Problem 6.1

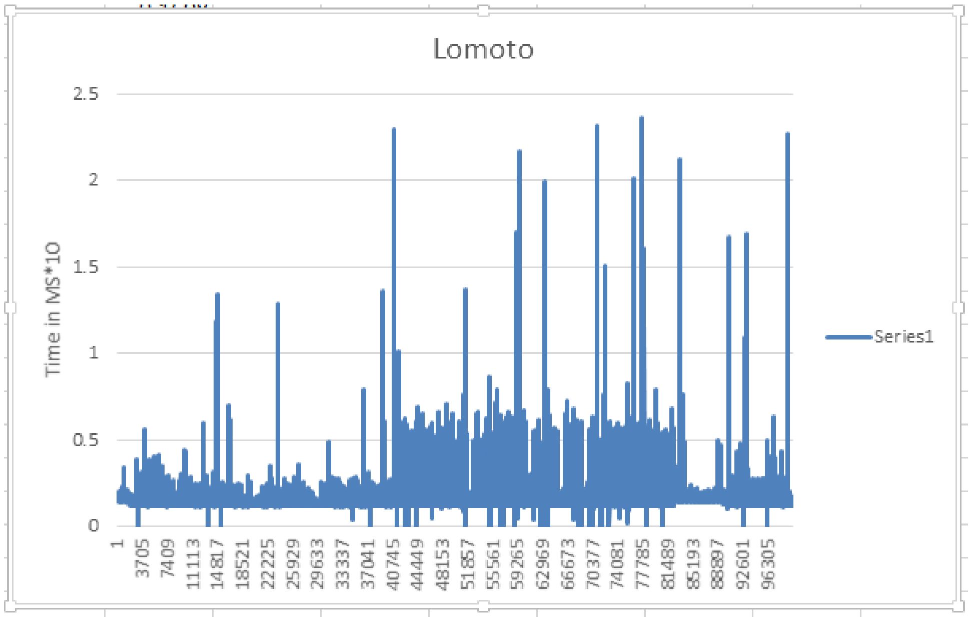
a) The implementation of the pseudocode is in LomotoTest.c.

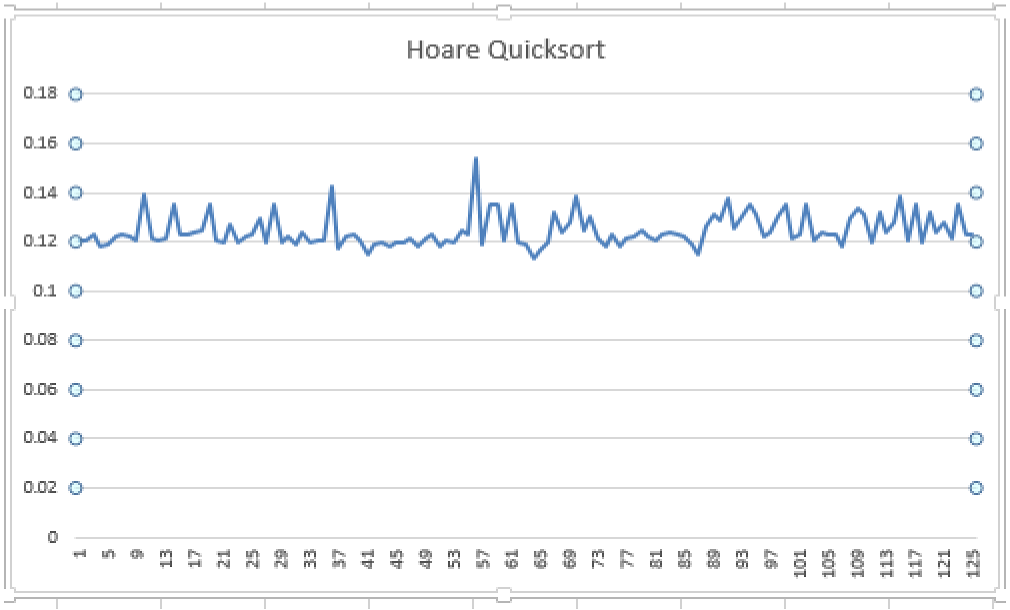
b) The implementation of the pseudocode is in HoareTest.c.

c) The implementation of the pseudocode is in MedianTest.c.

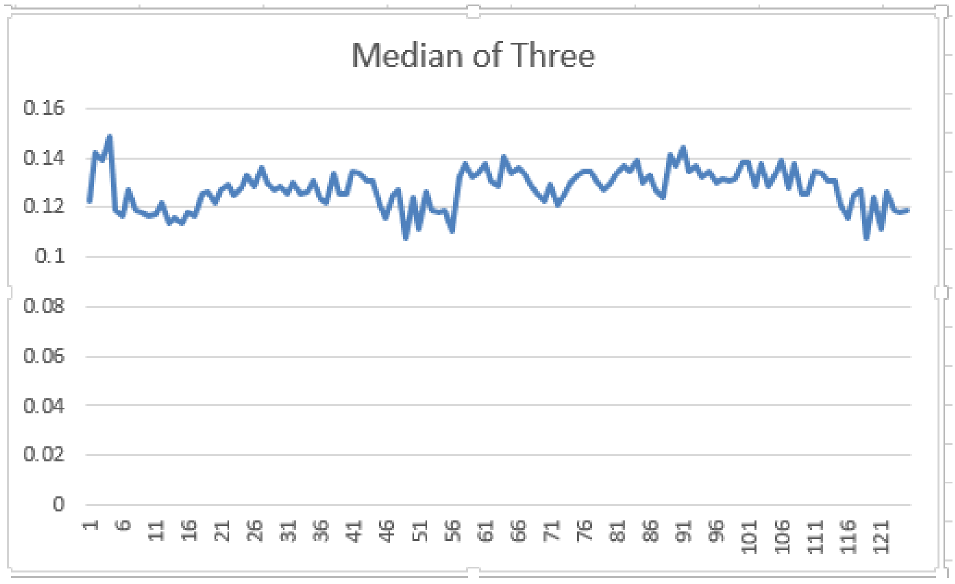
d) The implementation of the pseudocode is in LomotoData.c, HoareData.c and MedianData.c. Since my compiler cannot work properly if I take the loop from 1 to 100,000 at one-time, I made a loop with 1 to 800, and run the program for 125 times consecutively. I copied the results from each run process (1 to 800 for 125 times) and pasted to the excel sheet to calculate the average time taken. I calculated the average for each 1 to 800 process, added them together and calculated the average of 125 average results.

According to the time taken measurement for each quicksort function, Lomoto quicksort function takes around 0.144044 milliseconds in average for n=1000.





I calculated the average for each process (1 to 800), saved 125 average results in the spreadsheet, and calculated the average again. The figure above shows the average time taken for each running process.
The average time taken I got was around 0.124491 milliseconds.



The average time taken I got was around 0.127511 milliseconds.

According to these results, Lomoto method takes the longest time, and the time taken difference between the rest two methods are more closer to each other. Median of three might take less time than Hoare for different n, rather than 1000. Hoare is the most efficient method for quicksort.

Problem 6.2

a) The implementation of the pseudocode is in Threeway.c.

b) In the best case, the array will be evenly divided.). Also, the partitioning part will take O(n) times.

$T(n) = T(n\text{-}L\text{-}R\text{-}2) + T(L) + T(R) + O(n)$

where L,R will have the size of n/3. So, $T(n\text{-}L\text{-}R\text{-}2)$ will be approximately equal to n/3.

Hence, we can say that $T(n) = 3T(n/3) + O(n)$.

If we recursively divide this time complexity equation, the height of the recursion tree will be $\lg_3 n$, and the addition of each level will result n. If $c = 0.63>0$, $n \lg_3 n \leq 0.63*n \lg(n)$ if $n > n_0$. Hence, the time complexity is $T(n) = O(n \lg n)$.

In the worst case, the second element will be swapped with the rightmost element, and the first element will be remained in the same place. Also, the sorting is called recursively by dividing the elements in the middle of the array (n-2 elements). Also, the partitioning part will take O(n) times.

$T(n) = T(n\text{-}2) + T(1) + T(1) + O(n)$

The array will be divided for n/2 (changing to n-4, n-6,… in each step) times repeatedly. So, the addition of values from each level will be around $(n/2)*[(n/2)+1]$. Hence, the time complexity is $n^2/4 + n/2 + O(n)$, which will be $T(n) = O(n^2)$.

If $f(n) = n^2/4 + n/2 + O(n)$, $c = 0.5$ and $g(n) = n^2$,

Then $0 \leq f(n) \leq cg(n)$ for some $c>0$ and $n\geq n_0$. Hence, $T(n) = O(n^2)$.

c) The implementation of the pseudocode is in Random.c.

Problem 6.3

Let $f(n) = \lg(n!)$, $g(n) = n \lg(n)$, $c_1 = 1>0$

$\lg(n!) = \lg(1*2*\ldots*n) = \lg(1)+ \lg(2)+ \ldots+ \lg(n/2) + \ldots+ \lg(n)$

$n \lg(n) = \lg(n) + \lg(n) + \ldots + \lg(n)$

$c_1*g(n) = 1*n \lg(n) = n \lg(n)$

$0 \leq f(n) \leq c_1 g(n)$ if $n\geq n_0$ for some $c_1>0$ Hence, $\lg(n!) = O(n \lg n)$

$f(n) \geq \lg(\frac{n}{2})+ \lg(\frac{n}{2}+1) + \ldots +\lg(n)$

$f(n) \geq \frac{n}{2} \lg(\frac{n}{2}) = \frac{n}{2} (\lg(n)- \lg(2)) = \frac{n}{2} \lg(n) - \frac{n}{2} \lg(2)$

$f(n) \geq \frac{1}{2} *n \lg(n) - \frac{n}{2}$

So, $f(n) \geq \frac{1}{2} * n \lg(n)$ *(Ignoring n/2 constant)*

If $c_2 = 0.5$, $c_2 * g(n) = 0.5 * n \lg(n) = \frac{1}{2} * n \lg(n)$

$0 \leq c_2 g(n) \leq f(n)$ if $n \geq n_0$ for some $c_2 > 0$ Hence, $\lg(n!) = \Omega(n \lg n)$

$\frac{1}{2} * n \lg(n) \leq \lg(n!) \leq n \lg(n)$

$c_2 g(n) \leq f(n) \leq c_1 g(n)$ if $n \geq n_0$

Since $\lg(n!) = O(n \lg n) = \Omega(n \lg n)$, $\lg(n!) = \Theta(n \lg n)$ is also true.

References

Elhaiani, S. (2017, September 19). Dual pivot Quicksort. Retrieved March 14, 2019, from https://www.geeksforgeeks.org/dual-pivot-quicksort/
(I used this website to get ideas for Problem 5.2 a) and c).)

Joshi, R. (2014, June 19). Quicksort algorithm in Java – Code Example. Retrieved March 15, 2019, from https://examples.javacodegeeks.com/core-java/quicksort-algorithm-in-java-code-example/
(I used this website to get ideas for Problem 5.1 c).)

Quicksort. (2019, March 01). Retrieved March 14, 2019, from https://en.wikipedia.org/wiki/Quicksort
(I used this website to get ideas for Problem 5.1 b) and c).)