**Lab 9 Sorting**

Name: \_\_Witchayut Thongyoi\_\_\_\_\_\_\_\_

ID: \_\_\_6238197821\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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.

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 |  |  |  |
| Selection Sort |  |  |  |
| Insertion Sort |  |  |  |
| Merge Sort |  |  |  |
| Quicksort |  |  |  |

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

7.1) Bubble, Selection, Insertion

Insertion sort is the fastest because it takes only O(1) for the ordered dataset. O(1) is the fastest compare to other algorithms.

7.2) Merge, Quick

Quick sort is faster because it takes less time than merge sort in sorted and reversed dataset.

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**: Fastest when the data is already sorted, because it doesn’t need to swap every element in the array. Reversed data is come up with 2nd place by predicting the outcome of this comparison in advance, your processor can use its pipeline to process more iterations of the loop simultaneously, and finishes sorting earlier than it would have done with randomly ordered data.

**Selection Sort**: All of dataset require checking every element, so the time complexity is very similar.

**Insertion Sort**: Reversed dataset is the slowest because it has to insert every data. Random data is second fastest because some element is already in order and sorted data is the fastest as it doesn’t need to insert any data.

**Merge Sort**: Random dataset performs the worst. Reversed data is second fastest because of the same reason with bubble sort(pipeline and branch prediction) and sorted data is the fastest because it doesn’t need to swap the data.

**Quicksort**: Sorted and reversed dataset time is very close because they don’t have to swap the data and random data has to swap so it’s the slowest.

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!!!