Description

The objective of this programming lab is to test the performance of **six** different sorting algorithms (five of them discussed in class) by sorting a *big file of data*.

The purpose is to count the total time that each algorithm took to sort a dictionary. Write a class called `Sorting.java`, where will process a file containing a dictionary. The file:

dictionary.txt

That is located at the website, contains **641,392** words in English and Spanish. In the file `Sorting.java` you will process the time elapsed that each algorithm took to sort in increasing order the file `dictionary.txt`.

Use the following sorting methods:

- bubble,
- selection,
- insertion,
- merge,
- quick, and
- a one algorithm selected of your preference [1]

Test Cases (25 pts.)

Sort the file with the six different algorithms and check the performance. You can use the

System.nanoTime()

to calculate the execution time for each algorithm. In similar fashion, test also in nanoseconds sequential search and binary search for the following cases:

1. yellow-earth
2. AMARyYO
3. amarillo
4. yellow

[1] See video: 15 Sorting Algorithms in 6 Minutes and select one of your preference

Implementation and Test Cases (25 pts.)

The `SortAlgorithms.java` class

Create a file called `Sorting.java`, where will contain the six algorithms. Notice that the Tester program will call these static methods. Each method will only take **one** parameter, i.e., an array of `words`.

The implementation of the first five algorithms are in your book. Feel free to use them. The *merge* sort and the one of your preference are **not** in your book, however, you can use well known resources such as books in computer programming or the Wikipedia to reference your work. Whatever resource you wish to use, make sure you cite your references properly.

The `Tester.java` class

Implement a class named `Tester.java` that will test the performance of the six algorithms implemented in `Sorting.java`. In this class, you will

- read the file `dictionary.txt`
- process each line containing a word
- store each `word` in an array of `String`s
- sort the array of `String`s with the 6 different algorithms and check the performance.

Be aware that the array used by the first algorithm will be *sorted* after the method invocation. Therefore, it is recommended that you *clone* the array six times, and test each sorting algorithm with a different copy of the array.

Report (75 pts.)

Write a report using the IEEE format sample:

- see: IEEE Format (doc)
- see: IEEE Format (LaTeX)

explaining the performance of each sorting algorithm and the experiments you performed. You can include graphs showing the performance between the algorithms. In the report, show the Big–$\mathcal{O}$ notation for each algorithm. Explain the why the algorithm owns that complexity. You can also include the run time analysis for each sorting algorithm. NOTE: Notice that most of the points are in the report, therefore I expect a clear, concise and well structured report. I will check the following for the report:

1. **Introduction.** Here goes the description of the problem you are solving. What is the problem given to you? What is the challenge? What is the task you need to perform.

2. **The problem and our solving approach.** Here goes the description of the way you plan to address this problem: basically an informal description, along with your algorithms. You have to explain (justify) why your approach actually solves the given problem.

3. **Sixth Algorithm Selection.** Provide a description about the sixth algorithm you selected to test. For this algorithm, provide:

   (a) Description about the algorithm: What is the input, output, and main features of the algorithm

   (b) Historic details, i.e., who invented, what year was invented, when was created, under what circumstances was invented (make sure you cite your references properly.

   (c) Complexity: Provide the $\mathcal{O}(n)$, $\Theta(n)$, and the $\Omega(n)$

4. **Performance of our approach.** Here goes the theoretical study of your algorithm, i.e., run time analysis and description of the algorithm complexity (i.e., Big–$\mathcal{O}$)

5. **Testing strategy.** Describe the test cases that you include for testing your five different algorithms. Enumerate your test cases, e.g., unordered data, sorted data, duplicates, reverse order data.

6. **Experiments and Results.** Provide the experiments and the results of each experiment. You need to be as detail as possible. Provide tables, graphs, or any kind of ideas to demonstrate the work you did. For this specific lab, you can include the theoretical results (i.e., the run time analysis) and the experimental results (the actual execution time).

7. **Conclusion.** Summarize your work done in the report, recap everything you learned and justify the results of your experiments.

8. **Implementation Annexes** In this section you must include your code for your algorithms and your programs including the test cases.