Q1)

a) T(n) = 5T(n/3) + n logn , n =3^k
= 5T(3^k/3) + 3^k log 3^k
T (n/3)= 5 (5T(n/9) + n/3 log n/3) + nlogn
T (n/9)= 5^2 T (n/3^2) + 5n/3 log n/3 + n logn
= 5^3 T(n/3^3) + 5^3n/3^3 log n/3^3 + 5n/3 log n/3 + n logn
.
.
= 5^k T(n/3^k) + sigma (i=0 , k)(5/3)^k nlog (n/3^k), (k = $log_3 n$)
= 5 ^ ($log_3 n$) T(1) + sigma (i=0 , $log_3 n$)(5/3)^ (k) nlog (n/3^k)
= 5 ^($log_5 n/log_3 5$) + sigma (i=1 , k)(5/3)^i nlog(u/3i)
= n ^($log_3 5$) + sigma (i=1 , k) (5/3)^i nlog (u/3^i)
<= n ^($log_3 5$) + sigma (i=0 , k) (5/3)^k nlog n
<= n ^($log_3 5$) + ($log_3 n$) n ^ ($log_3 5$) n (logn)^2
= O(n^ ($log_3 5$))

T(n) = T(n - 1) + n^2, T(n-1) = T(n - 2) + (n - 1)^2 + n^2
T(n-2) = T(n - 3) + (n - 2)^2 + (n - 1)^2 + n^2
We observe that:
T(n) = T (n-k) + (n-k+1)^2 + (n-k+2)^2 + (n-k+2)^2 +….+ (n-1)^2 + n^2
Now assuming k = n - 1, we have:
T(n) = T (1) + 1^2 + 2^2 + … + (n - 1)^2 + n^2
T(n) = 1 + 1^2 + 2^2 + … +n^2 (using T(1) = 1)
T(n) = 1 + (n (n + 1) (2n+1)) / 6 (sum of squares)
O(n) = n^3

b) [44, 937, 13, 69,37, 80, 472, 49, 300, 183]
Merge sort
And refers to the fact that the elements were divided in the previous step.
[44, 937, 13, 69,37, 80, 472, 49, 300, 183] Divide(#1)
[44, 937, 13, 69,37] and [80, 472, 49, 300, 183] D(#2)
[44, 937, 13] and [69,37] || [80, 472, 49] and [300, 183] D(#3)
[44, 937] and [13] || [69] and [37]|| [80, 472] and [49] || [300]and [183] D(#4)
[44] and [937] || [13] || [69] || [37] ||[80] and [472] || [49] || [300] || [183] D(#5)

[44, 937]||[13] || [37,69] || [80, 472] || [49] || [183, 300] Compare and Merge (#1)
[13, 44, 937] || [37,69] || [49, 80, 472] || [183, 300] C&M(#2)
[13,37, 44, 69, 937] || [49, 80, 83, 300, 472] C&M(#3)
[13, 37, 44, 49, 69, 80, 83, 300, 472, 937] C&M(#4)

Insertion sort
(||) divides the sorted from unsorted, ({}) the current element.
All shifts were performed to the left and were shown as 1 only
[44||, {937}, 13, 69,37, 80, 472, 49, 300, 183] do nothing
[44, 937||, {13}, 69,37, 80, 472, 49, 300, 183] copy 13 and shift 2 steps
[13, 44, 937||, {69},37, 80, 472, 49, 300, 183] copy 69 and shift one step
[13, 44, 69, 937||, {37}, 80, 472, 49, 300, 183] copy 37 and shift 3 steps
[13, 37, 44, 69, 937||, {80}, 472, 49, 300, 183] copy 80 and shift 1 steps
[13, 37, 44, 69, 80, 937||, {472}, 49, 300, 183] copy 472 and shift 1 steps
[13, 37, 44, 69, 80, 472, 937||, {49}, 300, 183] copy 49 and shift 5 steps
[13, 37, 44, 49,69, 80, 472, 937||, {300}, 183] copy 300 and shift 2 steps
[13, 37, 44, 49,69, 80, 300, 472, 937||, {183}] copy 183 and shift 4 steps
[13, 37, 44, 49,69, 80, 183, 300, 472, 937,||]

c)
Worst case happens when the pivot is less the all the other elements resulting in recursive calls on n-1, where n is the number of elements in the previous step.
We know the T(0) = T(1) = 0
T(n) = n + T(n-1) Solving the relation:
T(n) = n + T(n-1)
T(n-1) = n + (n-1) + T(n-2)
T(n-2) = n + (n-1) + (n-2) + T(n-3)
T(n) = n + (n-1) + … + 3+ 2 + 1 (partial sum of the series is n (n+1))
T(n) = n^2 + n
O(n) = n^2

Q2)

```
abdul@abdul-Inspiron-5420: ~/Desktop
File  Edit  View  Search  Terminal  Help
1       17      43      57      58      92      93      99      100
1       17      43      57      58      92      93      99      100
1       17      43      57      58      92      93      99      100
-----------------------------------------------------
Part a - Time analysis of Quick Sort
Array Size              Time Elapsed (ms)               compCount               moveCount
2000                    1.344                           13927                   47141
4000                    3.668                           25893                   88355
6000                    5.19                            41685                   141111
8000                    9.544                           52453                   178551
10000                   15.58                           76845                   257143
12000                   22.44                           96399                   321057
14000                   32.564                          117679                  390561
16000                   41.888                          131531                  437265
18000                   52.902                          144345                  481127
20000                   70.4                            187006                  614290
-----------------------------------------------------
Part b - Time analysis of Insertion Sort
Array Size              Time Elapsed (ms)               compCount               moveCount
2000                    9.582                           1015628                 1019628
4000                    62.876                          3976005                 3984005
6000                    232.278                         8961802                 8973802
8000                    522.104                         16107587                16123587
10000                   1029.45                         24962621                24982621
12000                   1751.63                         35821519                35845519
14000                   2828.13                         48810292                48838292
16000                   4146.18                         63852925                63884925
18000                   5860.64                         81365184                81401184
20000                   8121.6                          100295387               100335387
-----------------------------------------------------
Part c - Time analysis of Hybrid Sort
Array Size              Time Elapsed (ms)               compCount               moveCount
2000                    0.532                           14686                   44986
4000                    2.136                           27402                   84582
6000                    5.016                           44077                   135145
8000                    9.248                           55601                   171189
10000                   14.58                           80897                   247607
12000                   21.792                          101295                  310037
14000                   30.184                          123167                  376995
16000                   40.336                          137885                  421957
18000                   52.398                          151348                  464666
20000                   67.06                           194829                  596107
```

Q3)

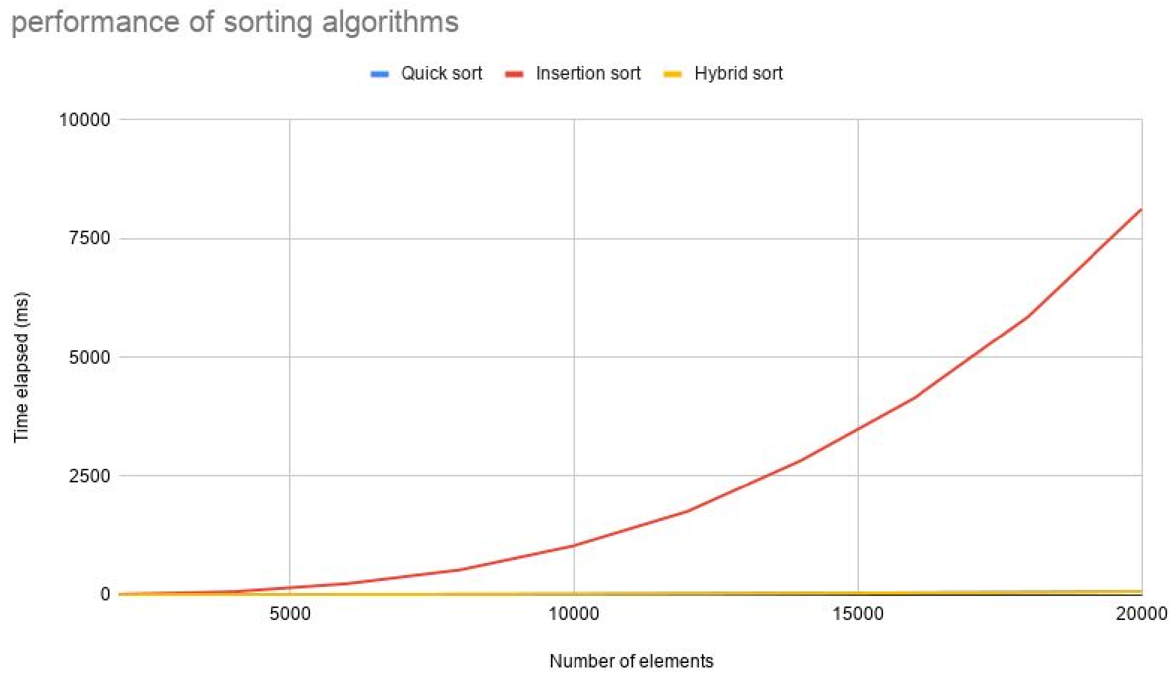

Figure 1: Sorting algorithms performance.

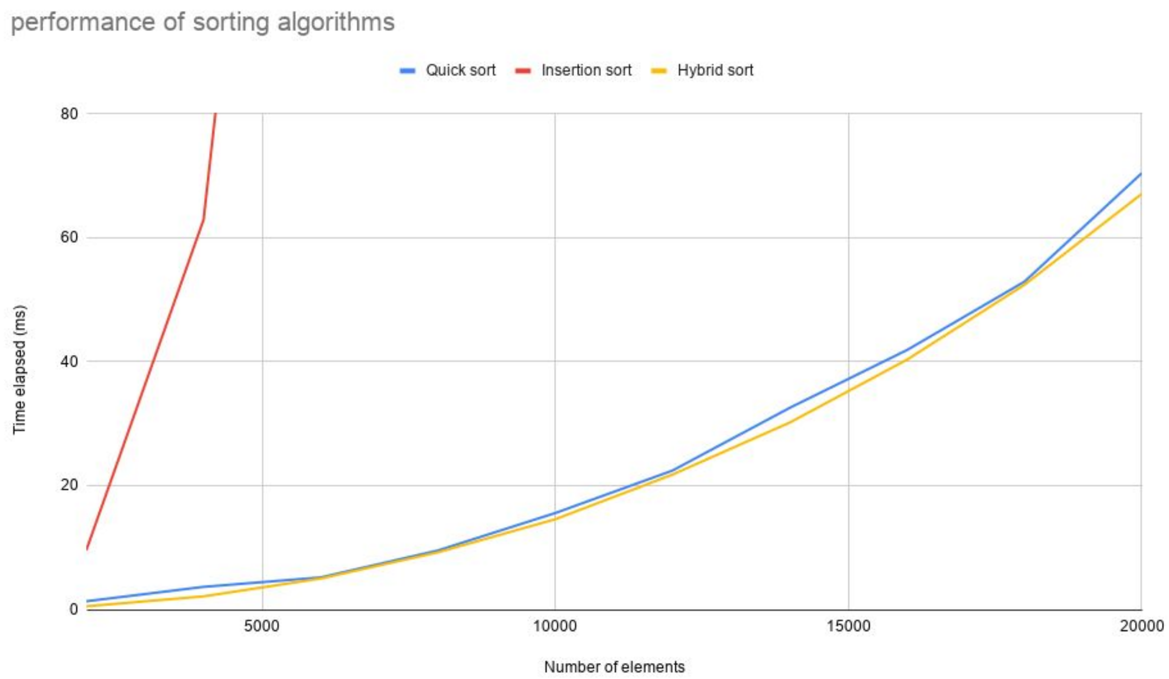

Figure 2: Close-up of the Sorting algorithms performance graph.

As we see in the graphs above Insertion Sort has significantly higher time compared to Quick sort and Hybrid Sort (figure 1). This matches the theory since worst-case for Insertion sort n n^2 and so is its average case. While for Quick Sort it is O(n logn) for average and n^2 for worst case scenario. We can realize that the worst-case does not always happen and that's why we had a huge difference between the two algorithms (difference between 20000^2 and 20000 * log 20000 is enormous), Since Hybrid is Quick sort for most of the process it is expected that the time is very close to Quick sort (figure 2). Hybrid sort, has the good of both, since the best case scenario for Insertion is O(n) and O(nlogn) for Quick sort, using Insertion for very small arrays is more efficient in terms of time, since the number of comparisons is small, resulting in less time in Insertion compared to Quick sort where we need to partition first then recursively call on itself.