Lan Gao(001568670)

Program Structures & Algorithms

Fall 2021

Assignment No 2

- ## Task

(Part 1) You are to implement three methods of a class called *Timer*. Please see the skeleton class that I created in the repository. *Timer* is invoked from a class called *Benchmark_Timer* which implements the *Benchmark* interface.

(Part 2) Implement *InsertionSort* (in the *InsertionSort* class) by simply looking up the insertion code used by *Arrays.sort.* You should use the *helper.swap* method although you could also just copy that from the same source code. You should of course run the unit tests in *InsertionSortTest*.

(Part 3) Implement a main program (or you could do it via your own unit tests) to actually run the following benchmarks: measure the running times of this sort, using four different initial array ordering situations: random, ordered, partially-ordered and reverse-ordered. I suggest that your arrays to be sorted are of type *Integer*. Use the doubling method for choosing *n* and test for at least five values of *n.* Draw any conclusions from your observations regarding the order of growth.

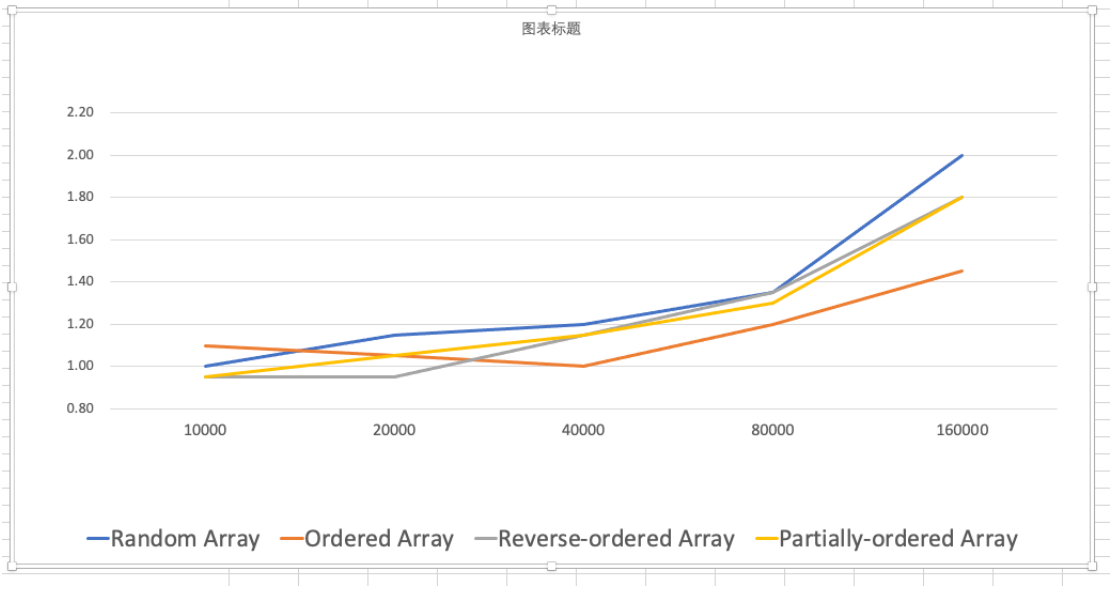
- ## Conclusion

With n increase, the running time of the insertion sort goes up too and the relationship between the time of sorting an ordered array and n is nearly linear.

In most situation, the running-time relationship among those arrays shows like: ” Random Array” > ”Reverse-ordered Array” > ”Partially-ordered Array” > ”Ordered Array”. But when n is small, it couldn’t appear to be such a stable relationship(like n under 40000).

- ## Graphical Representation

	10000	20000	40000	80000	160000
Random Array	1.00	1.15	1.20	1.35	2.00
Ordered Array	1.10	1.05	1.00	1.20	1.45
Reverse-ordered Array	0.95	0.95	1.15	1.35	1.80
Partially-ordered Array	0.95	1.05	1.15	1.30	1.80



- **Evidence to support the conclusion**

1 . Output

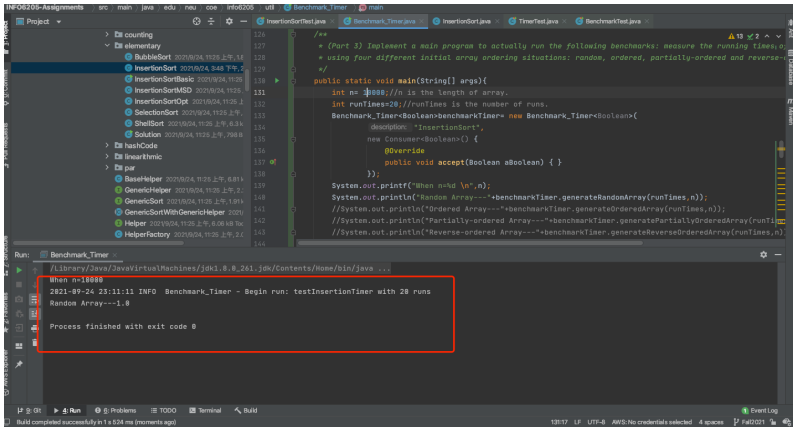
Table

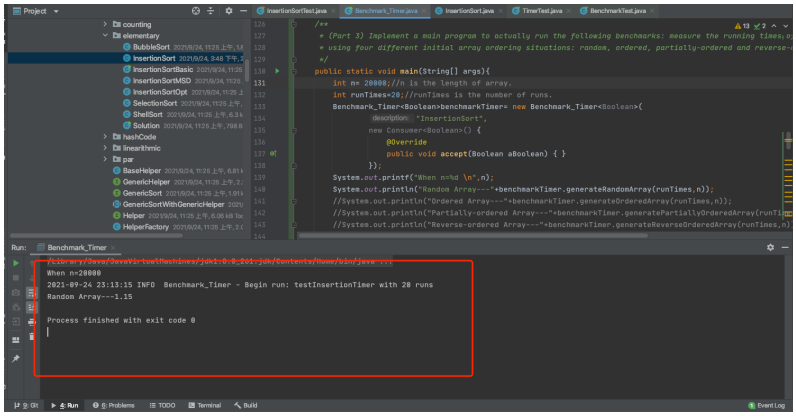
RunTimes = 20

Array type \ n	10000	20000	40000	80000	160000
Random Array	1.00	1.15	1.20	1.35	2.00
Ordered Array	1.10	1.05	1.00	1.20	1.45
Reverse-ordered Array	0.95	0.95	1.15	1.35	1.80
Partially-ordered Array	0.95	1.05	1.15	1.30	1.80

2. Snapshots

For Random Array:





```
/**
 * (Part 3) Implement a main program to actually run the following benchmarks: measure the running times, o
 * using four different initial array ordering situations: random, ordered, partially-ordered and reverse-
 */
public static void main(String[] args){
    int n= 40000;//n is the length of array.
    int runTimes=20;//runTimes is the number of runs.
    Benchmark_Timer<Boolean>benchmarkTimer= new Benchmark_Timer<Boolean>(
            description: "InsertionSort",
            new Consumer<Boolean>() {
                @Override
                public void accept(Boolean aBoolean) { }
            });
    System.out.printf("When n=%d \n",n);
    System.out.println("Random Array---"+benchmarkTimer.generateRandomArray(runTimes,n));
    //System.out.println("Ordered Array---"+benchmarkTimer.generateOrderedArray(runTimes,n));
    //System.out.println("Partially-ordered Array---"+benchmarkTimer.generatePartiallyOrderedArray(runTi
    //System.out.println("Reverse-ordered Array---"+benchmarkTimer.generateReverseOrderedArray(runTimes,n)
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_261.jdk/Contents/Home/bin/java ...
When n=40000
2021-09-24 23:14:22 INFO  Benchmark_Timer - Begin run: testInsertionTimer with 20 runs
Random Array---1.2

Process finished with exit code 0
```

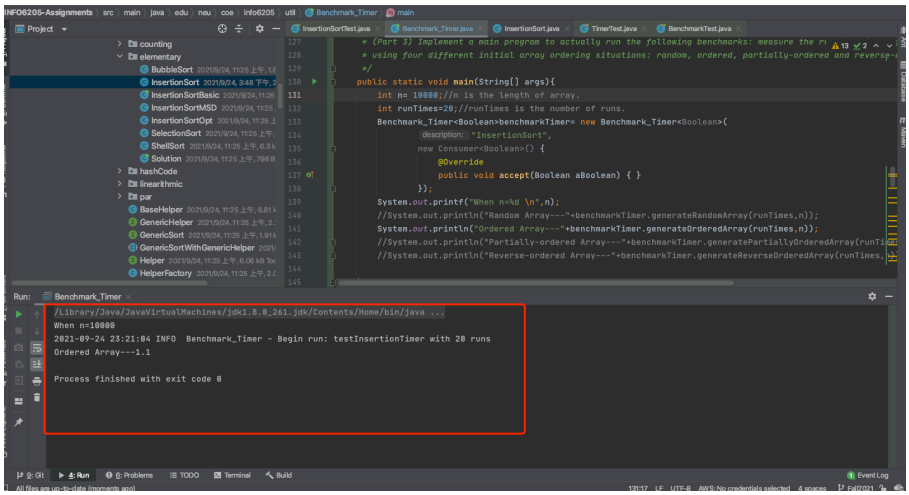
```
/**
 * (Part 3) Implement a main program to actually run the following benchmarks: measure the running times, o
 * using four different initial array ordering situations: random, ordered, partially-ordered and reverse-
 */
public static void main(String[] args){
    int n= 80000;//n is the length of array.
    int runTimes=20;//runTimes is the number of runs.
    Benchmark_Timer<Boolean>benchmarkTimer= new Benchmark_Timer<Boolean>(
            description: "InsertionSort",
            new Consumer<Boolean>() {
                @Override
                public void accept(Boolean aBoolean) { }
            });
    System.out.printf("When n=%d \n",n);
    System.out.println("Random Array---"+benchmarkTimer.generateRandomArray(runTimes,n));
    //System.out.println("Ordered Array---"+benchmarkTimer.generateOrderedArray(runTimes,n));
    //System.out.println("Partially-ordered Array---"+benchmarkTimer.generatePartiallyOrderedArray(runTi
    //System.out.println("Reverse-ordered Array---"+benchmarkTimer.generateReverseOrderedArray(runTimes,n)
```

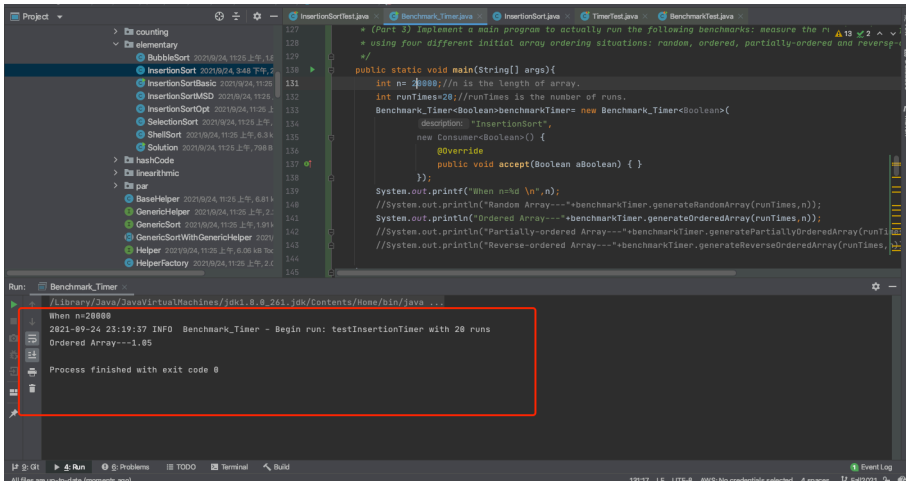
```
/Library/Java/JavaVirtualMachines/jdk1.8.0_261.jdk/Contents/Home/bin/java ...
When n=80000
2021-09-24 23:14:58 INFO  Benchmark_Timer - Begin run: testInsertionTimer with 20 runs
Random Array---1.35

Process finished with exit code 0
```

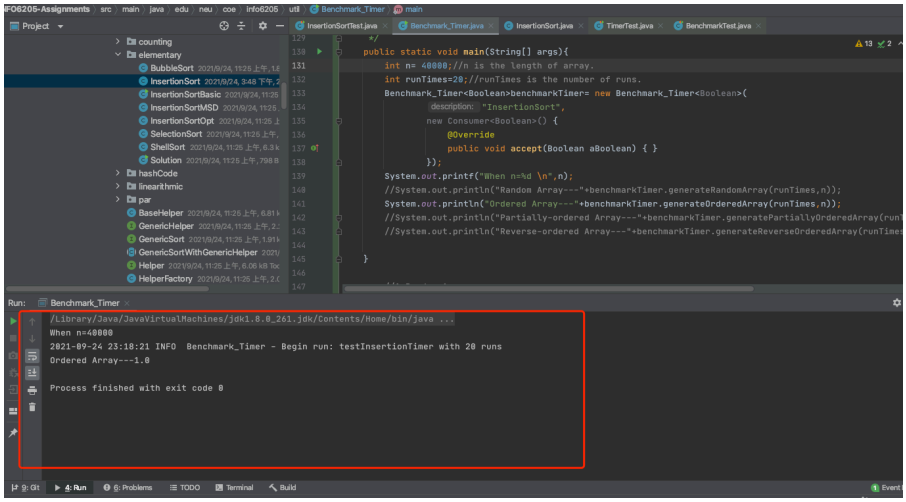
```
/**
 * (Part 3) Implement a main program to actually run the following benchmarks: measure the running times, o
 * using four different initial array ordering situations: random, ordered, partially-ordered and reverse-
 */
public static void main(String[] args){
    int n= 160000;//n is the length of array.
    int runTimes=20;//runTimes is the number of runs.
    Benchmark_Timer<Boolean>benchmarkTimer= new Benchmark_Timer<Boolean>(
            description: "InsertionSort",
            new Consumer<Boolean>() {
                @Override
                public void accept(Boolean aBoolean) { }
            });
    System.out.printf("When n=%d \n",n);
    System.out.println("Random Array---"+benchmarkTimer.generateRandomArray(runTimes,n));
    //System.out.println("Ordered Array---"+benchmarkTimer.generateOrderedArray(runTimes,n));
    //System.out.println("Partially-ordered Array---"+benchmarkTimer.generatePartiallyOrderedArray(runTi
    //System.out.println("Reverse-ordered Array---"+benchmarkTimer.generateReverseOrderedArray(runTimes,n)
```

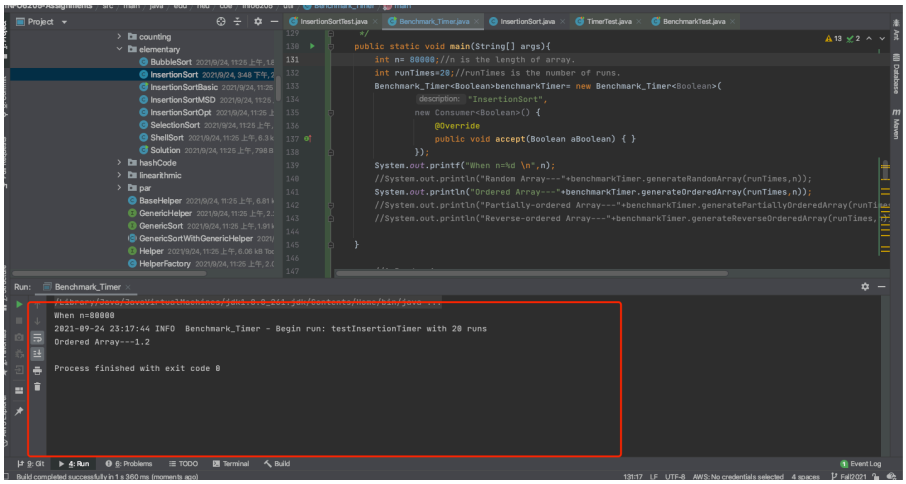
```
/Library/Java/JavaVirtualMachines/jdk1.8.0_261.jdk/Contents/Home/bin/java ...
When n=160000
2021-09-24 23:15:46 INFO  Benchmark_Timer - Begin run: testInsertionTimer with 20 runs
Random Array---2.8

Process finished with exit code 0
```

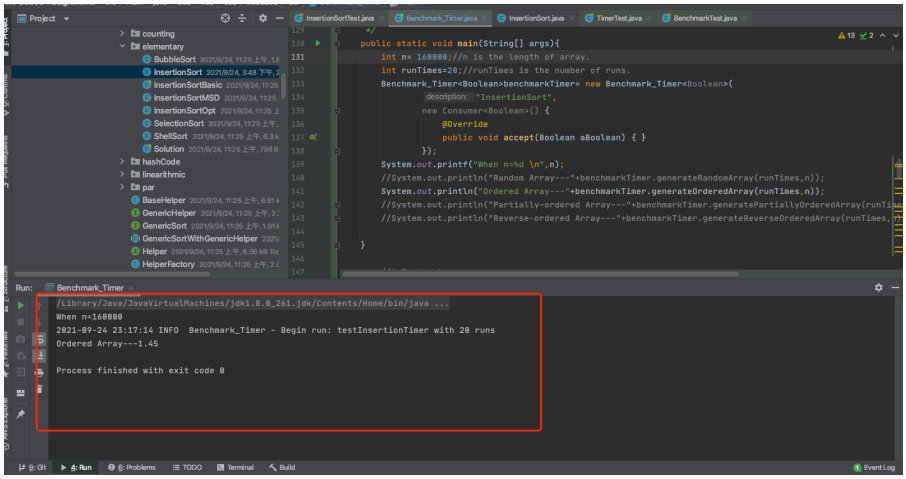
For Ordered Array:



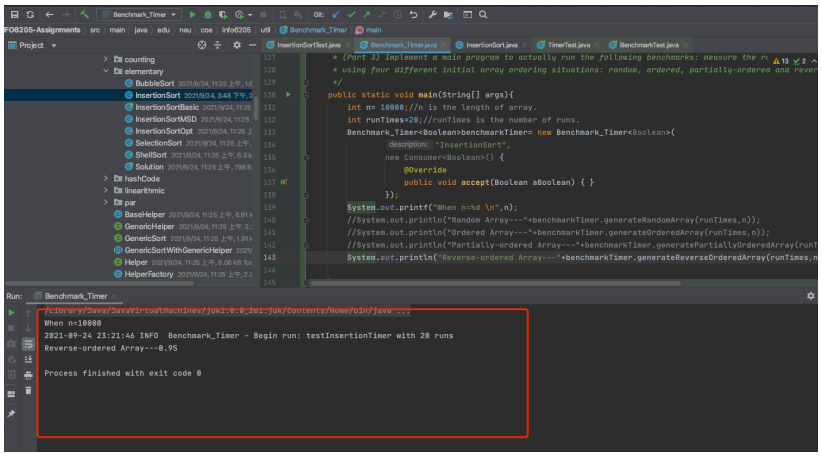


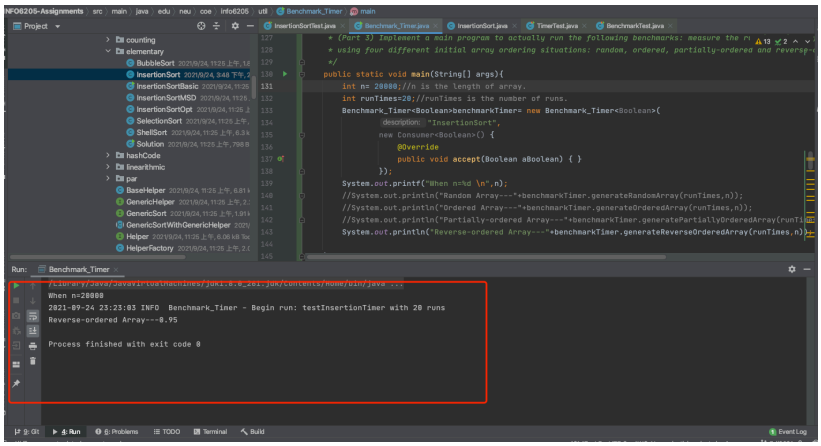


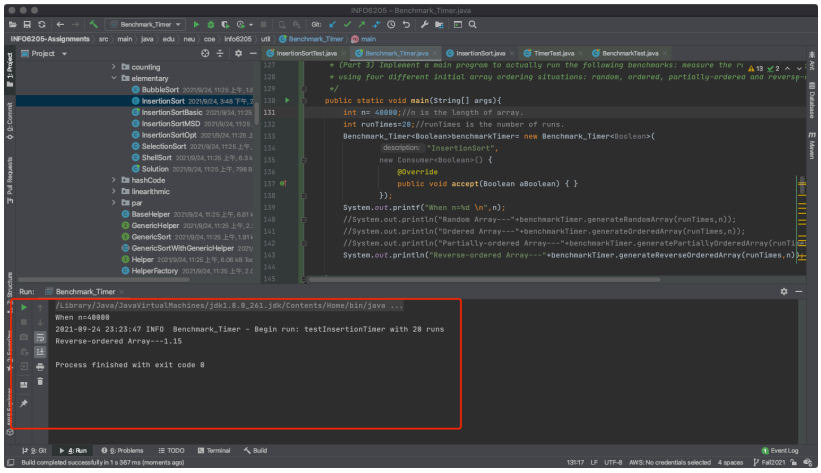


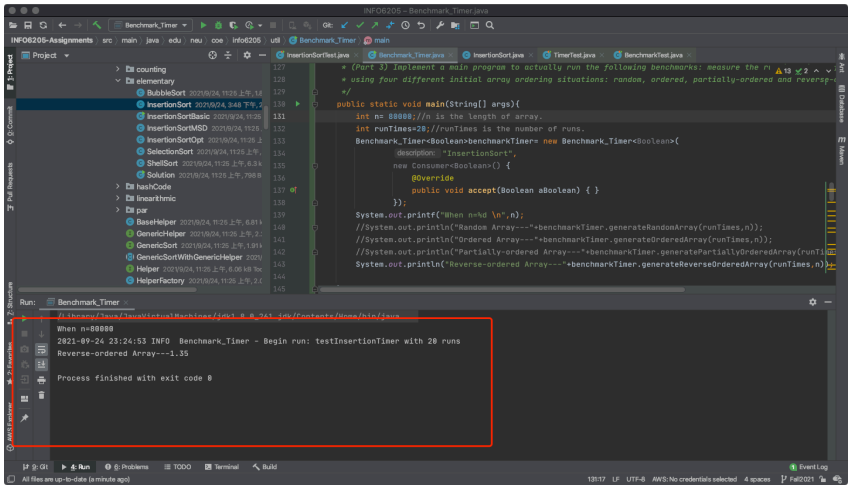


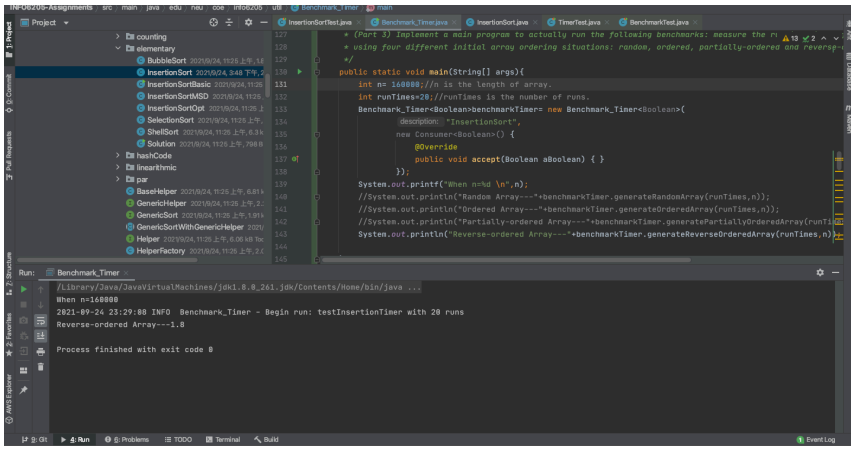
For Reverse-ordered Array:



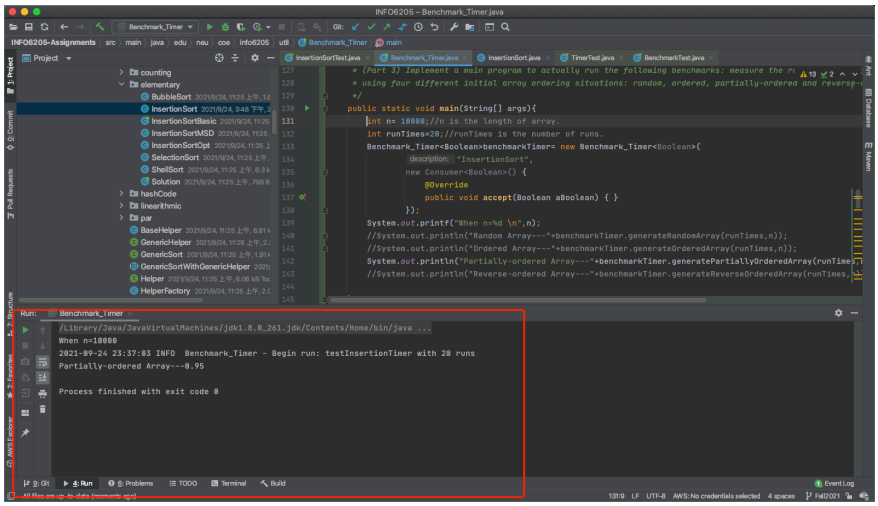

INFO6205-Assignments
src
main
java
edu
neu
coe
info6205
util
Benchmark_Timer
main
Project
counting
elementary
BubbleSort
InsertionSort
InsertionSortBasic
InsertionSortMSD
InsertionSortOpt
SelectionSort
ShellSort
Solution
hashCode
linearithmic
par
BaseHelper
GenericHelper
GenericSort
GenericSortWithGenericHelper
Helper
HelperFactory
InsertionSortTest.java
Benchmark_Timer.java
InsertionSort.java
TimerTest.java
BenchmarkTest.java
* (Part 3) Implement a main program to actually run the following benchmarks: measure the ru
* using four different initial array ordering situations: random, ordered, partially-ordered and reverse-
*/
public static void main(String[] args){
int n= 20000;//n is the length of array.
int runTimes=20;//runTimes is the number of runs.
Benchmark_Timer<Boolean>benchmarkTimer= new Benchmark_Timer<Boolean>(
description: "InsertionSort",
new Consumer<Boolean>() {
@Override
public void accept(Boolean aBoolean) { }
});
System.out.printf("When n=%d \n",n);
//System.out.println("Random Array---"+benchmarkTimer.generateRandomArray(runTimes,n));
//System.out.println("Ordered Array---"+benchmarkTimer.generateOrderedArray(runTimes,n));
//System.out.println("Partially-ordered Array---"+benchmarkTimer.generatePartiallyOrderedArray(runTi
System.out.println("Reverse-ordered Array---"+benchmarkTimer.generateReverseOrderedArray(runTimes,n)
Run:
Benchmark_Timer
/Library/Java/JavaVirtualMachines/jdk1.8.0_261.jdk/Contents/Home/bin/java ...
When n=20000
2021-09-24 23:23:03 INFO Benchmark_Timer - Begin run: testInsertionTimer with 20 runs
Reverse-ordered Array---0.95
Process finished with exit code 0
Event Log

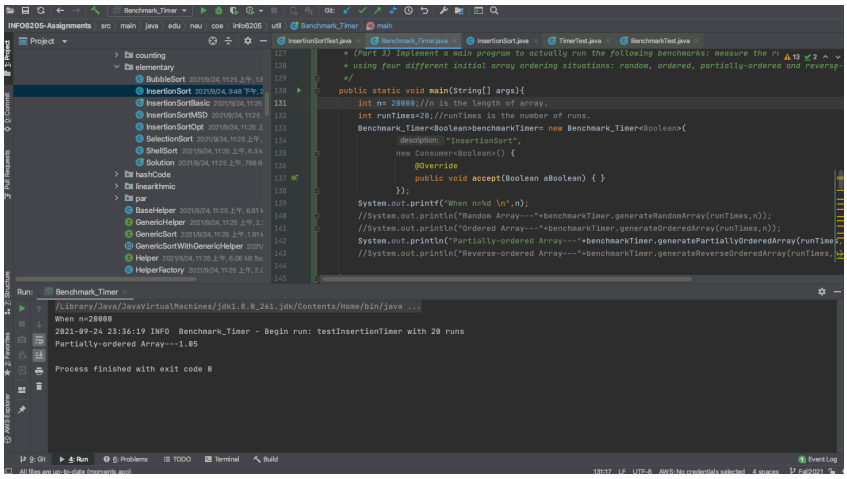

INFO6205 – Benchmark_Timer.java
INFO6205-Assignments
src
main
java
edu
neu
coe
info6205
util
Benchmark_Timer
main
Project
counting
elementary
BubbleSort
InsertionSort
InsertionSortBasic
InsertionSortMSD
InsertionSortOpt
SelectionSort
ShellSort
Solution
hashCode
linearithmic
par
BaseHelper
GenericHelper
GenericSort
GenericSortWithGenericHelper
Helper
HelperFactory
InsertionSortTest.java
Benchmark_Timer.java
InsertionSort.java
TimerTest.java
BenchmarkTest.java
* (Part 3) Implement a main program to actually run the following benchmarks: measure the ru
* using four different initial array ordering situations: random, ordered, partially-ordered and reverse-
*/
public static void main(String[] args){
int n= 40000;//n is the length of array.
int runTimes=20;//runTimes is the number of runs.
Benchmark_Timer<Boolean>benchmarkTimer= new Benchmark_Timer<Boolean>(
description: "InsertionSort",
new Consumer<Boolean>() {
@Override
public void accept(Boolean aBoolean) { }
});
System.out.printf("When n=%d \n",n);
//System.out.println("Random Array---"+benchmarkTimer.generateRandomArray(runTimes,n));
//System.out.println("Ordered Array---"+benchmarkTimer.generateOrderedArray(runTimes,n));
//System.out.println("Partially-ordered Array---"+benchmarkTimer.generatePartiallyOrderedArray(runTi
System.out.println("Reverse-ordered Array---"+benchmarkTimer.generateReverseOrderedArray(runTimes,n)
Run:
Benchmark_Timer
/Library/Java/JavaVirtualMachines/jdk1.8.0_261.jdk/Contents/Home/bin/java ...
When n=40000
2021-09-24 23:23:47 INFO Benchmark_Timer - Begin run: testInsertionTimer with 20 runs
Reverse-ordered Array---1.15
Process finished with exit code 0
Event Log
Build completed successfully in 1 s 367 ms (moments ago)

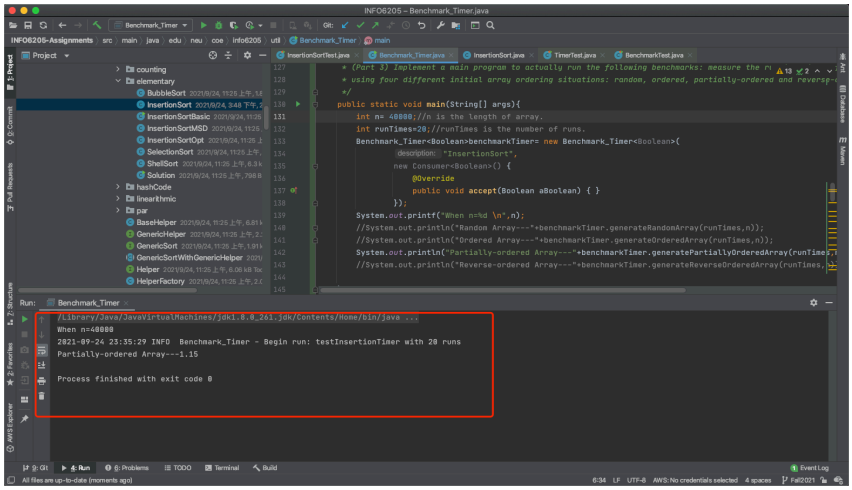

INFO6205 – Benchmark_Timer.java
INFO6205-Assignments
src
main
java
edu
neu
coe
info6205
util
Benchmark_Timer
main
Project
counting
elementary
BubbleSort
InsertionSort
InsertionSortBasic
InsertionSortMSD
InsertionSortOpt
SelectionSort
ShellSort
Solution
hashCode
linearithmic
par
BaseHelper
GenericHelper
GenericSort
GenericSortWithGenericHelper
Helper
HelperFactory
InsertionSortTest.java
Benchmark_Timer.java
InsertionSort.java
TimerTest.java
BenchmarkTest.java
* (Part 3) Implement a main program to actually run the following benchmarks: measure the ru
* using four different initial array ordering situations: random, ordered, partially-ordered and reverse-
*/
public static void main(String[] args){
int n= 80000;//n is the length of array.
int runTimes=20;//runTimes is the number of runs.
Benchmark_Timer<Boolean>benchmarkTimer= new Benchmark_Timer<Boolean>(
description: "InsertionSort",
new Consumer<Boolean>() {
@Override
public void accept(Boolean aBoolean) { }
});
System.out.printf("When n=%d \n",n);
//System.out.println("Random Array---"+benchmarkTimer.generateRandomArray(runTimes,n));
//System.out.println("Ordered Array---"+benchmarkTimer.generateOrderedArray(runTimes,n));
//System.out.println("Partially-ordered Array---"+benchmarkTimer.generatePartiallyOrderedArray(runTi
System.out.println("Reverse-ordered Array---"+benchmarkTimer.generateReverseOrderedArray(runTimes,n)
Run:
Benchmark_Timer
/Library/Java/JavaVirtualMachines/jdk1.8.0_261.jdk/Contents/Home/bin/java ...
When n=80000
2021-09-24 23:24:53 INFO Benchmark_Timer - Begin run: testInsertionTimer with 20 runs
Reverse-ordered Array---1.35
Process finished with exit code 0
Event Log
All files are up-to-date (a minute ago)

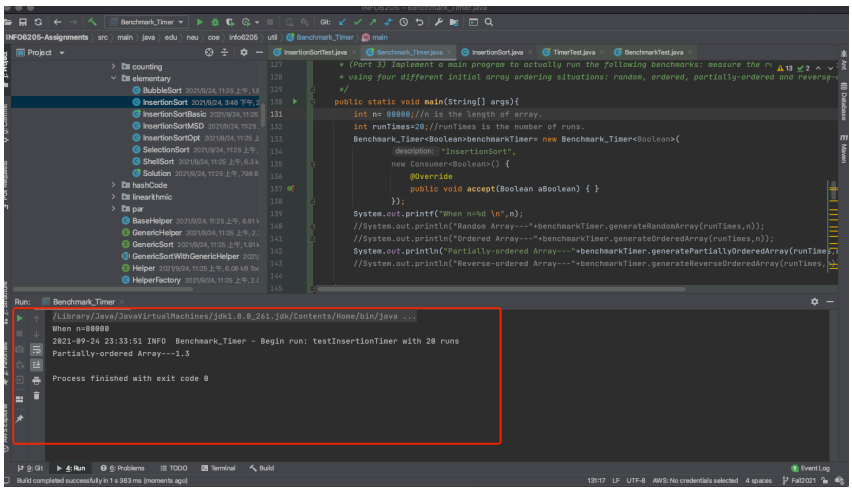


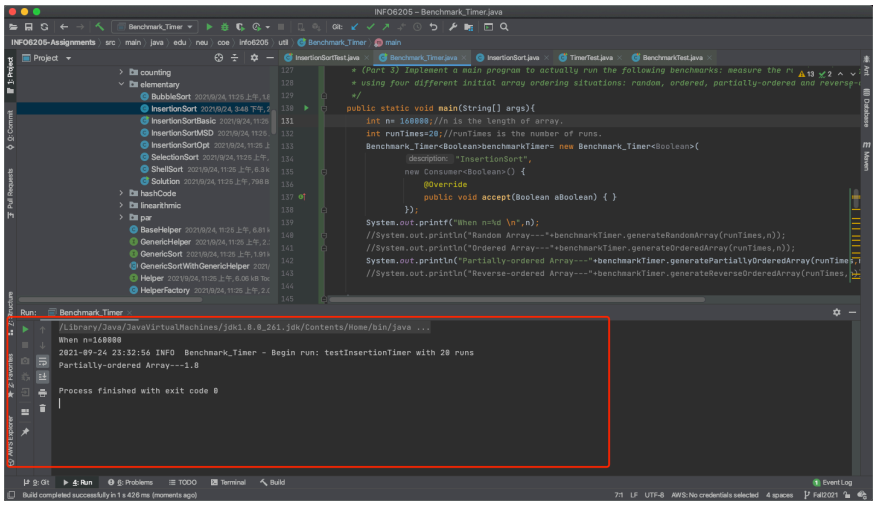
For Partially-ordered Array:





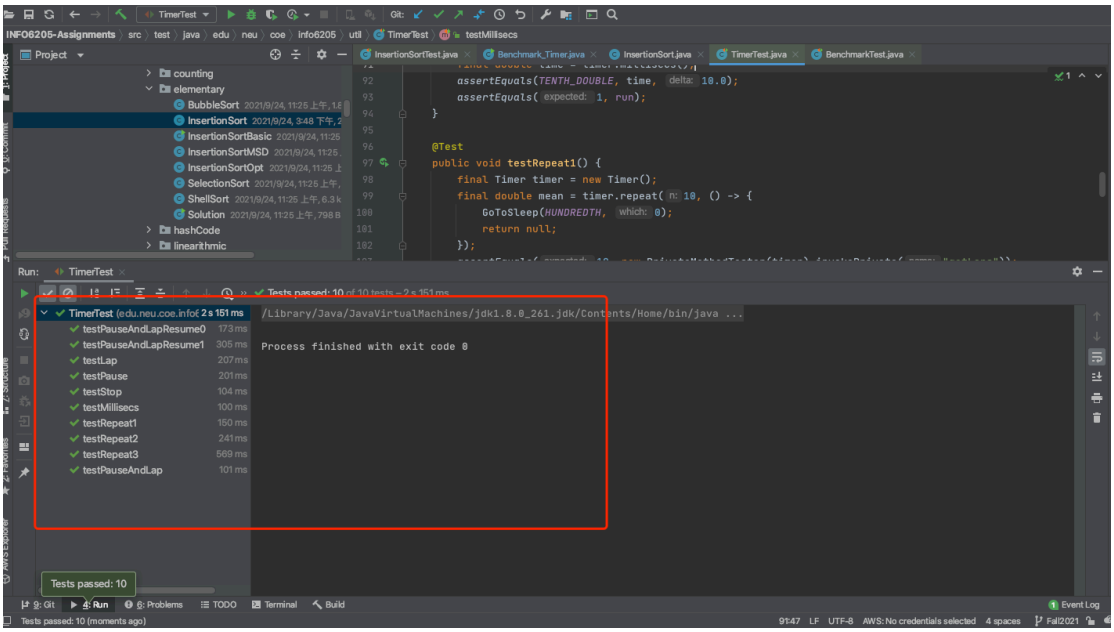

INFO6205 – Benchmark_Timer.java
* (Part 3) Implement a main program to actually run the following benchmarks: measure the ru
* using four different initial array ordering situations: random, ordered, partially-ordered and reversp
*/
public static void main(String[] args){
int n= 40000;//n is the length of array.
int runTimes=20;//runTimes is the number of runs.
Benchmark_Timer<Boolean>benchmarkTimer= new Benchmark_Timer<Boolean>{
description: "InsertionSort",
new Consumer<Boolean>() {
@Override
public void accept(Boolean aBoolean) { }
});
System.out.printf("When n=%d \n",n);
//System.out.println("Random Array---"+benchmarkTimer.generateRandomArray(runTimes,n));
//System.out.println("Ordered Array---"+benchmarkTimer.generateOrderedArray(runTimes,n));
System.out.println("Partially-ordered Array---"+benchmarkTimer.generatePartiallyOrderedArray(runTimes,
//System.out.println("Reverse-ordered Array---"+benchmarkTimer.generateReverseOrderedArray(runTimes,
Run: Benchmark_Timer
/Library/Java/JavaVirtualMachines/jdk1.8.0_261.jdk/Contents/Home/bin/java ...
When n=40000
2021-09-24 23:35:29 INFO Benchmark_Timer - Begin run: testInsertionTimer with 20 runs
Partially-ordered Array---1.15
Process finished with exit code 0


* (Part 3) Implement a main program to actually run the following benchmarks: measure the ru
* using four different initial array ordering situations: random, ordered, partially-ordered and reversp
*/
public static void main(String[] args){
int n= 80000;//n is the length of array.
int runTimes=20;//runTimes is the number of runs.
Benchmark_Timer<Boolean>benchmarkTimer= new Benchmark_Timer<Boolean>{
description: "InsertionSort",
new Consumer<Boolean>() {
@Override
public void accept(Boolean aBoolean) { }
});
System.out.printf("When n=%d \n",n);
//System.out.println("Random Array---"+benchmarkTimer.generateRandomArray(runTimes,n));
//System.out.println("Ordered Array---"+benchmarkTimer.generateOrderedArray(runTimes,n));
System.out.println("Partially-ordered Array---"+benchmarkTimer.generatePartiallyOrderedArray(runTimes,
//System.out.println("Reverse-ordered Array---"+benchmarkTimer.generateReverseOrderedArray(runTimes,
Run: Benchmark_Timer
/Library/Java/JavaVirtualMachines/jdk1.8.0_261.jdk/Contents/Home/bin/java ...
When n=80000
2021-09-24 23:33:51 INFO Benchmark_Timer - Begin run: testInsertionTimer with 20 runs
Partially-ordered Array---1.3
Process finished with exit code 0

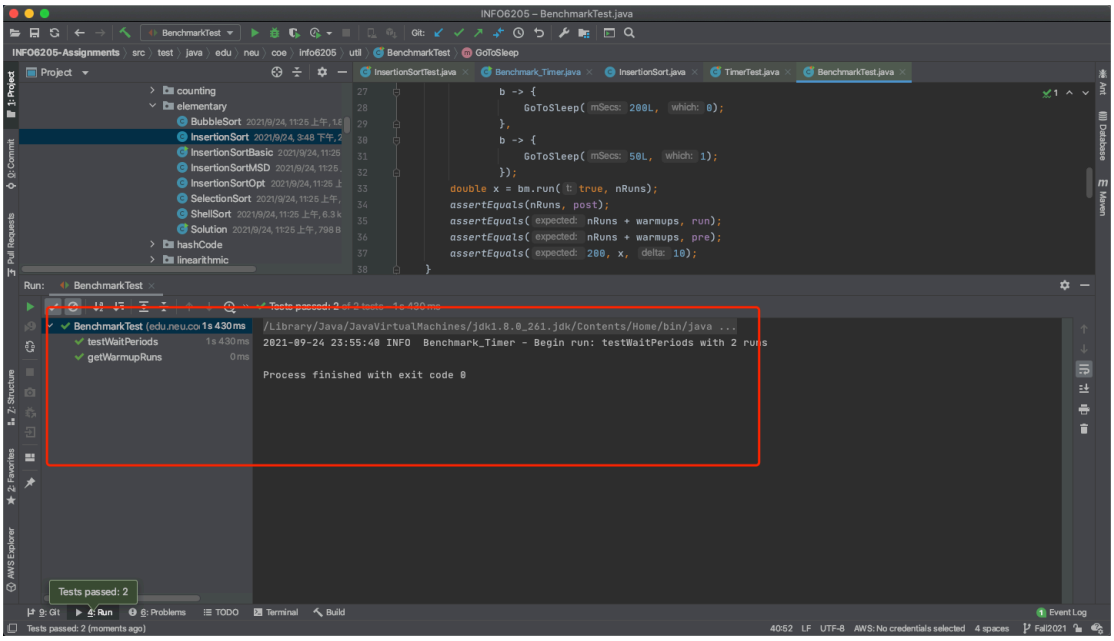

INFO6205 – Benchmark_Timer.java
* (Part 3) Implement a main program to actually run the following benchmarks: measure the ru
* using four different initial array ordering situations: random, ordered, partially-ordered and reversp
*/
public static void main(String[] args){
int n= 160000;//n is the length of array.
int runTimes=20;//runTimes is the number of runs.
Benchmark_Timer<Boolean>benchmarkTimer= new Benchmark_Timer<Boolean>{
description: "InsertionSort",
new Consumer<Boolean>() {
@Override
public void accept(Boolean aBoolean) { }
});
System.out.printf("When n=%d \n",n);
//System.out.println("Random Array---"+benchmarkTimer.generateRandomArray(runTimes,n));
//System.out.println("Ordered Array---"+benchmarkTimer.generateOrderedArray(runTimes,n));
System.out.println("Partially-ordered Array---"+benchmarkTimer.generatePartiallyOrderedArray(runTimes,
//System.out.println("Reverse-ordered Array---"+benchmarkTimer.generateReverseOrderedArray(runTimes,
Run: Benchmark_Timer
/Library/Java/JavaVirtualMachines/jdk1.8.0_261.jdk/Contents/Home/bin/java ...
When n=160000
2021-09-24 23:32:56 INFO Benchmark_Timer - Begin run: testInsertionTimer with 20 runs
Partially-ordered Array---1.8
Process finished with exit code 0

- **Unit test result**

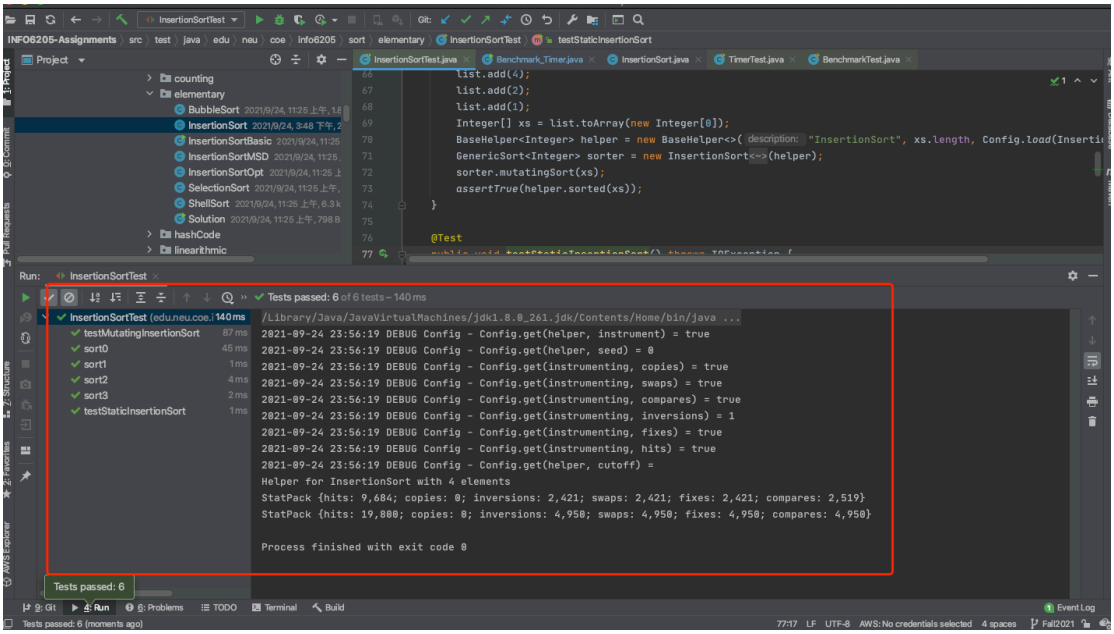
TimmerTest



BenchmarkTest



InsertionSortTest


InsertionSortTest
INFO6205-Assignments / src / test / java / edu / neu / coe / info6205 / sort / elementary / InsertionSortTest / testStaticInsertionSort
Project
counting
elementary
BubbleSort
InsertionSort
InsertionSortBasic
InsertionSortMSD
InsertionSortOpt
SelectionSort
ShellSort
Solution
hashCode
linearithmic
InsertionSortTest.java
Benchmark_Timer.java
InsertionSort.java
TimerTest.java
BenchmarkTest.java
list.add(4);
list.add(2);
list.add(1);
Integer[] xs = list.toArray(new Integer[0]);
BaseHelper<Integer> helper = new BaseHelper<>(description: "InsertionSort", xs.length, Config.load(Insertio
GenericSort<Integer> sorter = new InsertionSort<>(helper);
sorter.mutatingSort(xs);
assertTrue(helper.sorted(xs));
}
@Test
Run: InsertionSortTest
Tests passed: 6 of 6 tests – 140 ms
InsertionSortTest (edu.neu.coe.i 140 ms
testMutatingInsertionSort 87 ms
sort0 45 ms
sort1 1 ms
sort2 4 ms
sort3 2 ms
testStaticInsertionSort 1 ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_261.jdk/Contents/Home/bin/java ...
2021-09-24 23:56:19 DEBUG Config - Config.get(helper, instrument) = true
2021-09-24 23:56:19 DEBUG Config - Config.get(helper, seed) = 0
2021-09-24 23:56:19 DEBUG Config - Config.get(instrumenting, copies) = true
2021-09-24 23:56:19 DEBUG Config - Config.get(instrumenting, swaps) = true
2021-09-24 23:56:19 DEBUG Config - Config.get(instrumenting, compares) = true
2021-09-24 23:56:19 DEBUG Config - Config.get(instrumenting, inversions) = 1
2021-09-24 23:56:19 DEBUG Config - Config.get(instrumenting, fixes) = true
2021-09-24 23:56:19 DEBUG Config - Config.get(instrumenting, hits) = true
2021-09-24 23:56:19 DEBUG Config - Config.get(helper, cutoff) =
Helper for InsertionSort with 4 elements
StatPack {hits: 9,684; copies: 0; inversions: 2,421; swaps: 2,421; fixes: 2,421; compares: 2,519}
StatPack {hits: 19,800; copies: 0; inversions: 4,950; swaps: 4,950; fixes: 4,950; compares: 4,950}
Process finished with exit code 0
Tests passed: 6
9: Git
4: Run
6: Problems
TODO
Terminal
Build
Event Log
Tests passed: 6 (moments ago)
77:17 LF UTF-8 AWS: No credentials selected 4 spaces Fall2021