CS202 HW1 FALL 2019

Name: Fatih Sevban Uyanık
Id: 21602486

Q1)

A) $f_5 < f_4 < f_{10} < f_9 < f_6 < f_8 < f_7 < f_1 < f_2 < f_3$

B) Let us consider that we got n/2 from random(n). Thus,

T(n) = T(n/2) + T(n/2) = 2T(n/2)
= 2(2T(n/4)) = 4T(n/4)
= 4(2T(n/8)) = 8T(n/8)
= 2^log(n) * T(n / 2^log(n))
= 2^log(n) * T(n / n)
= n * T(1)
= n = **O(n)**

C)

Bubble Sort

	Original	-->	607, 1896, 1165, 2217, 675, 2492, 2706, 894, 743, 568
1.	Iteration	-->	607, 1165, 1896, 675, 2217, 2492, 894, 743, 568, 2706
2.	Iteration	-->	607, 1165, 675, 1896, 2217, 894, 743, 568, 2492, 2706
3.	Iteration	-->	607, 675, 1165, 1896, 894, 743, 568, 2217, 2492, 2706
4.	Iteration	-->	607, 675, 1165, 894, 743, 568, 1896, 2217, 2492, 2706
5.	Iteration	-->	607, 675, 894, 743, 568, 1165, 1896, 2217, 2492, 2706
6.	Iteration	-->	607, 675, 743, 568, 894, 1165, 1896, 2217, 2492, 2706
7.	Iteration	-->	607, 675, 568, 743, 894, 1165, 1896, 2217, 2492, 2706
8.	Iteration	-->	607, 568, 675, 743, 894, 1165, 1896, 2217, 2492, 2706
9.	Iteration	-->	568, 607, 675, 743, 894, 1165, 1896, 2217, 2492, 2706

Radix Sort

	Original	-->	607, 1896, 1165, 2217, 675, 2492, 2706, 894, 743, 568
1.	Iteration	-->	2492, 743, 894, 1165, 675, 1896, 2706, 607, 2217, 568
2.	Iteration	-->	2706, 607, 2217, 743, 1165, 568, 675, 2492, 894, 1896
3.	Iteration	-->	1165, 2217, 2492, 568, 607, 675, 2706, 743, 894, 1896
4.	Iteration	-->	568, 607, 675, 743, 894, 1165, 1896, 2217, 2492, 2706

Q2)

QUICK SORT

[22, 11, 6, 7, 30, 2, 27, 24, 9, 1, 20, 17]

[1, 2, 6, 7, 9, 11, 17, 20, 22, 24, 27, 30]

Move Count: 105
Comparison Count: 42

INSERTION SORT

[22, 11, 6, 7, 30, 2, 27, 24, 9, 1, 20, 17]

[1, 2, 6, 7, 9, 11, 17, 20, 22, 24, 27, 30]

Move Count: 58
Comparison Count: 36

HYBRID SORT

[22, 11, 6, 7, 30, 2, 27, 24, 9, 1, 20, 17]

[1, 2, 6, 7, 9, 11, 17, 20, 22, 24, 27, 30]

Move Count: 48
Comparison Count: 32

----------QUICK SORT----------

****
Array Size : 3000
Time Elapsed : 0.3453
Comparison Count: 38486
Move Count : 69846
****

****
Array Size : 4500
Time Elapsed : 0.5532
Comparison Count: 65022
Move Count : 109113
****
****
Array Size : 6000
Time Elapsed : 0.747
Comparison Count: 81159
Move Count : 140604
****
****
Array Size : 7500
Time Elapsed : 0.9418
Comparison Count: 110098
Move Count : 176274
****
****
Array Size : 9000
Time Elapsed : 1.1657
Comparison Count: 133403
Move Count : 225003
****
****
Array Size : 10500
Time Elapsed : 1.465
Comparison Count: 169456
Move Count : 304314
****
****
Array Size : 12000
Time Elapsed : 1.6605
Comparison Count: 187337
Move Count : 304431
****
****
Array Size : 13500
Time Elapsed : 1.8275
Comparison Count: 237722
Move Count : 402063
****
****
Array Size : 15000
Time Elapsed : 2.0411
Comparison Count: 244181
Move Count : 414942
****

----------HYBRID SORT----------

```
*********************************************
Array Size          : 3000
Time Elapsed        : 0.3003
Comparison Count: 38816
Move Count          : 57658
*********************************************
*********************************************
Array Size          : 4500
Time Elapsed        : 0.5019
Comparison Count: 62269
Move Count          : 108642
*********************************************
*********************************************
Array Size          : 6000
Time Elapsed        : 0.6411
Comparison Count: 80255
Move Count          : 126513
*********************************************
*********************************************
Array Size          : 7500
Time Elapsed        : 0.8493
Comparison Count: 106237
Move Count          : 165984
*********************************************
*********************************************
Array Size          : 9000
Time Elapsed        : 1.0734
Comparison Count: 136102
Move Count          : 225355
*********************************************
*********************************************
Array Size          : 10500
Time Elapsed        : 1.2178
Comparison Count: 153861
Move Count          : 232503
*********************************************
*********************************************
Array Size          : 12000
Time Elapsed        : 1.4385
Comparison Count: 186352
Move Count          : 316615
*********************************************
*********************************************
Array Size          : 13500
Time Elapsed        : 1.6202
Comparison Count: 202785
Move Count          : 321200
*********************************************
*********************************************
Array Size          : 15000
Time Elapsed        : 1.8545
```

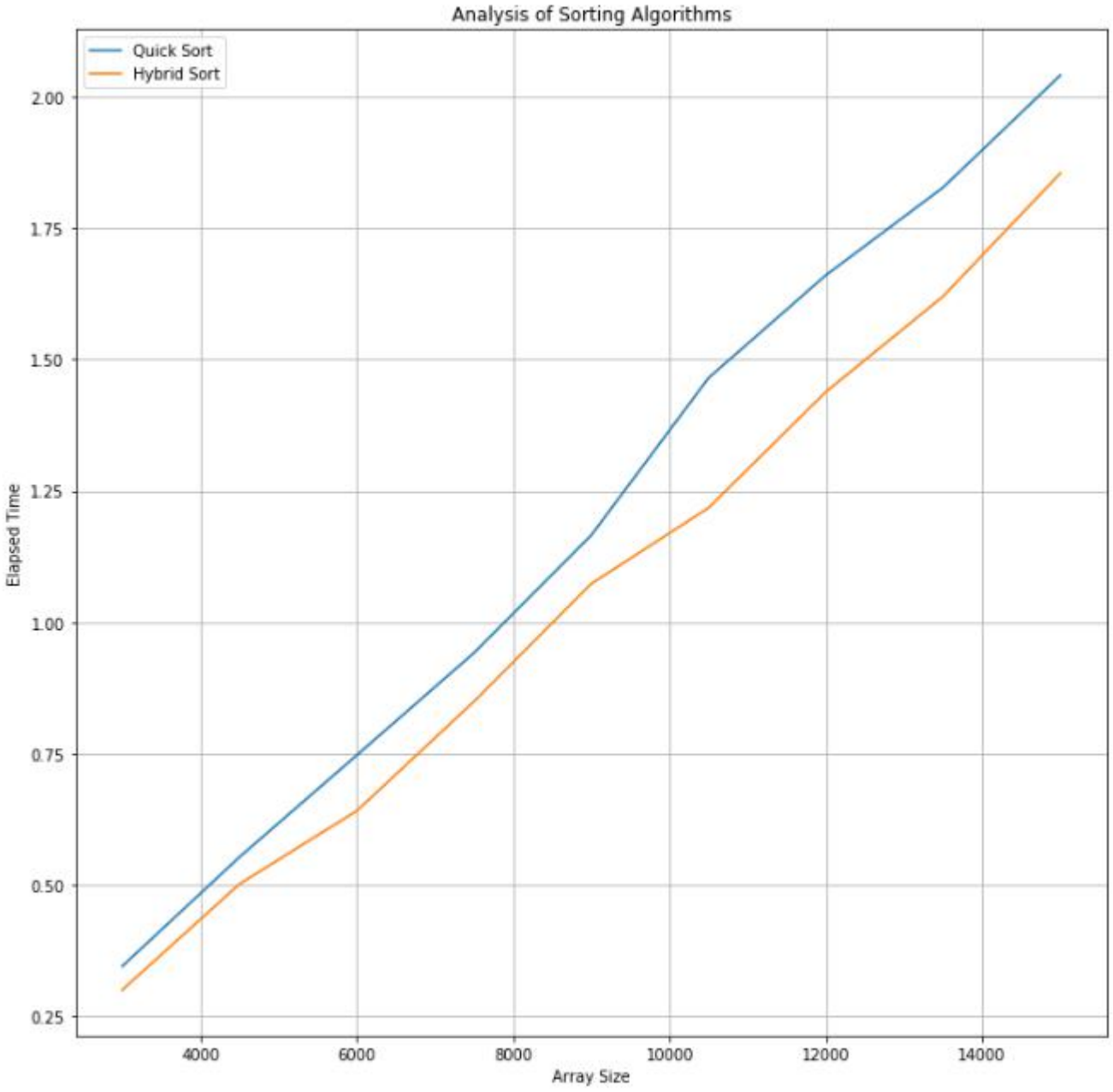
Comparison Count: 237120
Move Count : 389499
****

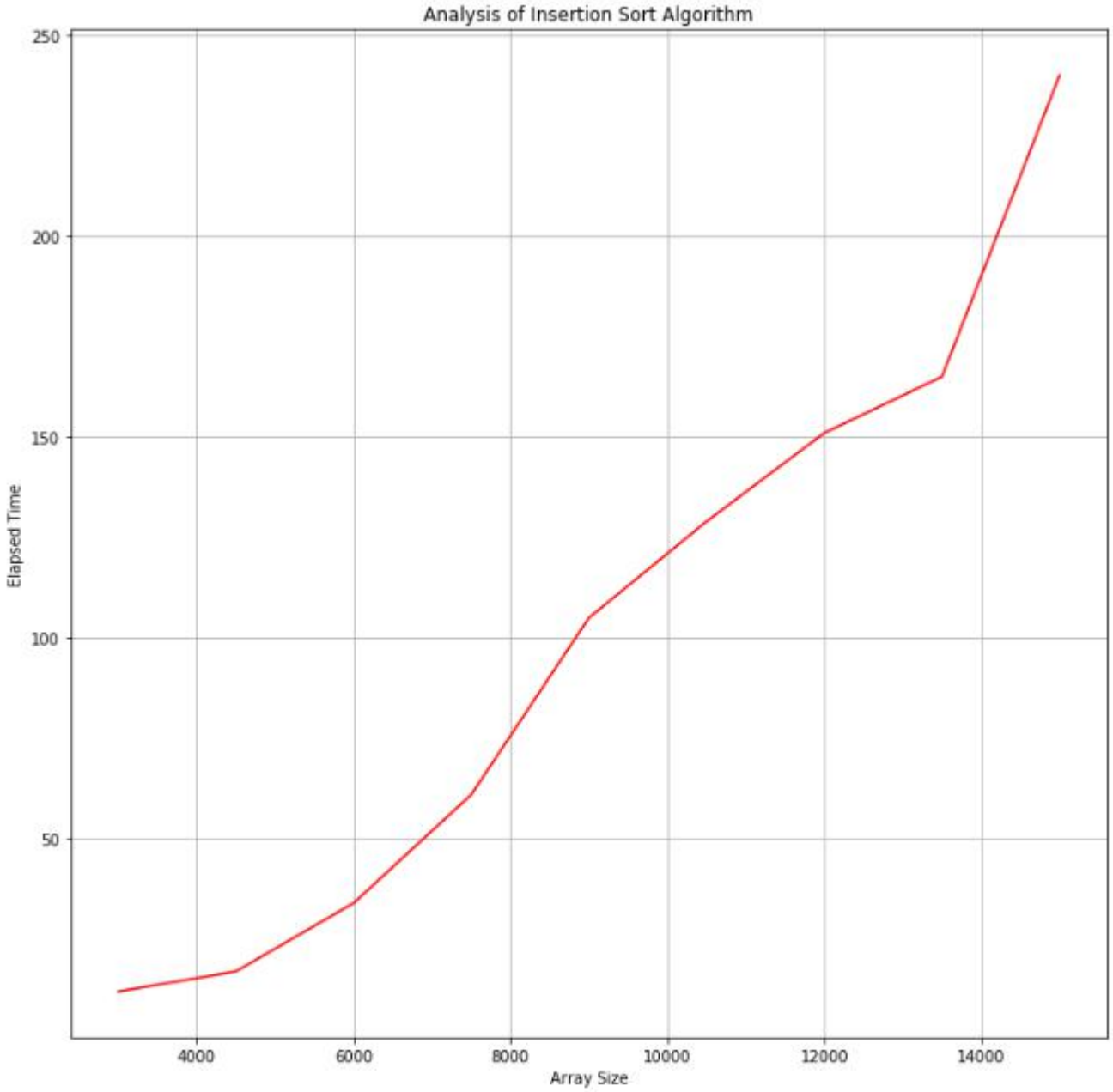
----------INSERTION SORT----------

****
Array Size : 3000
Time Elapsed : 12
Comparison Count: 2229474
Move Count : 2235472
****
****
Array Size : 4500
Time Elapsed : 17
Comparison Count: 5040461
Move Count : 5049459
****
****
Array Size : 6000
Time Elapsed : 34
Comparison Count: 8873564
Move Count : 8885562
****
****
Array Size : 7500
Time Elapsed : 61
Comparison Count: 13946591
Move Count : 13961589
****
****
Array Size : 9000
Time Elapsed : 105
Comparison Count: 20074592
Move Count : 20092590
****
****
Array Size : 10500
Time Elapsed : 129
Comparison Count: 27424225
Move Count : 27445223
****
****
Array Size : 12000
Time Elapsed : 151
Comparison Count: 36059777
Move Count : 36083775
****
****
Array Size : 13500

Time Elapsed : 165
Comparison Count: 45464675
Move Count : 45491673
****
****
Array Size : 15000
Time Elapsed : 240
Comparison Count: 56194328
Move Count : 56224326
****

Q3)


Analysis of Sorting Algorithms
Quick Sort
Hybrid Sort
Elapsed Time
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
4000
6000
8000
10000
12000
14000
Array Size



Because Quicksort is O(nlogn), its execution time is much lower in comparisan with insertion sort whose order of magnitude is O(n^2). In addition, combining these two algorithms increased the performance because insertion sort performs better in small sized arrays whereas quicksort performs bettter in large sized arrays. The reason why insertion sort performs better in small sized array is that it does not uses recursion and fills the stack with activation records.