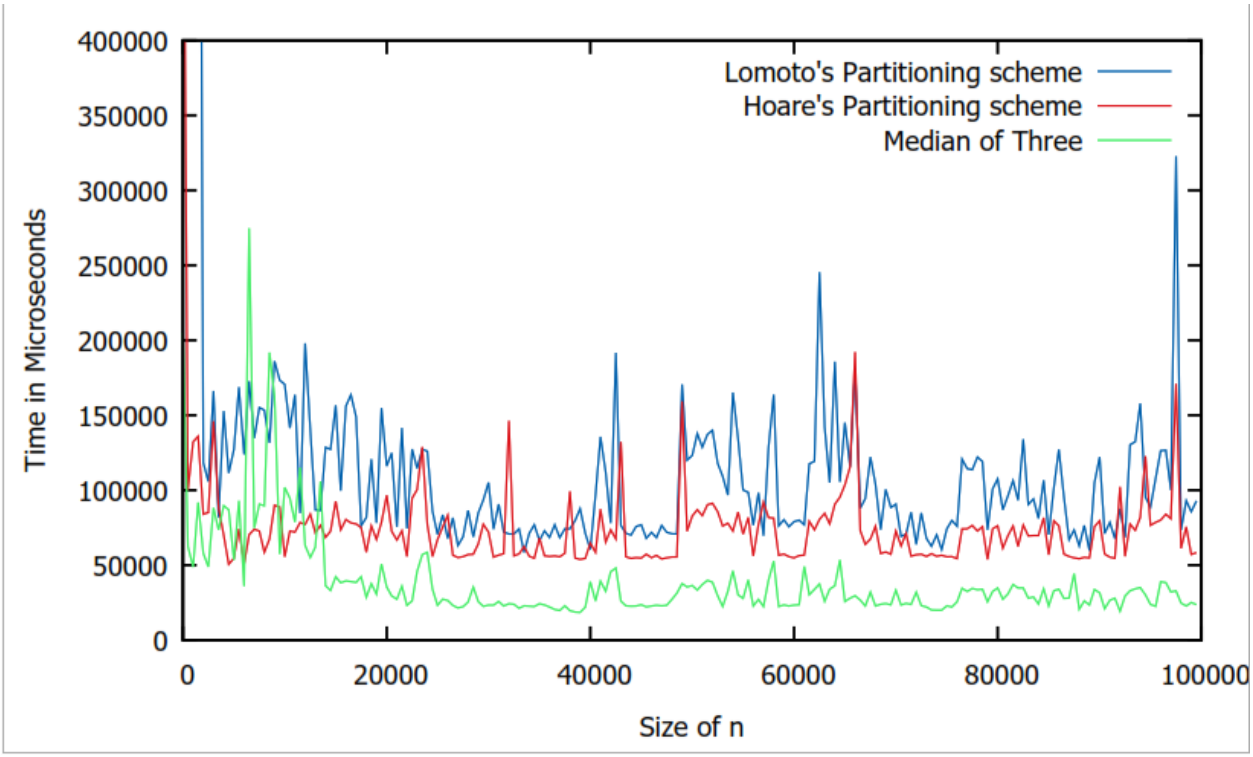
Homework 7: Algorithms and Data Structure

Blen Daniel Assefa

March 2020

Problem 7.1 Quicksort with Partition Versions

a) Implemented in Quicksort.java
b) Implemented in Quicksort.java
c) Implemented in Quicksort.java
d) Implemented in Quicksort.java:



Average Timings:
Lomotu Case: 122826
Hoare Case: 75905
Median Case: 38730

The result shows that Lomotu takes the most time. This is because of the constant for the time complexity. It has the highest as compared to Hoare's or Median partition algorithm.

Problem 7.2 Modified Quicksort

a) Implemented in ModifiedQuicksort.java
b) Best Case:
This will happen if there is an equal partition in all the recursive calls. As the partition iterates through the element only once, the complexity will be $O(n)$. Every recursive call divides the array into 3 intervals. We can write for the best case:

$T(n) = 3T(\frac{n}{3}) + O(n)$

By applying the master method we get $T(n) = O(nlg(n))$
Base of the lg is 3

Worst Case:
This will happen when the array is sorted in an ascending or descending order. If we take the first two elements as pivots, then they won't have any elements in between them so the pivots don't act as a pivot as there are no elements in between them. So if the pivot1 and pivot2 are in order, then the pivot2 in the end of the array has to come after the first element, so would come n-2 places back from the end.

$T(n) = T(n - 2) + T(1) + T(1) + O(n)$
In the end, after solving the recurrence we have:
$T(n) = O(n^2)$

c) Implemented in RandomQuickSort.java