**Module 3 Critical Thinking Option #1: Random Number Generator**

Youssef Jamal Diallo

Colorado State University Global

CSC506 Design and Analysis of Algorithms

Dr. Dong Nguyen

2/4/2024

Bubble sort is a simple comparison-based sorting algorithm. It works by repeatedly stepping through the list, comparing adjacent elements, and swapping them if they are in the wrong order. This process is repeated until the list is sorted.

Selection sort is another straightforward algorithm that repeatedly selects the minimum element from the unsorted portion and swaps it with the first unsorted element. It gradually builds the sorted portion of the list.

Merge sort is a divide-and-conquer algorithm that divides the input array into two halves, recursively sorts them, and then merges the two sorted halves to produce a single sorted array.

Bubble sort, selection sort, and merge sort are all sorting algorithms that rely on comparing elements to arrange them in a specific order. However, they differ significantly in terms of their time and space complexity, ultimately impacting their overall performance. Let's take a closer look at these three algorithms:

**Bubble Sort:**

**Time Complexity**:

Worst and average case: O(n^2)

Best case (already sorted): O(n)

Space Complexity: O(1) (constant)

**Performance:** Bubble sort is generally considered the least efficient of the three algorithms due to its quadratic time complexity in most cases. While it has a lower space complexity, its slow performance makes it unsuitable for large datasets.

**Selection Sort:**

Time Complexity: O(n^2) in all cases

Space Complexity: O(1) (constant)

**Performance:** Like bubble sort, selection sort also exhibits a quadratic time complexity in all scenarios. While its space complexity is low, its slow performance limits its practicality for large datasets.

**Merge Sort:**

Time Complexity: O(n log n) in all cases

Space Complexity: O(n)

**Performance:** Merge sort stands out for its superior performance with a time complexity of O(n log n) in all cases. This makes it significantly faster than bubble and selection sort, especially for large datasets. However, its increased space complexity of O(n) requires additional memory compared to the constant space usage of the other two algorithms.

This Python script showcases three sorting algorithms: bubble sort, selection sort, and merge sort. It also includes a function specifically designed to measure their execution time. The script's functionality can be summarized as follows:

**Random Data Generation:**

The generate\_random\_list function creates a list of random integers. This list serves as the test data for evaluating the sorting algorithms.

**Performance Measurement:**

The benchmark\_sort function takes a sorting algorithm function as input and measures its execution time on the generated random list. This allows for a fair comparison of the algorithms' speed.

**Algorithm Implementations:**

Dedicated functions bubble\_sort, selection\_sort, and merge\_sort implement the respective sorting algorithms within the script.

**Benchmarking and Results:**

The script utilizes the benchmark\_sort function to measure the performance of each sorting algorithm on the random list.

The execution times for each algorithm are then printed, providing a clear comparison of their efficiency.

**Fastest Algorithm Identification:**

Based on the benchmark results, the script identifies the algorithm that demonstrated the fastest execution time for the given data. In our case, as expected, the results were: 0.1019 for bubble Sort, 0.0270 for selection Sort and lastly 0.0050 for merge Sort.

While bubble sort and selection sort are simple to understand and implement, their quadratic time complexity makes them less practical for real-world applications involving large datasets. Merge sort, although requiring more space, offers a significant performance advantage due to its O(n log n) time complexity, making it a preferred choice for larger datasets. The choice of algorithm ultimately depends on the specific needs of your application, considering factors such as dataset size, performance requirements, and available memory.

**References:**

* Sah, K. N. K. (2023, July 22). Efficient Sorting Algorithms in Dart: Comparing Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, and Quick Sort. LinkedIn. <https://www.linkedin.com/pulse/efficient-sorting-algorithms-dart-comparing-bubble-sort-sah>
* Sepahyar, S., Vaziri, R., & Rezaei, M. (2019). Comparing four important sorting algorithms based on their time complexityLinks to an external site.. Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence. https://doi-org.csuglobal.idm.oclc.org/10.1145/3377713.3377808
* Tang, Y. (n.d.). Sorting algorithms. Python Algorithms. Retrieved from https://pythonalgos.com/resources-2/sorting-algorithms/
* zyBooks. (2023, November 16). Data Structures Essentials with Python Examples - zyBooks. https://www.zybooks.com/catalog/data-structures-essentials-python/