

CS 202
Section 1
Homework 1
Ömer Oktay Gültekin
21901413
08/03/2022

Question 1:

a) Show that $f(n) = 8n^4 + 5n^3 + 7$ is $O(n^5)$ by specifying appropriate c and n_0 values in Big-O definition

We need to find 2 positive constants c and n_0 such that:

$$0 \le 8n^4 + 5n^3 + 7 \le cn^5 \text{ for all } n \ge n_0$$

$$\frac{8}{n} + \frac{5}{n^2} + \frac{7}{n^5} \le c \text{ for all } n \ge n_0$$

Choose c = 20 and $n_0 = 1$:

$$8n^4 + 5n^3 + 7 \le 20n^5$$
 for all $n \ge 1$

Therefore
$$f(n) = 8n^4 + 5n^3 + 7 = O(n^5)$$

- **b)** Trace the following sorting algorithms to sort the array [22, 8, 49, 25, 18, 30, 20, 15, 35, 27] in ascending order. Use the array implementation of the algorithms as described in the textbook and show all major steps.
 - 1) Selection Sort

Initial Array: [22, 8, 49, 25, 18, 30, 20, 15, 35, 27]

Pass number 1:

Number of key comparisons: 9

Key comparison pairs: (8, 22), (49, 22), (25, 49), (18, 49), (30, 49), (20, 49), (15, 49), (35, 49), (27, 49)

Largest index and value: 2, 49

Last index and value: 9, 27

Swap between indexes 2 and 9: temp = 49, x = 27, y = 49

Total swap and moves so far: 1, 3

After iteration: [22, 8, 27, 25, 18, 30, 20, 15, 35, 49]

Pass number 2:

Number of key comparisons: 8

Key comparison pairs: (8, 22), (27, 22), (25, 27), (18, 27), (30, 27), (20, 30), (15, 30), (35, 30)

Largest index and value: 8, 35

Last index and value: 8, 35

Swap between indexes 8 and 8: temp = 35, x = 35, y = 35

Total swap and moves so far: 2, 6

After iteration: [22, 8, 27, 25, 18, 30, 20, 15, 35, 49]

Pass number 3:

Number of key comparisons: 7

Key comparison pairs: (8, 22), (27, 22), (25, 27), (18, 27), (30, 27), (20, 30), (15, 30)

Largest index and value: 5, 30

Last index and value: 7, 15

Swap between indexes 5 and 7: temp = 30, x = 15, y = 30

Total swap and moves so far: 3, 9

After iteration: [22, 8, 27, 25, 18, 15, 20, 30, 35, 49]

Pass number 4:

Number of key comparisons: 6

Key comparison pairs: (8, 22), (27, 22), (25, 27), (18, 27), (15, 27), (20, 27)

Largest index and value: 2, 27 Last index and value: 6, 20

Swap between indexes 2 and 6: temp = 27, x = 20, y = 27

Total swap and moves so far: 4, 12

After iteration: [22, 8, 20, 25, 18, 15, 27, 30, 35, 49]

Pass number 5:

Number of key comparisons: 5

Key comparison pairs: (8, 22), (20, 22), (25, 22), (18, 25), (15, 25)

Largest index and value: 3, 25 Last index and value: 5, 15

Swap between indexes 3 and 5: temp = 25, x = 15, y = 25

Total swap and moves so far: 5, 15

After iteration: [22, 8, 20, 15, 18, 25, 27, 30, 35, 49]

Pass number 6:

Number of key comparisons: 4

Key comparison pairs: (8, 22), (20, 22), (15, 22), (18, 22)

Largest index and value: 0, 22 Last index and value: 4, 18

Swap between indexes 0 and 4: temp = 22, x = 18, y = 22

Total swap and moves so far: 6, 18

After iteration: [18, 8, 20, 15, 22, 25, 27, 30, 35, 49]

Pass number 7:

Number of key comparisons: 3

Key comparison pairs: (8, 18), (20, 18), (15, 20)

Largest index and value: 2, 20 Last index and value: 3, 15

Swap between indexes 2 and 3: temp = 20, x = 15, y = 20

Total swap and moves so far: 7, 21

After iteration: [18, 8, 15, 20, 22, 25, 27, 30, 35, 49]

Pass number 8:

Number of key comparisons: 2

Key comparison pairs: (8, 18), (15, 18)

Largest index and value: 0, 18

```
Last index and value: 2, 15
 Swap between indexes 0 and 2: temp = 18, x = 15, y = 18
 Total swap and moves so far: 8, 24
After iteration: [15, 8, 18, 20, 22, 25, 27, 30, 35, 49]
Pass number 9:
 Number of key comparisons: 1
 Key comparison pairs: (8, 15)
 Largest index and value: 0, 15
 Last index and value: 1, 8
 Swap between indexes 0 and 1: temp = 15, x = 8, y = 15
 Total swap and moves so far: 9, 27
After iteration: [8, 15, 18, 20, 22, 25, 27, 30, 35, 49]
2) Bubble Sort
Initial Array: [ 22, 8, 49, 25, 18, 30, 20, 15, 35, 27 ]
Pass number 1:
 Number of key comparisons: 9
 Swap between indexes 0 and 1: temp = 22, x = 8, y = 22
 Total swap and moves so far: 1, 3
 After swap: [8, 22, 49, 25, 18, 30, 20, 15, 35, 27]
 Swap between indexes 2 and 3: temp = 49, x = 25, y = 49
 Total swap and moves so far: 2, 6
 After swap: [8, 22, 25, 49, 18, 30, 20, 15, 35, 27]
 Swap between indexes 3 and 4: temp = 49, x = 18, y = 49
 Total swap and moves so far: 3, 9
 After swap: [8, 22, 25, 18, 49, 30, 20, 15, 35, 27]
 Swap between indexes 4 and 5: temp = 49, x = 30, y = 49
 Total swap and moves so far: 4, 12
 After swap: [8, 22, 25, 18, 30, 49, 20, 15, 35, 27]
 Swap between indexes 5 and 6: temp = 49, x = 20, y = 49
 Total swap and moves so far: 5, 15
 After swap: [8, 22, 25, 18, 30, 20, 49, 15, 35, 27]
 Swap between indexes 6 and 7: temp = 49, x = 15, y = 49
 Total swap and moves so far: 6, 18
 After swap: [8, 22, 25, 18, 30, 20, 15, 49, 35, 27]
 Swap between indexes 7 and 8: temp = 49, x = 35, y = 49
  Total swap and moves so far: 7, 21
 After swap: [8, 22, 25, 18, 30, 20, 15, 35, 49, 27]
```

Swap between indexes 8 and 9: temp = 49, x = 27, y = 49

Total swap and moves so far: 8, 24

After swap: [8, 22, 25, 18, 30, 20, 15, 35, 27, 49]

After pass: [8, 22, 25, 18, 30, 20, 15, 35, 27, 49]

Pass number 2:

Number of key comparisons: 8

Swap between indexes 2 and 3: temp = 25, x = 18, y = 25

Total swap and moves so far: 9, 27

After swap: [8, 22, 18, 25, 30, 20, 15, 35, 27, 49]

Swap between indexes 4 and 5: temp = 30, x = 20, y = 30

Total swap and moves so far: 10, 30

After swap: [8, 22, 18, 25, 20, 30, 15, 35, 27, 49]

Swap between indexes 5 and 6: temp = 30, x = 15, y = 30

Total swap and moves so far: 11, 33

After swap: [8, 22, 18, 25, 20, 15, 30, 35, 27, 49]

Swap between indexes 7 and 8: temp = 35, x = 27, y = 35

Total swap and moves so far: 12, 36

After swap: [8, 22, 18, 25, 20, 15, 30, 27, 35, 49]

After pass: [8, 22, 18, 25, 20, 15, 30, 27, 35, 49]

Pass number 3:

Number of key comparisons: 7

Swap between indexes 1 and 2: temp = 22, x = 18, y = 22

Total swap and moves so far: 13, 39

After swap: [8, 18, 22, 25, 20, 15, 30, 27, 35, 49]

Swap between indexes 3 and 4: temp = 25, x = 20, y = 25

Total swap and moves so far: 14, 42

After swap: [8, 18, 22, 20, 25, 15, 30, 27, 35, 49]

Swap between indexes 4 and 5: temp = 25, x = 15, y = 25

Total swap and moves so far: 15, 45

After swap: [8, 18, 22, 20, 15, 25, 30, 27, 35, 49]

Swap between indexes 6 and 7: temp = 30, x = 27, y = 30

Total swap and moves so far: 16, 48

After swap: [8, 18, 22, 20, 15, 25, 27, 30, 35, 49]

After pass: [8, 18, 22, 20, 15, 25, 27, 30, 35, 49]

Pass number 4:

Number of key comparisons: 6

Swap between indexes 2 and 3: temp = 22, x = 20, y = 22

Total swap and moves so far: 17, 51

After swap: [8, 18, 20, 22, 15, 25, 27, 30, 35, 49]

Swap between indexes 3 and 4: temp = 22, x = 15, y = 22

Total swap and moves so far: 18, 54

After swap: [8, 18, 20, 15, 22, 25, 27, 30, 35, 49]

After pass: [8, 18, 20, 15, 22, 25, 27, 30, 35, 49]

Pass number 5:

Number of key comparisons: 5

Swap between indexes 2 and 3: temp = 20, x = 15, y = 20

Total swap and moves so far: 19, 57

After swap: [8, 18, 15, 20, 22, 25, 27, 30, 35, 49]

After pass: [8, 18, 15, 20, 22, 25, 27, 30, 35, 49]

Pass number 6:

Number of key comparisons: 4

Swap between indexes 1 and 2: temp = 18, x = 15, y = 18

Total swap and moves so far: 20, 60

After swap: [8, 15, 18, 20, 22, 25, 27, 30, 35, 49]

After pass: [8, 15, 18, 20, 22, 25, 27, 30, 35, 49]

Pass number 7:

Number of key comparisons: 3

After pass: [8, 15, 18, 20, 22, 25, 27, 30, 35, 49]

Question 2:

c)

```
Analysis of Insertion Sort:
Unsorted Array: [ 9, 6, 7, 16, 18, 5, 2, 12, 20, 1, 16, 17, 4, 11, 13, 8 ]
Sorted Array: [ 1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 16, 16, 17, 18, 20 ]
Key Comparisons: 69
Number of Data Moves: 88
Analysis of Bubble Sort:
Unsorted Array: [ 9, 6, 7, 16, 18, 5, 2, 12, 20, 1, 16, 17, 4, 11, 13, 8 ]
Sorted Array: [ 1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 16, 16, 17, 18, 20 ]
Key Comparisons: 110
Number of Data Moves: 174
Sorted Array: [ 1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 16, 16, 17, 18, 20 ]
Analysis of Merge Sort:
Unsorted Array: [ 9, 6, 7, 16, 18, 5, 2, 12, 20, 1, 16, 17, 4, 11, 13, 8 ]
Sorted Array: [ 1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 16, 16, 17, 18, 20 ]
Key Comparisons: 47
Number of Data Moves: 128
Analysis of Quick Sort:
Unsorted Array: [ 9, 6, 7, 16, 18, 5, 2, 12, 20, 1, 16, 17, 4, 11, 13, 8 ]
Sorted Array: [ 1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 16, 16, 17, 18, 20 ]
Key Comparisons: 50
Number of Data Moves: 125
```

d)

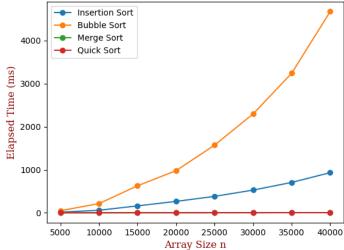
Analysis using Rand	lom Arrays:		
Analysis of Inserti			
Array Size	Elapsed Time	compCount	moveCount
5000	15	6180381	6185439
10000	59	25003650	25013660
15000	161	56545220	56560230
20000	265	100107502	100127508
25000	382	156927036	156952043
30000	528	224312219	224342224
35000	706	304313988	304348997
40000	933	398672705	398712714
Analysis of Bubble	Sort		
Array Size	Elapsed Time	compCount	moveCount
5000	49	12494725	18526155
10000	214	49978710	74980986
15000	627	112487550	169590696
20000	979	199970694	300262530
25000	1572	312473805	470706135
30000	2298	449969775	672846678
35000	3242		
		612480847	912836997
40000	4676	799950597	1195898148
Analysis of Merge S	Gort		
Analysis of Merge S Array Size		compCount	moveCount
		compCount 55250	moveCount 123616
Array Size	Elapsed Time		
Array Size 5000	Elapsed Time 1	55250	123616
Array Size 5000 10000	Elapsed Time 1 2	55250 120451	123616 267232
Array Size 5000 10000 15000	Elapsed Time 1 2 3	55250 120451 189396	123616 267232 417232
Array Size 5000 10000 15000 20000	Elapsed Time 1 2 3	55250 120451 189396 260890	123616 267232 417232 574464
Array Size 5000 10000 15000 20000 25000 30000	Elapsed Time 1 2 3 3 4	55250 120451 189396 260890 334037 408493	123616 267232 417232 574464 734464 894464
Array Size 5000 10000 15000 20000 25000 30000	Elapsed Time 1 2 3 3 4	55250 120451 189396 260890 334037 408493	123616 267232 417232 574464 734464
Array Size 5000 10000 15000 20000 25000 30000	Elapsed Time 1 2 3 3 4	55250 120451 189396 260890 334037 408493	123616 267232 417232 574464 734464 894464 1058928
Array Size 5000 10000 15000 20000 25000 30000	Elapsed Time 1 2 3 3 4	55250 120451 189396 260890 334037 408493	123616 267232 417232 574464 734464 894464 1058928
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 1 2 3 3 4 5	55250 120451 189396 260890 334037 408493	123616 267232 417232 574464 734464 894464 1058928
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 1 2 3 3 4 5 5	55250 120451 189396 260890 334037 408493 484377 561800	123616 267232 417232 574464 734464 894464 1058928 1228928
Array Size 5000 10000 15000 20000 25000 30000 35000 40000 Analysis of Quick S	Elapsed Time 1 2 3 3 4 5 5 Cort Elapsed Time	55250 120451 189396 260890 334037 408493 484377 561800	123616 267232 417232 574464 734464 894464 1058928 1228928
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 1 2 3 3 4 5 5 Elapsed Time 0	55250 120451 189396 260890 334037 408493 484377 561800 	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 130738
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 1 2 3 3 4 5 5 Elapsed Time 0 1	55250 120451 189396 260890 334037 408493 484377 561800 	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 130738 253443
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 1 2 3 3 4 5 5 Cort Elapsed Time 0 1	55250 120451 189396 260890 334037 408493 484377 561800 	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 130738 253443 447996
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 1 2 3 3 4 5 5 Elapsed Time 0 1 2 2	55250 120451 189396 260890 334037 408493 484377 561800 	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 130738 253443 447996 495275
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 1 2 3 3 4 5 5 5 Cort Elapsed Time 0 1 2 2 2 3	55250 120451 189396 260890 334037 408493 484377 561800 compCount 84796 150404 247245 314140 437987	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 130738 253443 447996 495275 644850
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 1 2 3 3 4 5 5 Cort Elapsed Time 0 1 2 2 3 3	55250 120451 189396 260890 334037 408493 484377 561800 compCount 84796 150404 247245 314140 437987 517122	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 130738 253443 447996 495275 644850 897860
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 1 2 3 3 4 5 5 Cort Elapsed Time 0 1 2 2 3 3 4	55250 120451 189396 260890 334037 408493 484377 561800 compCount 84796 150404 247245 314140 437987 517122 630932	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 130738 253443 447996 495275 644850 897860 968971
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 1 2 3 3 4 5 5 Cort Elapsed Time 0 1 2 2 3 3	55250 120451 189396 260890 334037 408493 484377 561800 compCount 84796 150404 247245 314140 437987 517122	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 130738 253443 447996 495275 644850 897860

Analysis using Almo	ost Sorted Arrays:		
Analysis of Insert	ion Sort		
Array Size	Elapsed Time	compCount	moveCount
5000	2	771965	776964
10000	13	3288505	3298504
15000	16	7152715	7167714
20000	26	11179787	11199786
25000	74	19544639	19569642
30000	84	29242049	29272048
35000	120	35670985	35705984
40000	98	38707459	38747460
Analysis of Bubble	Sort		
Array Size	Elapsed Time	compCount	moveCount
5000	30	12488589	2300898
10000	133	49970910	9835518
15000	389	112377540	21413148
20000	565	199960839	33479364
25000	984	312039569	58558932
30000	1382	449737544	87636150
35000	1612	608339619	106907958
40000	1982	765936374	116002386
Analysis of Merge	 Sort		
Analysis of Merge		compCount	moveCount
Array Size	Elapsed Time		moveCount 123616
Array Size 5000	Elapsed Time O	51125	123616
Array Size 5000 10000	Elapsed Time 0 1	51125 111083	123616 267232
Array Size 5000 10000 15000	Elapsed Time 0 1 2	51125 111083 173694	123616 267232 417232
Array Size 5000 10000 15000 20000	Elapsed Time 0 1 2 2	51125 111083 173694 237876	123616 267232 417232 574464
Array Size 5000 10000 15000 20000 25000	Elapsed Time 0 1 2 2	51125 111083 173694 237876 308858	123616 267232 417232 574464 734464
Array Size 5000 10000 15000 20000 25000 30000	Elapsed Time 0 1 2 2 2 2	51125 111083 173694 237876 308858 379022	123616 267232 417232 574464 734464 894464
Array Size 5000 10000 15000 20000 25000 30000	Elapsed Time 0 1 2 2 2 2 3	51125 111083 173694 237876 308858 379022 435514	123616 267232 417232 574464 734464 894464 1058928
Array Size 5000 10000 15000 20000 25000 30000	Elapsed Time 0 1 2 2 2 2	51125 111083 173694 237876 308858 379022	123616 267232 417232 574464 734464 894464
Array Size 5000 10000 15000 20000 25000 30000	Elapsed Time 0 1 2 2 2 2 3	51125 111083 173694 237876 308858 379022 435514	123616 267232 417232 574464 734464 894464 1058928
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 0 1 2 2 2 2 3 3	51125 111083 173694 237876 308858 379022 435514	123616 267232 417232 574464 734464 894464 1058928
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 0 1 2 2 2 3 3	51125 111083 173694 237876 308858 379022 435514 483003	123616 267232 417232 574464 734464 894464 1058928 1228928
Array Size 5000 10000 15000 20000 25000 30000 35000 40000 Analysis of Quick	Elapsed Time 0 1 2 2 2 2 3 3 Sort Elapsed Time	51125 111083 173694 237876 308858 379022 435514 483003	123616 267232 417232 574464 734464 894464 1058928 1228928
Array Size 5000 10000 15000 20000 25000 30000 35000 40000 Analysis of Quick: Array Size 5000	Elapsed Time 0 1 2 2 2 3 3 Sort Elapsed Time 1	51125 111083 173694 237876 308858 379022 435514 483003	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 185452
Array Size 5000 10000 15000 20000 25000 30000 35000 40000 Analysis of Quick Array Size 5000 10000	Elapsed Time 0 1 2 2 2 3 3 3 Sort Elapsed Time 1	51125 111083 173694 237876 308858 379022 435514 483003 	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 185452 467113
Array Size 5000 10000 15000 20000 25000 30000 35000 40000 Analysis of Quick: Array Size 5000 10000	Elapsed Time 0 1 2 2 2 3 3 3 Sort Elapsed Time 1 1	51125 111083 173694 237876 308858 379022 435514 483003 	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 185452 467113 745382
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 0 1 2 2 2 3 3 Sort Elapsed Time 1 1 2 3	51125 111083 173694 237876 308858 379022 435514 483003 	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 185452 467113 745382 1277869
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 0 1 2 2 2 2 3 3 Sort Elapsed Time 1 1 2 3 4	51125 111083 173694 237876 308858 379022 435514 483003 	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 185452 467113 745382 1277869 1523612
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 0 1 2 2 2 3 3 Sort Elapsed Time 1 1 2 3 4 7	51125 111083 173694 237876 308858 379022 435514 483003 	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 185452 467113 745382 1277869 1523612 2299925
Array Size 5000 10000 15000 20000 25000 30000 35000 40000	Elapsed Time 0 1 2 2 2 2 3 3 Sort Elapsed Time 1 1 2 3 4	51125 111083 173694 237876 308858 379022 435514 483003 	123616 267232 417232 574464 734464 894464 1058928 1228928 moveCount 185452 467113 745382 1277869 1523612

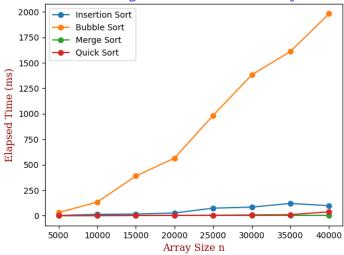
Analysis using Almost Unsorted Arrays:	Analysis wains Alma	at Uncented Appays:					
Array Size	Analysis using Almos	st unsurted Arrays.					
Array Size	Analysis of Insertic	on Sort					
5000 28 11779819 11784840 10900 115 46912641 46922672 15900 255 105300999 105316026 20000 459 188897569 188917580 25900 762 293409738 293434760 35000 1133 421376972 421407008 35000 1609 577427615 577464882 40900 2277 760074594 760121838 Analysis of Bubble Sort 15000 53 12497500 35324526 2000 239 49995000 1407088022 19078020 159602 159600 315858084 36060 315858084 36060 12497500 315858084 36060 12407500 880154286 36062746 28060 1240798500 124070802 124070802 124070802 124070802 124070802 124070802 124070802 124070802 124070802 124070802 124070802 124070802 124070802 124070802 1240708	•		compCount	moveCount			
15000		28		11784840			
20000	10000	115	46912641	46922672			
25000 762 293409738 293434760 30000 1133 421376972 421407008 35000 1609 577427615 577464882 40000 2277 760074594 760121838 Analysis of Bubble Sort Array Size Elapsed Time compCount moveCount 5000 556 112492500 315858084 20000 971 199989999 566632746 25000 1597 312487500 880154286 30000 2341 449985000 124041030 35000 3357 612482500 1732184652 40000 4099 79980000 -2014841776 Analysis of Merge Sort Array Size Elapsed Time compCount moveCount 5000 6 49547 123616 10000 1 108691 267232 15000 2 2 172531 417232 20000 2 3 3565 3 376312 894464 35000 3 3 435721 1058928 40000 1 121483 191358 10000 1 1 367813 59528 40000 1 1 367813 59528 40000 1 367813 59528 20000 2 571278 892085 20000 2 571278 892085 20000 2 571278 892085 20000 2 571278 892085 20000 2 571278 892085 20000 2 571278 892085 20000 3 873315 1406946 35000 3 873315 1406946 35000 3 873315 1406946 35000 3 873315 1406966 35000 3 873315 1406966 35000 3 873315 1406966	15000	255	105300999	105316026			
30000 1133 421376972 421407008 35000 1609 577427615 577464882 40000 2277 760074594 760121838 Analysis of Bubble Sort Array Size Elapsed Time compCount moveCount 5000 63 12497500 35324526 10000 239 49975000 140708822 15000 556 112492500 315858084 20000 971 199989999 566632746 25000 1597 312487500 880154286 30000 2341 449985000 1264041030 35000 3357 612482500 1732184652 40000 4099 79998000 -2014841776 Analysis of Merge Sort Array Size Elapsed Time compCount moveCount 5000 0 49547 123616 18000 1 1 108691 267232 15000 2 2 376312 417232 28000 2 336583 734464 30000 2 3376312 894464 35000 3 435721 1058928 40000 4 484490 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 1 221483 19158 40000 1 1 126928 40000 1 1 121483 19158 40000 1 1 121483	20000	459	188897569	188917580			
35000 1609 577427615 577464882 40000 2277 760074594 760121838	25000	762	293409738	293434760			
Analysis of Bubble Sort Aray Size	30000	1133	421376972	421407008			
Analysis of Bubble Sort Array Size	35000	1609	577427615	577464882			
Array Size	40000	2277	760074594	760121838			
Array Size							
Array Size							
5000 63 12497500 35324526 10000 239 49995000 140708022 15000 556 112492500 315858084 20000 971 19989999 566632746 25000 1597 312487500 880154286 30000 2341 449985000 1264041030 35000 3357 612482500 1732184652 40000 4099 79998000 -2014841776 Analysis of Merge Sort Array Size Elapsed Time compCount moveCount 5000 1 108691 2 237617 57464 25000 2 306583 734464 35000 3 435721 1058928 40000 4 484490 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 4 484490 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 121483 191358 10000 1 367813 585540 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 121483 191358 10000 1 121483 191358 10000 1 121483 191358 19285 19000 2 568573 914815 125000 3 873315 1406946 35000 16 4944236 7538032 Analysis of Quick Sort Analysis of Quick Sort Analysis of Quick Sort Array Size Elapsed Time compCount moveCount move	Analysis of Bubble S	Sort					
10000	Array Size	Elapsed Time	compCount	moveCount			
15000 556 112492500 315858084 20000 971 199989999 566632746 25000 1597 312487500 880154286 30000 2341 449985000 1264041030 355000 3357 612482500 1732184652 40000 4099 799980000 -2014841776	5000	63	12497500	35324526			
20000 971 199989999 566632746 25000 1597 312487500 880154286 30000 2341 449985000 1264041030 35000 3357 612482500 1732184652 40000 4099 799980000 -2014841776	10000	239	49995000	140708022			
25000 1597 312487500 880154286 30000 2341 449985000 1264041030 35000 3357 612482500 1732184652 40000 4099 799980000 -2014841776	15000	556	112492500	315858084			
30000 2341 449985000 1264041030 35000 3357 612482500 1732184652 40000 4099 799980000 -2014841776	20000	971	199989999	566632746			
355000 3357 612482500 1732184652 40000 4099 799980000 -2014841776	25000	1597	312487500	880154286			
Analysis of Merge Sort Array Size Elapsed Time compCount moveCount 5000 0 49547 123616 10000 1 108691 267232 15000 2 172531 417232 20000 2 237617 574464 25000 2 306583 734464 30000 2 376312 894464 35000 3 435721 1058928 40000 4 484490 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 121483 191358 10000 1 1 367813 585540 15000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032	30000	2341	449985000	1264041030			
Analysis of Merge Sort Array Size	35000	3357	612482500	1732184652			
Array Size	40000	4099	799980000	-2014841776			
Array Size							
Array Size	Apalysis of Manga						
5000 0 49547 123616 10000 1 108691 267232 15000 2 172531 417232 20000 2 237617 574464 25000 2 306583 734464 30000 2 376312 894464 35000 3 435721 1058928 40000 4 484490 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 121483 191358 10000 1 367813 585540 15000 2 571278 892085 20000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032			oompCoup+	mayaCaup+			
10000 1 108691 267232 15000 2 172531 417232 20000 2 237617 574464 25000 2 306583 734464 30000 2 376312 894464 35000 3 435721 1058928 40000 4 4 484490 1228928							
15000 2 172531 417232 20000 2 237617 574464 25000 2 306583 734464 30000 2 376312 894464 35000 3 435721 1058928 40000 4 484490 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 121483 191358 10000 1 367813 585540 15000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032							
20000 2 237617 574464 25000 2 306583 734464 30000 2 376312 894464 35000 3 435721 1058928 40000 4 484490 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 121483 191358 10000 1 367813 585540 15000 2 571278 892085 20000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032							
25000 2 376312 894464 35000 3 435721 1058928 40000 4 484490 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 121483 191358 10000 1 367813 585540 15000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032							
30000 2 376312 894464 35000 3 435721 1058928 40000 4 484490 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 121483 191358 10000 1 367813 585540 15000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032							
35000 3 435721 1058928 40000 4 484490 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 121483 191358 10000 1 367813 585540 15000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032							
40000 4 484490 1228928 Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 121483 191358 10000 1 367813 585540 15000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032							
Analysis of Quick Sort Array Size Elapsed Time compCount moveCount 5000 1 121483 191358 10000 1 367813 585540 15000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032							
Array SizeElapsed TimecompCountmoveCount50001121483191358100001367813585540150002571278892085200002568573914815250004105505917272863000038733151406946350001649442367538032	40000	4	484490	1228928			
Array SizeElapsed TimecompCountmoveCount50001121483191358100001367813585540150002571278892085200002568573914815250004105505917272863000038733151406946350001649442367538032							
Array SizeElapsed TimecompCountmoveCount50001121483191358100001367813585540150002571278892085200002568573914815250004105505917272863000038733151406946350001649442367538032	Analysis of Osish Cont						
5000 1 121483 191358 10000 1 367813 585540 15000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032							
10000 1 367813 585540 15000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032	*						
15000 2 571278 892085 20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032							
20000 2 568573 914815 25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032							
25000 4 1055059 1727286 30000 3 873315 1406946 35000 16 4944236 7538032			571278				
30000 3 873315 1406946 35000 16 4944236 7538032			568573				
35000 16 4944236 7538032	25000		1055059	1727286			
	30000	3	873315	1406946			
40000 88 27405350 41251057	35000	16	4944236	7538032			
	40000	88	27405350	41251057			

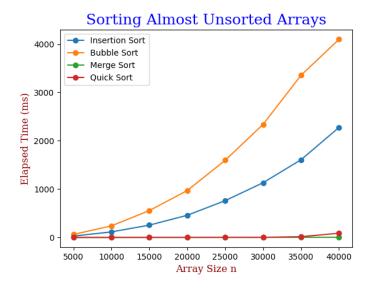
Question 3:



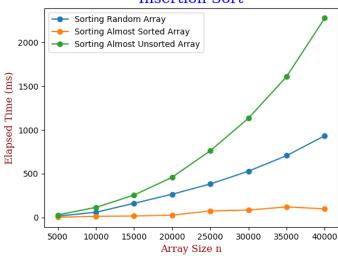


Sorting Almost Sorted Arrays

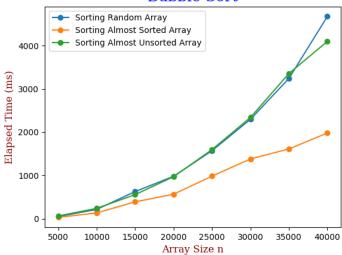


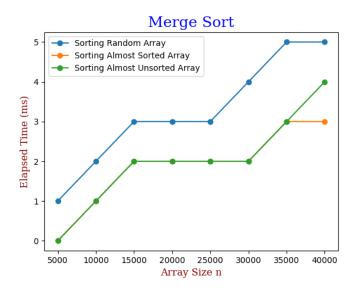


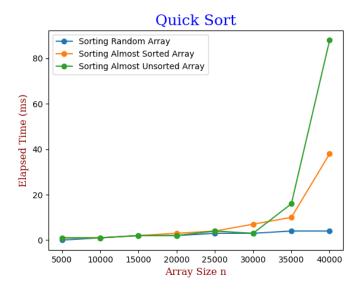
Insertion Sort



Bubble Sort







Report

Theoretically, we expect that insertion sort is O(n) for best case, $O(n^2)$ for average and worst cases. From experimental results, we see that for random array (average case) it looks like parabola, for almost sorted array (best case) it looks like line and for almost unsorted array (worst case) it even more parabola than average case. That is theoretical = experimental values.

Theoretically, we expect that bubble sort is O(n) for best case, $O(n^2)$ for average and worst cases. From experimental results, we see that for random array (average case) it looks like parabola, for almost sorted array (best case) it looks like line but it's slope much larger than insertion sort's best case slope and for almost unsorted array (worst case) it very much resembles average case line. That is theoratical = experimental values.

Theoretically, merge sort is $O(n * \log n)$ for all possible cases. From experimental results, ladder-like lines suggest that it is not linear but some multiplication of n and for our case, factor should be $\log n$. Therefore, merge sort is also experimentally correct.

Theoretically, we expect that quick sort is O(n * logn) for best and average case, $O(n^2)$ for worst case. From experimental results, almost sorted array (worst case) shows parabola and we expect that the others should be ladder-like lines. Our claim is true for random array, but not for almost unsorted array which has some outlier value at the end. Ignoring the last value of almost unsorted array we can see that quick sort is experimentally correct.

Between 4 sorting algorithms we see that merge and quick sort is much more efficient that insertion and bubble sorts which we expect from their time complexities. It seems among them the worst algorithm is bubble sort and the best is merge sort. This homework clearly shows that how O(n * logn) and O(n) much more efficient than $O(n^2)$ since even in very high values the first two ones not change much.