Competency#10:Sorting

Sorting	**Mastered** 1 point	**Progressing** 0.001 points	**Novice** 0 points	**Criterion Score**
Can explain time efficiency of various sort algorithms				/ 1
Can implement and analyze quicksort				/ 1
Can implement and analyze merge sort				/ 1
Can implement and analyze heap sort				/ 1
Can implement and analyze shell sort				/ 1
Can implement and analyze radix sort				/ 1

Sort

Goal: Compare and analyze different sort algorithms

Problem Description

In this assignment you will explore sort algorithms and their time efficiency using research and experimentation.
Here are some suggested sites for research:

https://visualgo.net/en/sorting
https://www.geeksforgeeks.org/

For experimentation, download the Java code and Excel spreadsheet from D2L.

Merge Sort

Answer the following questions:

1. Research Merge Sort. In your own words, how does it work? What is the runtime (big O) of the algorithm?

- works by the divide & conquer concept. First, it divides the problem into smaller problems, second, each smaller problem conquers into small problem solutions, third, solve the smaller problems as base cases, fourth, combined all the solutions for the smaller problems to find the solution for original problem. Merge sort continuous to cut the list into smaller-lists until each item is individual, then merge those individuals into sorted list.
- O(N log N) runtime

2. Run the provided code 10 times, recording the responses in the Excel Spreadsheet. Looking at the graph, how do the experimental results compare to the big O? Explain.
   - the graph reflects accurately, the graph have seconds as time measurement while the experimentation have milliseconds as time measurement. The graph shows the time as nearly to the 0 number line, while in milliseconds from the experimentation, its reflection is correct
   - However, the time doesn't always increase with increased size. Based on the experimentation, some of the larger sizes have smaller times than smaller sizes. The graph should reflect the ups and downs of the time as the sizes change up. Graph would be more visually different if the time column is measured in milliseconds

3. Copy the timesort method and adapt it to create a sorted array. Run the algorithm. Do you see a difference in the runtime when it starts sorted?

- yes, there is a difference

4. Do you have any other observations?

- merge sort works best for larger arrays

Quick Sort

Answer the following questions:

1. Research Quick Sort. In your own words, how does it work? What is the runtime (big O) of the algorithm?

- the fastest sorting method. It works by breaking an array into smaller individuals and swap with individuals, with the comparison based on the picked pivot. It can also be explained as a sorting technique that divides larger data array into smaller ones.
- First, set an index value of the array as pivot. Second, partition the array according to the pivot. Third, recursively sort of the left partition. Fourth, recursively sort to the correct partition
- O(N log N) runtime

2. Run the provided code 10 times, recording the responses in the Excel Spreadsheet. Looking at the graph, how do the experimental results compare to the big O? Explain.
   - the graph reflects accurately, the graph have seconds as time measurement while the experimentation have milliseconds as time measurement. The graph shows the time as nearly to the 0 number line, while in milliseconds from the experimentation, its reflection is correct
   - if the time in the graph is measured by milliseconds, the graph would have a high->low->high line. As low and high sizes have higher times, but with median sizes have lowest times.The graph should reflect the ups and downs of the time as the sizes change up. Graph would be more visually different if the time column is measured in milliseconds

3. Copy the timesort method and adapt it to create a sorted array. Run the algorithm. Do you see a difference in the runtime when it starts sorted?

- yes, there is a difference

4. Do you have any other observations?

- its lower

Shell Sort

Answer the following questions:

1. Research Shell Sort. In your own words, how does it work? What is the runtime (big O) of the algorithm?

- based on insertion sort algorithm. Metro begins by sorting elements distanced from each other, then narrow the gaps between the compared elements. It first sorts them and then sorts the less wide spaced elements. This method allows exchange of far items, making the array to be #-sorted for the large value of #, then reduce the # value until it hits 1
- O(N log^2 N) runtime

2. Run the provided code 10 times, recording the responses in the Excel Spreadsheet. Looking at the graph, how do the experimental results compare to the big O? Explain.

- the graph reflects accurately, the graph have seconds as time measurement while the experimentation have milliseconds as time measurement. The graph shows the time as nearly to the 0 number line, while in milliseconds from the experimentation, its reflection is correct
 - if the time in the graph is measured by milliseconds, the graph would have a high->low->increase in time line, it would be a visual “V” form. As based on the experimentation, the initial size results in a high number, but as the size increases, the time decreases for second size. Then as for third size, as the size grows, the time also grow consistently.The graph should reflect the ups and downs of the time as the sizes change up. Graph would be more visually different if the time column is measured in milliseconds

3. Copy the timesort method and adapt it to create a sorted array. Run the algorithm. Do you see a difference in the runtime when it starts sorted?

- yes, there is a difference

4. Do you have any other observations?

- faster