**Brandon Bardwell**

**Discrete Structures**

**2-7-2020**

**Programming Practicum 2: Algorithm Running Time**

**A single question describing what the question is asking**

I am tasked with writing a program that fills and sorts an array full of numbers, then displays how long it took to do that.

**A list of other useful resources and references.**

I made note of the two references that are listed in the code, they helped me understand it a lot better.

<https://www.baeldung.com/java-math-pow>

<https://stackoverflow.com/questions/180158/how-do-i-time-a-methods-execution-in-java>

**A list of key assumptions and general observations**

I understood the start time and the end time and how that would work, I had to use the reference to learn about the nano time and how it was divided.

**A copy of your calculations and other work**

I have no other work.

**Your code**

**import** java.util.Arrays;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

/\*\* Plot sort time vs n \*/

/\* for n = 10,

\* 100, 1,000, 10,000, 100,000,

\* 1,000,000, 10,000,000 and 100,000,000.

\*/

// notes: help from:

//https://stackoverflow.com/questions/180158/how-do-i-time-a-methods-execution-in-java

// and

// https://www.baeldung.com/java-math-pow

**int** arraySize = 1;

**long** startTime = System.*nanoTime*();

**long** endTime = System.*nanoTime*();

**long** duration = (endTime - startTime);

//divide by 1000000 to get milliseconds.

System.***out***.println("array size, Sort time, Parallel Sort Time ");

**for**(**int** zeroCount = 1; zeroCount < 9; zeroCount++)

{

arraySize = (**int**)Math.*pow*(10, (**double**)zeroCount);

System.***out***.print(arraySize + ", ");

**int**[] ourArray = **new** **int**[arraySize];

**int**[] ourSecondArray = **new** **int**[arraySize];

*fillArray*(ourArray);

*fillArrayCopy*(ourSecondArray, ourArray);

startTime = System.*nanoTime*();

Arrays.*sort*(ourArray);

endTime = System.*nanoTime*();

duration = (endTime - startTime);

//divide by 1000000 to get milliseconds.

System.***out***.print(duration + ", ");

startTime = System.*nanoTime*();

Arrays.*parallelSort*(ourSecondArray);

endTime = System.*nanoTime*();

duration = (endTime - startTime);

//divide by 1000000 to get milliseconds.

System.***out***.print(duration + ", ");

System.***out***.println();

}

System.***out***.println("Thanks for coming to my TED Talk :)");

}

**private** **static** **void** fillArrayCopy(**int**[] ourSecondArray, **int**[] ourArray) {

**for**(**int** index = 0; index < ourArray.length; index++)

{

ourSecondArray[index] = ourArray[index];

}

}

**private** **static** **void** fillArray(**int**[] ourArray) {

**for**(**int** index = 0; index < ourArray.length; index++)

{

ourArray[index] = (**int**)(Math.*random*()\*ourArray.length\*10);

}

}

}

**Your code output**

array size, Sort time, Parallel Sort Time

10, 151800, 8100,

100, 34700, 50600,

1000, 407600, 355100,

10000, 1955800, 8138000,

100000, 12855200, 55548800,

1000000, 95244000, 11265300,

10000000, 949917000, 125453800,

100000000, 10456410000, 1431802000,

Thanks for coming to my TED Talk :)

**A discussion of your solution, and why you believe it is correct**

I believe this solution is correct because it displays everything asked for in the question.