Ashutosh Rai and Vitalii Stadnyk

CS 310 Project 1

Instruction on how to run the program:

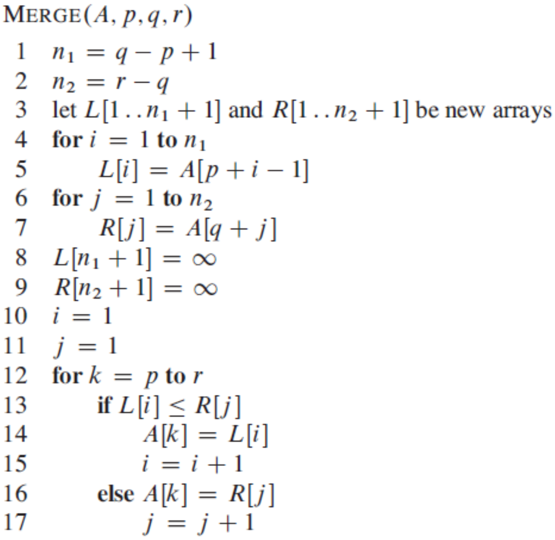
The program is pretty straight forward and also instructs on screen. It asks you for the size of the first array you want to generate and then the sorting algorithms will be implemented for those. Then the size of the array will progressively get bigger by a multiple of 10. The maximum array size is capped at 1M.

Design of the program:

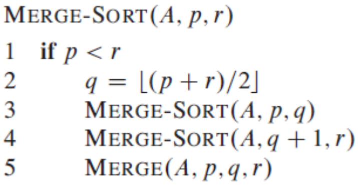
1. First ask the user for the size of the first array
2. Write a function to generate a list of random floats and output it to a text file named list.txt
3. Convert the list of numbers from list.txt to a dynamic array
4. Make copies of the original array for each algorithm separately for consistency
5. Write functions for each algorithm based on the pseudocode provided below
6. Execute all the functions for the algorithms
7. Increase the size of the array by a multiple of 10 and repeat all the above steps. The maximum size of the array has been capped at 1M

Pseudocode for the algorithms:

Merge sort:



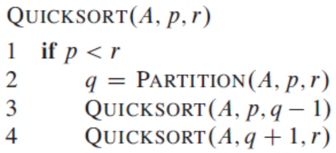
```
MERGE($A, p, q, r$)
$n_1 = q - p + 1$
$n_2 = r - q$
let $L[1 \,..\, n_1 + 1]$ and $R[1 \,..\, n_2 + 1]$ be new arrays
for $i = 1$ to $n_1$
    $L[i] = A[p + i - 1]$
for $j = 1$ to $n_2$
    $R[j] = A[q + j]$
$L[n_1 + 1] = \infty$
$R[n_2 + 1] = \infty$
$i = 1$
$j = 1$
for $k = p$ to $r$
    if $L[i]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



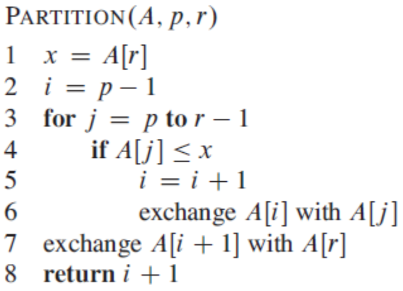
```
MERGE-SORT($A, p, r$)
if $p < r$
    $q = \lfloor (p + r)/2 \rfloor$
    MERGE-SORT($A, p, q$)
    MERGE-SORT($A, q + 1, r$)
    MERGE($A, p, q, r$)
```

Ashutosh Rai and Vitalii Stadnyk

Quick sort:

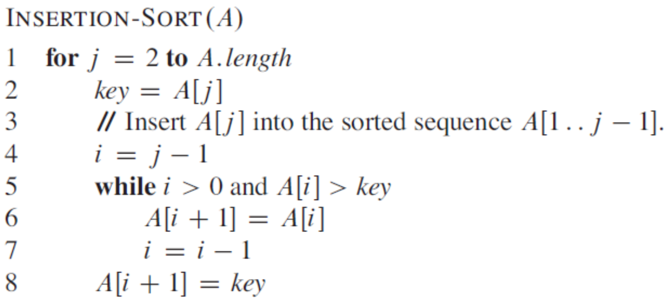


```
QUICKSORT$(A, p, r)$
if $p < r$
    $q = $ PARTITION$(A, p, r)$
    QUICKSORT$(A, p, q - 1)$
    QUICKSORT$(A, q + 1, r)$
```



```
PARTITION$(A, p, r)$
$x = A[r]$
$i = p - 1$
for $j = p$ to $r - 1$
    if $A[j] \leq x$
        $i = i + 1$
        exchange $A[i]$ with $A[j]$
exchange $A[i + 1]$ with $A[r]$
return $i + 1$
```

Insertion sort:



```
INSERTION-SORT$(A)$
for $j = 2$ to $A.length$
    $key = A[j]$
    // Insert $A[j]$ into the sorted sequence $A[1 \,.\,.\, j - 1]$.
    $i = j - 1$
    while $i > 0$ and $A[i] > key$
        $A[i + 1] = A[i]$
        $i = i - 1$
    $A[i + 1] = key$
```

Bubble sort:

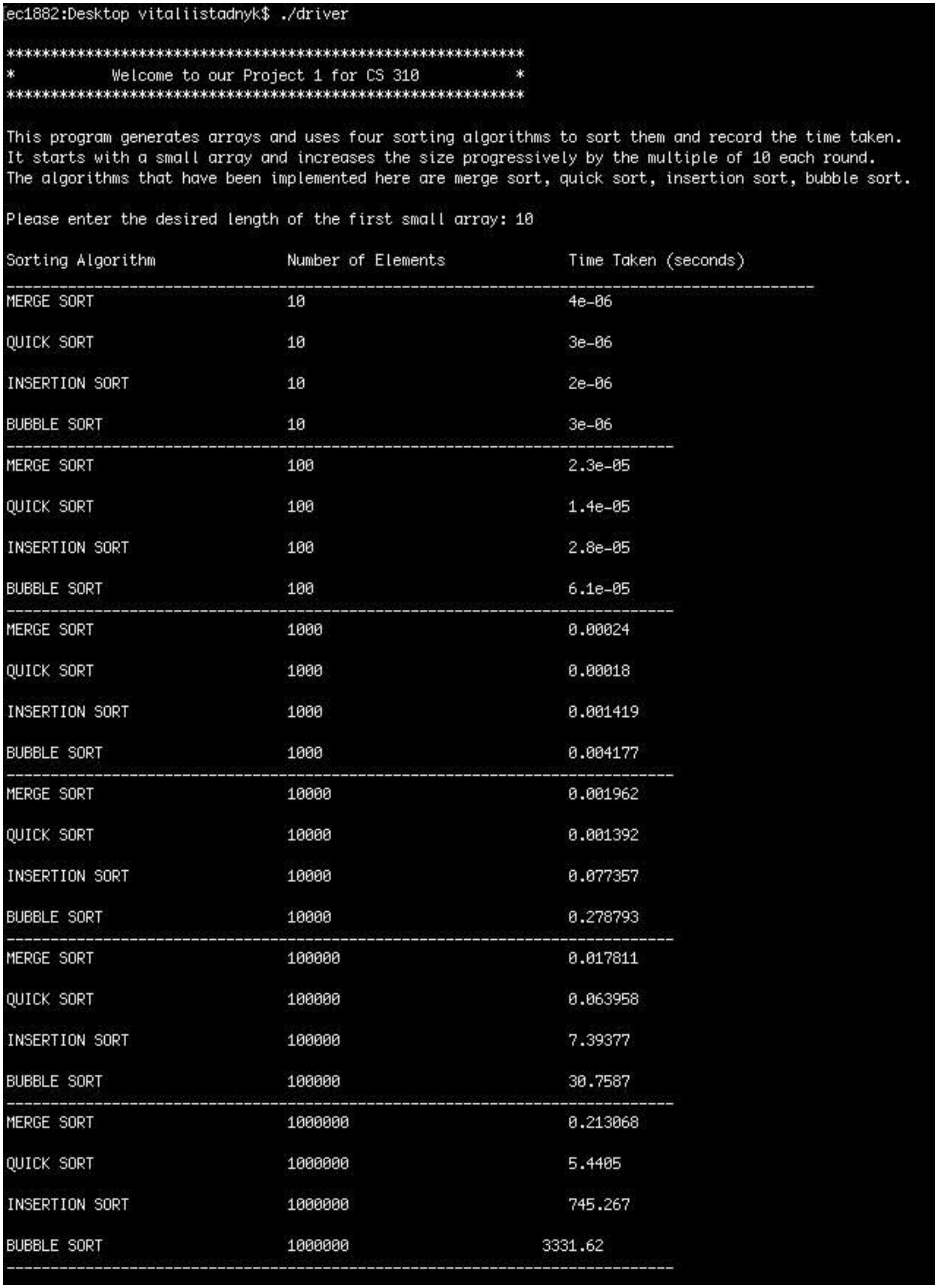
```
Bubble-Sort(A)
  for i = (A.length -1) to 1
    for n = 1 to i
        if A[n] > A[n-1]
          swap A[n] and A[n-1]
```

<u>Individual work in the project</u>

- Ash worked on generating the array (creating list.txt), insertion sort, and bubble sort.
- Vitalii worked on quicksort, output of the program and the for loop for increasing the size of array.
- Both of us worked on merge sort, keeping track of time of the processes, documentation and debugging.

Ashutosh Rai and Vitalii Stadnyk

Our running time: (Execution of the program)



```
[ec1882:Desktop vitaliistadnyk$ ./driver

***********************************************************
*           Welcome to our Project 1 for CS 310           *
***********************************************************

This program generates arrays and uses four sorting algorithms to sort them and record the time taken.
It starts with a small array and increases the size progressively by the multiple of 10 each round.
The algorithms that have been implemented here are merge sort, quick sort, insertion sort, bubble sort.

Please enter the desired length of the first small array: 10

Sorting Algorithm               Number of Elements              Time Taken (seconds)
--------------------------------------------------------------------------------------------
MERGE SORT                      10                              4e-06

QUICK SORT                      10                              3e-06

INSERTION SORT                  10                              2e-06

BUBBLE SORT                     10                              3e-06
----------------------------------------------------------------------------
MERGE SORT                      100                             2.3e-05

QUICK SORT                      100                             1.4e-05

INSERTION SORT                  100                             2.8e-05

BUBBLE SORT                     100                             6.1e-05
----------------------------------------------------------------------------
MERGE SORT                      1000                            0.00024

QUICK SORT                      1000                            0.00018

INSERTION SORT                  1000                            0.001419

BUBBLE SORT                     1000                            0.004177
----------------------------------------------------------------------------
MERGE SORT                      10000                           0.001962

QUICK SORT                      10000                           0.001392

INSERTION SORT                  10000                           0.077357

BUBBLE SORT                     10000                           0.278793
----------------------------------------------------------------------------
MERGE SORT                      100000                          0.017811

QUICK SORT                      100000                          0.063958

INSERTION SORT                  100000                          7.39377

BUBBLE SORT                     100000                          30.7587
----------------------------------------------------------------------------
MERGE SORT                      1000000                         0.213068

QUICK SORT                      1000000                         5.4405

INSERTION SORT                  1000000                         745.267

BUBBLE SORT                     1000000                      3331.62
----------------------------------------------------------------------------
```