Program Structures and Algorithms

Summer I 2023(SEC – 1)

Assignment -3 (Insertion Sort)

NAME: Amrita Dubey

NUID: 002760658

**Task:**

* **Implement three (3) methods (repeat, getClock, and toMillisecs) of a class called Timer.**
* **Implement Insertion Sort.**
* **Implement Main program to run benchmarks.**
* **Measure the running times of this sort, using four different initial array ordering situations: random, ordered, partially ordered and reverse-ordered.**
* **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.**

**Observations regarding the order of growth:**

As per benchmarking data following graph is deduced:

All experiments were run for data that were:

* Random order.
* Sorted on ascending order.
* Partially sorted in ascending order
* Reverse order sorted.

From the above graph we can deduce that “Reverse Ordered” data takes the most time to sort whereas “Ordered” data takes the least time. The measured timings, which range from 0.98 ms for N = 500 to 469.17 ms for N = 16000, are longer than with random or partially sorted data.

As the input size grows, the time goes up dramatically, suggesting a higher time complexity.

The worst-case time complexity of insertion sort if O(n^2) which means it takes quadratic time. This is when every element is inverted and needs to be swapped to move to its right position.

The best-case time complexity is O(n) which means that the array is already sorted, and no swap is required. The observed times are significantly lower than with random data, ranging from 0.48 ms for N = 500 to 4.58 ms for N = 16000. The time still increases with the input size, but the growth is significantly slower compared to random data.

In terms of time required, sorting partially sorted data falls between sorting randomly generated data, and ordered data. The observed times are often a little bit slower but closer than the times for random data. Like random data, the time climbs as the input size does.

Overall, the benchmarking data shows that insertion sort works rather well with ordered data but grows quadratic with random or reversed ordered data.

**Spreadsheet showing timing observations and their respective graphs:**

****

**Screenshots of Unit Tests and Benchmarking:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**