CSci 1933 Lab 9

November 10, 2015

1. Introduction

The goal of this lab is to compare the running time of a Merge Sort implementation and a Insertion Sort implementation. You will do this by adding in counters to the two implementations that count the number of comparisons and the number of writes.

2. Getting started

Create a new java project in IntelliJ name Lab9 and import the files: AbstractSorter.java, CompareByIntegerValue.java, InsertionSort.java, MergeSort.java, Sorter.java,SortResults.java into the src/ directory of the project.

These files do the following:

- AbstractSorter.java is an abstract class that implements the Sorter interface. You will provide implementations of the methods from Sorter.
- CompareByIntegerValue.java is a simple comparator to sort two Integers in ascending order.
- InsertionSort.java is a subclass of AbstractSorter.java that implements the insertion sort algorithm in it's sort method.
- MergeSort.java is a subclass of AbstractSorter.java that implements the merge sort algorithm in it's sort method.
- Sorter.java is an interface that defines a "loggable" Sorter class. Implementing classes must implement a sort method and a set of methods to count the number of comparisons and the number of writes done in a sort.
- SortResults.java is a class that simply has a main method that creates four arrays, sorts them with each algorithm, and then displays the results.

3. Your tasks

Your job is to implement the methods in AbstractSorter.java, and add counters into the sort implementations. You will then analyze the results that are printed out:

3.1. Implement the methods within AbstractSorter.java. See the source for the specific method signatures.

3.2. Modify InsertionSort.java and MergeSort.java to include method calls to `incNumComparisons()` and `incNumWrites()` in the proper places to keep track of the two different counts. Note: there may be multiple different places in each file that each method will need to be called. You will need to call each method in the appropriate places, as defined below:

3.2.1. A write is defined as: consisting of writing to the array being sorted or (in the case of merge sort), the temporary arrays used in merging.

3.2.2. A comparison is defined as: every time the compare method of a comparator is invoked.

3.3. Run SortResults.java to see the results. Four different lists are being sorted in this example. There are two large lists of size 100,000 and two small lists of size 10. There is one small list and one large list that are already sorted and one small and one large list that contain random values. Look over the results and try to understand what the results mean. Remember the `big-O()` running time of each algorithm and compute what the results should be by hand. Compare the results between the large and small lists between the two algorithms and compare the results between the sorted and random lists between the two algorithms. Notice how the algorithms do compared to each other in each case.

Solution will be available in last 15 mins.