, summarized data will be useful.

6. Conclusions with support from the data. (0 + 15%) Consideration of errors and discussion of possible additional work.

Getting Started:

1. The package `java.util.Date` is convenient for capturing elapsed time for sections of code. The following code times `Arrays.sort()`:

```
Date start=new Date();
Arrays.sort(arr);
Date stop=new Date();
double seconds=(stop.getTime()- start.getTime())/1000.0;
System.out.format("Arrays.sort for %d ints took %f seconds\n",n,seconds);
```

   `System.nanoTime()` may also be used.

2. The class `java.util.Random` is convenient for generating random data. The time to generate random data is ***not*** of interest.

3. Random data is inappropriate for the suffix array construction evaluation. Text files of a consistent nature, such as source files or books from Project Gutenberg, are useful. You should replace linebreaks by a space. The mean of the `lcp` values is a good measure of the degree of repetition in a text.

Grading Rubric:

C1: Questions and related background show that student clearly understands the issues to examine.

5: Summary indicates that student understands the issues and what should be explored.
3: Summary indicates that student should learn something from their experiments.
1: Not clear that student will be performing an organized experiment.

C2: Student has decomposed the problem into one or more experimentally testable hypotheses.

5: Hypotheses indicate that student performed initial work leading to testable hypotheses for the given problem.
3: Hypotheses indicate that initial work was hastily performed.
1: Hypotheses are flawed and may not be testable.

C3: The components (e.g. code for 2320) have been implemented, tested, and could be used by others.

5: Components are designed to be useful for anyone performing related experiments.
3: Components are insufficient for performing necessary experiments.
1: Components are incorrect.

C4: The nature of the collected data is accurately described.

5: Quantity of data is appropriate, but not overwhelming, for drawing conclusions.
3: Quantity of data is barely sufficient for drawing conclusions.
1: Experimental set-up precludes obtaining data.

C5: Data is presented appropriately in tables or graphical forms.

5: Trends in the data are clearly identifiable.
3: Trends appear, but are not obvious.
1: Data is very incomplete.

C6: Conclusion(s) regarding hypotheses are presented with support from the data. Possible errors are noted. Remaining issues are discussed, along with additional experimental work that could be performed.

5: Conclusions for all hypotheses, with support from data.
3: Conclusions are related to hypotheses, but questionable support from data.
1: Conclusions related to hypotheses are lacking.