Alex Glavascu

Professor Gillenwater

CSCI 2210 – Data Structures

21 October 2024

Performance Comparison of Iterative and Recursive Sorting Algorithms

The purpose of this experiment is to compare the performance of two sorting algorithms: the iterative Bubble Sort and the recursive Merge Sort. The performance of these algorithms is evaluated by putting their performance into a variety of assessments, including differences in dataset sizes, list ordering, and data types (integers and Book objects). Determining which algorithm performs better and under what conditions is the primary objective.

The experiment assessed the algorithms on five different list orders (random, reverse, ordered, and almost ordered) and five different dataset sizes (10, 100, 1000, 10000, and 100000 elements). There was use of both reference types (Book objects) and value types (integers). A stopwatch was used to measure the runtimes of each algorithm, which were performed five times per dataset. The average time was calculated for comparison.

**Graph 1: Average Runtime vs. Multiple Factors***(Graph 1 shows: Number of elements,* ***integer*** *data,* ***iterative*** *approach, and value type)*

**Graph 2: Average Runtime vs. Multiple Factors***(Graph 2 shows: Number of elements,* ***integer*** *data,* ***recursive*** *approach, and value type)*

**Graph 3: Average Runtime vs. Multiple Factors***(Graph 3 shows: Number of elements,* ***book*** *data,* ***iterative*** *approach, and reference type)*

**Graph 4: Average Runtime vs. Multiple Factors***(Graph 4 shows: Number of elements,* ***book*** *data,* ***recursive*** *approach, and reference type)*

**Analysis**

**How Conditions Affected Runtime**

1. **Dataset Size**:
   * On larger datasets (1000+ elements), the recursive Merge Sort performed significantly better than the iterative Bubble Sort, especially for integers.
   * For smaller datasets (10 and 100 elements), both algorithms performed similarly.
2. **List Order**:
   * Both algorithms struggled with reverse-ordered lists, but Merge Sort handled it more efficiently than Bubble Sort.
   * Sorted or almost sorted lists had faster performance, especially with Merge Sort.
3. **Data Type**:
   * When sorting integers, Merge Sort consistently performed better for larger datasets.
   * Sorting Book objects revealed inefficiencies, particularly for recursive sorting. This suggests that Merge Sort for Books data may not be optimal for complex datasets.

**Best-Performing Conditions**

* **Merge Sort**: Best performance was observed with large, randomly ordered integer datasets.
* **Bubble Sort**: Performed well with small datasets (10 or 100 elements), especially when the list was already sorted or nearly sorted.

**Worst-Performing Conditions**

* **Merge Sort**: Struggled with Book objects, likely due to the complex comparisons required in the CompareTo method.
* **Bubble Sort**: Performed poorly on larger datasets and reverse-ordered lists due to its quadratic time complexity.

**Reasoning Behind Results**

* **Integer Sorting**: Merge Sort’s divide-and-conquer approach allowed it to handle large datasets more efficiently than Bubble Sort.
* **Book Sorting**: The CompareTo method likely added overhead (additional resources needed to complete the task) for Merge Sort, as it had to compare multiple fields for each Book object.

The experiment shows the advantages and disadvantages of recursive and iterative sorting algorithms. Recursive algorithms (Merge Sort) perform significantly better than iterative ones (Bubble Sort) for large datasets of simple data types like integers. However, problems were revealed with the recursive Merge Sort when dealing with more complex objects like Books.

In conclusion, the size, order, and complexity of the data to be sorted have a significant part in the decision between recursive and iterative sorting algorithms. Iterative sorting algorithms are capable of handling smaller, simpler datasets, but recursive sorting methods are usually more efficient for bigger datasets.