CS1713 ALGORITHM DESIGN AND ANALYSIS

# Assignment 3

## Idea

In this assignment you will write code for sort methods to sort integer arrays. Then you will time them under different conditions to assess and compare their runtimes.

## Method

1. Write 6 sort methods in one file, **Sort**, with no **main**. Use the pseuodocode given in the book for Selection Sort (P. 99), Insertion Sort (P. 134), Bubble Sort (P.100), Merge Sort (P. 172), Shellsort (handout) and regular Quicksort (P 176).

2. Create **SortTest** whose **main** will test the six sorts. Test each sort on 3 integer arrays of size 20. One array will be a sorted array, one a reverse sorted array and the third a random filled array. Display the array before and after each sort.

3. Create **Assign3** that will time 4 sorts (do not do Quick Sort and Merge Sort). Each sort will operate on three different types of array – sorted (k = 1), reverse sorted (k = 2) and randomly filled (k = 3). You will determine the size of the arrays. You should choose a starting size N and time each sort (and each array) for size N, 2N, 4N, 8N.

The pseudocode for **Assign3** will be as follows:

**For (Size = N, 2N, 4N, 8N) {**

**Declare array(s)**

**For (Type of Array, k = 1, 2, 3) {**

**For each sort {**

### Fill array depending on k and Size

**Start Timer**

**Sort Array**

**Stop Timer**

**Calculate and display elapsed time**

**}**

**}**

**}**

4. Run your program three times, record times and average your results. For each of the 4 sorts, summarize your results in an Excel table as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Size | k | Time1 | Time2 | Time3 | Ave Time  12 lines in table for each sort |
|  |  |  |  |  |  |

## General Notes

1. Timing should be done to 3 decimal places. When running your timing program first close down all other applications and take all your measurements during the same session. Allow plenty of time (2 – 3 hours).

2. The choice of N is very important. It will depend on the speed of your computer and must be low enough to make sure that the slow sorts are not too long but high enough that we can time the faster sorts. You may still have a problem timing the faster sorts in which case you can try rerunning **Assign3** with higher array sizes and commenting out slower sorts.

3. The sorted and reverse sorted arrays should contain the integers from 1 up to the current size of the array. The random-filled array should contain integers that are randomly generated (duplicates allowed) in that same range – **do not hardcode your own.**

**Help**

1. Do not use any pre-written data structures or methods from libraries and collections - create your own regular arrays and write sorts from scratch. In C++ the arrays should be dynamic and deleted when no longer needed. The **Sort** methods should be **public** and **static** - a static method is invoked through its classname Ex. **Sort.ShellSort(A)** where A is an array.
2. Both languages have ways to time an interval.

#### Turning in the Assignment

1. Print out your Excel table, Assignment Information Sheet, all source code. Put them in a folder and turn in.
2. Upload all source and compiled files to CourseWeb (preferably zipped). Do not forget to upload the **SortTest** program.

Assignment due Monday 30th September.