CS202-2019 Fall

Algorithm Efficiency and Sorting

Muhammed Naci Dalkıran

21601736

Sec: 01

Assigment: 01

Question 1

a)

n^ (1/logn) < logn < √ n < nlogn < log(n!) < n ^(log(logn)) < n^3 < e^n < 10^n < n!

F5 < F4 < F10 < F6 < F2 < F9 < F8 < F7 < F1 < F3

b)

ik = (1,2,3...k) k is elements of natural number and these number is generated in each step.

T(n) = T(i1) +T(n-1-i1)

=[T(i2) + T(i1-1-i2)]+ [T(i3) + (n -1-i1-1-i3)] = ...

In every stage the number ik is generated and it provide to determine orocessing time. When looked at the avarage case, ik is genarally half of n. When reorganized notation:

T(n) = T(n/2) + T(n – 1 – n/2)

=[T(n/2) + T(n/2-1-n/2)] + [T(n/2) + T(n-1-1-n/2-n/2)]

= T[(n/2) + O(1)] + [T(n/2) +O(1)]...(after n iteration)

= 2^n

c)

***In Bubble Sort**

Initial case

607, 1896, 1165, 2217, 675, 2492, 2706, 894, 743, 568

Step 1 : compare 607 and 1896 and go on. And swap 1896 with 1165

607, 1165, 1896, 2217, 675, 2492, 2706, 894, 743, 568

And swap 2217 with 675 and 2792 with 894, 743, 568. These means our biggest number is 2492 and until thise number, array is unsorted.

607, 1165, 1896, 675, 2217 , 2492, 2706, 894, 743, 568

607, 1165, 1896, 675, 2217 , 2492, 894, 743, 568, 2706,

Then, swap 1896 with 675 and swap 2492 step by step with 894,743 and 568.

607, 1165, 675, 1896, 2217 , 2492, 894, 743, 568, 2706

607, 1165, 675, 1896, 2217 894, 743, 568, 2492, 2706

Swap 1165 with 675 and 2217 with 894,743,568.

607, 675, 1165, 1896, 894, 743, 568, 2217,   2492, 2706

607, 675, 1165, 894, 743, 568, 1896, 2217,   2492, 2706

607, 675, 894, 743, 568, 1165, 1896, 2217,   2492, 2706

Swap with 894 with 743 and 568 step by step

607, 675, 743, 568, 894, 1165, 1896, 2217,   2492, 2706

607, 675, 568, 743, 894, 1165, 1896, 2217,   2492, 2706

Swap 675 with 568 and swap 607 with 568

568, 607, 675, 743, 894, 1165, 1896, 2217,   2492, 2706

***In radix Sort**

Initial case : 607, 1896, 1165, 2217, 675, 2492, 2706, 894, 743, 568

Look at less magnitude digit and sort it

2492,743,894,1165,675,1896,2706,607,2217,568

Look at second less magnitude digit and sort it

2706,607,2217,743,1165,568,675,2492,894,1896

Look at third less magnitude digit and sort it

1165,2217,2492,568,607,675,2706,743,894,1896

Look at fourth digit and get sorted array

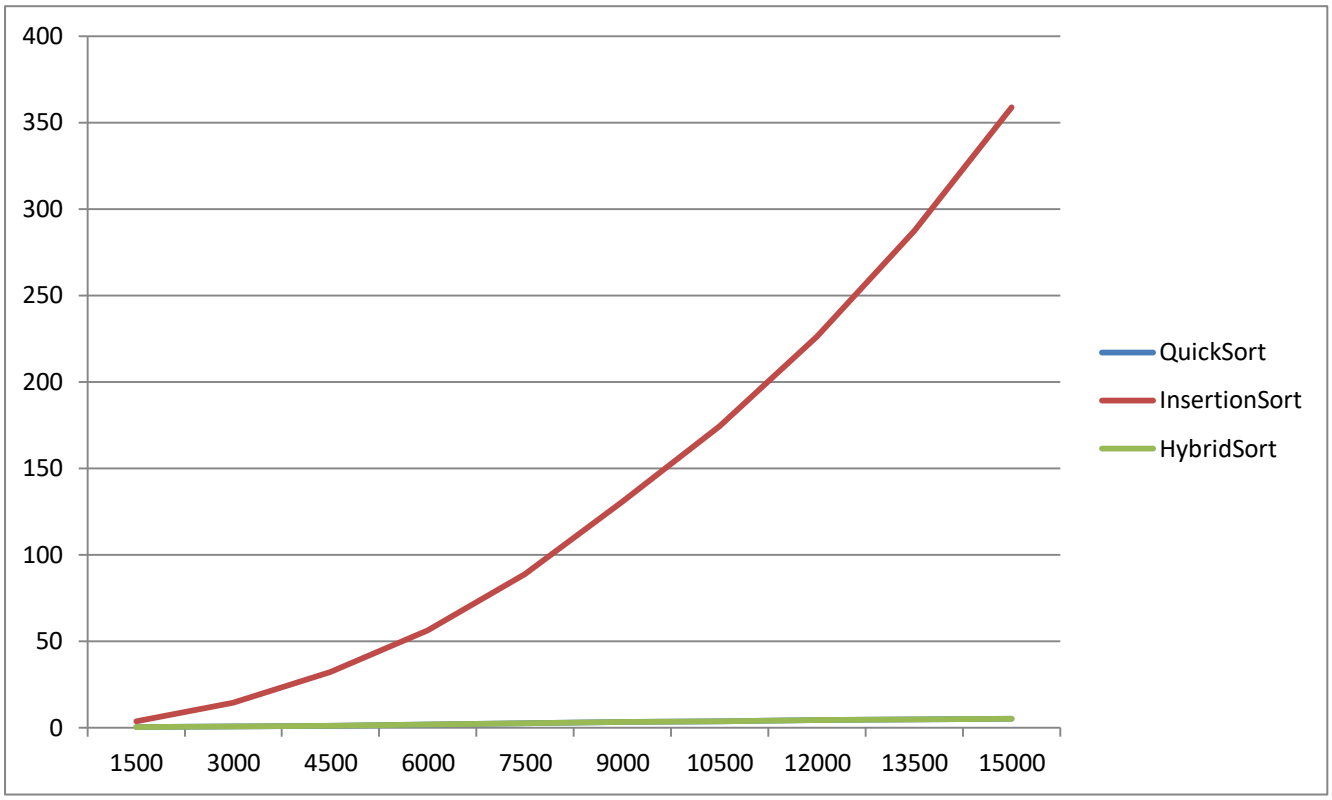
568, 607, 675, 743, 894, 1165, 1896, 2217,   2492, 2706

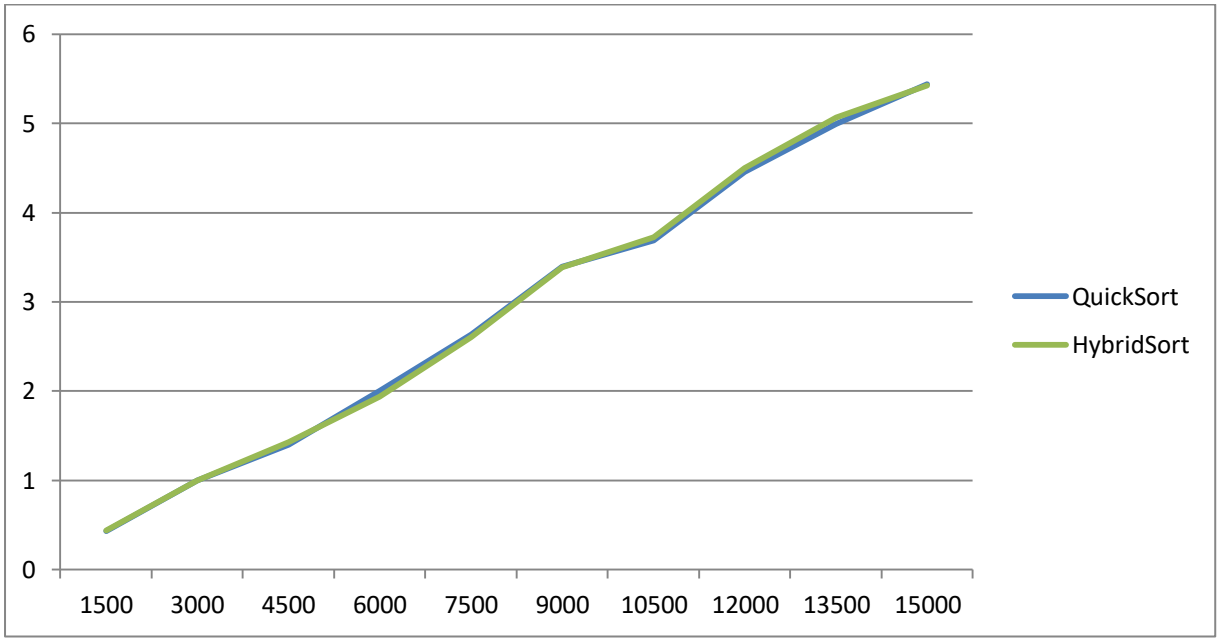
Question 2

```
./hw1
22 11 6 7 30 2 27 24 9 1 20 17
22 11 6 7 30 2 27 24 9 1 20 17
22 11 6 7 30 2 27 24 9 1 20 17
1 2 6 7 9 11 17 20 22 24 27 30
1 2 6 7 9 11 17 20 22 24 27 30
1 2 6 7 9 11 17 20 22 24 27 30
-----------------------------------
```

```
Part a - Time analysis of Quick Sort
Array Size      Time Elapsed    compCount       moveCount
1500            0.44125         27790           29491
3000            0.998949                65999           75482
4500            1.40238         94363           91603
6000            1.99726         134471          139179
7500            2.6391          172776          186893
9000            3.39104         231204          259005
10500           3.69601         248202          256103
12000           4.45219         296658          323091
13500           4.98598         345471          350286
15000           5.42891         357080          376861
-----------------------------------------------------
Part b - Time analysis of Insertion Sort
Array Size      Time Elapsed    compCount       moveCount
1500            3.73073         582538          582538
3000            14.4735         2269885         2269885
4500            32.5296         5113881         5113881
6000            56.4238         8879246         8879246
7500            87.9312         13831444        13831444
9000            130.289         20318380        20318380
10500           175.213         27293656        27293656
12000           228.802         35642677        35642677
13500           290.726         45287290        45287290
15000           361.134         56498125        56498125
-----------------------------------------------------
Part b - Time analysis of Hybrid Sort
Array Size      Time Elapsed    compCount       moveCount
1500            0.43825         28108           31341
3000            0.998752                65986           75481
4500            1.43254         97099           95849
6000            1.93782         131050          131077
7500            2.59957         167748          186185
9000            3.38951         229192          255579
10500           3.72183         253280          259507
12000           4.49349         302074          326486
13500           5.05757         349366          360227
15000           5.41766         359578          376550
-----------------------------------------------------
```

Question 3





When comparing theoretical and empirical result, sometimes there is a little difference, but sometimes there might be huge differences. Since power of computer or operating system can be changeable and it cannot be the most ideal form, theoretical and empirical result can be different. Also, Other reason of differences is that computer can be affected by temperature or other application which is working on background and that when computer is working, it is getting hot and this affect computer's devices, cables and naturally its performance.

Asymptotic notation of QuickSort algorithm is O (nlogn) and Asymptotic notation of InsertionSort algorithm is O (n^2); therefore QuickSort is more efficient than InsertionSort. When compared with QuickSort, HybridSort is more efficient. HybridSort has InsertionSort for smaller size of array and also has QuickSort for bigger size of array. This combination provides HybridSort with opportunities to become more efficient. Also, InsertionSort is more effective than QuickSort until size of array is less than 10.