AUBURN UNIVERSITY

CPSC 3273 Programming Assignment 2

Objectives of this assignment:

- to explore time complexity and "real time"

What you need to do:

1. Implement the ***Merge-Sort*** algorithm to sort an array. (See Appendix for the *Merge-Sort* algorithm)
2. Collect the execution time T(n) as a function of n
3. Plot the functions $T(n)/\log_2(n)$, $T(n)/n.\log_2(n)$, and $T(n)/n^2.\log_2(n)$ on the same graph.
4. In Module 4 (next module), we will establish that the running time T(n) of *Merge-Sort* is $\Theta(n.\log(n))$. Discuss T(n) in light of the graph you plotted above.

Objective:

The objective of this programming assignment is to design and implement in Java the Merge-Sort algorithm presented in the lecture to sort a list of numbers. We are interested in exploring the relationship between the time complexity and the "real time". For this exploration, you will collect the execution time T(n) as a function of n and plot the functions $T(n)/\log_2(n)$, $T(n)/n.\log_2(n)$, and $T(n)/n^2.\log_2(n)$ on the same graph. 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 0xffffffe.
    for n = 5,000 to L (with step 1,000)
        copy in Array A n first values from Array G

        Start timing // We time the sorting of Array A of length n
        Merge-Sort(A,0,n-1)
        Store the value n and the values $T(n)/\log_2(n)$, $T(n)/n.\log_2(n)$, and
        $T(n)/n^2.\log_2(n)$ in a file F where T(n) is the execution time
```

Data Analysis

Use any plotting software (e.g., Excel) to plot the values $T(n)/\log_2(n)$, $T(n)/n.\log_2(n)$, and $T(n)/n^2.\log_2(n)$ 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. (**Hint**: is T(n) closer to $K.\ \log_2(n)$, $K.\ n.\log_2(n)$, or $K.\ n^2.\log_2(n)$ where K is a constant?)

Report

- Write a report that will contain, explain, and discuss the plot. The report should not exceed one page.
- In addition, your report must contain the following information:
    - whether the program works or not (this must be just ONE sentence)
    - the directions to compile and execute your program
- 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
- Electronic copy of the report (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.

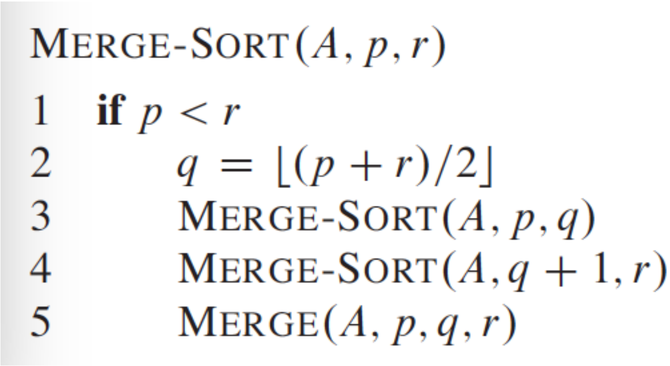
Grading

- Program is worth 30% if it works and provides data to analyze
- Quality of the report is worth 70% distributed as follows: good plot (25%), explanations of plot (10%), discussion

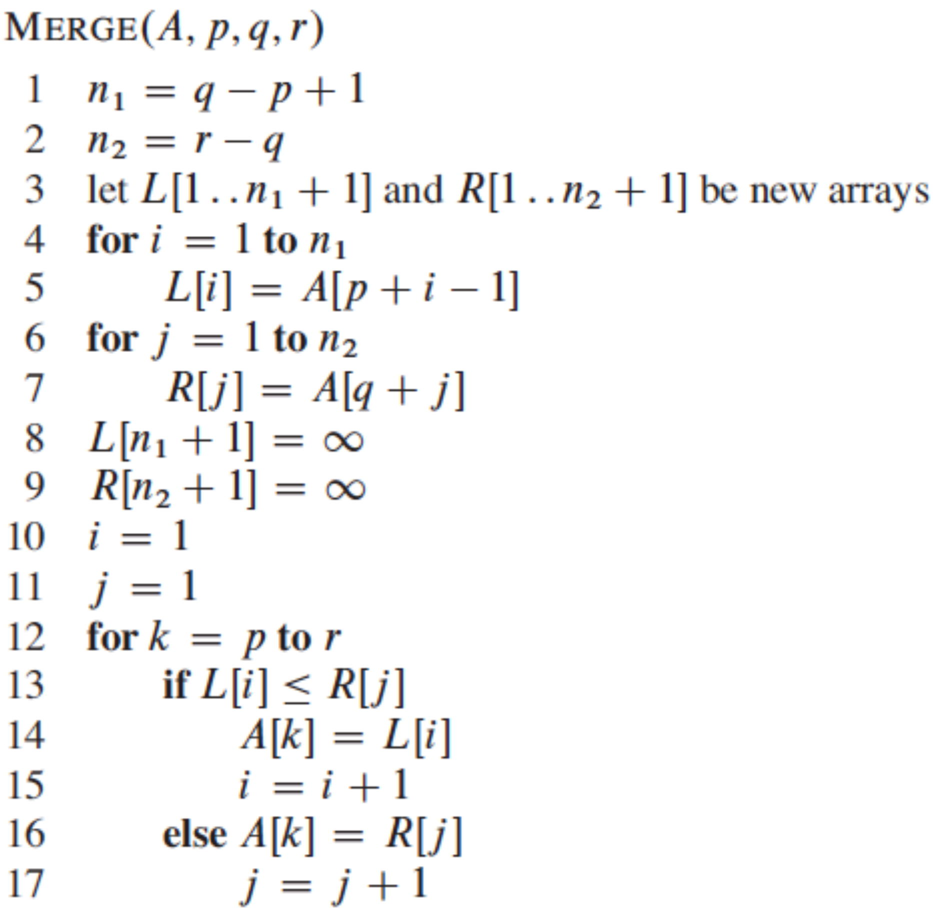
and conclusion (35%).

Appendix: Merge-Sort Algorithm.

At this stage, you do NOT need to understand Merge-Sort (It will be presented and explained in Module 4)). Implement Merge-Sort exactly the way it is described below. Replace the infinity value (∞) with 0xffffffff.



```
MERGE-SORT($A, p, r$)
if $p < r$
    $q = \lfloor (p + r)/2 \rfloor$
    MERGE-SORT($A, p, q$)
    MERGE-SORT($A, q + 1, r$)
    MERGE($A, p, q, r$)
```



```
MERGE($A, p, q, r$)
$n_1 = q - p + 1$
$n_2 = r - q$
let $L[1 \,..\, n_1 + 1]$ and $R[1 \,..\, n_2 + 1]$ be new arrays
for $i = 1$ to $n_1$
    $L[i] = A[p + i - 1]$
for $j = 1$ to $n_2$
    $R[j] = A[q + j]$
$L[n_1 + 1] = \infty$
$R[n_2 + 1] = \infty$
$i = 1$
$j = 1$
for $k = p$ to $r$
    if $L[i] \leq R[j]$
        $A[k] = L[i]$
        $i = i + 1$
    else $A[k] = R[j]$
        $j = j + 1$
```