Wei Chen(001562214)

Program Structures & Algorithms

Fall 2021

Assignment No.2

1. 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. If you have the instrument = true setting in test/resources/config.ini, then you will need to use the helper methods for comparing and swapping (so that they properly count the number of swaps/compares). The easiest is to use the helper.swapStableConditional method, continuing if it returns true, otherwise breaking the loop. Alternatively, if you are not using instrumenting, then you can write (or copy) your own compare/swap code. Either way, you must 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.

1. Relationship Conclusion:

when the length of the array is double of the original length, time to sort the array will change to above 3 times and round to 4 times of the original time.

RandomArray and OrderedArray need less time than PartiallyOrderedArray and ReverseOrderedArray. Besides, ReverseOrderedArray needs the most time to sort.

1. Evidence to support the conclusion:

Output:

文本

描述已自动生成

Graphical Representation:

图形用户界面, 图表, 应用程序, 折线图

描述已自动生成

1. Unit Test Result

TimerTest:

图形用户界面, 文本

描述已自动生成

BenchMarkTest:

文本

描述已自动生成