Problem 7.1 Quicksort with Partition Versions

d) The average time of Lomuto Quicksort: 109 microseconds

The average time of Hoare Quicksort: 89 microseconds

The average time of Median of Three Quicksort: 106 microseconds

From the results we can understand that Hoare Quicksort performs better. This happens because on average Hoare partition does three times fewer swaps that Lomuto partition. However, for an already sorted array, the complexity time of both is $O(n^2)$. Median of Three Quicksort gives a better estimation of the optimal pivot than selecting any single random element. We choose for pivot the median of the first, middle and last element of the partition. So, in an already sorted array, the worst-case behavior is avoided. The expected number of comparisons needed to sort n elements in Median of Three Quicksort is 1.188 nlogn, but as we can see from the results it still performs slower than Hoare Quicksort.

Problem 7.2 Modified Quicksort

b) Time complexity

Best case – These two pivots divide the whole array into 3 equal parts. So, we have n elements and the 3 subarrays formed from the partitions have the size of n/3.

The recurrence: $T(n) = 3T(n/3) + O(n)$

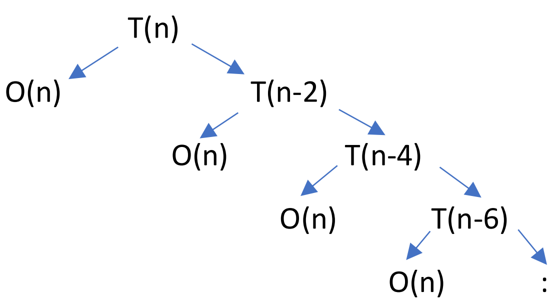
Using Master Method:

$a = 3$	$n^{\log_b a} = n^{\log_3 3} = n^1$
$b = 3$	$f(n) = O(n)$

This falls into case 2. Therefore, T(n) is **O(n logn)**.

Worst case – This case happens when, after dividing the array into 3 parts, the two pivots are positioned one in the start of the array and the other in the end of the array. So, in an array of n elements, the partition 1 and 3 have 0 elements while the partition 2 between first pivot and second pivot has n-2 elements.

The recurrence: $T(n) = T(n-2) + O(n)$ *(n is a constant so it takes linear time O(n))*



➔ 0 + 2 + 4 + … + n-2 + n = n (n+2) / 4

$T(n) = n\,(n+2) / 4 = \mathbf{O(n^2)}$

:

$$\begin{tikzcd} & T(2) \arrow[dl] \arrow[dr] & \\ O(n) & & T(0) \end{tikzcd}$$

$T(n) = O(n^2) + O(n)$

$= \mathbf{O(n^2)}$

References:

https://en.wikipedia.org/wiki/Quicksort