**Program Structures and Algorithms**

**Spring 2023(SEC 03) Assignment 03**

**NAME: Shivam Thabe**

**NUID: 002765286**

**Task: Benchmark (Insertion Sort)**

Part 1:You are to implement three (3) methods (repeat, getClock, and toMillisecs) of a class called Timer.

Part 2: Implement InsertionSort(in the InsertionSort class) by simply looking up the insertion code used byArrays.sort.

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

**Relationship Conclusion:**

The benchmark test was run for the insertion sort algorithm. The algorithm was tested by the doubling method with input array sizes as (1000, 2000, 4000, 8000 and 16000) and each of the input was run for 100 times with 100 iterations. Further, the input was classified in four different types: Ordered Input, partially ordered input, reverse ordered input and random input.

The benchmark results indicate that the average time required to sort the ordered input is the lowest. This is because the input is already sorted, and the number of swaps needed are zero.

The average time for the partially sorted input is greater than the ordered input, however it is lower than the reverse sorted input.

The average time for the randomly sorted input is intermediate between the partially sorted and reverse sorted array.

Also, the average time across all the input types is proportional to the input array size. It can be concluded that the time required to sort the array with insertion sort algorithm increases with the number of unordered elements. i.e., Time required to sort the input: Ordered array < partially ordered < random ordered < reverse ordered

**Evidence to support that conclusion (Timing observations):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Input size** | **Ordered** | **Partially Ordered** | **Random Ordered** | **Reverse Ordered** |
|  | **N** | **Time ms** | **Time ms** | **Time ms** | **Time ms** |
| 1 | 1000 | 0.06802912 | 0.10170917 | 0.02816503 | 0.08838754 |
| 2 | 2000 | 0.01425169 | 0.05872581 | 0.07106128 | 0.07620836 |
| 3 | 4000 | 0.02840876 | 0.19919457 | 0.25910789 | 0.25561085 |
| 4 | 8000 | 0.05748545 | 0.7102762 | 0.67066835 | 0.93678499 |
| 5 | 16000 | 0.08569629 | 2.26819255 | 3.00083332 | 3.44194499 |
| 6 | 32000 | 0.07252716 | 9.14479587 | 11.9918242 | 20.5362225 |

**Graphical Representation:**

Chart, line chart

Description automatically generated

**Unit Test Screenshots:**

1. Text

   Description automatically generatedInsertionSortTest 2. BenchmarkTest

Text

Description automatically generated

1. TimerTest

Text

Description automatically generated