CO322: Data Structures and Algorithms

Lab01

Wijesooriya H.D

E/17/407

Theoretical Results

- Big Oh notation was used to compare the performance of 3 sorting algorithms theoreticaly.
- Run times of those sorting algorithms are shown below. (N-number of elements in the array)

Algorithms	Run time	
	Best case	Worst case
Bubble sort	O(N)	O(N^2)
Selection sort	O(N^2)	O(N^2)
Insertion sort	O(N)	O(N^2)

- According to the table above,we can see that all three sorting algorithms have the same running time for the worst case scenario.
- If we double the array size ,the run time would be 4 times bigger than the previous one in the worst case scenario.

Empirical results

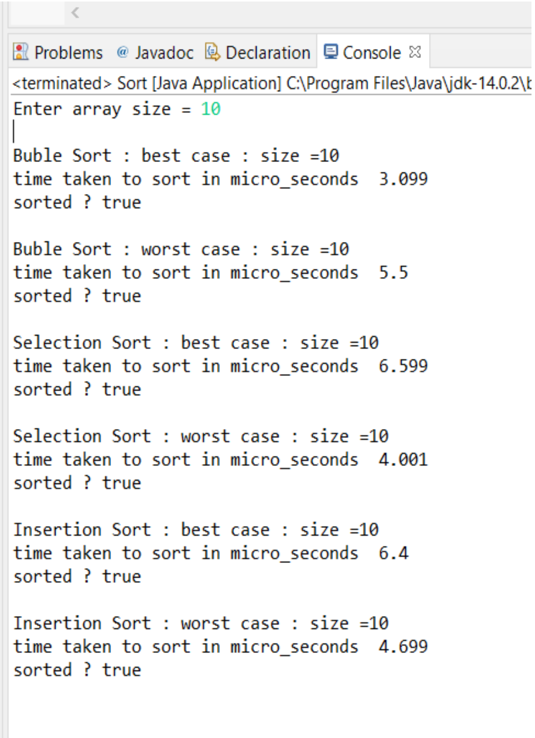
- The performance of the 3 algorithms for different input sizes were measured.
- Some empirical results are shown below.

Test case 1: array size=10

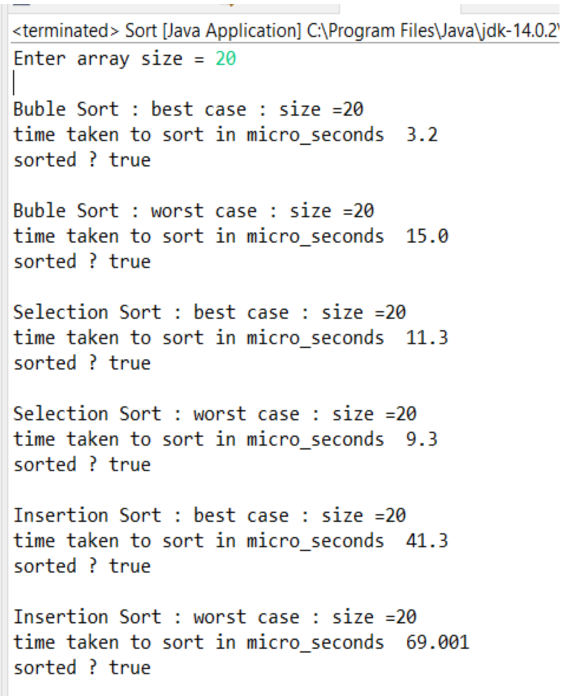
Algorithm	Time taken to sort (µs)	
	Best case	Worst case
Bubble sort	3.099	5.5
Selection sort	6.599	4.001
Insertion sort	6.4	4.699

Test case 2: array size=20

Algorithm	Time taken to sort (µs)	
	Best case	Worst case
Bubble sort	3.2	15.0
Selection sort	11.3	9.3
Insertion sort	41.3	69.001



```
<terminated> Sort [Java Application] C:\Program Files\Java\jdk-14.0.2\b
Enter array size = 10

Buble Sort : best case : size =10
time taken to sort in micro_seconds  3.099
sorted ? true

Buble Sort : worst case : size =10
time taken to sort in micro_seconds  5.5
sorted ? true

Selection Sort : best case : size =10
time taken to sort in micro_seconds  6.599
sorted ? true

Selection Sort : worst case : size =10
time taken to sort in micro_seconds  4.001
sorted ? true

Insertion Sort : best case : size =10
time taken to sort in micro_seconds  6.4
sorted ? true

Insertion Sort : worst case : size =10
time taken to sort in micro_seconds  4.699
sorted ? true
```



```
<terminated> Sort [Java Application] C:\Program Files\Java\jdk-14.0.2\
Enter array size = 20

Buble Sort : best case : size =20
time taken to sort in micro_seconds  3.2
sorted ? true

Buble Sort : worst case : size =20
time taken to sort in micro_seconds  15.0
sorted ? true

Selection Sort : best case : size =20
time taken to sort in micro_seconds  11.3
sorted ? true

Selection Sort : worst case : size =20
time taken to sort in micro_seconds  9.3
sorted ? true

Insertion Sort : best case : size =20
time taken to sort in micro_seconds  41.3
sorted ? true

Insertion Sort : worst case : size =20
time taken to sort in micro_seconds  69.001
sorted ? true
```

- **Test case 1 and test case 2 were used to measure the performance of algorithms when the array size is small.**

- **According to the above tables we can see that the bubble sort algorithm has shown a high performance relative to the other algorithms in the best case scenario. This matches with the theoretical analysis. (bubble sort ->best case-> run time ->O(N))**
- **Selection sort algorithm has shown a high performance in the worst case scenario.**

- **In test case 2 the array size was doubled relative to the test case 1. So the running time of each algorithm should be 4 times bigger than the previous one. (according to the theoretical analysis, run time for worst case->O(N^2))**
- **In this case the empirical results I got, does not agree with the theoretical analysis.**

  Eg: Insertion sort

  Array size =10: run time (worst case) =4.699µs

  Array size =20:

  theoretical run time (worst case) =4x4.699=18.796µs

  empirical run time (worst case) =69.001µs (>> 4x4.699)
- **Reasons for the above observation:**

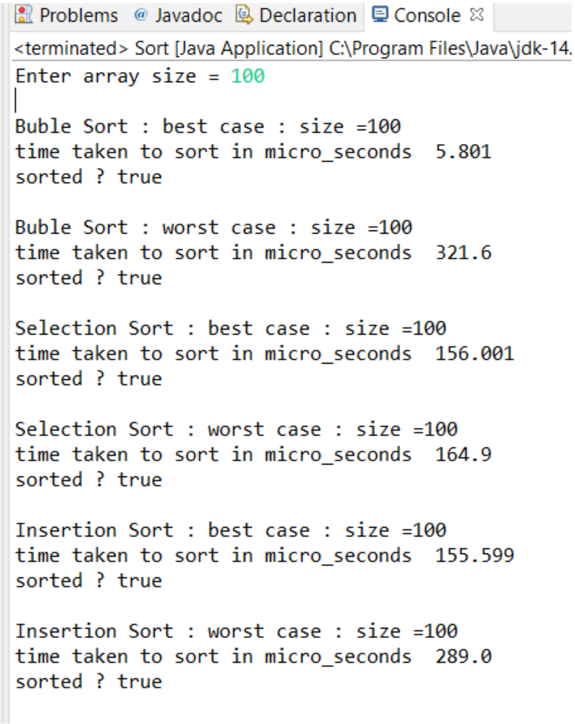
  Performance of the cache

Test case 3: array size=100

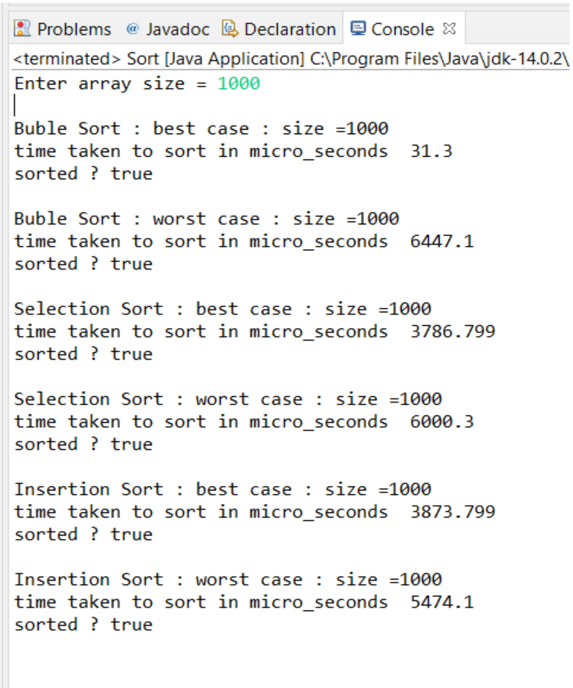
Algorithm	Time taken to sort (µs)	
	Best case	Worst case
Bubble sort	5.801	321.6
Selection sort	156.001	164.9
Insertion sort	155.599	289.0

Test case 4: array size=1000

Algorithm	Time taken to sort (µs)	
	Best case	Worst case
Bubble sort	31.3	6447.1
Selection sort	3786.799	6000.3
Insertion sort	3873.799	5474.1



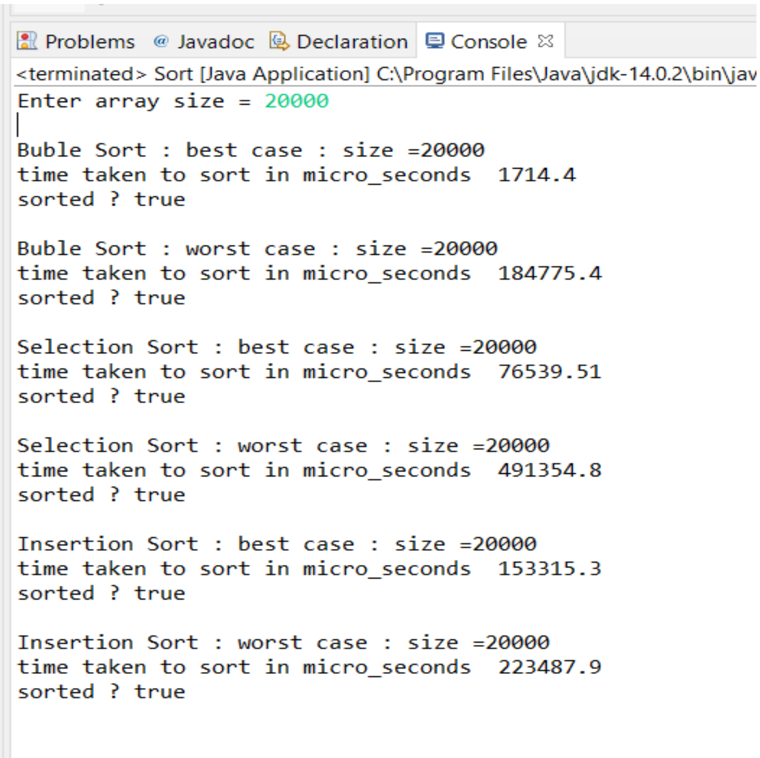
```
<terminated> Sort [Java Application] C:\Program Files\Java\jdk-14.
Enter array size = 100

Buble Sort : best case : size =100
time taken to sort in micro_seconds  5.801
sorted ? true

Buble Sort : worst case : size =100
time taken to sort in micro_seconds  321.6
sorted ? true

Selection Sort : best case : size =100
time taken to sort in micro_seconds  156.001
sorted ? true

Selection Sort : worst case : size =100
time taken to sort in micro_seconds  164.9
sorted ? true

Insertion Sort : best case : size =100
time taken to sort in micro_seconds  155.599
sorted ? true

Insertion Sort : worst case : size =100
time taken to sort in micro_seconds  289.0
sorted ? true
```



```
<terminated> Sort [Java Application] C:\Program Files\Java\jdk-14.0.2\
Enter array size = 1000

Buble Sort : best case : size =1000
time taken to sort in micro_seconds  31.3
sorted ? true

Buble Sort : worst case : size =1000
time taken to sort in micro_seconds  6447.1
sorted ? true

Selection Sort : best case : size =1000
time taken to sort in micro_seconds  3786.799
sorted ? true

Selection Sort : worst case : size =1000
time taken to sort in micro_seconds  6000.3
sorted ? true

Insertion Sort : best case : size =1000
time taken to sort in micro_seconds  3873.799
sorted ? true

Insertion Sort : worst case : size =1000
time taken to sort in micro_seconds  5474.1
sorted ? true
```

Test case 5: array size=20000

Algorithm	Time taken to sort (µs)	
	Best case	Worst case
Bubble sort	1714.4	184775.4
Selection sort	76539.51	491354.8
Insertion sort	153315.3	223487.9



```
<terminated> Sort [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\jav
Enter array size = 20000

Buble Sort : best case : size =20000
time taken to sort in micro_seconds  1714.4
sorted ? true

Buble Sort : worst case : size =20000
time taken to sort in micro_seconds  184775.4
sorted ? true

Selection Sort : best case : size =20000
time taken to sort in micro_seconds  76539.51
sorted ? true

Selection Sort : worst case : size =20000
time taken to sort in micro_seconds  491354.8
sorted ? true

Insertion Sort : best case : size =20000
time taken to sort in micro_seconds  153315.3
sorted ? true

Insertion Sort : worst case : size =20000
time taken to sort in micro_seconds  223487.9
sorted ? true
```

- **Test case 3,4 and 5 were used to measure the performance of algorithms when the array size is large.**

- **According to the above tables we can see that the bubble sort algorithm shows a high performance relative to the other**

algorithms in best case scenario. This matches with the theoretical analysis. (bubble sort ->best case-> run time ->O(N))

- **When the array size varies between 100-1000, selection sort algorithm has shown a high performance in the worst case scenario.**
- **When the array size varies between 1000-10000, insertion sort and bubble sort algorithms have shown a high performance in the worst case scenario.**
- **When the array size was grater than 10000 ,bubble sort algorithm has shown a high performance in the worst case scenario.**
- **So it is clear that we can't say that a particular algorithm is better than another algorithm just looking at the array size.**

- **When we run the same program with the same array size in different times it showed different sorting times.The reason for that would be the cache performance.**