CSE 2046 ANALYSIS OF ALGORITHM

HOMEWORK 2

Mücahit Tanacıoğlu 150115006

Experiment Design:

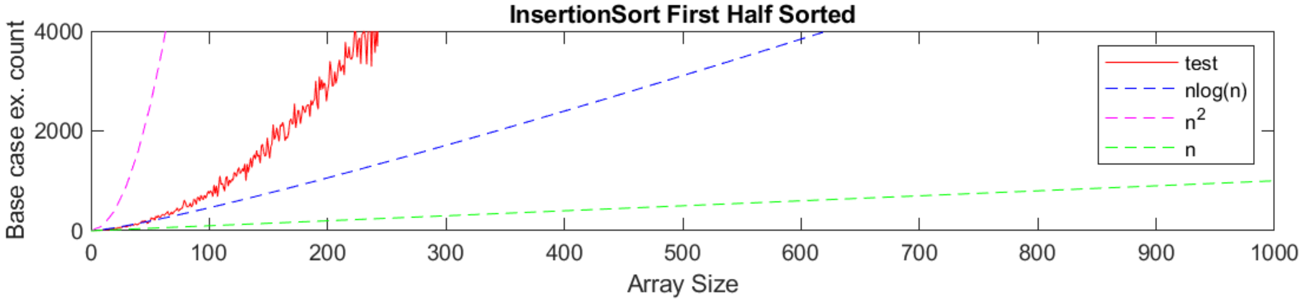
The inputs: Inputs are consist of integer arrays.
There is 5 different type of arrays :
1.) Sorted arrays
2.) Reverse Sorted arrays
3.) Random arrays
4.) First half sorted arrays(other half is random)
5.) Second half sorted arrays(other half is random)

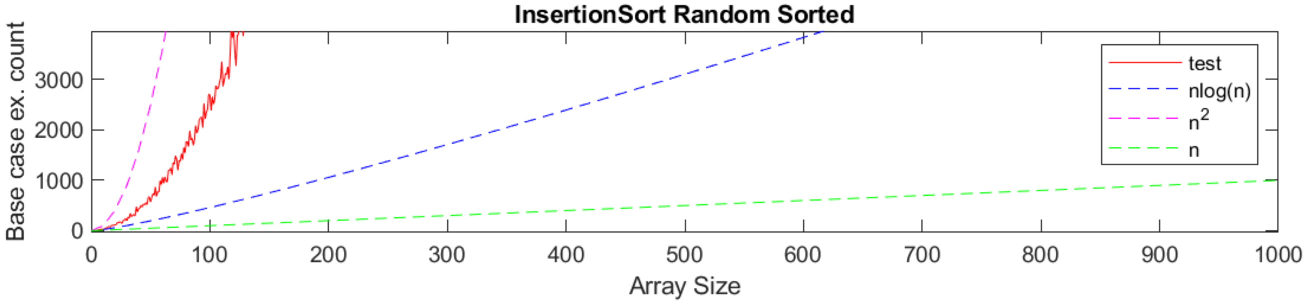
For each type I have created 1000 array and their lengths differ by 1 from 1 to 1000.
Since array get sorted after one algorithm work on it, recreated another same type array for each algorithm, the arrays are identical for sorted parts as random parts I have used java's random class and range is pretty big including negative numbers, therefore I think random parts are okay for this test, for the sorted part its simply fill array with 1 to length of array with increasing order by 1.

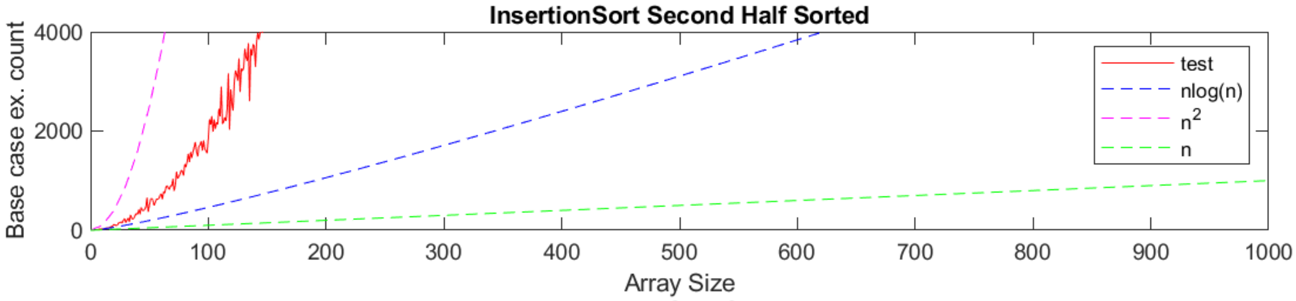
For efficiency measurement I have used base case method because measuring time between algorithm's start and end line didn't look like reliable even with doing same array with same algorithm 50 times and taking average of them.

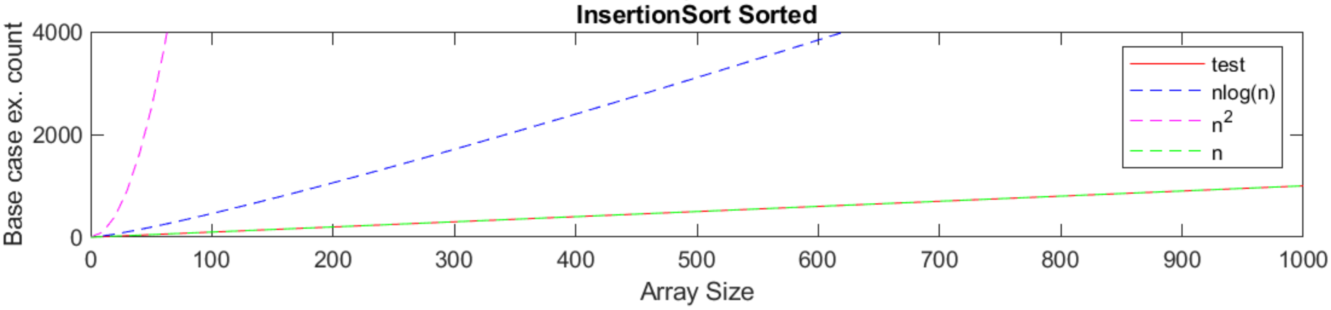
For drawing graphs I have saved data from algorithms to an array list and write them into text file with two column first column indicates number of element in array second column indicates result from algorithm , file name identify which type of array and which algorithm used(ex."InsertionSort_Sorted.txt").Finally I draw graphs with using Matlab apart from test result I have draw expected functions to see is result really fit theoretical function. I had problem with this problem despite I had 1000 different result I couldn't make it draw because for the case test linearly increase n^2 function grow so fast and we cant see where test result lies, so I have to limit y value between 0-4000 and this is effects some graphs where test result goes edge off the graph after a few point, also its effects n*log(n) vs n functions.

Insertion Sort:





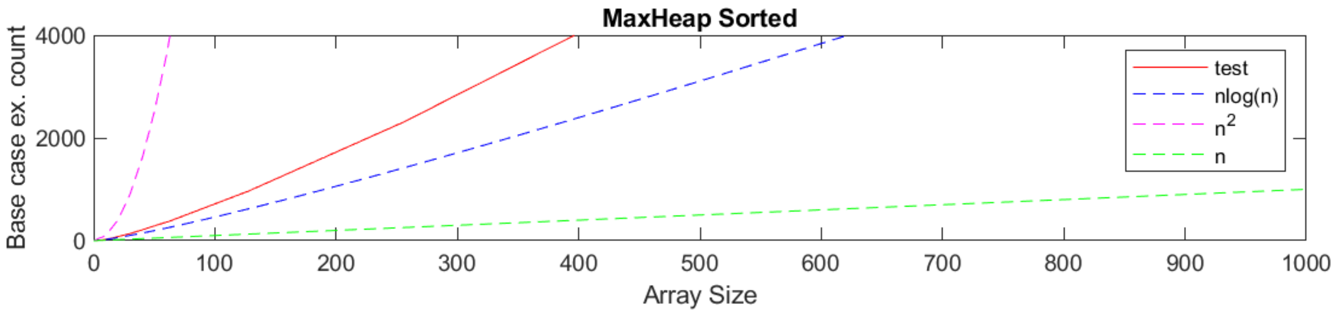


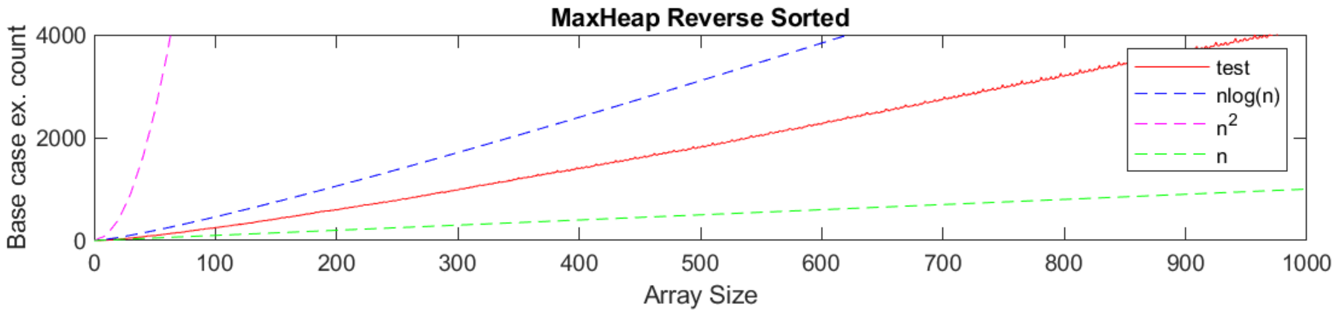


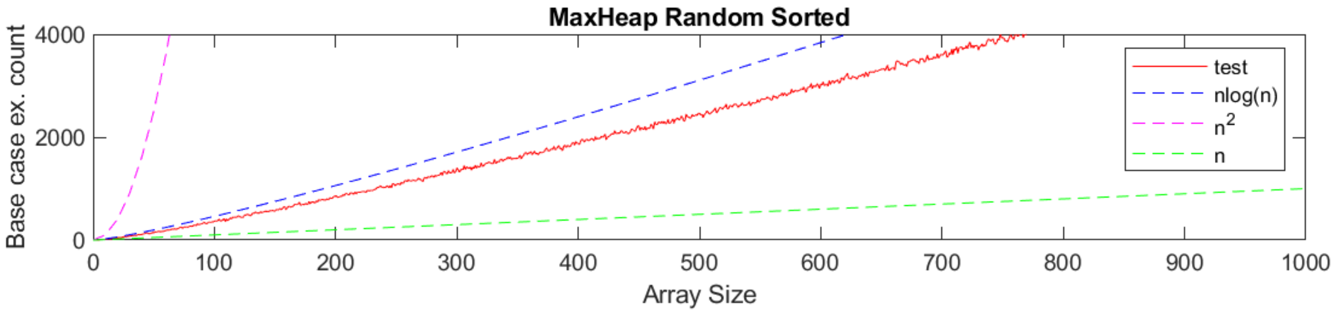
	Best Case	Average Case	Worst Case
Insertion	O(n)	O(n^2)	O(n^2)

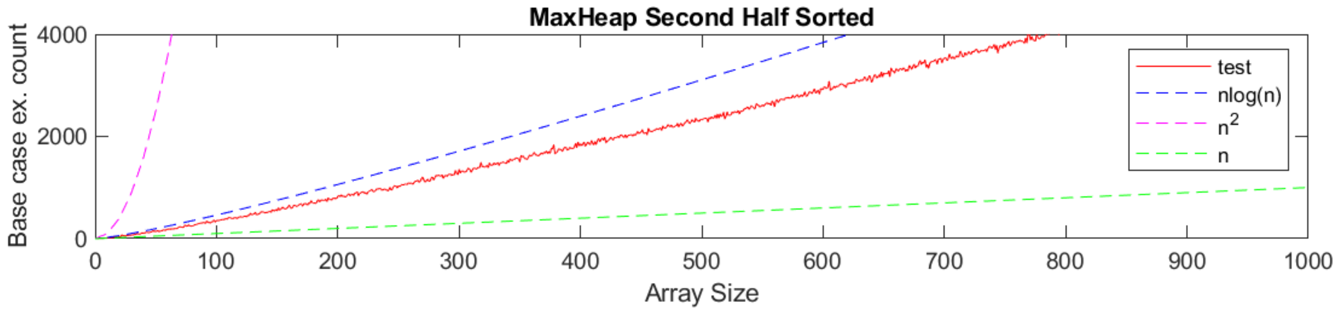
As graphs show its perfectly fit the theoretical functions, sorted array hit the best case and others hit worst case.

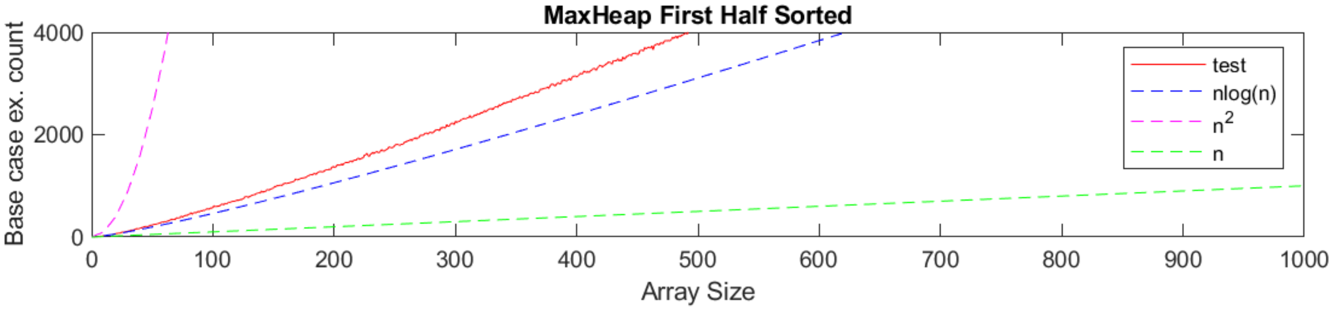
Max Heap:







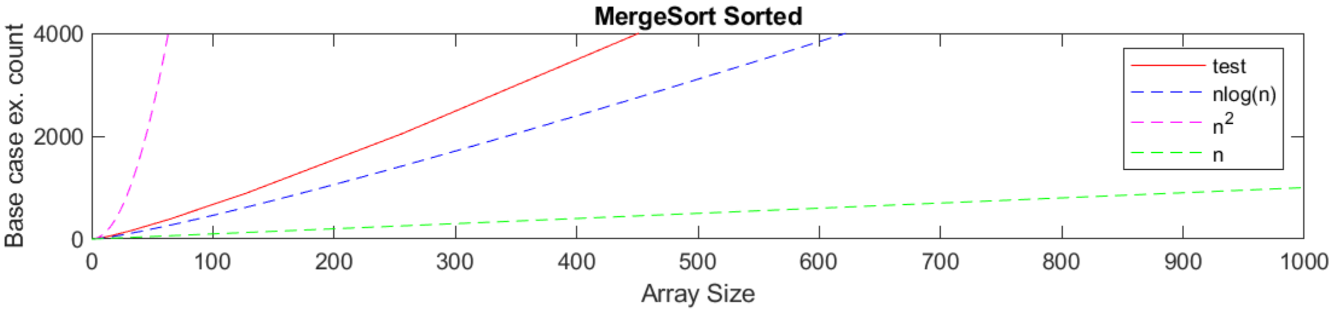


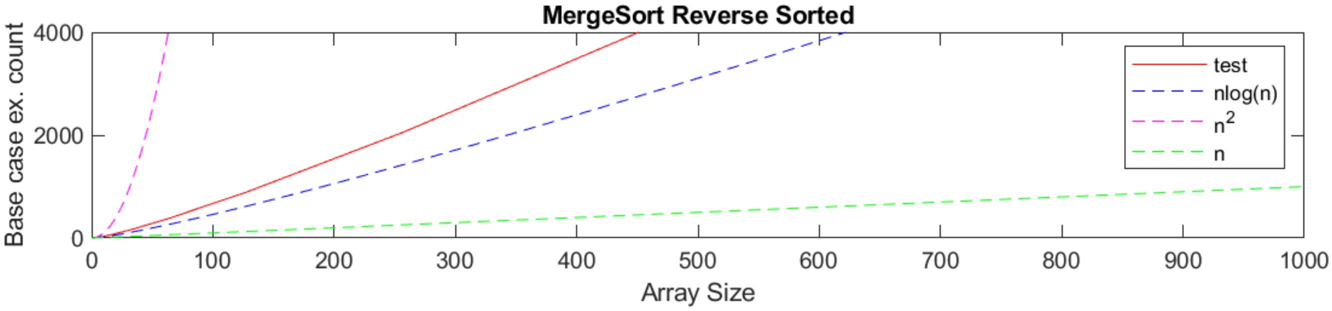


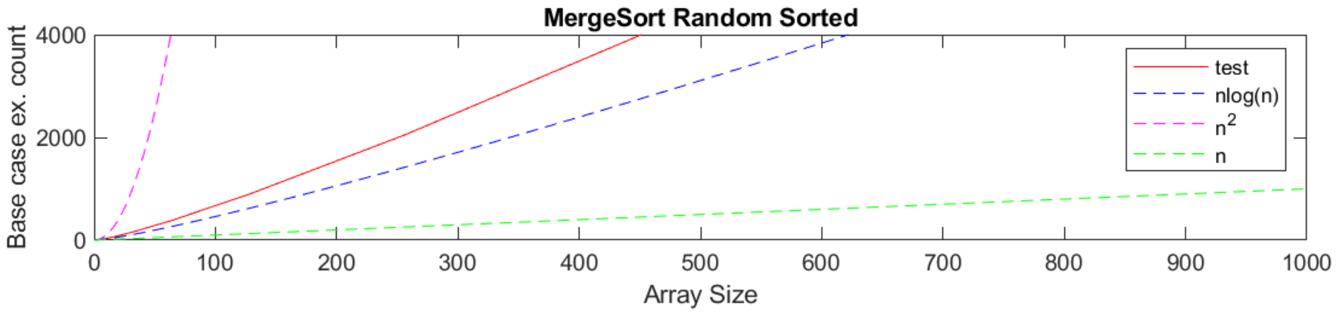
	Best Case	Average Case	Worst Case
Max Heap	O(n log(n))	O(n log(n))	O(n log(n))

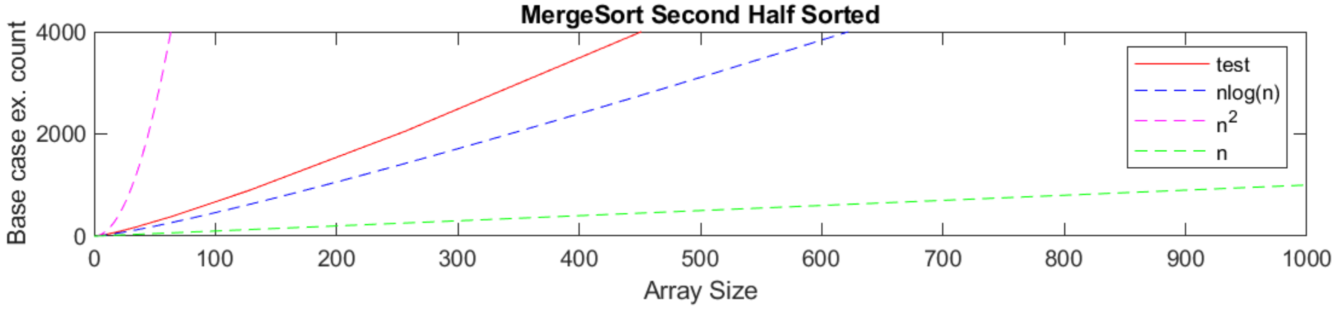
For this case we can see none of them n^2 but problem is its look like linear function since n*log(n) looks pretty much like n function the I don't know how to proof exactly that graphs are n*log(n) but when I first draw this without n^2 function test data goes with n*log(n) way bigger than n function I cant draw all the test results because of n^2 function get bigger so fast and other 3 looks like straight line around zero. So I assume these is fit theoretical functions.

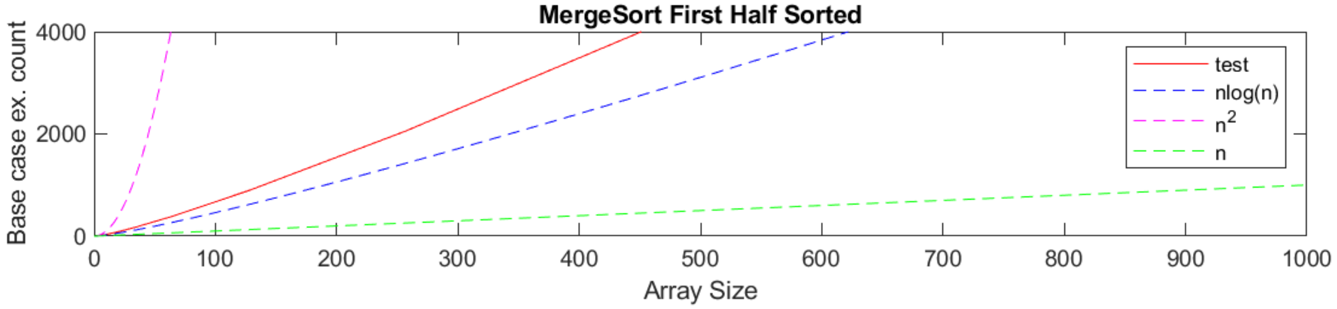
Merge Sort:






MergeSort Random Sorted

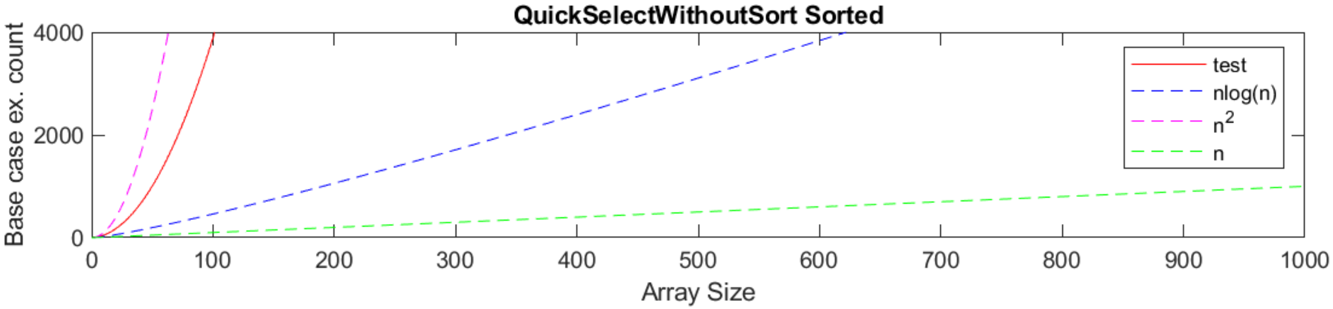

MergeSort Second Half Sorted

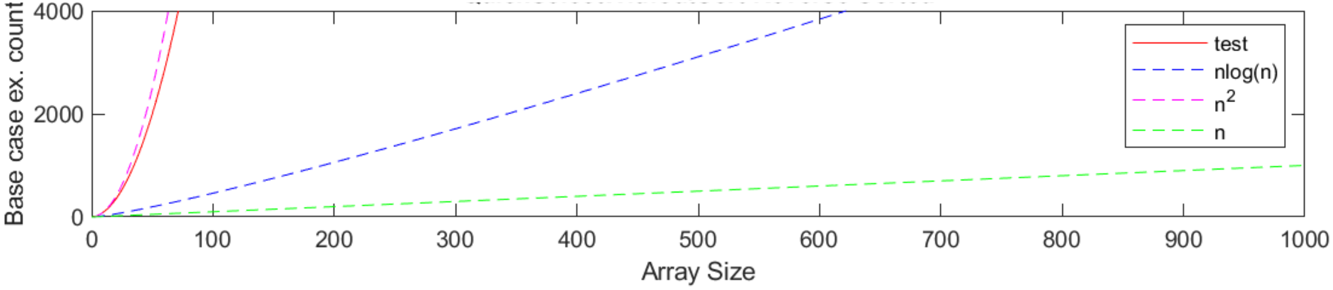

MergeSort First Half Sorted

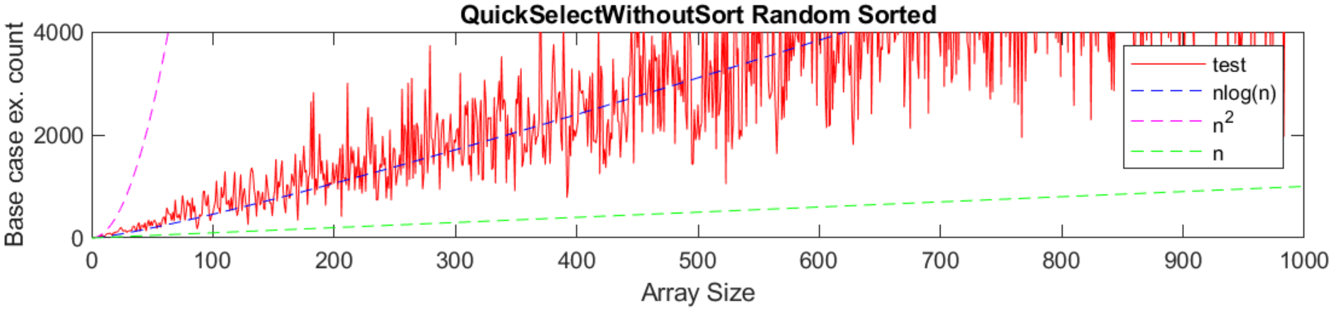
	Best Case	**Average Case**	**Worst Case**
Merge Sort	O(n log(n))	O(n log(n))	O(n log(n))

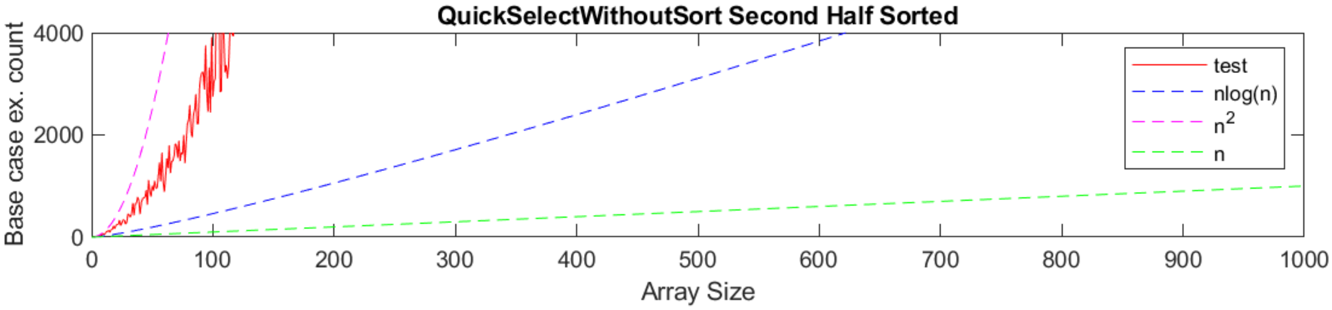
As we can clearly see again its definitely not quadratic, like explained in Max Heap, I assume its fit the theoretical functions.

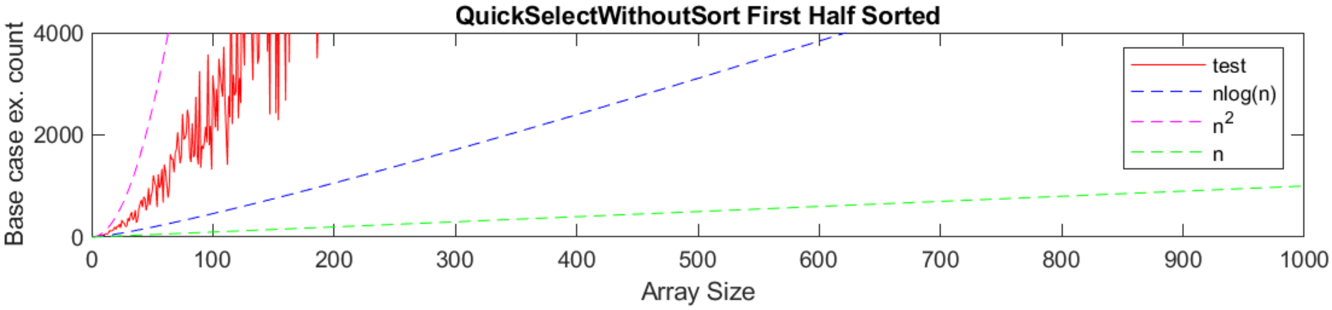
Quick Select:







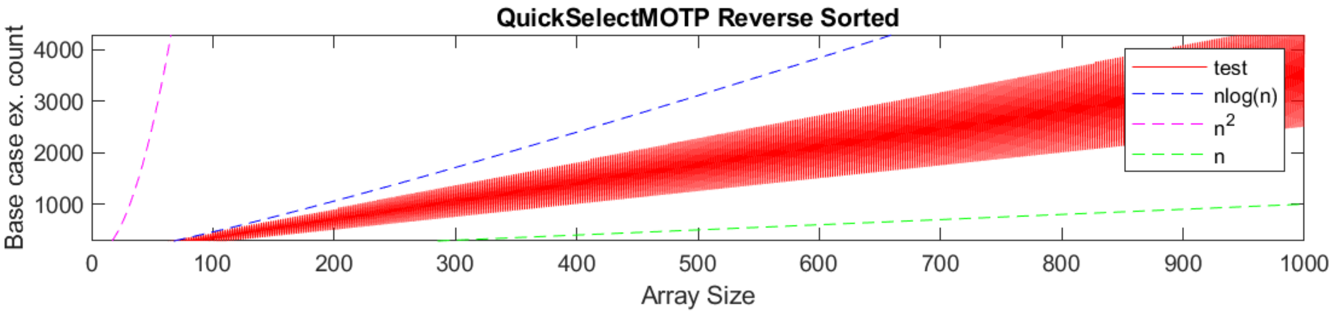


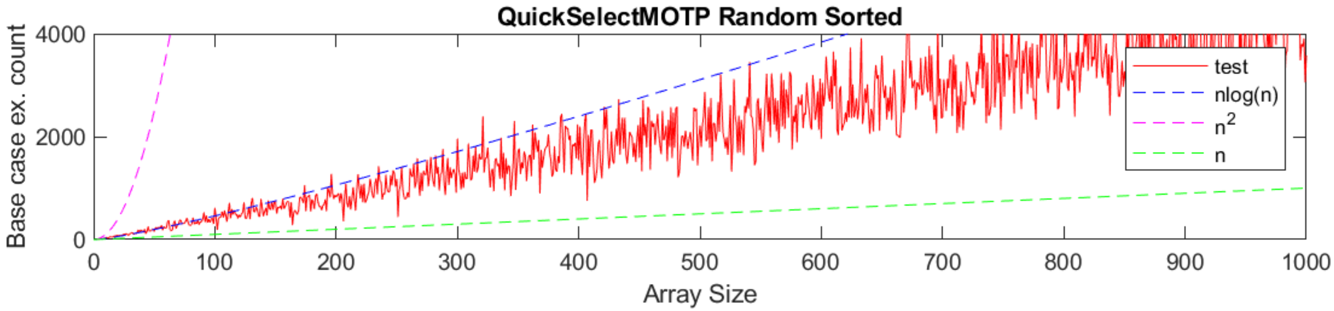


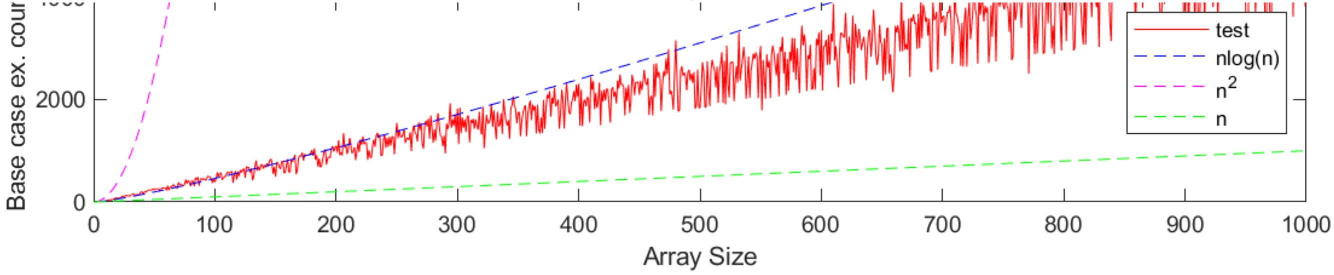
	Best Case	**Average Case**	**Worst Case**
Quick	O(n log(n))	O(n log(n))	O(n^2)

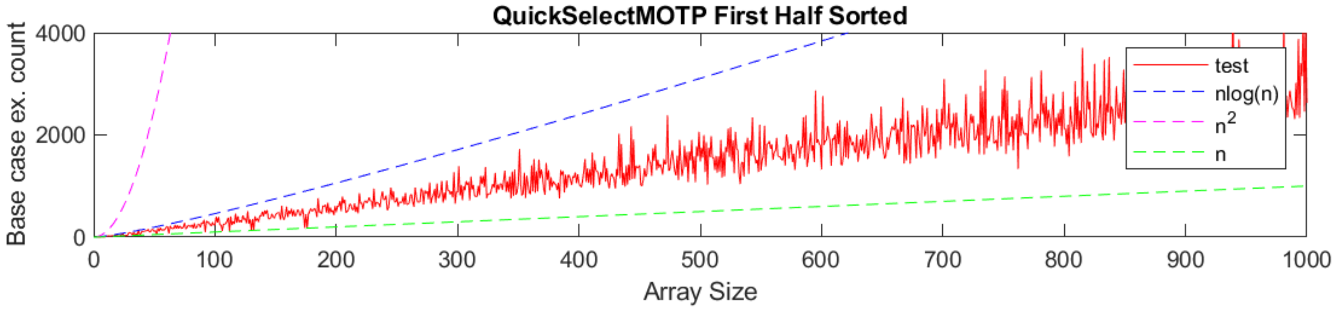
As we can see quick select getting quadratic if array is sorted or partially sorted and its hits n*log(n) if array is randam.

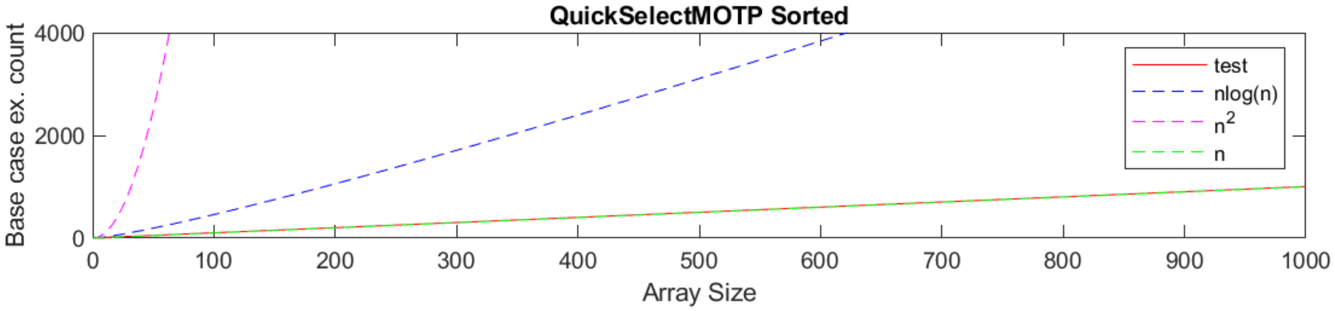
Quick Select MOTP:


QuickSelectMOTP Reverse Sorted


QuickSelectMOTP Random Sorted




QuickSelectMOTP First Half Sorted



For this algorithm I couldn’t find about complexity on the Ethernet but it seems n*log(n) on worst case and n with best case which is sorted maybe for reverse sorted it can be n.