Part 1:

Bubble Sort:

Start: [18, 4, 47, 24, 15, 24, 17, 11, 31, 23]

1st run:

Step 1: [4, 18, 47, 24, 15, 24, 17, 11, 31, 23]

Step 2: [4, 18, 47, 24, 15, 24, 17, 11, 31, 23]

Step 3: [4, 18, 24, 47, 15, 24, 17, 11, 31, 23]

Step 4: [4, 18, 24, 15, 47, 24, 17, 11, 31, 23]

Step 5: [4, 18, 24, 15, 24, 47, 17, 11, 31, 23]

Step 6: [4, 18, 24, 15, 24, 17, 47, 11, 31, 23]

Step 7: [4, 18, 24, 15, 24, 17, 11, 47, 31, 23]

Step 8: [4, 18, 24, 15, 24, 17, 11, 31, 47, 23]

Step 9: [4, 18, 24, 15, 24, 17, 11, 31, 23, 47]

2nd run:

Step 1: [4, 18, 24, 15, 24, 17, 11, 31, 23, 47]

Step 2: [4, 18, 24, 15, 24, 17, 11, 31, 23, 47]

Step 3: [4, 18, 15, 24, 24, 17, 11, 31, 23, 47]

Step 4: [4, 18, 15, 24, 24, 17, 11, 31, 23, 47]

Step 5: [4, 18, 15, 24, 17, 24, 11, 31, 23, 47]

Step 6: [4, 18, 15, 24, 17, 11, 24, 31, 23, 47]

Step 7: [4, 18, 15, 24, 17, 11, 24, 31, 23, 47]

Step 8: [4, 18, 15, 24, 17, 11, 24, 23, 31, 47]

Step 9: [4, 18, 15, 24, 17, 11, 24, 23, 31, 47]

3rd run:

Step 1: [4, 18, 15, 24, 17, 11, 24, 23, 31, 47]

Step 2: [4, 15, 18, 24, 17, 11, 24, 23, 31, 47]

Step 3: [4, 15, 18, 24, 17, 11, 24, 23, 31, 47]

Step 4: [4, 15, 18, 17, 24, 11, 24, 23, 31, 47]

Step 5: [4, 15, 18, 17, 11, 24, 24, 23, 31, 47]

Step 6: [4, 15, 18, 17, 11, 24, 24, 23, 31, 47]

Step 7: [4, 15, 18, 17, 11, 24, 23, 24, 31, 47]

Step 8: [4, 15, 18, 17, 11, 24, 23, 24, 31, 47]

Step 9: [4, 15, 18, 17, 11, 24, 23, 24, 31, 47]

4th run:

Step 1: [4, 15, 18, 17, 11, 24, 23, 24, 31, 47]

Step 2: [4, 15, 18, 17, 11, 24, 23, 24, 31, 47]

Step 3: [4, 15, 17, 18, 11, 24, 23, 24, 31, 47]

Step 4: [4, 15, 17, 11, 18, 24, 23, 24, 31, 47]

Step 5: [4, 15, 17, 11, 18, 24, 23, 24, 31, 47]

Step 6: [4, 15, 17, 11, 18, 23, 24, 24, 31, 47]

Step 7: [4, 15, 17, 11, 18, 23, 24, 24, 31, 47]

Step 8: [4, 15, 17, 11, 18, 23, 24, 24, 31, 47]

Step 9: [4, 15, 17, 11, 18, 23, 24, 24, 31, 47]

5th run:

Step 1: [4, 15, 17, 11, 18, 23, 24, 24, 31, 47]

Step 2: [4, 15, 17, 11, 18, 23, 24, 24, 31, 47]

Step 3: [4, 15, 11, 17, 18, 23, 24, 24, 31, 47]

Step 4: [4, 15, 11, 17, 18, 23, 24, 24, 31, 47]

Step 5: [4, 15, 11, 17, 18, 23, 24, 24, 31, 47]

Step 6: [4, 15, 11, 17, 18, 23, 24, 24, 31, 47]

Step 7: [4, 15, 11, 17, 18, 23, 24, 24, 31, 47]

Step 8: [4, 15, 11, 17, 18, 23, 24, 24, 31, 47]

Step 9: [4, 15, 11, 17, 18, 23, 24, 24, 31, 47]

6th run:

Step 1: [4, 15, 11, 17, 18, 23, 24, 24, 31, 47]

Step 2: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 3: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 4: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 5: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 6: [4, 11, 15, 17, 18, 23, 24, 31, 47]

Step 7: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 8: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 9: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

7th run:

Step 1: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 2: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 3: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 4: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 5: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 6: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 7: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 8: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 9: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

End Result: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Selection Sort:

Start: [18, 4, 47, 24, 15, 24, 17, 11, 31, 23]

Step 1: [4, 18, 47, 24, 15, 24, 17, 11, 31, 23]

Step 2: [4, 11, 47, 24, 15, 24, 17, 18, 31, 23]

Step 3: [4, 11, 15, 24, 47, 24, 17, 18, 31, 23]

Step 4: [4, 11, 15, 17, 47, 24, 24, 18, 31, 23]

Step 5: [4, 11, 15, 17, 18, 24, 24, 47, 31, 23]

Step 6: [4, 11, 15, 17, 18, 23, 24, 47, 31, 24]

Step 7: [4, 11, 15, 17, 18, 23, 24, 47, 31, 24]

Step 8: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Step 9: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

End Result: [4, 11, 15, 17, 18, 23, 24, 24, 31, 47]

Part 2:

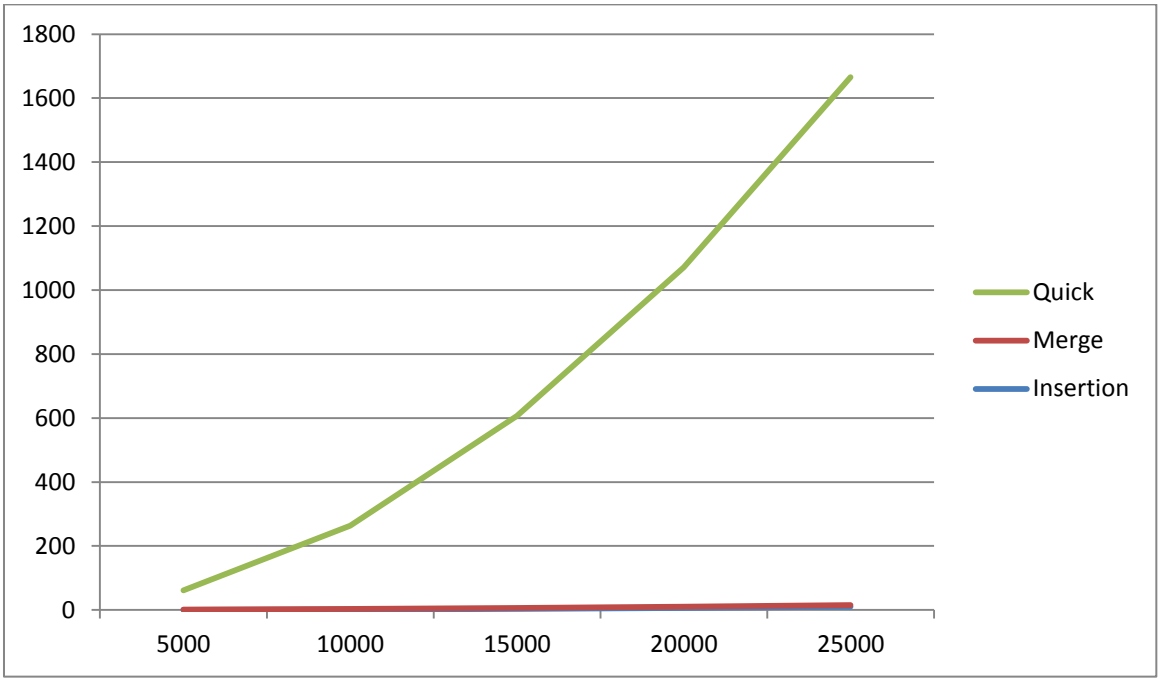
```
[betul.ciftci@dijkstra defnefinal]$ ./hw1
Insertion sort:
0       2       3       5       6       7       8       9       9       11      1
1       14      15      16      17      18
Number of key comparisons: 71
Number of move counts: 59
Merge sort:
0       2       3       5       6       7       8       9       9       11      1
1       14      15      16      17      18
Number of key comparisons: 64
Number of move counts: 32
Quick sort:
0       2       3       5       6       7       8       9       9       11      1
1       14      15      16      17      18
Number of key comparisons: 120
Number of move counts: 15
Random arrays:
-----------------------------------------------------
Part c -Time analysis of InsertionSort
Array Size      Time Elapsed    compCount       moveCount
5000            0.4ms           6741323         6241431
10000           1.6ms           25746364                24746470
15000           3.6ms           57568307                56068415
20000           6.3ms           102533056               100533173
```

Cont.on next page

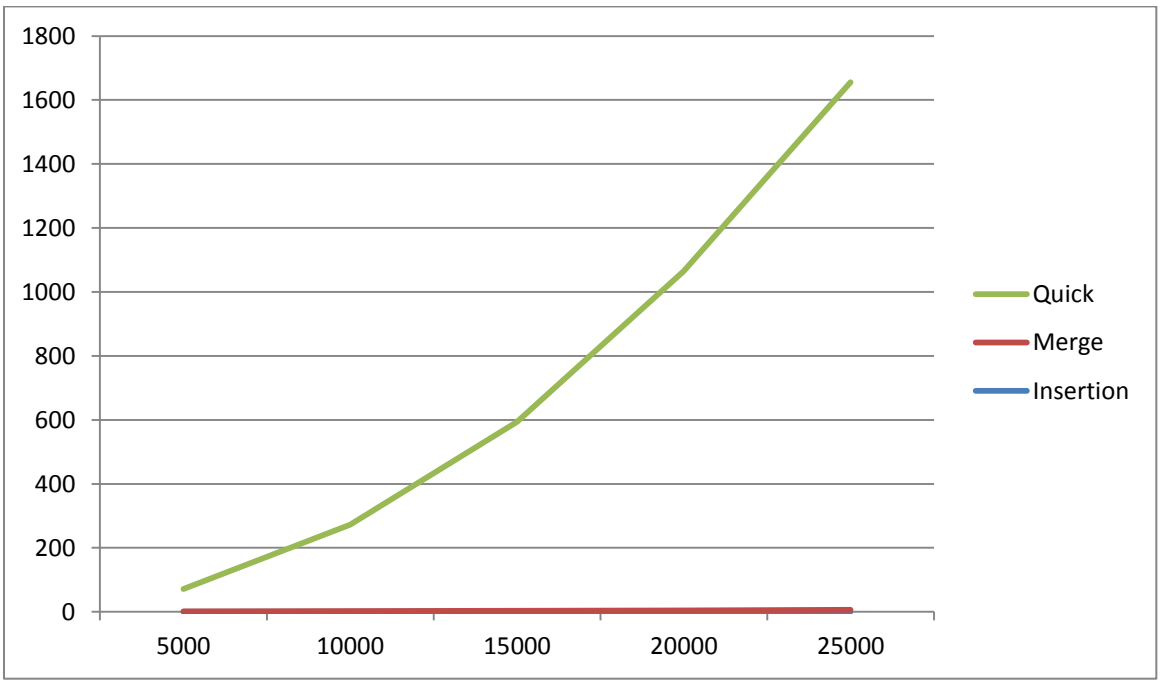
```
20000           6.3ms           102533056               100533173
25000           9.8ms           158664069               156164177
Part c - Time analysis of Merge Sort
Array Size      Time Elapsed    compCount       moveCount
5000            1ms             6180800         3200400
10000           2.1ms           13361600                6900800
15000           3.1ms           20861600                10636400
20000           4.3ms           28723200                14801600
25000           5.5ms           36723200                18847600
Part c - Time analysis of Quick Sort
Array Size      Time Elapsed    compCount       moveCount
5000            60ms            12497500                4999
10000           260ms           49995000                9999
15000           600ms           112492500               14999
20000           1060ms          199990000               19999
25000           1650ms          312487500               24999
Already sorted arrays:
------------------------------------------------------
Part c -Time analysis of InsertionSort
Array Size      Time Elapsed    compCount       moveCount
5000            0.1ms           499900          0
10000           0.1ms           999900          0
15000           0.1ms           1499900         0
20000           0.2ms           1999900         0
25000           0.2ms           2499900         0
Part c - Time analysis of Merge Sort
Array Size      Time Elapsed    compCount       moveCount
5000            1ms             6180800         3200400
10000           2ms             13361600                6900800
15000           3.2ms           20861600                10636400
20000           4.3ms           28723200                14801600
25000           5.6ms           36723200                18847600
Part c - Time analysis of Quick Sort
Array Size      Time Elapsed    compCount       moveCount
5000            70ms            12497500                4999
10000           270ms           49995000                9999
15000           590ms           112492500               14999
20000           1060ms          199990000               19999
25000           1650ms          312487500               24999
Nearly sorted arrays:
------------------------------------------------------
Part c -Time analysis of InsertionSort
Array Size      Time Elapsed    compCount       moveCount
5000            0.3ms           4248595         3748703
10000           1ms             15908833                14908939
15000           2.1ms           34986965                33487071
20000           3.8ms           61607986                59608091
25000           5.9ms           95908029                93408136
Part c - Time analysis of Merge Sort
Array Size      Time Elapsed    compCount       moveCount
5000            1ms             6180800         3200400
10000           2.1ms           13361600                6900800
15000           3.2ms           20861600                10636400
20000           4.4ms           28723200                14801600
25000           5.6ms           36723200                18847600
Part c - Time analysis of Quick Sort
Array Size      Time Elapsed    compCount       moveCount
5000            60ms            12497500                4999
10000           260ms           49995000                9999
15000           600ms           112492500               14999
20000           1060ms          199990000               19999
25000           1650ms          312487500               24999
[betul.ciftci@dijkstra defnefinal]$
```

Part 3:

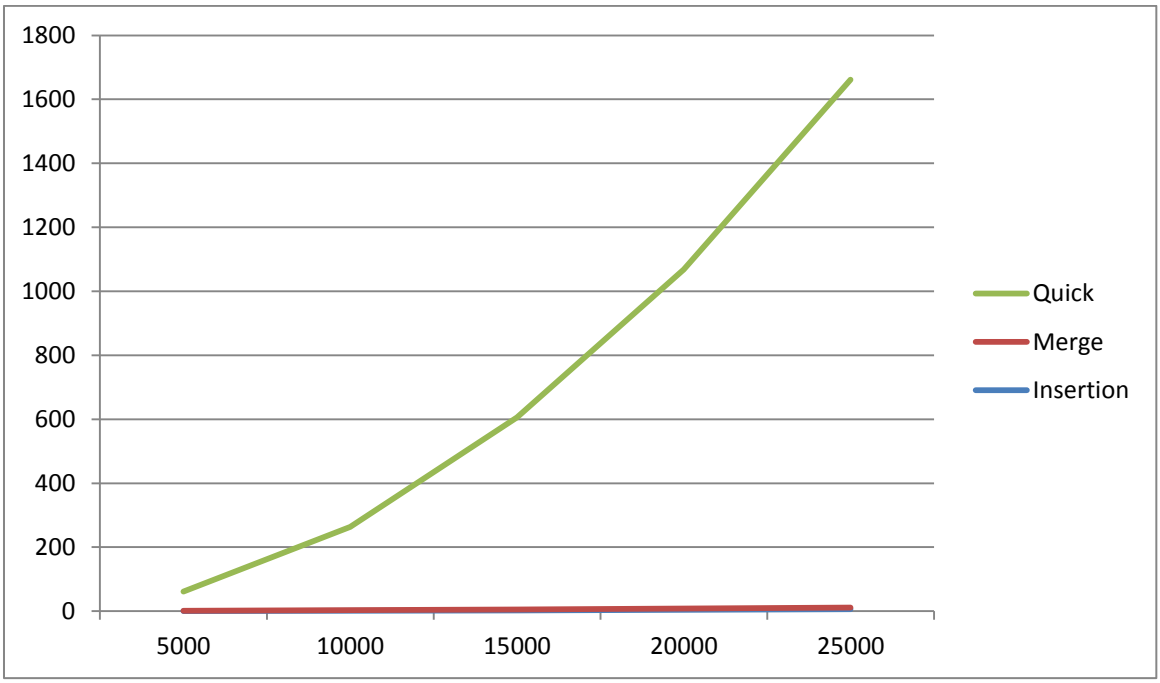
Random arrays:



Already sorted arrays:



Nearly sorted arrays:


1800
1600
1400
1200
1000
800
600
400
200
0
5000
10000
15000
20000
25000
Quick
Merge
Insertion

Part 4:

During my experiment of it, I have set K as the half of the size – for this value of K, the results did not seem to change significantly. Had I tried other K's, not necessarily close to the middle, the results would still not have changed in a significant way, although it had to not increase logarithmically in some cases even though my experiment results show otherwise.

Merge sort seems the most efficient when a randomly sorted array is used and insertion sort seems the most efficient when an already sorted array is used