CSE 241 Assigment #8 - Sorting Spring 2015

Due date: Jun. 13 20:00

Sorting

In this assignment you will implement several sorting algorithms. Once you have implemented one of the algorithms, an animation will allow you to visualize how it works on random input.

Sequence interface

You will be sorting an instance of the Sequence interface, which is defined as follows:

```
public interface Sequence<E> {
    public int size();
    public void put(int index, E value);
    public E get(int index);
    public void swap(int index1, int index2);
}
```

1. The type parameter E represents the type of element to be stored in the sequence.
2. The size method returns the number of elements in the sequence.
3. The put method stores a value at a the given index in the sequence.
4. The get method retrieves the value of an element at the given index and returns it.
5. The swap method swaps the values of the elements located in the two locations referred to by index1 and index2.

Sort interface

Your sorting algorithm implementations will be classes that implement the Sort interface, which is defined as follows:

```
import java.util.Comparator;

public interface Sort<E> {
    public void sort(Sequence<E> sequence, Comparator<E> comparator);
}
```

The sort method sorts the given Sequence, using the given Comparator object to compare elements in the sequence.

Sorting algorithms

You should implement the following sorting algorithms in each java source file

1. BubbleSort.java
2. SelectionSort.java
3. InsertionSort.java
4. ShellSort.java
5. QuickSort.java

except SelectionSort.java. In order to show how your sorting classes should work with the provided classes that implement the interfaces, selection sorting algorithm is already implemented in the provided source code.

The other class skeleton codes are also provided for you (/home/cse241/assign8/), with empty sort methods in each for you to fill in with actual code.

As some of the algorithms were not covered in our lectures, you need to google and study the algorithms for yourself.

As for the Shell sort, you should use the gap originally proposed by Shell. That is, $\lfloor N/2^k \rfloor$.

And, as for the Quick sort, you should choose the middle $\lfloor (left\_index + right\_index)/2 \rfloor$ element of the partition as the pivot.

Getting Started

To compile, simply type "make".

```
$ make
javac -g BubbleSort.java
javac -g QuickSort.java
javac -g SelectionSort.java
javac -g ShellSort.java
javac -g InsertionSort.java
javac -g SortDemoSequence.java
javac -g TextSequence.java
$
```

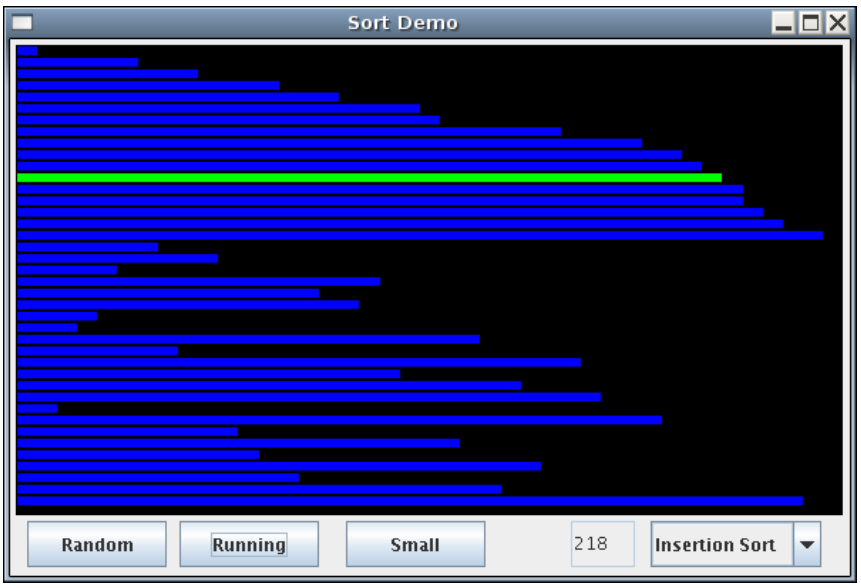
To test your algorithm, you can run either the SortDemo program or SortText program. In order to run SortDemo - GUI JFrame application, you should install Xming in your Windows PC, which you can download from http://sourceforge.net/projects/xming/

After installation, click Xming shortcut from the desktop and open Properties window, and modify the options in the Target text box as follows.

```
Xming\Xming.exe":0 -clipboard -multiwindow -ac
```

If you are using Mac OS or Linux, just login to class machine with ssh -X option.

After login, run java SortDemo. You will see a window that looks like this:



The "Random" button randomly shuffles the input. The button labeled "Large" or "Small" chooses between two input sizes. Note that Insertion Sort and Bubble Sort will take a long time to complete on the Large input size. The button labeled "Sort!" starts the sort algorithm selected by the combo box at the lower right. (It will display "Running" while the sort algorithm is running.) The text box with the counter displays how many operations (calls to get and put) the sorting algorithm has performed.

As this GUI window may slow down network connection to our class machine when many of you students are viewing their applications at the same time, I strongly recommend you not only to use uni06 but to use uni07, uni08, uni09, or uni10.

Instead of the GUI demo, you can also test your algorithm using SortText program. SortText class creates TextSequence instance that implements Sequence interface, reads input text file, run your sorting algorithms, and prints out the sorted text.

```
$ java SortText input.txt Selection
Sorting...
Bruce
Elayna
Lisa
Marcus
Amber
Ryan
Dorian
Joel
Kelly

Done sorting
Amber
Bruce
Dorian
Elayna
Joel
Kelly
Lisa
Marcus
Ryan
Number Of Operations: 12
```

Submitting Your Code

You must submit all files in your working directory using the "oopsubmit" command as follows:

```
$ cd <your working directory>
$ oopsubmit assign8 .
```

First you should go to your working directory that contains all project java source files. And, do not forget putting the last argument of oopsubmit command - dot (.).

If you do not follow this submission guideline, you will lose 10 points out of 100.

You should also submit a hard copy of your code to TA. Your report must have a cover page with your student ID and name. **Please do not print out other provided class source codes. Only the sorting algorithm classes should be included in the hard copy.** Of course, in the report your code must be well commented to explain your class.