CS202
Title: Sorting and Algorithm Efficiency
Author: Deniz Semih Özal
ID: 21802414
Assignment: 1
Description: Comparing Empirical and Theoretical Results of Different Sorting Algorithms

Question 1

Selection Sort:

Inital Array: 6 1 5 3 7 2 8 4

1th pass:

swap 4 , 8

6 1 5 3 7 2 4 |8

2th pass:

swap 4 , 7

6 1 5 3 4 2 |7 8

3th pass:

swap 2 , 6

2 1 5 3 4 |6 7 8

4th pass:

swap 4 , 5

2 1 4 3 |5 6 7 8

5th pass:

swap 3 , 4

2 1 3 |4 5 6 7 8

6th pass:

swap 3 , 3

2 1 |3 4 5 6 7 8

7th pass:

swap 1 , 2

1 |2 3 4 5 6 7 8

Key Comparisons: 28

Data Moves: 21

Insertion Sort:

Inital Array: 6 1 5 3 7 2 8 4

1th pass: 1 |6 5 3 7 2 8 4

2th pass: 1 5| 6 3 7 2 8 4

3th pass: 1 3 5| 6 7 2 8 4

4th pass: 1 3 5 6| 7 2 8 4

5th pass: 1 2 3 5 6 |7 8 4

6th pass: 1 2 3 5 6 7| 8 4

7th pass: 1 2 3 4 5 6 7| 8

Key Comparisons: 19

Data Moves: 26

Bubble Sort:

Inital Array: 6 1 5 3 7 2 8 4

1th pass:

swap 6 , 1

1 6 5 3 7 2 8 4

swap 6 , 5

1 5 6 3 7 2 8 4

swap 6 , 3

1 5 3 6 7 2 8 4

swap 7 , 2

1 5 3 6 2 7 8 4

swap 8 , 4

1 5 3 6 2 7 4 8

2th pass:

swap 5 , 3

1 3 5 6 2 7 4 8

swap 6 , 2

1 3 5 2 6 7 4 8

swap 7 , 4

1 3 5 2 6 4 7 8

3th pass:

swap 5 , 2

1 3 2 5 6 4 7 8

swap 6 , 4

1 3 2 5 4 6 7 8

4th pass:

swap 3 , 2

1 2 3 5 4 6 7 8

swap 5 , 4

1 2 3 4 5 6 7 8

5th pass:

1 2 3 4 5 6 7 8

Merge Sort:

Inital Array: 6 1 5 3 7 2 8 4

MergeSort called

First portion

6 1 5 3

Second portion

7 2 8 4

MergeSort called

First portion

6 1

Second portion

5 3

MergeSort called

First portion

6

Second portion

1

6

6 1

Merge called

First portion

1

Second portion

6

1 6

MergeSort called

First portion

5

Second portion

3

1 6 5

1 6 5 3

Merge called

First portion

3

Second portion

5

1 6 3 5

Merge called

First portion

1 3

Second portion

5 6

1 3 5 6

MergeSort called

First portion

7 2

Second portion

8 4

MergeSort called

First portion

7

Second portion

2

1 3 5 6 7

1 3 5 6 7 2

Merge called

First portion

2

Second portion

7

1 3 5 6 2 7

MergeSort called

First portion

8

Second portion

4

1 3 5 6 2 7 8

1 3 5 6 2 7 8 4

Merge called

First portion

4

Second portion

8

1 3 5 6 2 7 4 8

Merge called

First portion

2 4

Second portion

7 8

1 3 5 6 2 4 7 8

Merge called

First portion

1 2 3 4

Second portion

5 6 7 8

1 2 3 4 5 6 7 8

Key Comparisons: 16

Data Moves: 48

Quick Sort:

Inital Array: 6 1 5 3 7 2 8 4

Partition 1

6 as pivot

Checking if 1 < 6 (pivot)

Checking if 5 < 6 (pivot)

Checking if 3 < 6 (pivot)

Checking if 7 < 6 (pivot)

Checking if 2 < 6 (pivot)

Swap (2,7)

6 1 5 3 2 7 8 4

Checking if 8 < 6 (pivot)

Checking if 4 < 6 (ivot)

6 1 5 3 2 4 8 7

Swap (6,4)

4 1 5 3 2 6 8 7

4 as pivot

Checking if 1 < 4

Checking if 5 < 4

Checking if 3 < 4

Swap (3,5)

4 1 3 5 2 6 8 7

Checking if 2 < 4

Swap (2,5)

4 1 3 2 5 6 8 7

Swap (2,4)

2 1 3 4 5 6 8 7

Partition 3

2 as pivot

Checking if 1 < 2

Checking if 3 < 2

Swap (1,2)

1 2 3 4 5 6 8 7

Selecting 8 as pivot

Checking if 7 < 8

Swap (7,8)

1 2 3 4 5 6 7 8

Question 2

Output in Dijkstra

Sorting on Random Arrays

Analysis of Insertion Sort

Array Size	Time Elapsed	countComp	countMove
5000	39.7417	6235702	6240701
10000	160.708	25103015	25113014
15000	357.831	56068725	56083724
20000	638.308	100093616	100113615
25000	991.085	155658723	155683722
30000	1423.08	223610730	223640729
35000	1954.83	307218965	307253964
40000	2543.71	399797892	399837891

Analysis of Merge Sort

Array Size	Time Elapsed	countComp	countMove
5000	1.81195	55259	123616
10000	3.76633	120454	267232
15000	6.01854	189263	417232
20000	8.15281	260876	574464
25000	10.37	334116	734464
30000	12.6657	408650	894464
35000	14.5552	484406	1058928
40000	16.8962	561938	1228928

Analysis of Quick Sort

Array Size Time Elapsed countComp countMove

5000	1.19592	66414	119349
10000	2.61497	154480	249224
15000	4.00936	236334	390377
20000	5.59431	347663	565425
25000	6.94984	424185	650555
30000	8.88886	536588	944008
35000	10.3557	632630	1027266
40000	11.7163	716116	1157486

Sorting on Ascending Arrays

Analysis of Insertion Sort

Array Size	Time Elapsed	countComp	countMove
5000	81.0274	12502499	6240701
10000	319.801	50004999	50014998
15000	721.981	112507499	112522498
20000	1275.25	200009999	200029998
25000	1999.04	312512499	312537498
30000	2864.22	450014999	450044998
35000	3903.32	612517499	612552498
40000	5112.17	800019999	800059998

Analysis of Merge Sort

Array Size	Time Elapsed	countComp	countMove
5000	1.1795	29804	123616
10000	2.44087	64608	267232
15000	3.89338	102252	417232
20000	5.30756	139216	574464

25000 6.71511 178756 734464

30000 8.12256 219504 894464

35000 9.3512 260100 1058928

40000 10.8148 298432 1228928

Analysis of Quick Sort

Array Size	Time Elapsed	countComp	countMove
5000	107.12	12497500	18769996
10000	426.053	49995000	75039996
15000	961.115	112492500	168809996
20000	1703.88	199990000	300079996
25000	2670.63	312487500	468849996

Sorting on Descending Arrays

Analysis of Insertion Sort

Array Size	Time Elapsed	countComp	countMove
5000	0.049233	4999	6240701
10000	0.096304	9999	19998
15000	0.146081	14999	29998
20000	0.193313	19999	39998
25000	0.239998	24999	49998
30000	0.291311	29999	59998
35000	0.335511	34999	69998
40000	0.388189	39999	79998

Analysis of Merge Sort

Array Size	Time Elapsed	countComp	countMove

5000	1.18923	32004	123616
10000	2.4523	69008	267232
15000	3.90375	106364	417232
20000	5.33883	148016	574464
25000	6.74078	188476	734464
30000	8.16517	227728	894464
35000	9.38636	269364	1058928
40000	10.8665	316032	1228928

Analysis of Quick Sort

Array Size	Time Elapsed	countComp	countMove
5000	52.9152	12497500	19996
10000	211.594	49995000	39996
15000	475.346	112492500	59996
20000	850.368	199990000	79996
25000	1320.85	312487500	99996

ScreenShot from Dijkstra

semih.ozal@dijkstra:~/CS202/HW1

```
[semih.ozal@dijkstra HW1]$ ./hw1
           Sorting on Random Arrays
--------------------------------------------------
Analysis of Insertion Sort
Array Size      Time Elapsed    countComp       countMove
5000            39.7417         6235702         6240701
10000           160.708         25103015        25113014
15000           357.831         56068725        56083724
20000           638.308         100093616       100113615
25000           991.085         155658723       155683722
30000           1423.08         223610730       223640729
35000           1954.83         307218965       307253964
40000           2543.71         399797892       399837891
--------------------------------------------------
Analysis of Merge Sort
Array Size      Time Elapsed    countComp       countMove
5000            1.81195         55259           123616
10000           3.76633         120454          267232
15000           6.01854         189263          417232
20000           8.15281         260876          574464
25000           10.37           334116          734464
30000           12.6657         408650          894464
35000           14.5552         484406          1058928
40000           16.8962         561938          1228928
--------------------------------------------------
Analysis of Quick Sort
Array Size      Time Elapsed    countComp       countMove
5000            1.19592         66414           119349
10000           2.61497         154480          249224
15000           4.00936         236334          390377
20000           5.59431         347663          565425
25000           6.94984         424185          650555
30000           8.88886         536588          944008
35000           10.3557         632630          1027266
40000           11.7163         716116          1157486
--------------------------------------------------
--------------------------------------------------
           Sorting on Ascending Arrays
--------------------------------------------------
Analysis of Insertion Sort
Array Size      Time Elapsed    countComp       countMove
5000            81.0274         12502499        6240701
10000           319.801         50004999        50014998
15000           721.981         112507499       112522498
20000           1275.25         200009999       200029998
25000           1999.04         312512499       312537498
30000           2864.22         450014999       450044998
35000           3903.32         612517499       612552498
40000           5112.17         800019999       800059998
--------------------------------------------------
```

semih.ozal@dijkstra:~/CS202/HW1

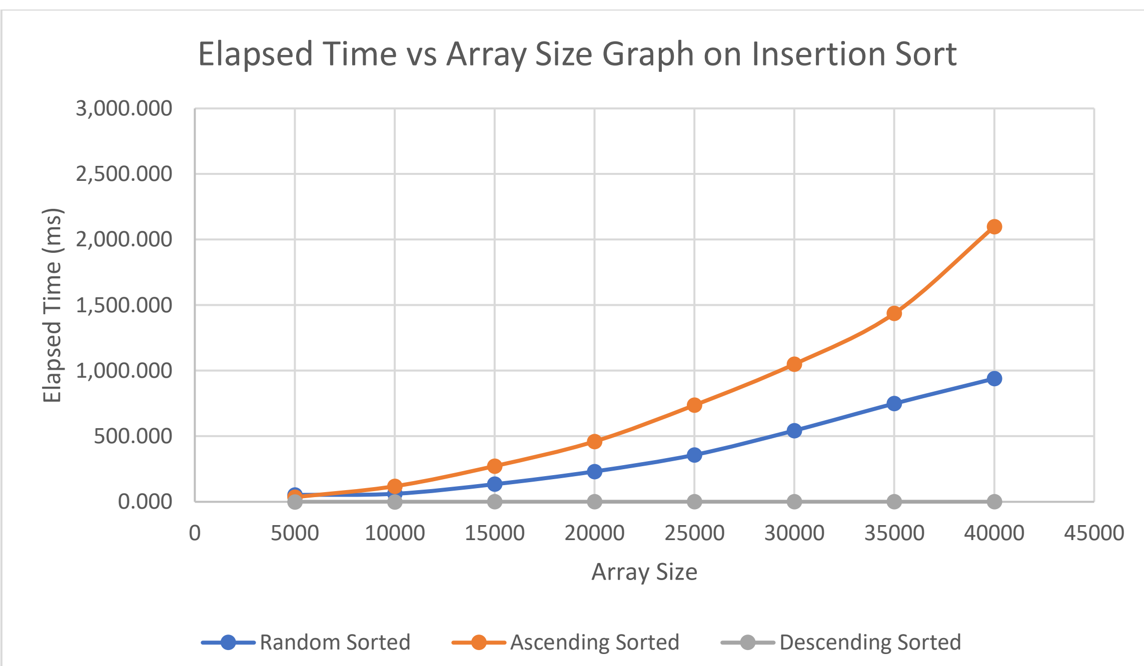
```
--------------------------------------------------
--------------------------------------------------
           Sorting on Ascending Arrays
--------------------------------------------------
Analysis of Insertion Sort
Array Size      Time Elapsed    countComp       countMove
5000            81.0274         12502499        6240701
10000           319.801         50004999        50014998
15000           721.981         112507499       112522498
20000           1275.25         200009999       200029998
25000           1999.04         312512499       312537498
30000           2864.22         450014999       450044998
35000           3903.32         612517499       612552498
40000           5112.17         800019999       800059998
--------------------------------------------------
Analysis of Merge Sort
Array Size      Time Elapsed    countComp       countMove
5000            1.1795          29804           123616
10000           2.44087         64608           267232
15000           3.89338         102252          417232
20000           5.30756         139216          574464
25000           6.71511         178756          734464
30000           8.12256         219504          894464
35000           9.3512          260100          1058928
40000           10.8148         298432          1228928
--------------------------------------------------
Analysis of Quick Sort
Array Size      Time Elapsed    countComp       countMove
5000            107.12          12497500        18769996
10000           426.053         49995000        75039996
15000           961.115         112492500       168809996
20000           1703.88         199990000       300079996
25000           2670.63         312487500       468849996
--------------------------------------------------
--------------------------------------------------
           Sorting on Descending Arrays
--------------------------------------------------
Analysis of Insertion Sort
Array Size      Time Elapsed    countComp       countMove
5000            0.049233        4999            6240701
10000           0.096304        9999            19998
15000           0.146081        14999           29998
20000           0.193313        19999           39998
25000           0.239998        24999           49998
30000           0.291311        29999           59998
35000           0.335511        34999           69998
40000           0.388189        39999           79998
--------------------------------------------------
```

semih.ozal@dijkstra:~/CS202/HW1

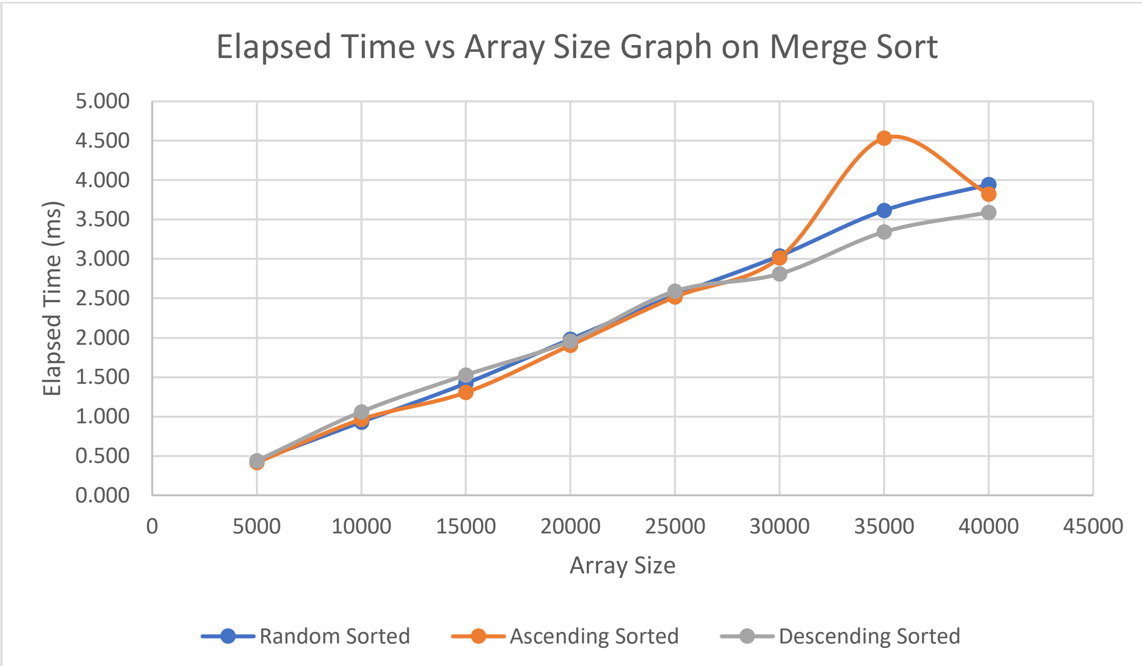
```
5000            1.1795          29804           123616
10000           2.44087         64608           267232
15000           3.89338         102252          417232
20000           5.30756         139216          574464
25000           6.71511         178756          734464
30000           8.12256         219504          894464
35000           9.3512          260100          1058928
40000           10.8148         298432          1228928
--------------------------------------------------
Analysis of Quick Sort
Array Size      Time Elapsed    countComp       countMove
5000            107.12          12497500        18769996
10000           426.053         49995000        75039996
15000           961.115         112492500       168809996
20000           1703.88         199990000       300079996
25000           2670.63         312487500       468849996
--------------------------------------------------
--------------------------------------------------
           Sorting on Descending Arrays
--------------------------------------------------
Analysis of Insertion Sort
Array Size      Time Elapsed    countComp       countMove
5000            0.049233        4999            6240701
10000           0.096304        9999            19998
15000           0.146081        14999           29998
20000           0.193313        19999           39998
25000           0.239998        24999           49998
30000           0.291311        29999           59998
35000           0.335511        34999           69998
40000           0.388189        39999           79998
--------------------------------------------------
Analysis of Merge Sort
Array Size      Time Elapsed     countComp      countMove
5000            1.18923         32004           123616
10000           2.4523          69008           267232
15000           3.90375         106364          417232
20000           5.33883         148016          574464
25000           6.74078         188476          734464
30000           8.16517         227728          894464
35000           9.38636         269364          1058928
40000           10.8665         316032          1228928
--------------------------------------------------
Analysis of Quick Sort
Array Size      Time Elapsed    countComp       countMove
5000            52.9152         12497500        19996
10000           211.594         49995000        39996
15000           475.346         112492500       59996
20000           850.368         199990000       79996
25000           1320.85         312487500       99996
[semih.ozal@dijkstra HW1]$
```

Question3

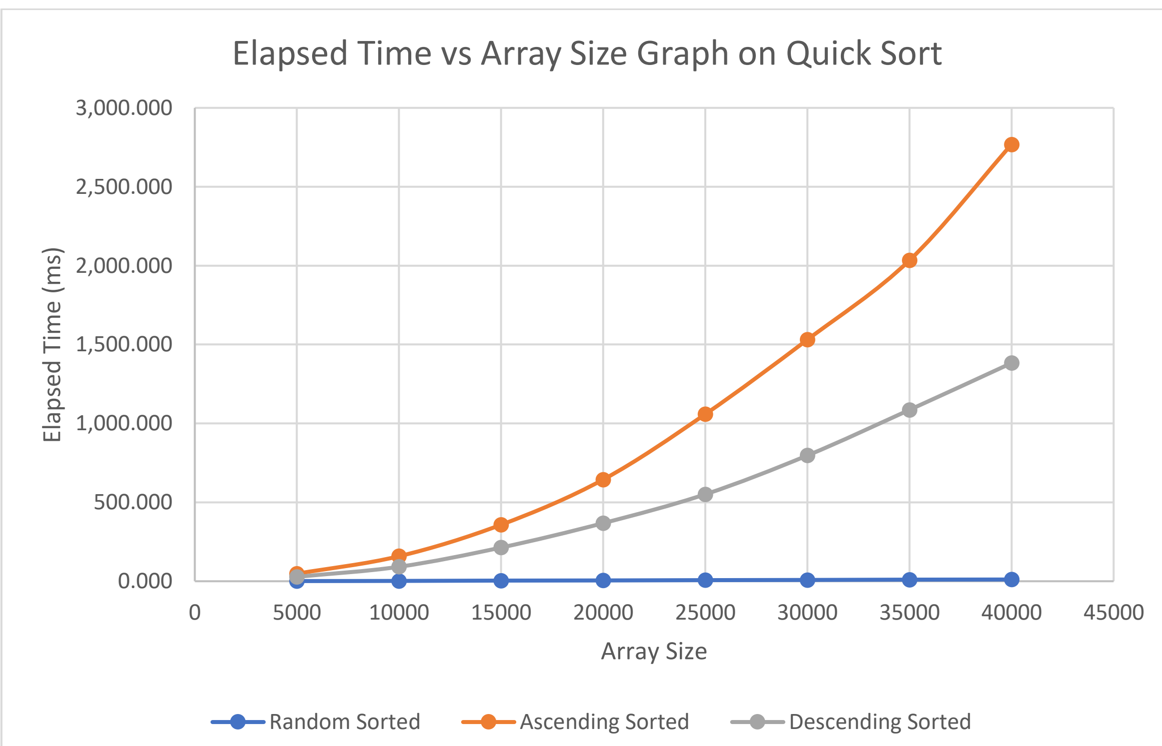
A) Elapsed Time vs Array Size Graph on Insertion Sort



B) Elapsed Time vs Array Size Graph on Merge Sort



C) Elapsed Time vs Array Size Graph on Quick Sort



Analysis:

Insertion Sort Analysis:

As it can be seen from the first graph, the greatest elapsed time belongs to ascending sorted array and if we compare our empirical results to the thereotical one it is quite reasonable. According to the theoretical result, ascending order is the worst-case since our insertion sort is implemented in a descending way and its time complexity is O(n^2). Whereas the best case is when the array is already descending sorted and its time complexity is O(n). Also as we expected the random sorted array gives an averages-case result and its time complexity again O(n^2). Therefore, I can argue that thereotical and empirical results match each other. The reason of why ascending order is the worst case is that is in reverse order, also the number of moves and the number of key comparisons are both in O(n^2) manner.

Merge Sort Analysis:

As it can be seen from the second graph, all elapsed times are quite close to each other and if we compare out empirical results to the theoretical one it is quite reasonable. In Merge Sort algorithm, both worst case and average cases are in O(nlogn) manner so there are very minor differences between ascending, descending, and random sorted arrays. Merge Sort is a quite efficient algorithm if it is compared with insertion, selection or bubble sort. However, by the principle of memory-time tradeoff, this algorithm requires an extra array whose size equals to the size of the original. Therefore, for many large sizes this algorithm could produce a problem and may give an error.

Quick Sort Analysis:

As it can be seen from the third graph, the greatest elapsed times belongs to ascending sorted array and if we compare our empirical results to the theoretical one it is quite reasonable. According to the theoretical result, quicksort is slow when the array is already sorted and if we choose element as the pivot. Similarly, we choose first element as a pivot and descending array is the worst case our implementation works in a descending manner. Therefore, for the sorted arrays the complexity is O(n^2) and both ascending and desceding sorted arrays work in this time complexity. On the other hand random sorted array takes quite minor time compared to sorted arrays and its time complexity is O(nlogn). Furthermore, when I try to compile after the size exceeding 25000, the compiler gives a stack overflow error for both ascending and descending sorted arrays. By my observation, I can argue that quick sort is a quite efficient algorithm if the array is not sorted, because if it is sorted compilers can’t calculate excessively recursive and swap operations.

Note: I have added std=c++11 when I try to compile my file because chrono functions works only in this compiler version.