

INFO 6205

Program Structures & Algorithms

Summer Full 2018

Assignment 3

In this Assignment, I will form different types of arrays and will sort it by Selection and Insertion sort.

Selection sort and Insertion sort have different complexities in different scenarios. N is the number of elements in the array.

Sorts:	<u>Best</u>	<u>Average</u>	<u>Worst</u>
Selection	$\frac{1}{2} N^2$	$\frac{1}{2} N^2$	$\frac{1}{2} N^2$
Insertion	N	$\frac{1}{4} N^2$	$\frac{1}{2} N^2$

Here, I have created 4 different arrays as follows:

- Random Array
- Sorted Array
- Reverse Array
- Partially Sorted Array

1. CONCLUSION:

Some useful abbreviations:

- n – Number of elements in the Array.

I ran the experiment for various “n” like 1000,2000,4000,8000,16000 etc.

While doing the experiment, the mean pairs that were calculated each time for n which also ran in an incremental manner.

Experiments taken: 100 time.

So, 1st experiment ran for 100*1000 times.

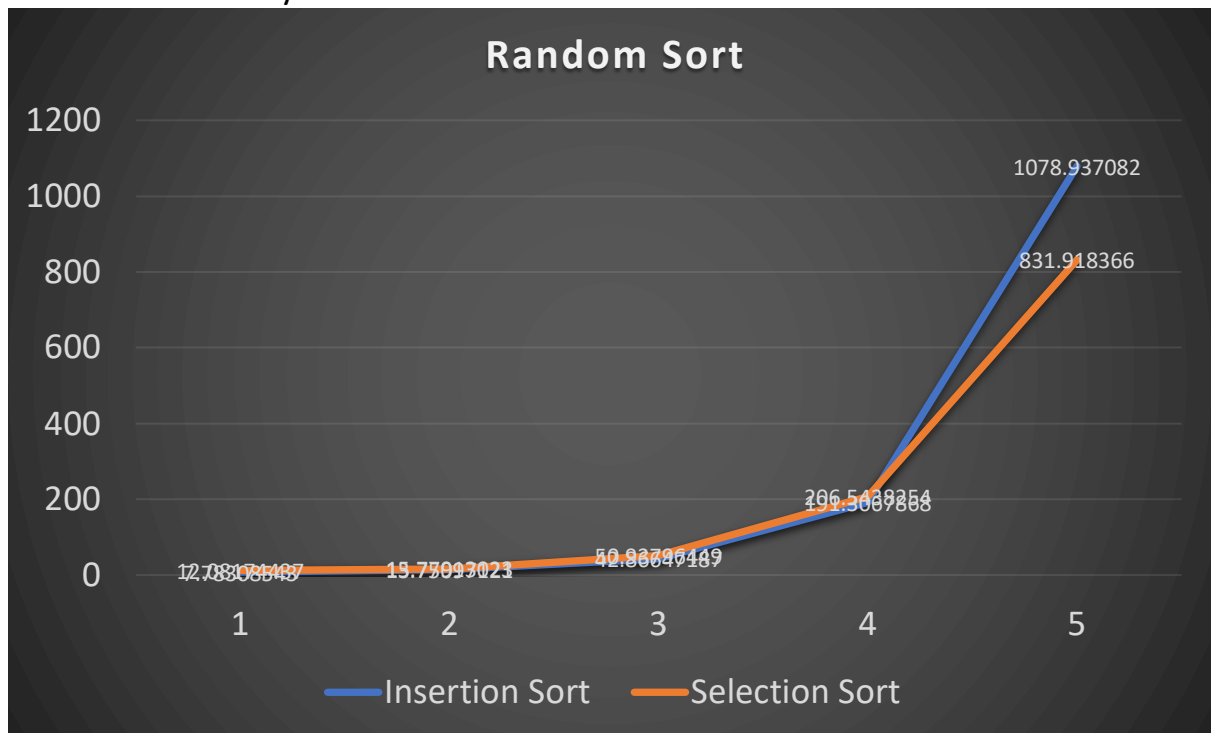
Please find observations, screenshots, graphs, examples below:

Observations:

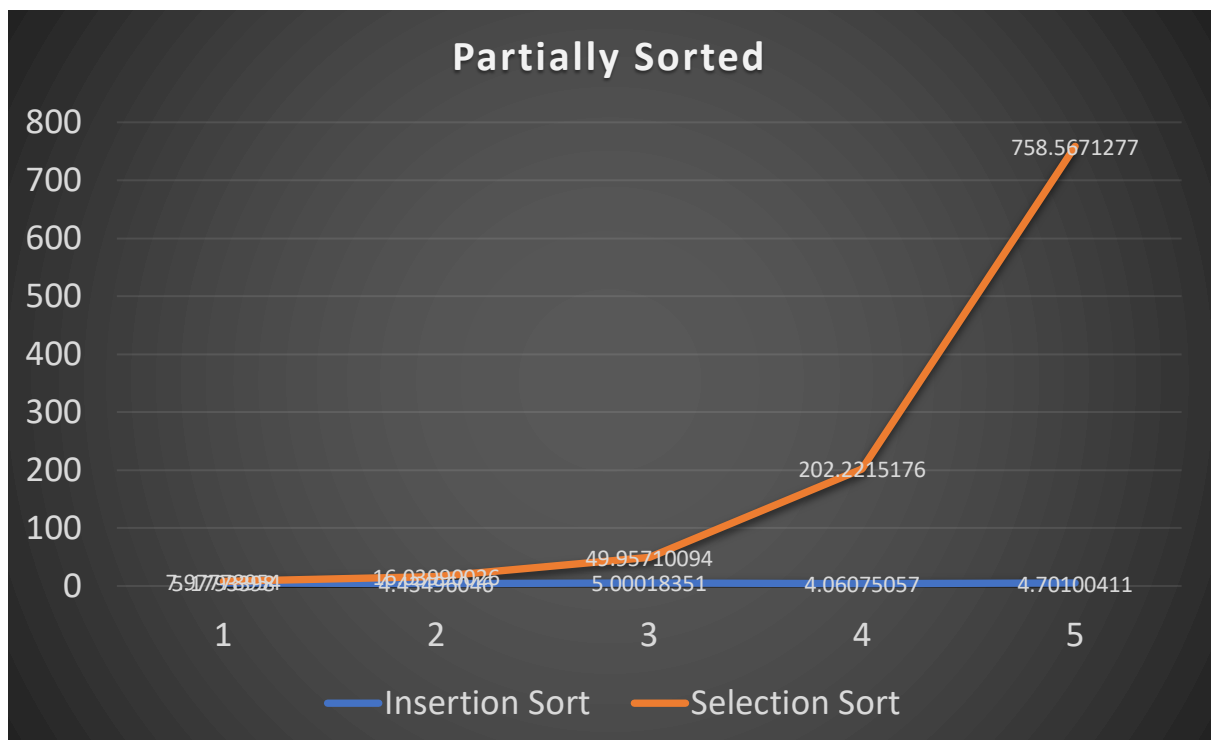
Array	N (No. of Elements)	Selection Sort time(millisecons)	Insertion Sort time(millisecons)	Result
Sorted	1000	10.30749803	7.43517455	Insertion
Reverse	1000	10.25576403	10.1069063	Selection
Random	1000	12.08174437	7.78308543	Insertion
Partial Sort	1000	7.97778954	5.1793398	Insertion
Sorted	2000	15.93404995	4.519438650001	Insertion
Reverse	2000	15.79157095	21.87396804	Selection
Random	2000	15.77913023	13.75097121	Insertion
Partial Sort	2000	16.03990026	4.43496046	Insertion
Sorted	4000	51.52672263	4.33987573	Insertion
Reverse	4000	50.36611286996	68.9221657	Selection
Random	4000	50.937964490006	42.86647187	Insertion
Partial Sort	4000	49.957100940004	5.00018351	Insertion
Sorted	8000	190.90480757	4.06110498	Insertion
Reverse	8000	188.1818505	299.9949137603	Selection
Random	8000	206.5438253802	191.30678682	Insertion
Partial Sort	8000	202.22151758	4.06075057	Insertion
Sorted	16000	782.06934952	4.95224215	Insertion
Reverse	16000	788.76900499	1203.3181278701	Selection
Random	16000	831.91836596	1078.93708227	Insertion
Partial Sort	16000	758.56712769	4.70100411	Insertion

2. Graph of different arrays:

- Random Sort Array:

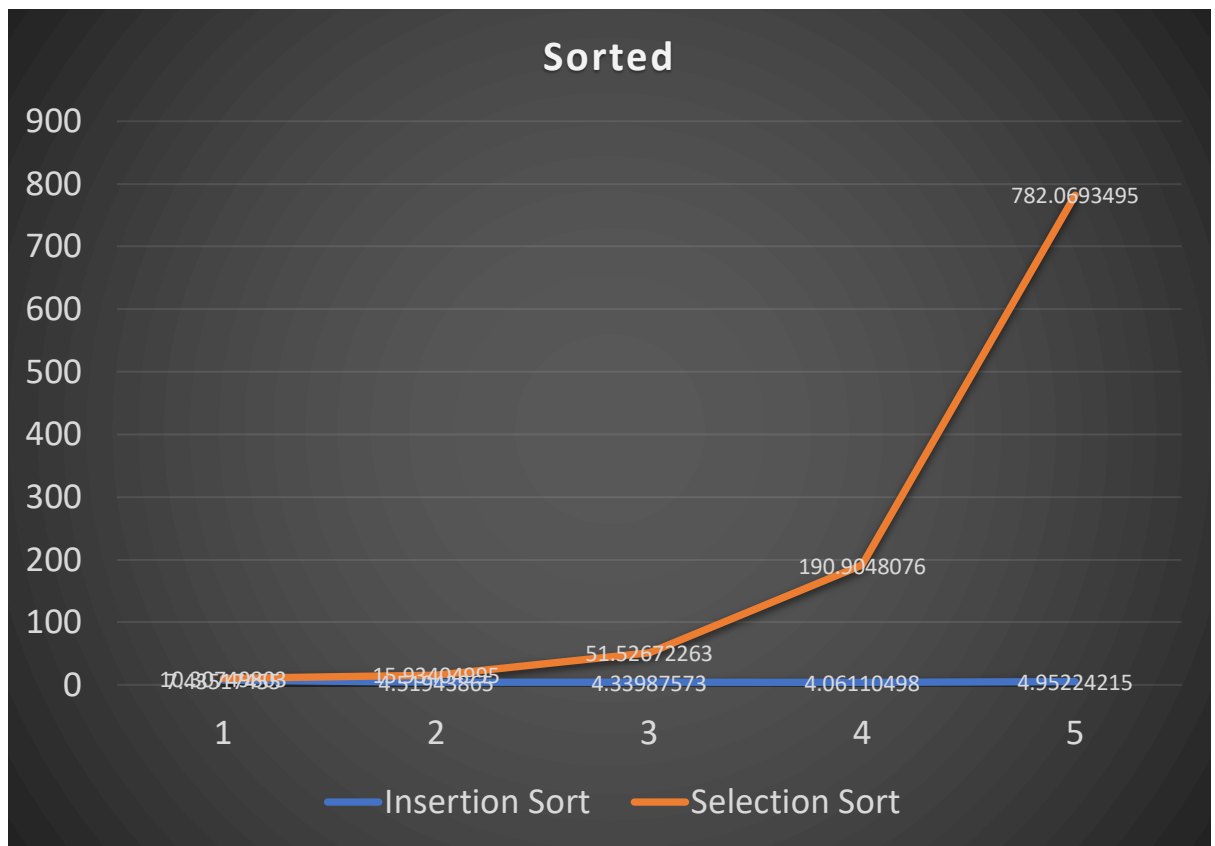


- Partially Sort Array

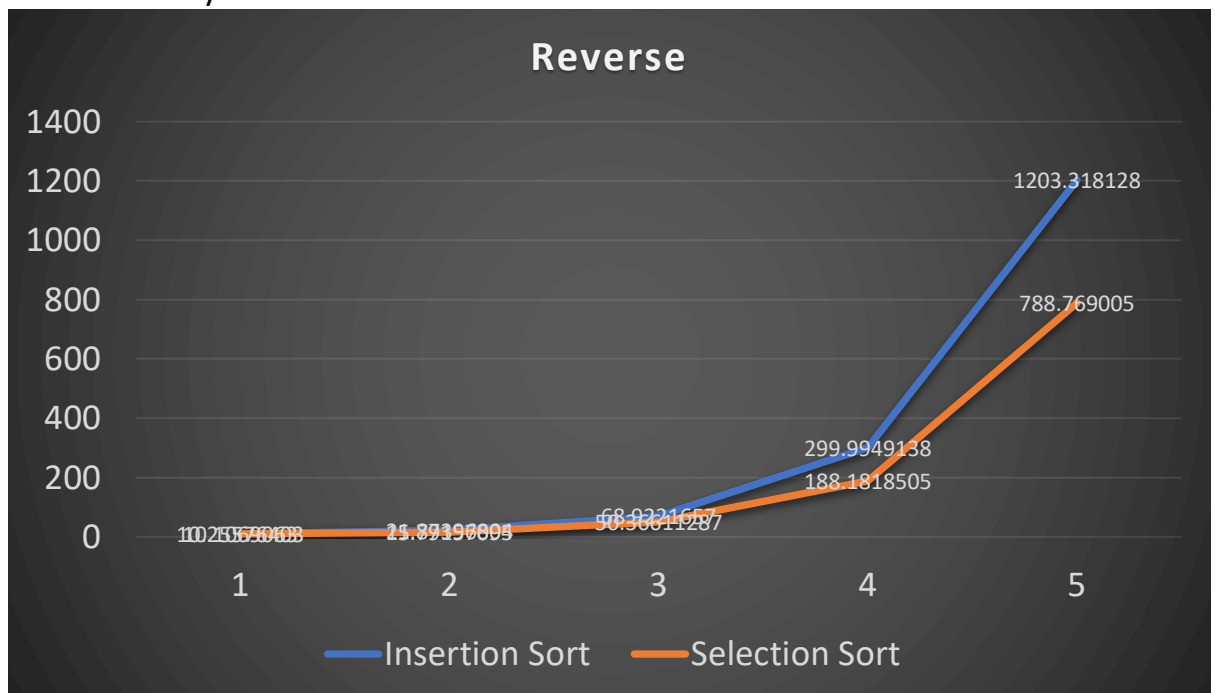


Goel, Ajay: Nu Id (001897443)

- Sorted Array:



- Reverse Array:



Goel, Ajay: Nu Id (001897443)

Please find examples below:

Let's take an example to prove the complexity:

Examples:

(a). For Sorted Array:

Take $n = 2000$

In the screenshot, when I have taken:

Experiments: 100

Time Complexity of Selection Sort: N^2

Time Complexity of Insertion Sort: N

According to observation:

For Selection sort, time through code: 15.93404995 milliseconds.

For Insertion sort, time through code: 4.51943865001 milliseconds.

Insertion Sort is taking very less time as compared to Selection sort as its complexity is very lower than Selection sort.

Hence Proved

(b). For Reverse Array:

Take $n = 2000$

In the screenshot, when I have taken:

Experiments: 100

Time Complexity of Selection Sort: N^2

Time Complexity of Insertion Sort: N^2

According to observation:

For Insertion sort, time through code: 21.87396804 milliseconds.

For Selection sort, time through code: 15.79157095 milliseconds.

Selection sort is taking less time as compared to Insertion sort.

Hence Proved

(c). For Random Array:

Take $n = 2000$

In the screenshot, when I have taken:

Experiments: 100

Time Complexity of Selection Sort: $\frac{1}{2} N^2$

Time Complexity of Insertion Sort: $\frac{1}{4} N^2$

According to observation:

For Selection sort, time through code: 15.77913023 milliseconds.

For Insertion sort, time through code: 13.75097121 milliseconds.

Insertion Sort is better as it is taking less time and have time complexity as $\frac{1}{4} N^2$ and $\frac{1}{2} N^2$ respectively.

Hence Proved

(a). For Partial Sort Array:

Take $n = 2000$

In the screenshot, when I have taken:

Experiments: 100

Time Complexity of Selection Sort: N^2

Time Complexity of Insertion Sort: N

According to observation:

For Selection sort, time through code: 16.03990026 milliseconds.

For Insertion sort, time through code: 4.43496046 milliseconds.

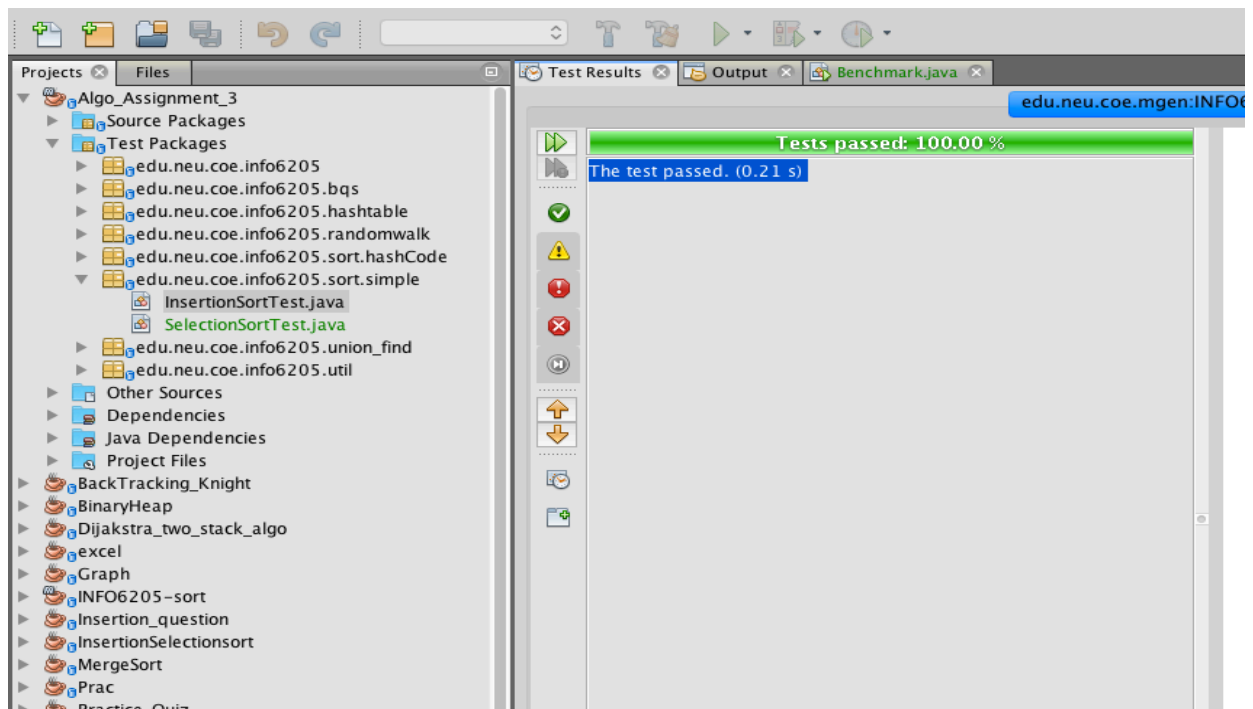
Insertion Sort is taking very less time as compared to Selection sort as its complexity is very lower than Selection sort.

Hence Proved

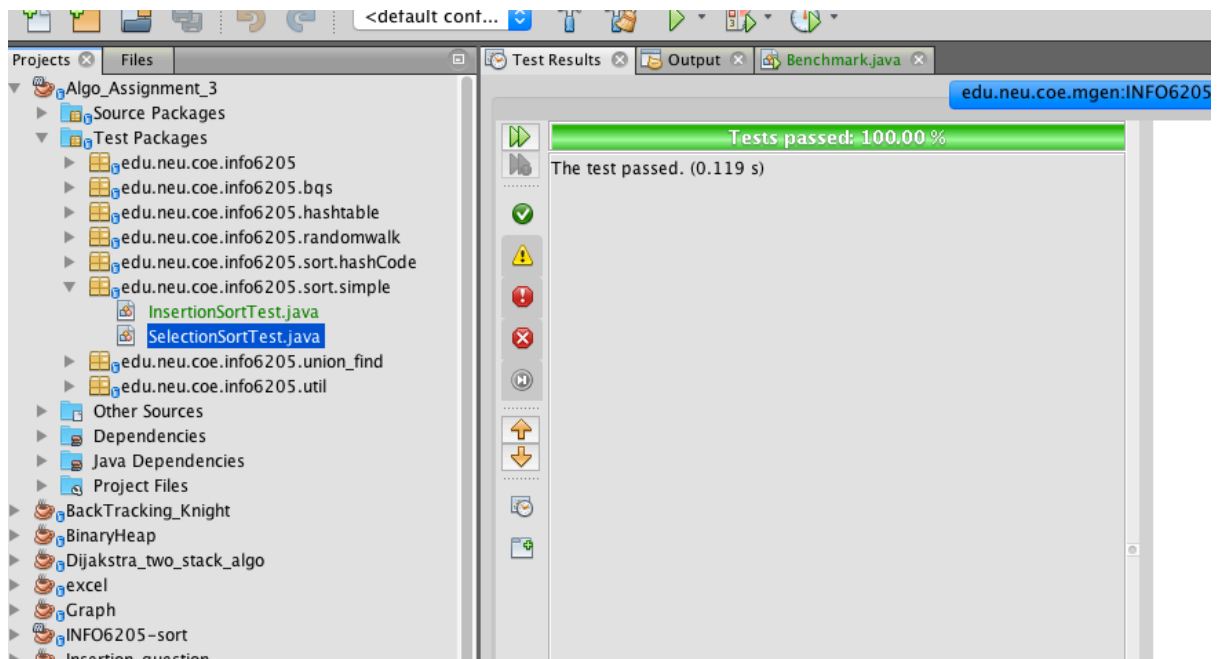
Goel, Ajay: Nu Id (001897443)

Test Cases: NETBEANS

- **Insertion Sort**



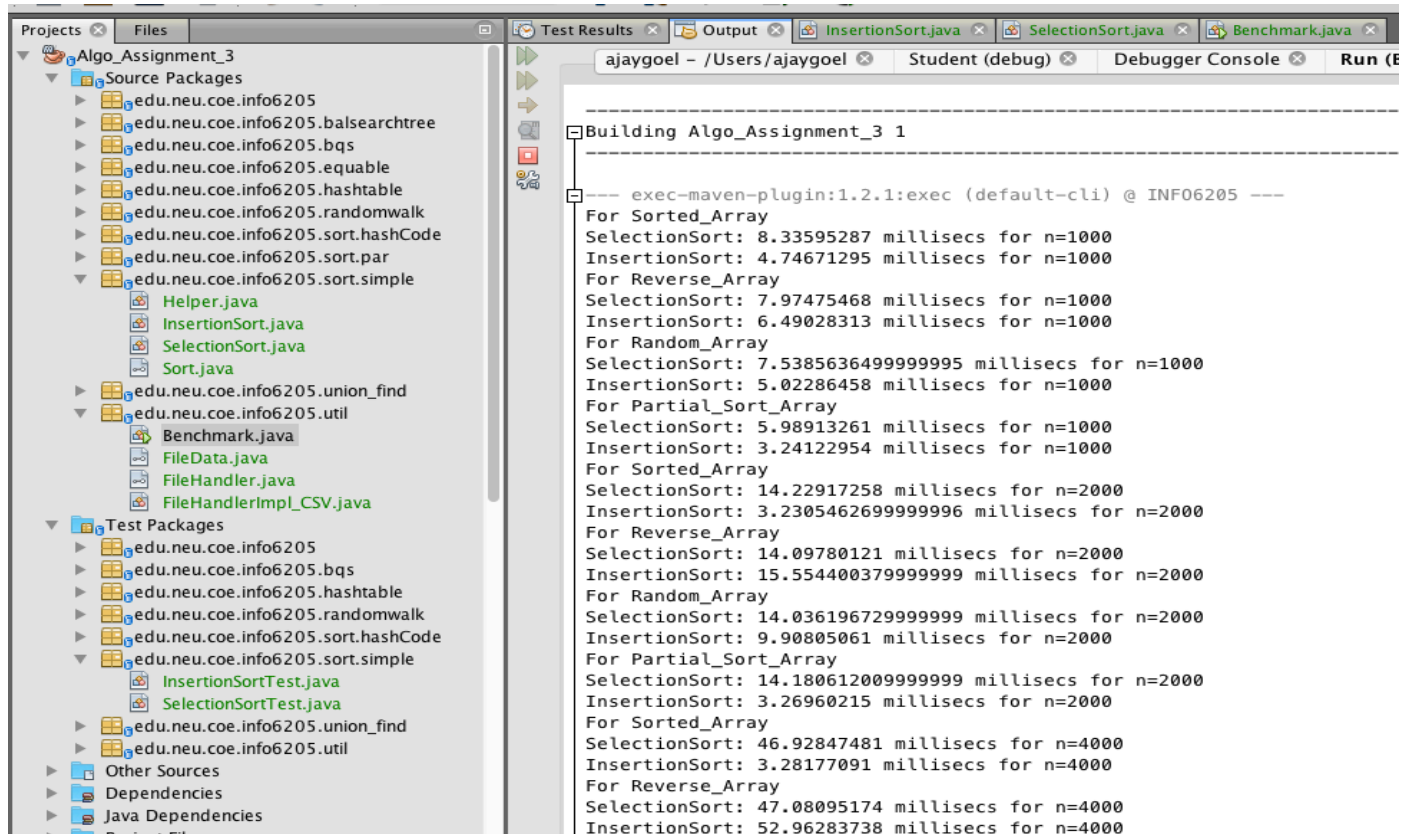
- **Selection Sort**



OUTPUT: NETBEANS

Result:

Goel, Ajay: Nu Id (001897443)



The screenshot displays an IDE interface with a project named 'Algo_Assignment_3'. The left sidebar shows the project structure, including source packages and test packages. The right sidebar shows the output of a Maven exec plugin, which has executed benchmarks for SelectionSort and InsertionSort across various array types and sizes.

Source Packages:

- edu.neu.coe.info6205
- edu.neu.coe.info6205.balsearchtree
- edu.neu.coe.info6205.bqs
- edu.neu.coe.info6205.equable
- edu.neu.coe.info6205.hashtable
- edu.neu.coe.info6205.randomwalk
- edu.neu.coe.info6205.sort.hashCode
- edu.neu.coe.info6205.sort.par
- edu.neu.coe.info6205.sort.simple
 - Helper.java
 - InsertionSort.java
 - SelectionSort.java
 - Sort.java
- edu.neu.coe.info6205.union_find
- edu.neu.coe.info6205.util
 - Benchmark.java
 - FileData.java
 - FileHandler.java
 - FileHandlerImpl_CSV.java

Test Packages:

- edu.neu.coe.info6205
- edu.neu.coe.info6205.bqs
- edu.neu.coe.info6205.hashtable
- edu.neu.coe.info6205.randomwalk
- edu.neu.coe.info6205.sort.hashCode
- edu.neu.coe.info6205.sort.simple
 - InsertionSortTest.java
 - SelectionSortTest.java
- edu.neu.coe.info6205.union_find
- edu.neu.coe.info6205.util

Output:

```
Building Algo_Assignment_3 1
--- exec-maven-plugin:1.2.1:exec (default-cli) @ INF06205 ---
For Sorted_Array
SelectionSort: 8.33595287 millisecs for n=1000
InsertionSort: 4.74671295 millisecs for n=1000
For Reverse_Array
SelectionSort: 7.97475468 millisecs for n=1000
InsertionSort: 6.49028313 millisecs for n=1000
For Random_Array
SelectionSort: 7.5385636499999995 millisecs for n=1000
InsertionSort: 5.02286458 millisecs for n=1000
For Partial_Sort_Array
SelectionSort: 5.98913261 millisecs for n=1000
InsertionSort: 3.24122954 millisecs for n=1000
For Sorted_Array
SelectionSort: 14.22917258 millisecs for n=2000
InsertionSort: 3.2305462699999996 millisecs for n=2000
For Reverse_Array
SelectionSort: 14.09780121 millisecs for n=2000
InsertionSort: 15.554400379999999 millisecs for n=2000
For Random_Array
SelectionSort: 14.036196729999999 millisecs for n=2000
InsertionSort: 9.90805061 millisecs for n=2000
For Partial_Sort_Array
SelectionSort: 14.180612009999999 millisecs for n=2000
InsertionSort: 3.26960215 millisecs for n=2000
For Sorted_Array
SelectionSort: 46.92847481 millisecs for n=4000
InsertionSort: 3.28177091 millisecs for n=4000
For Reverse_Array
SelectionSort: 47.08095174 millisecs for n=4000
InsertionSort: 52.96283738 millisecs for n=4000
```