İlke Kaş

21803184

Section 3

Question 1

(a) We need to find two positive constants: c and n_0 such that:

$$\mathbf{5n^3 + 4n^2 + 10 \leq c \cdot n^4 \; for \; \forall n \geq n_0}$$

$$\mathbf{\frac{5}{n} + \frac{4}{n^2} + \frac{10}{n^4} \leq c \; for \; \forall n \geq n_0}$$

If we choose c = 19 and $n_0 = 1$, we will show that this function is order of $O(n^4)$.

(b)

Insertion Sort: Red will show the next element
Blue will show the sorted sub list
Green will show shifted elements

Initial Array: 24, 8, 51, 28, 20, 29, 21, 17, 38, 27 (Copy 8)
24, 24, 51, 28, 20, 29, 21, 17, 38, 27 (Shift 24)
8, 24, 51, 28, 20, 29, 21, 17, 38, 27 (Insert 8; copy 51, insert 51 on top)
8, 24, 51, 28, 20, 29, 21, 17, 38, 27 (Copy 28)
8, 24, 51, 51, 20, 29, 21, 17, 38, 27 (Shift 51)
8, 24, 28, 51, 20, 29, 21, 17, 38, 27(Insert 28; copy 20)
8, 24, 24, 28, 51, 29, 21, 17, 38, 27 (Shift 24, 28, 51)
8, 20, 24, 28, 51, 29, 21, 17, 38, 27 (Insert 20; copy 29)
8, 20, 24, 28, 51, 51, 21, 17, 38, 27 (Shift 51)
8, 20, 24, 28, 29, 51, 21, 17, 38, 27 (Insert 29; Copy 21)
8, 20, 24, 24, 28, 29, 51, 17, 38, 27 (Shift 24, 28, 29, 51)
8, 20, 21, 24, 28, 29, 51, 17, 38, 27 (Insert 21; Copy 17)
8, 20, 20, 21, 24, 28, 29, 51, 38, 27 (Shift 20, 21, 24, 28, 29, 51)
8, 17, 20, 21, 24, 28, 29, 51, 38, 27 (Insert 17; Copy 38)
8, 17, 20, 21, 24, 28, 29, 51, 51, 27 (Shift 51)
8, 17, 20, 21, 24, 28, 29, 38, 51, 27 (Insert 38; Copy 27)
8, 17, 20, 21, 24, 28, 28, 29, 38, 51 (Shift 28, 29,38, 51)

Sorted Array: 8, 17, 20, 21, 24, 27, 28, 29, 38, 51 (Insert 27)

Bubble Sort: Blue will show the sorted sub list

Green will show the compared elements

1st pass: 24, 8, 51, 28, 20, 29, 21, 17, 38, 27 (Swap 24 and 8; increment index)

8, 24, 51, 28, 20, 29, 21, 17, 38, 27 (Increment index)

8, 24, 51, 28, 20, 29, 21, 17, 38, 27 (Swap 51 and 28; increment index)

8, 24, 28, 51, 20, 29, 21, 17, 38, 27 (Swap 51 and 20; increment index)

8, 24, 28, 20, 51, 29, 21, 17, 38, 27 (Swap 29 and 51; increment index)

8, 24, 28, 20, 29, 51, 21, 17, 38, 27 (Swap 51 and 21; increment index)

8, 24, 28, 20, 29, 21, 51, 17, 38, 27 (Swap 51 and 17; increment index)

8, 24, 28, 20, 29, 21, 17, 51, 38, 27 (Swap 51 and 38; increment index)

8, 24, 28, 20, 29, 21, 17, 38, 51, 27 (Swap 51 and 27)

8, 24, 28, 20, 29, 21, 17, 38, 27, 51 (Increment pass; still unsorted)

2nd pass: 8, 24, 28, 20, 29, 21, 17, 38, 27, 51 (Increment index)

8, 24, 28, 20, 29, 21, 17, 38, 27, 51 (Increment index)

8, 24, 28, 20, 29, 21, 17, 38, 27, 51 (Swap 28 and 20; increment index)

8, 24, 20, 28, 29, 21, 17, 38, 27, 51 (Increment index)

8, 24, 20, 28, 29, 21, 17, 38, 27, 51 (Swap 29 and 21; increment index)

8, 24, 20, 28, 21, 29, 17, 38, 27, 51 (Swap 29 and 17; increment index)

8, 24, 20, 28, 21, 17, 29, 38, 27, 51 (Increment index)

8, 24, 20, 28, 21, 17, 29, 38, 27, 51 (Swap 38 and 27; increment index)

8, 24, 20, 28, 21, 17, 29, 27, 38, 51 (Increment pass; still unsorted)

3rd pass: 8, 24, 20, 28, 21, 17, 29, 27, 38, 51 (Increment index)

8, 24, 20, 28, 21, 17, 29, 27, 38, 51 (Swap 24 and 20; increment index)

8, 20, 24, 28, 21, 17, 29, 27, 38, 51 (Increment index)

8, 20, 24, 28, 21, 17, 29, 27, 38, 51 (Swap 28 and 21; increment index)

8, 20, 24, 21, 28, 17, 29, 27, 38, 51 (Swap 28 and 17; increment index)

8, 20, 24, 21, 17, 28, 29, 27, 38, 51 (Increment index)

8, 20, 24, 21, 17, 28, 29, 27, 38, 51 (Swap 29 and 27; increment index)

8, 20, 24, 21, 17, 28, 27, 29, 38, 51 (Increment pass; still unsorted)

4th pass: 8, 20, 24, 21, 17, 28, 27, 29, 38, 51 (Increment index)

8, 20, 24, 21, 17, 28, 27, 29, 38, 51 (Increment index)

8, 20, 24, 21, 17, 28, 27, 29, 38, 51 (Swap 24 and 21; increment index)

8, 20, 21, 24, 17, 28, 27, 29, 38, 51 (Swap 24 and 17; increment index)

8, 20, 21, 17, 24, 28, 27, 29, 38, 51 (Increment index)

8, 20, 21, 17, 24, 28, 27, 29, 38, 51 (Swap 28 and 27; increment index)

8, 20, 21, 17, 24, 27, 28, 29, 38, 51 (Increment pass; still unsorted)

5th pass: 8, 20, 21, 17, 24, 27, 28, 29, 38, 51 (Increment index)

8, 20, 21, 17, 24, 27, 28, 29, 38, 51 (Increment index)

8, 20, 21, 17, 24, 27, 28, 29, 38, 51 (Swap 21 and 17; increment index)

8, 20, 17, 21, 24, 27, 28, 29, 38, 51 (Increment index)

8, 20, 17, 21, 24, 27, 28, 29, 38, 51 (Increment index)

8, 20, 17, 21, 24, 27, 28, 29, 38, 51 (Increment pass; still unsorted)

6th pass 8, 20, 17, 21, 24, 27, 28, 29, 38, 51 (Increment index)

8, 20, 17, 21, 24, 27, 28, 29, 38, 51 (Swap 20 and 17; increment index)

8, 17, 20, 21, 24, 27, 28, 29, 38, 51 (Increment index)

8, 17, 20, 21, 24, 27, 28, 29, 38, 51 (Increment index)

8, 17, 20, 21, 24, 27, 28, 29, 38, 51 (Increment pass; still unsorted)

7th pass: 8, 17, 20, 21, 24, 27, 28, 29, 38, 51 (Increment index)

8, 17, 20, 21, 24, 27, 28, 29, 38, 51 (Increment index)

8, 17, 20, 21, 24, 27, 28, 29, 38, 51 (Increment index)

Sorted Array: 8, 17, 20, 21, 24, 27, 28, 29, 38, 51 (Since no swapping is happened, sorted)

Question 2-

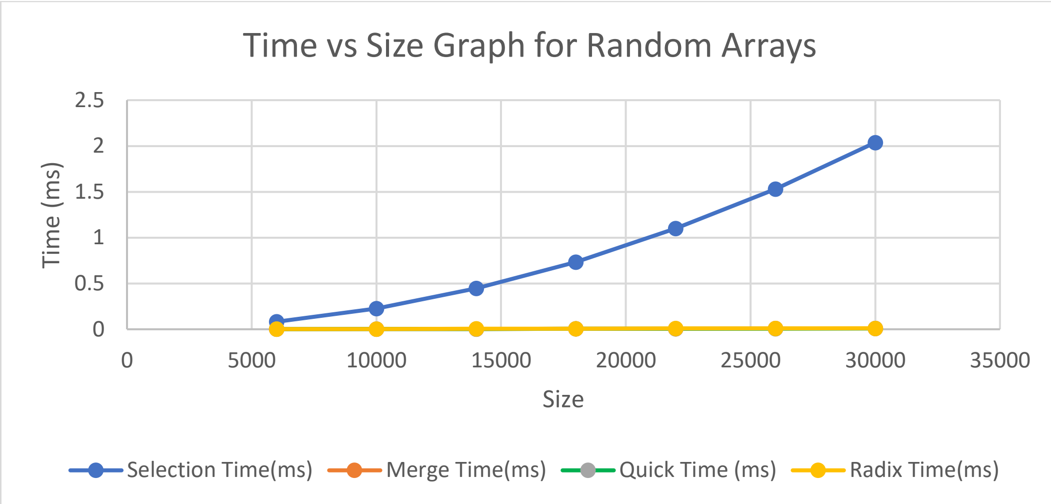
```
Selection Sort:
Number of key comparisons: 120
Number of data moves: 45
[ 3    5    6    7    8    9    11    12    12    14    14    17    18    19    20    21    ]
Merge Sort:
Number of key comparisons: 46
Number of data moves: 128
[ 3    5    6    7    8    9    11    12    12    14    14    17    18    19    20    21    ]
Quick Sort:
Number of key comparisons: 45
Number of data moves: 102
[ 3    5    6    7    8    9    11    12    12    14    14    17    18    19    20    21    ]
Radix Sort:
[ 3    5    6    7    8    9    11    12    12    14    14    17    18    19    20    21    ]
```

```
Random numbers
-------------------------------------------------------
Analysis of Selection Sort
Array Size   Elapsed time      compCount      moveCount
6000      0.0823695 ms      17997000         17997
10000       0.227535 ms      49995000          29997
14000       0.444914 ms      97993000          41997
18000       0.734621 ms     161991000          53997
22000        1.09656 ms     241989000          65997
26000        1.53086 ms     337987000          77997
30000        2.03777 ms     449985000          89997
-------------------------------------------------------
Analysis of Merge Sort
Array Size   Elapsed time      compCount      moveCount
6000     0.00168932 ms         67937         151616
10000      0.00292439 ms        120530          267232
14000       0.0042005 ms        175442          387232
18000      0.00554928 ms        231913          510464
22000      0.00693143 ms        289967          638464
26000      0.00826858 ms        349065          766464
30000      0.00967875 ms        408738          894464
-------------------------------------------------------
Analysis of Quick Sort
Array Size   Elapsed time      compCount      moveCount
6000     0.00151016 ms         88420         143487
10000      0.00262434 ms        152042          252252
14000      0.00389301 ms        228545          400448
18000      0.00496117 ms        295369          470188
22000      0.00626315 ms        359793          609209
26000      0.00750323 ms        428422          741152
30000      0.00892066 ms        545397          903365
-------------------------------------------------------
Analysis of Radix Sort
Array Size   Elapsed time
6000       0.0022997 ms
10000      0.00381524 ms
14000      0.00532546 ms
18000      0.00685549 ms
22000      0.00842575 ms
26000      0.00991601 ms
30000       0.0114758 ms
****************************
Ascending numbers
-------------------------------------------------------
```

```
****************************
Ascending numbers
-------------------------------------------------------
Analysis of Selection Sort
Array Size   Elapsed time      compCount      moveCount
6000       0.106849 ms      17997000         17997
10000       0.305934 ms      49995000         29997
14000       0.614448 ms      97993000         41997
18000         1.0521 ms     161991000         53997
22000        1.63075 ms     241989000         65997
26000        2.38738 ms     337987000         77997
30000        3.33232 ms     449985000         89997
-------------------------------------------------------
Analysis of Merge Sort
Array Size   Elapsed time      compCount      moveCount
6000    0.000981936 ms         39216         151616
10000     0.00171879 ms         69233         267232
14000     0.00247432 ms         99740         387232
18000     0.00328243 ms        131325         510464
22000     0.00406959 ms        165888         638464
26000     0.00485235 ms        198194         766464
30000     0.00568286 ms        229642         894464
-------------------------------------------------------
Analysis of Quick Sort
Array Size   Elapsed time      compCount      moveCount
6000      0.0765316 ms      17997000         23996
10000       0.212466 ms      49995000         39996
14000       0.416633 ms      97993000         55996
18000       0.688839 ms     161991000         71996
22000        1.02898 ms     241989000         87996
26000        1.43713 ms     337987000        103996
30000        1.91337 ms     449985000        119996
-------------------------------------------------------
Analysis of Radix Sort
Array Size   Elapsed time
6000     0.00226691 ms
10000     0.00378693 ms
14000     0.00528943 ms
18000     0.00680855 ms
22000     0.00833408 ms
26000      0.0098559 ms
30000      0.0114265 ms
****************************
Descending numbers
```

```
Descending numbers
-------------------------------------------------------
Analysis of Selection Sort
Array Size   Elapsed time      compCount      moveCount
6000       0.093594 ms       17997000         17997
10000       0.263357 ms       49995000         29997
14000       0.516373 ms       97993000         41997
18000       0.868021 ms      161991000         53997
22000        1.31474 ms      241989000         65997
26000        1.86038 ms      337987000         77997
30000        2.53335 ms      449985000         89997
-------------------------------------------------------
Analysis of Merge Sort
Array Size   Elapsed time      compCount      moveCount
6000    0.000970478 ms          36656         151616
10000     0.00168374 ms          64608         267232
14000     0.00242785 ms          94256         387232
18000     0.00320875 ms         124640         510464
22000     0.00401121 ms         154208         638464
26000     0.00475445 ms         186160         766464
30000     0.00554483 ms         219504         894464
-------------------------------------------------------
Analysis of Quick Sort
Array Size   Elapsed time      compCount      moveCount
6000       0.151437 ms       17716652       26602904
10000       0.416652 ms       48759940       73185871
14000       0.808735 ms       94539588      141872871
18000        1.32608 ms      155042383      232644288
22000        1.96359 ms      229427768      344239068
26000        2.71142 ms      316901431      475465727
30000        3.56769 ms      417053491      625709509
-------------------------------------------------------
Analysis of Radix Sort
Array Size   Elapsed time
6000     0.00227071 ms
10000     0.00378364 ms
14000     0.00529326 ms
18000     0.00680967 ms
22000     0.00833337 ms
26000     0.00985568 ms
30000      0.0114226 ms
[ilke.kas@dijkstra HW1]$
```

Question 3- **REPORT**



As we can see in the graph above, selection sort takes more time than other algorithms. Its graph seems as quadratic and this fits well to the theoretical results that implies that selection sort is $O(n^2)$. However, in the graph merge sort, quick sort and radix sort are close to each other, in according to results quick sort takes less time than the others. However, in theory, radix sort should take less time than quick sort and merge sort. The reason why quick sort takes less time than others is quick sort is faster than merge sort when the data is stored in memory, even in theory their best case is same and O(nlogn). Empirical relationship between radix and quick sort differs from theoretical one. In theory, radix sort should take less time than quick sort until the values of k in O(2*k*n) is less than O(nlogn). In this situation, since random generator can generate max number as 32767. So, k is equal 5 at most. On the other hand, logn is at least 12 when input is 6000. In this situation, radix sort should be less than quick sort but we get opposite as the result. This can be related to the randomness of the dataset. When we look at the ordered arrays, ascending and descending, we can easily notice that quicksort takes much more time in these cases and elapsed time is nearly the same as selection sort. Also, in theory, worst case of quick sort is given as $O(n^2)$ because in these situations, quick sort partition off two array whose size are 0 and n-1. In each partition, it divides to arrays n, n-1, n-2…..1. On the other hand, there are slight changes in elapsed times of merge, radix and selection sorts compared to quick sort. This is because, as in theory, their worst cases have the same time complexity with their average cases. When we look at the comparison counts, in each three cases, selection sort's comparison counts are greater than merge and radix because it has quadratic time complexity. In ascending and descending cases, quick sort's comparison counts are also quadratic. Looking to data moves, we can say that merge sort has disadvantage over quick because there is additional tempArray. However, selection sort again has the biggest value in data moves with complexity O(3n). To sum up, Quick sort has the biggest advantages in unordered arrays but comes up with the disadvantage in already sorted arrays. Selection sort would not be a good choice for arrays. Merge sort has stable and better time complexity but needs additional memory. Finally, radix sort has the best complexity but is only useful for fix length elements and this make it rarely used.