**Instructions**: To complete this lab, you will be required to apply your knowledge of various sorting algorithms and compare them through testing and inspection. In additional you will need to read from and write to files.

**Goal:**

The goal for this lab is to explore a variety of different sorting algorithms. These algorithms will be the three that we cover in class: Insertion, Selection and Bubble Sorts, plus **one more of your choosing.**

For each sorting algorithm you will be comparing algorithmic efficient through 3 different metrics: iterations taken, time taken and effort to implement (based on your subjective experience).

These comparisons will take place by using the algorithms to sort varying text files of data of a variety of lengths and formats (see the table in the **Test** **Data** section below). Your goal will be to determine the answer to a variety of inquiry questions listed in the **Analysis Questions** section**.**

**Additional Challenge**

Add another data order type: Clumped (where the full list of data is made of small, presorted sections of data: eg. 567123789)

Add another extra sorting algorithm **(or try and design your own!)**

**Requirements**

1. You will need to test each of the 3 sorting algorithms plus one of your choices (that you find online) (see the **Test Data** section) using data generated by the provided methods (stored in text files).
2. Your results should be recorded to a text file (**time taken** and **iterations taken**) for each test (each data order, each file size, each sort) in an understandable and readable format.
3. You need to write the code for all of your sorts (other than insertion).
4. You need to answer each of the questions found in the **Analysis Questions** Section.
5. You will need to submit the all the code you wrote (as per usual), the text file containing your results, your answers to the analysis questions, and your success criteria reflection.

**Test Data**

The algorithms will each be tested on text files of varying sizes and orders of data. Below is a table of the test data that you will test each algorithm on with. Please use the provided methods (see D2L) to generate the text files, so that when I test your programs our source files will match up. **Note that there may be duplicate elements present. At some point your sorts may take to long to complete. Feel free to place a reasonable limit on execution time (just note what happened).**

|  |  |  |
| --- | --- | --- |
| **Ascending Order** | **Descending Order** | **Random Order** |
| 10 | 10 | 10 |
| 100 | 100 | 100 |
| 1000 | 1000 | 1000 |
| 10 000 | 10 000 | 10 000 |
| 1 000 000 | 1 000 000 | 1 000 000 |
| 10 000 000 (may be too large for some sorts) | 10 000 000 (may be too large for some sorts) | 10 000 000 (may be too large for some sorts) |

**Analysis Questions**

**Answer these questions for each sorting algorithm:**

1. What was the algorithms best case (in terms of data order)?   
   (**eg.** The Bogo sort’s best case is Ascending Order).

Insertion: Ascending

Selection: Same for all

Bubble: Ascending

Quick: Descending

1. What was the algorithms worst case (in terms of data order)?   
   (**eg**. The Bogo sort’s worst case is every other case).

Insertion: Descending

Selection: Same for all

Bubble: Descending

Quick: Ascending

1. What was the algorithms average case (in terms of data order)?

Insertion: Random

Selection: Same for all

Bubble: Random

Quick: Random

1. What is the big O efficiency of each algorithm? (**Note:** You may be able to calculate this, check our class notes, or do online research)   
   (**eg.** Bogo sort is O(unbounded))

Insertion: O(n^2)

Selection: O(n^2)

Bubble: O(n^2)

Quick: O(n^2) – average: O(n log (n))

1. Explain the “use case” situations for use each algorithm  
   (**eg**. I might use a Bogo sort if I didn’t care how long it took to sort my data, and I wanted to entertain the user with a display of repeated failure)

You would use insertion sort if the array were already sorted or is close to sorted.

You would use selection sort if you wanted a consistent sorting algorithm and don’t care about sacrificing time.

You would use bubble sort if the array were sorted or close to sorted.

You would use quick sort for most cases especially with larger data sets

1. For each algorithm, what are the Pre and Post conditions?

|  |  |
| --- | --- |
| Pre-Conditions | Post-Conditions |
| There is an array of ints  The program needs to be run on a java machine  The array isn’t too large | Preconditions were true  Array will be sorted  An amount of time will pass |

**Answer the questions below just once.**

1. Which algorithm did you find most efficient in terms of difficulty to implement (ie. Which algorithm was the most “worth it” based on the effort and results)? Why was this the case?

The insertion sort algorithm is the most effective for the difficulty of setup because it only took 2 loops an few variable and is very easy to understand and setup quicks. However for larger datasets quick sort is the best because even though it took longer to understand now that I do understand it, it is very easy to set up because it just takes 3 loops and a handful of variables and because it is much more efficient than the other algorithms it is worth it for the extra setup time.

1. For each data order type, what algorithm(s) was the fastest in run time and iterations?

Random: Quick sort

Ascending: Insertion sort and bubble sort took same time

Descending: Quick sort

1. For each data order type, what algorithm(s) was the slowest in run time and iterations?

Random: Bubble sort

Ascending: Quick sort and selection sort took same time

Descending: Bubble

**Marking Scheme**

|  |  |  |
| --- | --- | --- |
| **Success Criteria:**  You show effective, useful, appropriate and thoughtful use of this skill or concept | **Marks**  **(KCAT)** | **Checklist** |
| I have successfully created and used an insertion sorting algorithm. | /5  **Knowledge** | 5/5 |
| I have successfully created and used a selection sorting algorithm. | /5  **Knowledge** | 5/5 |
| I have successfully created and used a bubble sorting algorithm. | /5  **Knowledge** | 5/5 |
| I have successfully created and used another sorting algorithm (based on my own design or research) | /5  **Thinking** | 5/5 |
| I have successfully answered all of the analysis questions. | /9  **Thinking** | 9/9 |
| I have successfully completed and recorded the results (time and iterations taken) for all tests (ascending, descending and random for each file size) | /5  **Application** | 5/5 |
| I have followed our style guide conventions. | /5  **Communication** | 5/5 |
| I have successfully met the requirements, and my program has no bug, and is fully error trapped (if needed). | /5  **Application** | 5/5 |
| I have reflected on this success criteria checklist. I.e., Read this checklist, checked it off and made sure that my program meets the requirements. | /4  **Thinking** | 4/4 |
| **Total** | /45 | 45/45 |