Lab 9 Sorting

Name: ____

ID: ____

No coding in this lab! :-D

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

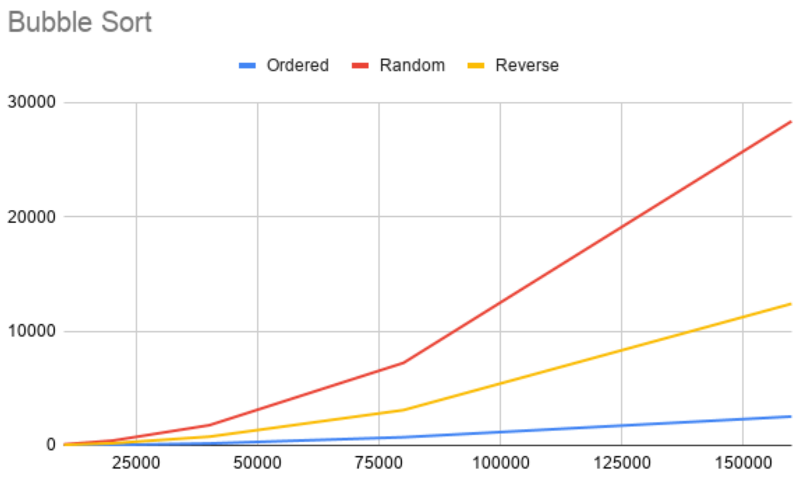
The provided program contains all sorting algorithms. The code in the program may look a little different from the code in lecture slides but they uses the same principles.

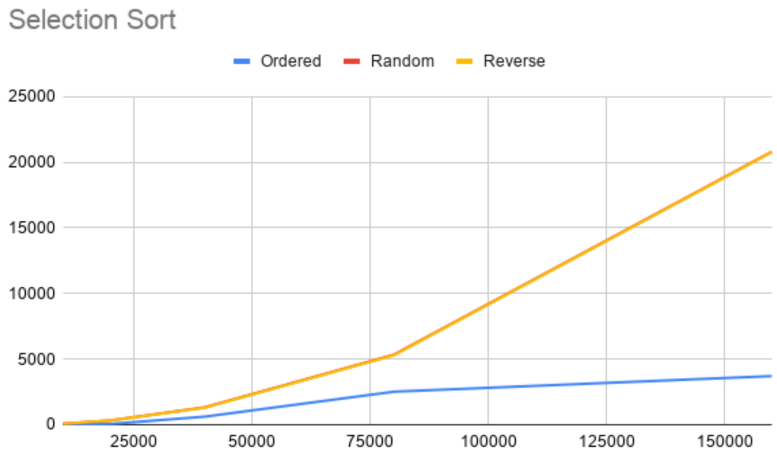
To help you understand how each algorithm works:

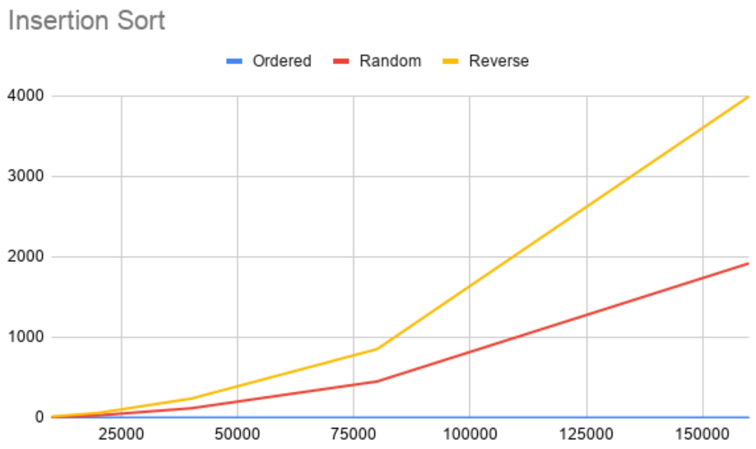
- Watch this video http://img-9gag-fun.9cache.com/photo/aPyoG4P_460sv_v1.mp4

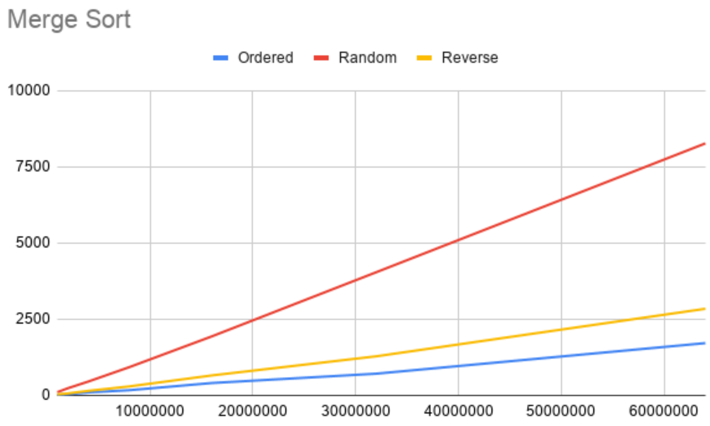
Follow the instructions and answer question 5-8:

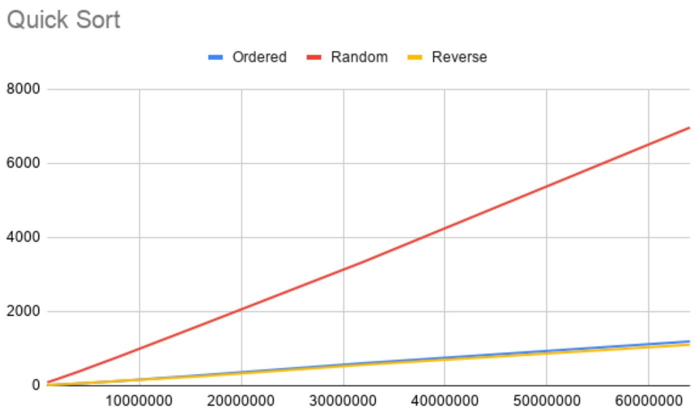
1. Select an algorithm from the list in line 34-40 of the given code by putting // in front of other algorithms. The algorithms to be used in this lab are bubble sort, selection sort, insertion sort, merge sort and quicksort.
2. Select one of the for loops in line 15 and 16 based on the selected algorithm.
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 Sort


Selection Sort


Insertion Sort





6. 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	O(n^2)	O(n^2)	O(n^2)
Selection Sort	O(n^2)	O(n^2)	O(n^2)
Insertion Sort	O(1)	O(n^2)	O(n^2)
Merge Sort	O(n)	O(nlogn)	O(n)
Quicksort	O(n)	O(nlogn)	O(n)

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

   7.1) Bubble, Selection, Insertion

   Insertion sort is the fastest, because the ordered data only uses O(1). Overall, the time complexity for insertion sort to complete the task is the least compare to the other sort in this group.

   7.2) Merge, Quick

   Quick sort is faster, because in all cases the time taken is all lesser than the time used by merge sort.

8. 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:

   Bubble sort runs fastest when the data is ordered, because the data would not need to be bubble swap at all. Reversed data is the second fastest, because branch prediction aid in processing reversed data. Ordered data is the fastest as it does not go through the swapping process.

   Selection Sort:

   Random and reversed data performed similarly as both require running through every data. However, the ordered value is fastest as the smallest value is already the first data.

   Insertion Sort:

   Reversed data performed the worst as it has to individually insert every data. Random data do not need to be inserted as several might be already in order. Ordered data is the fastest as it does not need to insert any data.

   Merge Sort:

   Random data performed the worst as expected. Reversed data perform faster as the branch prediction aids the process. Ordered data is the fastest, because it does not need to swap the data.

   Quicksort:

   Ordered and reversed data performed similarly fast. The ordered data do not need to be swapped, so it takes a small amount of time. Reversed data performed similar, because of

branch prediction and pipeline. Random data performed as expected to the normal runtime.

9. Submit this file. Name it YourID_Lab08_Sorting, where YourID is your student ID.

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