Objectives of this assignment:

* To compare the running time performance of NaïveSort and MergeSort. *NaïveSort* is the naïve sorting algorithm studied and analyzed in the lectures. Ask the instructor if you have any doubt.
* USE THIS FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **DO NOT DELETE ANYTHING FROM THIS FILE:** JUST **INSERT** YOUR ANSWERS.
* IF USING HAND WRITING (STRONGLY DISCOURAGED), **USE THIS FILE** BY CREATING SUFFICIENT SPACE AND WRITE IN YOUR ANSWERS.
* FAILING TO FOLLOW TURN IN DIRECTIONS /GUIDELINES WILL COST **A 30% PENALTY.**

What you need to do (*Insert in this file your answers*):

1. (20 points) Implement the *NaïveSort* and *MergeSort* algorithms to sort an array.

*submit your source code in your preferred language (as long as it is supported on Tux machines).*

*Provide here the instructions to compile and execute your code*

**Navigate to the directory (I was on bmm0066@tux251) The file is just in that home directory.**

**programming\_5.java should be there**

**type in “javac programming\_5.java”**

**After compilation finishes, type in “java programming\_5”**

**The program should run and it will output to the system how many iterations are done. It runs till 150000. Because the naïve sorting algorithm is so slow, this took about five minutes to execute on my machine.**

**Once done, there should be a file called “Data5.csv” with all the output data.**

1. ( 5 points) Collect the execution time T(n) as a function of n for the two algorithms

*No need to submit a table of the values you collect. Just state here if you collected data and submit a cvs file containing the values n, TNaive(n), and TMerge(n) where TNaive(n), and TMerge(n) are respectively the execution time of* NaïveSort and MergeSort for an array of size n.

**I collected and submitted a csv file containing the values of n, TNaive(n) (marked column Naïve), and TMerge(n) (marked column Merge).**

1. (20 points) Plot on the **same** graph the running time *TNaive(n), and TMerge(n)* of each algorithm.

*Insert here the plot...*

Chart, scatter chart

Description automatically generated

1. (20 points) Using a **pertinent graph/plot** with your data (**hint**: look at previous programming assignments how we can determine the shape of T(n) and its asymptotic growth), show/illustrate what the time complexity of *NaïveSort* is.

*Insert here the plot...*

Chart, scatter chart

Description automatically generated

*Discuss here the pertinent plot that determines the asymptotic growth of Naive Sort*

**Based on our in class study (slide attached below) of the naïve sorting algorithm, we know the time complexity of the naïve sort grows as n2. That means, when we collect the execution time of the naïve sorting algorithm and graph it against the values of n (for very large values of n), we expect the graph to be the same shape as a graph of y=x2. I attached below a graph of y=x2 and if we compare that image to our plot we can see that they both have the same basic shape of a curve upwards. Because of this, we can confirm that the naïve sorting algorithm has a time complexity that grows (Θ) as n2.**

Text

Description automatically generatedA picture containing diagram

Description automatically generated

1. (20 points) Using a **pertinent graph/plot** with your data (**hint**: look at previous programming assignments), show/illustrate what the time complexity of *MergeSort* is.

*Insert here the plot...*

Chart, scatter chart

Description automatically generated

*Discuss here the plot plot that determines the asymptotic growth of MergeSort*

**In class we studied the merge sort algorithm and learned that no matter the case, the time complexity of merge sort grows as nlog(n). I attached the slide below. Based off that information, when we plot the execution time of the algorithm as a function of n for a very large n, the resulting plot should match the basic shape of the graph of y=xlog(x). I attached an image of y=xlog(x) below. The basic shape is a gradual curve upwards. Looking at the plot of our data, it does fit that general shape, the curve is just very gradual. Since we know our plot matches the graph of y=xlog(x), our data supports the idea that merge sort’s time complexity grows (Θ) as nlog(n).**

Table

Description automatically generated

Chart, line chart

Description automatically generated

1. (15 points) Discuss the results comparing the two algorithms.

*Discuss here the plot. Which algorithm is better?*

**Examining our plot with the data from both the naïve and the merge sort algorithm on it (from 3.), we can tell that the naïve algorithm grows to much larger values than merge sort as n increases. In fact, using a term from class, the time complexity of the naïve algorithm dwarfs that of merge sort. Based on that information and the plot, we know that the naïve algorithm takes much longer to run than the merge sort algorithm, especially so for large values of n. Therefore, the merge sort algorithm is more efficient, and better, than the naïve sorting algorithm.**

**Objective**:

The objective of this programming assignment is to implement in Java the *NaïveSort* and *MergeSort* algorithms presented in the lectures to sort a list of numbers. We are interested in comparing the two algorithms. For this exploration, you will collect the execution time T(n) as a function of n and plot on the same graph the execution times T(n) of the two algorithms. Finally, discuss your results.

**Program to implement**

collectData()

Generate an array G of **HUGE** length L (as huge as your language allows) with **random** values capped at 0x7ffffffe.

for n = 4000 to L (with step 1,000)

for each algorithm *NaïveSort* and *MergeSort do*

copy in Array A **n** first values from Array G

Start timing // We time the sorting of Array A of length n

Sort A using one of the two algorithms.

Store the value n and the value T(n) in a file **F** where T(n) is the execution time

//Think here about value(s) to collect for the **pertinent** graph/plot for questions 4, 5, and 6

**What should be L?** L must be as large as possible such that 1) you collect enough data to produce meaningful plots, 2) it would not take too much time to collect data, and 3) your machine can handle.

**Data Analysis**

Use any plotting software (e.g., Excel) to plot the values T(n) in File F as a function of n. File F is the file produced by the program you implemented. Discuss your results based on the plots.

**Report**

* Your report is this file in which you inserted your answers
* Good writing is expected.
* Recall that answers must be well written, documented, justified, and presented to get full credit.

**What you need to turn in:**

* Electronic copy of your source program (standalone) that collects data
* csv file containing n, *TNaive(n), and TMerge(n)*
* Electronic copy of the report (this file including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.

**Grading**

1. (**20 points**) Implement the *NaïveSort* and *MergeSort* algorithms to sort an array A.
2. ( **5 points**) Collect the execution time T(n) as a function of n for the two algorithms
3. (**20 points**) Plot on the same graph the running time of the two algorithms.
4. (**20 points**) Using a **pertinent graph/plot** with your data (**hint**: look at previous programming assignments), show/illustrate what the time complexity of *NaïveSort* is.
5. (**20 points**) Using a **pertinent graph/plot** with your data (**hint**: look at previous programming assignments), show/illustrate what the time complexity of *MergeSort* is.
6. (**15 points**) Discuss the results comparing the two algorithms.