|  |  |
| --- | --- |
| CSC 210 | Sort Comparisons Lab |
|  | Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Copy the Lab4-Sorts folder from the K: drive common area to your K: drive folder. It contains a copy of this document along with a number of Java sorting demos. The two you will be using to complete this worksheet are SlowSort.java, which performs a bubble sort, and QuickSort\_v2.java.

1. Modify the slow sort in the following ways and then run it:
2. Have it print your name once at the top of the output.
3. Have it count *the number of comparisons* that are made and the *number of swaps* that are made. You may just use two global variables to accumulate these counts.
4. Have it print out these two variables, with appropriate labels, along with printing the number of elements it sorted and the number of seconds the sort took.
5. Capture this output screen for the 25,000 element run and paste it into a Word file.
6. Modify the faster sort the same way you did the slow sort. However, instead of counting the number of comparisons and swaps, have it count the number of swaps and the *number of times the recursive sort routine is called*. (Do this by counting once each time it is *entered* rather than where it was called from.) Capture this output screen for the 25,000 element run and paste it into the same Word file.

***Warning:*** When you run the sort to do the timings make sure you are not running any other applications at the same time. If, for example, you check your email or surf the web while a sort is running in the background, the applications will be sharing the CPU and you will get an inaccurate timing for your sort.

3. Run each of the two sort for all the requested number of elements, and use the output to complete the following two charts either neatly by hand or in Word. Print this sheet to hand in along with the Word file holding your run outputs.

**Bubble Sort**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Compares** | **Swaps** | **Time (sec.)** | **Time as F(N)** |
| N = 25,000 | 312,487,500 | 157,324,708 | 1.08 | /////////////////////// |
| N = 50,000 | 1,249,975,000 | 624,849,007 | 4.52 | /////////////////////// |
| N = 125,000 | 7,812,437,500 | 3,903,717,884 | 29.31 | /////////////////////// |
| N = 250,000 | 31,249,875,000 | 15,637,314,002 | 116.23 | /////////////////////// |
| **N** | /////////////////////// | /////////////////////// | /////////////////////// |  |

**QuickSort**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Fn Calls** | **Swaps** | **Time (sec.)** | **Time as F(N)** |
| N = 25,000 | 33,575 | 459,222 | 0.02 | /////////////////////// |
| N = 50,000 | 68,939 | 867,282 | 0.02 | /////////////////////// |
| N = 125,000 | 193,039 | 2,433,996 | 0.04 | /////////////////////// |
| N = 250,000 | 434,859 | 4,952,602 | 0.06 | /////////////////////// |
| N = 500,000 | 934,471 | 10,409,484 | 0.09 | /////////////////////// |
| N = 1,000,000 | 1,934,469 | 19,063,350 | 0.15 | /////////////////////// |
| N = 2,000,000 | 3,934,469 | 38,235,430 | 0.26 | /////////////////////// |
| N = 4,000,000 | 7,934,469 | 67,673,096 | 0.75 | /////////////////////// |
| N = 8,000,000 | 15,934,469 | 152,391,272 | 1.60 | /////////////////////// |
| N = 16,000,000 | 31,934,469 | 282,480,992 | 3.62 | /////////////////////// |
| N = 32,000,000 | 63,934,469 | 618,188,642 | 11.13 | /////////////////////// |
| N = 64,000,000 | 127,934,469 | 1,474,882,574 | 55.95 | /////////////////////// |
| **N** | /////////////////////// | /////////////////////// | /////////////////////// |  |

If you have time, take a look at **Sorting Algorithms Visually Shown via folk dances**

http://www.youtube.com/results?search\_query=sorting+folk+dances&aq=f



