CS350 TERM PROJECT (PROPOSAL)

BEN CARR AND RUSSELL MILLER

Our term project will focus on sorting algorithms. We would like to compare several sorting algorithms:

- Bubble, average $O(n^2)$, worst-case $O(n^2)$
- Insertion, average $O(n^2)$, worst-case $O(n^2)$
- Quick, average $O(nlgn)$ worst-case $O(n^2)$
- Merge, average and worst-case $O(nlgn)$

(We may also try bucket sort (average $O(n)$, worst-case $O(n^2)$), time permitting, as it is more complex to implement.)

Our goal is to implement each of these algorithms in Python, Java, C++ and Haskell, (approximately 20 algorithms in all). Russell will write the Python and Java code, and Ben will write C++ and Haskell. We will test our algorithms on unordered lists of integers. Each algorithm will sort the same 1000 lists of size $n = \{10\ 50\ 100\ 500\ 1,000\ 5,000\ 10,000\ 50,000\ 100,000\ 500,000\ 1,000,000\ 5,000,000\ 1,000,000,000\}$ that will be read from an external data file. The data we collect from these tests will be the execution time taken for the algorithm to sort the list. Once we have collected this data, several comparisons can be made. I would like to compare the following:

- How algorithm A in programming language L behaves as the size n of the list gets larger.
- How algorithm A_n compares to A_n in languages L_n, L_m. (Same algorithm, different language)
- The consistency or average of the algorithms after executing on list of size n.

Date: February 17, 2011.

We are creating a generator program that will aide in this process by:

- creating the sample data files (lists of n integers)
- executing and timing our algorithms
- writing the results to a .csv file

Our final report will have a discussion of our algorithms, and comparisons of their performance. I would like to graph the comparisons of the algorithms against each other, and also create "consistency graphs" that show that the expected complexities are consistent. We will also discuss how complexity theory shows up in practice.

Our goals for the next three weeks are as follows:

WEEK 1:

1. make a generator that:
    - creates sample data in an external file
    - executes and times our algorithms (in any language)
    - writes the results to a .csv file
2. Finish at least two algorithms in C++ and Python that we can use to test the generator.
3. Test .csv file in spreadsheet program

WEEK 2:

1. Finish the rest of the C++ and Python algorithms
2. Graph results of all implemented algorithms
3. Start Java and Haskell implementation

WEEK 3:

1. Finish writing and graphing Java and Haskell algorithms
2. Analyze/Discuss our findings
3. Write up our results for term paper.

So far we have been able to start the generator program that will run the executables, and also have the random list-generator working.

If you'd like to see our current progress, please visit our git repository:
`https://github.com/millertime-homework/cs350report`