Merge, Quick, and Radix Sort

Chapter 4 provides an introduction, analysis, and pseudo code for various sorting algorithms. These include Insertion Sort, Bubble Sort, Shell Sort, Radix Sort, Heap Sort, Merge Sort, and Quick Sort. Comparisons and criteria of applicability are discussed. Being one of the fundamental computer applications, it is important to understand each of the techniques.

Assignment

Merge and Quick Sort

Implement the Merge and Quick Sort algorithms using the pseudo code on pages 107/109 and 115/116 respectively. Instrument each algorithm, counting comparisons, as follows:

- Merge Sort: one comparison for each time through the while loop within MergeLists
- Quick Sort: one comparison for each time through the for loop within PivotList

Test your code for correctness adding print statements as necessary

Monte Carlo Simulation

Merge Sort

- Run the program on data sets consisting of random integers in the range of 0 to 100000. Run data sizes of 10, 100, 1000, 10000, 100000, 1000000 and plot size vs. count using Microsoft Excel or some other plotting package. Note that you should seed the random number generator so that each test uses the same random sequence.

Quick Sort

- Set the pivot point to be the first location of the list and run the program on data sets consisting of random integers in the range of 0 to 100000. Run data sizes of 10, 100, 1000, 10000, 100000, 1000000 and plot size vs. count using Microsoft Excel or some other plotting package. Note that you should seed the random number generator so that each test uses the same random sequence.
- Set the pivot point to be the last location of the list and run the program on data sets consisting of random integers in the range of 0 to 100000. Run data sizes of 10, 100, 1000, 10000, 100000, 1000000 and plot size vs. count using Microsoft Excel or some other plotting package. Note that you should seed the random number generator with the same value used above so that each test uses the same random sequence.
- Set the pivot point to be a random location in the list and run the program on data sets consisting of random integers in the range of 0 to 100000. Run data sizes of 10, 100, 1000, 10000, 100000, 1000000 and plot size vs. count using Microsoft Excel or some other plotting

package. Note that you should seed the random number generator so that each test uses the same random sequence.

- Set the pivot point to be the first location of the list and run the program on data sets consisting of in sequence (sorted smallest to largest) of 0 to 100000 for a list of 100000. Run the program and report the number of comparisons.
- Set the pivot point to be the first location of the list and run the program on data sets consisting of in sequence (sorted largest to smallest) of 100000 to 0 for a list of 100000. Run the program and report the number of comparisons.

Radix Sort (optional)

Implement the Radix Sort algorithm using the pseudo code given in the text but instead of sorting integers, implement it to alphabetize words. You may assume words consist of lower case letters only (not case sensitive) and have no punctuation marks. Demonstrate your implementation on the following list of words:

zero, one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen, seventeen, eighteen, nineteen, twenty

Deliverables

- Source code
- A reflection document including
    - Plot size vs. count for both sort algorithms for various sized lists of uniformly distributed random numbers (two separate plots)
    - Counts for ordered data lists in Quick Sort algorithm
    - Description of changes to the pseudo code to make radix sort operate on words (if implemented – optional)
    - Essay describing successes and difficulties

Reflection document must be a PDF file. Do not submit documents of type MSWord, Pages, OpenOffice, etc.