**Lab 9 Sorting**

Name: \_\_\_\_SARUN NUNTAVIRIYAKUL\_\_\_\_\_\_\_

ID: 6031851521

No coding in this lab! :-D

The objective of this lab is to analyze time complexity of sorting algorithms by experiments.

The provided program contains all sorting algorithms. The code in the program may look a little different from the code in lecture slides but they uses the same principles.

To help you understand how each algorithm works:

* Watch this video <http://img-9gag-fun.9cache.com/photo/aPyoG4P_460sv_v1.mp4>

Follow the instructions and answer question 5-8:

1. Select an algorithm from the list in line 34-40 of the given code by putting // in front of other algorithms. The algorithms to be used in this lab are bubble sort, selection sort, insertion sort, merge sort and quicksort.
2. Select one of the for loops in line 15 and 16 based on the selected algorithm.
3. Run the Sorting program. Do not run other applications while the Sorting program is running.
4. The program will create an array of size n, populate the array with data, sort the array, check the results, and print out the execution time for sorting. It will vary the size of the array and also vary the initial order of data (sorted, random, and reversed order).
5. For each algorithm and each initial order, create a line graph between data size and execution time. There will be 15 lines in total but you can put the graphs of the same algorithm in one plot. You can copy the output from Eclipse into the Excel file to create graphs.

**Bubble**

**Selection**

**Insertion**

Merge

Quick

1. Based on the experimental result, determine the time complexity of each algorithm in terms of Big O and fill in the table.

**Time Complexity**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ordered | Random | Reverse |
| Bubble Sort | N | N^2 | N^2 |
| Selection Sort | N | N^2 | N^2 |
| Insertion Sort | 1 | N^2 | N^2 |
| Merge Sort | n log(n) | n log(n) | n log(n) |
| Quicksort | N log (n) | N^2 | N log (N) |

1. Which algorithm in each group is the fastest? What is the reason?

7.1) Bubble, Selection, Insertion

Insertion sort because if the data is sorted the insertion sort move through the array without changing any data in the array.

7.2) Merge, Quick

Quicksort is the fastest but the outcome will be different because the pivot we choose. And because we arrange the data in a way that it will be already sorted in the sub array.

1. For each algorithm, how is it sensitive to the initial order of data? (Does it run much faster or slower when the data is initially sorted, random, or reversed?) Why?

Bubble Sort:

Sorted data is fastest because the program have to only compare the value doesn’t have to swap any value.

Random it take the longest time because it will have to compare and swap data which may be very far from each other .

Reverse take faster time than random because the similar value data is close together .

Selection Sort: Sorted data is fastest because we don’t have to swap any value. For both Random and Reversed it take about the same amount of time because we have to search all the value and swap it. Which take the same amount of time.

Insertion Sort: Sorted data is fastest . For Reverse take the longest time because we have to swap every value in the array.

Merge Sort: Sorted is fastest because we have to only split the data and combine . But for Random and Reverse take longer time because we have to rearrange the data then combine.

Quicksort: Sorted and Reverse take about the same amount of time. Random is slowest because we have to choose the right position for the pivot.

1. Submit this file. Name it YourID\_Lab08\_Sorting, where YourID is your student ID.

NOTE: A program may take a long time to run!!!