

CO322 LAB01  
SORTING ALGORITHMS

E/17/379

S.P.D.D.S WEERASINGHE

Steps

- Implemented the bubble, selection and insertion sort algorithms.
- tested implementations for various test cases and input sizes using ‘Issorted’ method.
- Measured the performance of the 3 algorithms for different input sizes for the worst and the best cases Using time functions.

Observations

Algorithms worked fine

```
Testing Bubble Sort

input = 1000 random array bubble sort test passed? true
input = 2000 random array bubble sort test passed? true
input = 4000 random array bubble sort test passed? true
input = 8000 random array bubble sort test passed? true
input = 1000 best array bubble sort test passed? true
input = 1000 worst array bubble sort test passed? true

Testing Selection Sort

input = 1000 random array selection sort test passed? true
input = 2000 random array selection sort test passed? true
input = 4000 random array selection sort test passed? true
input = 8000 random array selection sort test passed? true
input = 1000 best array selection sort test passed? true
input = 1000 worst array selection sort test passed? true

Testing Insertion Sort

input = 1000 random array insertion sort test passed? true
input = 2000 random array insertion sort test passed? true
input = 4000 random array insertion sort test passed? true
input = 8000 random array insertion sort test passed? true
input = 1000 best array insertion sort test passed? true
input = 1000 worst array insertion sort test passed? true
```

Timing Values

- Bubble Sort (Without Optimization

```
Bubble Sort timing results

Number of inputs =1000
Average bestcase running time in ns :613121
Average worstcase running time in ns :605372
Average random running time in ns :612321

Number of inputs =2000
Average bestcase running time in ns :2986983
Average worstcase running time in ns :2973822
Average random running time in ns :2981869

Number of inputs =4000
Average bestcase running time in ns :12315129
Average worstcase running time in ns :12369257
Average random running time in ns :12488409

Number of inputs =8000
Average bestcase running time in ns :49107105
Average worstcase running time in ns :49445629
Average random running time in ns :50221661
```

Selection Sort

```
Selection Sort timing results

Number of inputs =1000
Average bestcase running time in ns :438493
Average worstcase running time in ns :435295
Average random running time in ns :431983

Number of inputs =2000
Average bestcase running time in ns :2057335
Average worstcase running time in ns :2052740
Average random running time in ns :2053859

Number of inputs =4000
Average bestcase running time in ns :8236490
Average worstcase running time in ns :8329157
Average random running time in ns :8328035

Number of inputs =8000
Average bestcase running time in ns :33375791
Average worstcase running time in ns :33284988
Average random running time in ns :33122071
```

- Insertion Sort

```
Insertion Sort timing results

Number of inputs =1000
Average bestcase running time in ns :2058
Average worstcase running time in ns :4950
Average random running time in ns :3637

Number of inputs =2000
Average bestcase running time in ns :6297
Average worstcase running time in ns :18967
Average random running time in ns :12173

Number of inputs =4000
Average bestcase running time in ns :14526
Average worstcase running time in ns :83008
Average random running time in ns :42309

Number of inputs =8000
Average bestcase running time in ns :29900
Average worstcase running time in ns :269092
Average random running time in ns :138159

C:\Users\dilan\Desktop>javac Sort.java
```

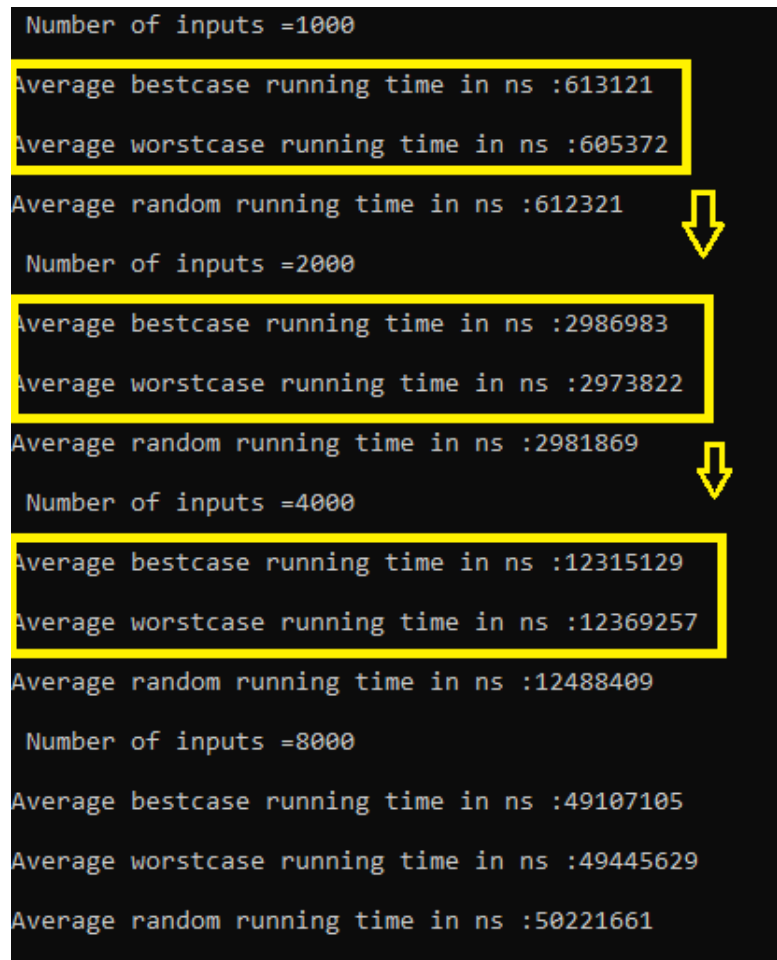
## Discussion

1). How does the performance vary with the input size

- When number of inputs were multiplied by 2;

1. In Bubble sort

```
Number of inputs =1000
Average bestcase running time in ns :613121
Average worstcase running time in ns :605372
Average random running time in ns :612321
Number of inputs =2000
Average bestcase running time in ns :2986983
Average worstcase running time in ns :2973822
Average random running time in ns :2981869
Number of inputs =4000
Average bestcase running time in ns :12315129
Average worstcase running time in ns :12369257
Average random running time in ns :12488409
Number of inputs =8000
Average bestcase running time in ns :49107105
Average worstcase running time in ns :49445629
Average random running time in ns :50221661
```



Best case and worst case run times were approximately multiplied by 4.

2. In Selection Sort

```

Number of inputs =1000
Average bestcase running time in ns :438493
Average worstcase running time in ns :435295
Average random running time in ns :431983
Number of inputs =2000
Average bestcase running time in ns :2057335
Average worstcase running time in ns :2052740
Average random running time in ns :2053859
Number of inputs =4000
Average bestcase running time in ns :8236490
Average worstcase running time in ns :8329157
Average random running time in ns :8328035
Number of inputs =8000
Average bestcase running time in ns :33375791
Average worstcase running time in ns :33284988
Average random running time in ns :33122071

```

Best case and worst case run times were approximately multiplied by 4.

### 3.In Insertion sort

```

Number of inputs =1000
Average bestcase running time in ns :2058
Average worstcase running time in ns :4950
Average random running time in ns :3637
Number of inputs =2000
Average bestcase running time in ns :6297
Average worstcase running time in ns :18967
Average random running time in ns :12173
Number of inputs =4000
Average bestcase running time in ns :14526
Average worstcase running time in ns :83008
Average random running time in ns :42309
Number of inputs =8000
Average bestcase running time in ns :29900
Average worstcase running time in ns :269092
Average random running time in ns :138159

```

Best case run times were approximately multiplied by 2.

worst case run times were approximately multiplied by 4.

- Insertion method has the best performance.
- Insertion and selection sort algorithms seems to be better than bubble sort algorithm. The time taken to sort the array using bubble sort algorithm was higher than the selection and insertion sort algorithms for almost all sizes of arrays.
- Insertion Sort is faster than selection sort both in best case as well as worst case data arrangements.
- Insertion Sort requires on average half as many comparisons. With Insertion Sort, we have comparisons and shifts averaging up to *half* of the sorted elements; with Selection Sort, we have to search for the smallest element in *all* unsorted elements in each step.
- Selection Sort has significantly fewer write operations, so Selection Sort can be faster when writing operations are expensive.

2)Does the empirical results you get agree with theoretical analysis?

According to the theoretical analysis time complexity of all three algorithms is mentioned in the below table

mentioned in the below table

Algorithm	Best Case time Complexity	Worst Case time Complexity
Bubble sort( Not optimized)	$O(n^2)$	$O(n^2)$
Selection sort	$O(n^2)$	$O(n^2)$
Insertion sort	$O(n)$	$O(n^2)$

When the above theoretical results are analyzed it is approximately equal to the empirical results of the run times of time complexities.

Run time does not only depend on the algorithm or the worst and best case arrays. It also depends on some other factors such as the applications running on the computer behind this program.

Therefore empirical results are not exact with theoretical results.

3) How did/should you test your code. Explain the test cases you used and the rationale for use them.

- Use 'isSorted' method to check whether the outputs of every algorithms for different number of inputs and arrangements were correctly sorted
- array sizes were multiplied by two to check how the performance vary with the array size.
- arrays with best, worst and random array data arrangements were used to test best and worst case performances of all sorting algorithms in each size of array.

## Conclusions

- Bubble sort can be more time consuming in a case where all the elements are in reverse order. Since algorithm has to go through all the elements 'Bubbling', the number of swaps are higher.

Worst case for the bubble sort can be where the last element of the list is also the smallest. Thus it has to pass down the list all through each element.

In the aspect of the CPU hardware, Bubble sort is considered to be less efficient since its cache misses are twice as high as insertion sort.

In Overall, Bubble sort performance is considered to be the worst of all three

- Selections sort efficiency is independent of the data being sorted, which is an advantage in real-time applications.

This perform well for complex algorithms where the auxiliary memory is low. Because of its divide and conquer nature.

Selections sort perform twice as high comparisons when compared to the insertion sort.

- In insertion, Number of comparisons are lower when compared to the selection sort.

The primary advantage of insertion sort over selection sort is that selection sort must always scan all remaining elements to find the absolute smallest element in the unsorted portion of the list, while insertion sort requires only a single comparison when the  $k+1$ th element is greater than the  $k$ th element.

A disadvantage of insertion sort over selection sort is that it requires more writes due to the fact that, on each iteration, inserting the  $k+1$ th element into the sorted portion of the array requires many element swaps to shift all of the following elements, while only a single swap is required for each iteration in selection sort.