Explaining Advanced Sorting Algorithms

Merge, Quick, and Heap

Blake Gateley

Computer Science 3
Allen High School
09/01/2023

Contents

1 Merge Sort

Merge sort works by taking the input array and dividing it into smaller arrays until it has n number of arrays with 1 index. It then starts to merge each array together in order from smallest to largest. It does this using recursive calls to sort each indice while merging the arrays. The time complexity of merge sort is interesting in that it is constant regardless of the order. This means that the worst case, best case, and average time complexity are O(n*logn). The space complexity is O(n) which is relatively large compared to other algorithms.

2 Quick Sort

Quicksort is another divide-and-conquer algorithm with the following steps: Pivot: Choose a pivot element from the array. Partition: Rearrange the elements such that elements less than the pivotare on the left, and elements greater than the pivot are on the right. Recursion: Recursively apply Quicksort to the left and right subarrays. Time Complexity: Best-case: O(nlogn) - When the pivot choice results in balanced partitions. Average-case: O(nlogn) Worst-case: O(n) - Occurs when the pivot choice consistently results in unbalanced partitions. Quicksort is not stable by default but can be modified to maintain stability. Space Complexity: Quicksort is an in-place sorting algorithm, meaning it does not require additional memory for temporary storage. Its space complexity is O(log n) for the stack during recursion.

3 Heap Sort

Heapsort is an in-place comparison-based sorting algorithm that works by creating a binary heap and repeatedly extracting the maximum (for ascending order) element from the heap and placing it in the sorted portion of the array. Time Complexity: Best-case: O(nlogn) Average-case: O(nlogn) Worst-case: O(nlogn) Heapsort is not a stable sort. Space Complexity: Heapsort has a space complexity of O(1) because it doesn't require additional memory for temporary storage. It operates directly on the input array.

Performance Comparison: Time Complexity: Merge Sort, Quicksort, and Heapsort all have similar time complexities in the average and worst cases, O(n log n). Therefore, they are all efficient for large datasets. Merge Sort is consistent in performance across all cases, while Quicksort's worst-case can be problematic. Heapsort has a consistent O(n logn) time complexity but is usually slower in practice compared to Merge Sort and Quicksort because of its higher constant factors.

Space Complexity: Merge Sort requires additional memory proportional to the size of the input (O(n)). Quicksort has a better space complexity of O(log n) due to in-place sorting, making it more memory efficient. Heapsort is the most memory-efficient with a space complexity of O(1). In summary, the choice of sorting algorithm depends on the specific requirements of your application. If you need a stable sort or have concerns about worst-case performance, Merge Sort might be a good choice. Quicksort is a solid general-purpose algorithm with good average-case performance and in-place sorting. Heapsort is efficient in terms of space but may be slower in practice.