GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Department of Computing & Mathematics

CP2SD – Data Structures & Algorithms
Lab 8: Sorting Algorithms

Sorting algorithms are an important topic in computer science and are used for applications as diverse as adversarial search of a game tree to arranging table views in a GUI. In this lab we will examine four different search algorithms, three of which execute in $\mathbf{O}(n^2)$ time and one with an average running time of $\mathbf{O}(n \log n)$. While there is virtually no difference in the running time for small sets of input data, as the input size increases, the more efficient algorithms exhibit a far superior running time.

- Download the Zip archive ***lab8-sorts.zip*** from Moodle and extract the files into a folder on your hard disk. As the classes involved are all packaged, ensure that you create the correct directory structure and study the source code before you compile.

The Zip archive contains the following files:

- ***characters.txt:*** a small data set containing a list of characters from a well-known book(s).
- ***dictionary.txt:*** a sorted dictionary of 234,936 words.
- ***Word.java:*** A wrapper class for a string.
- ***Dictionary.java:*** Reads and parses a file of strings into an *ArrayList* of *Word* objects. Note that the class uses the ***Collections.shuffle()*** method to randomly shuffle the dictionary.
- ***Sortable.java:*** An interface that defines a comment ***sort()*** method implemented by each of the sorting algorithms.
- ***SelectionSort.java:*** Sorts an array by making several passes through the array, selecting a next smallest item in the array each time and placing it where it belongs in the array.
- ***BubbleSort.java:*** Compares adjacent array elements and swaps their values if they are out of order.
- ***InsertionSort.java:*** partitions an array into sorted and unsorted parts and sequentially inserts each element into its correct position.
- ***QuickSort.java:*** A divide an conquer sort algorithm with an O(n log n) time complexity.
- ***SortRunner.java:*** Instantiates the Dictionary object and executes the sort algorithms.

Examine each of the sorting algorithms and familiarise yourself with the their basic idea of operation. Will we discuss each of these algorithms in detail at lectures. Apart from the quick sort, all the algorithms have a quadratic running time, $\mathbf{O}(n^2)$. Make sure you understand the parts of the algorithm that contribute significantly to the time complexity.

- Execute the class *SortRunner.java*, after modifying the *Dictionary* to parse the file *characters.txt*.
- Change the *Dictionary* class to parse the file ***dictionary.txt*** and execute the quick sort. The algorithm should sort the dictionary of words in a reasonable amount of time.
- Sort the dictionary of words using the quadratic sorting algorithms. Call *System.getCurrentTimeMillis()* to calculate efficiency of each of the algorithms.