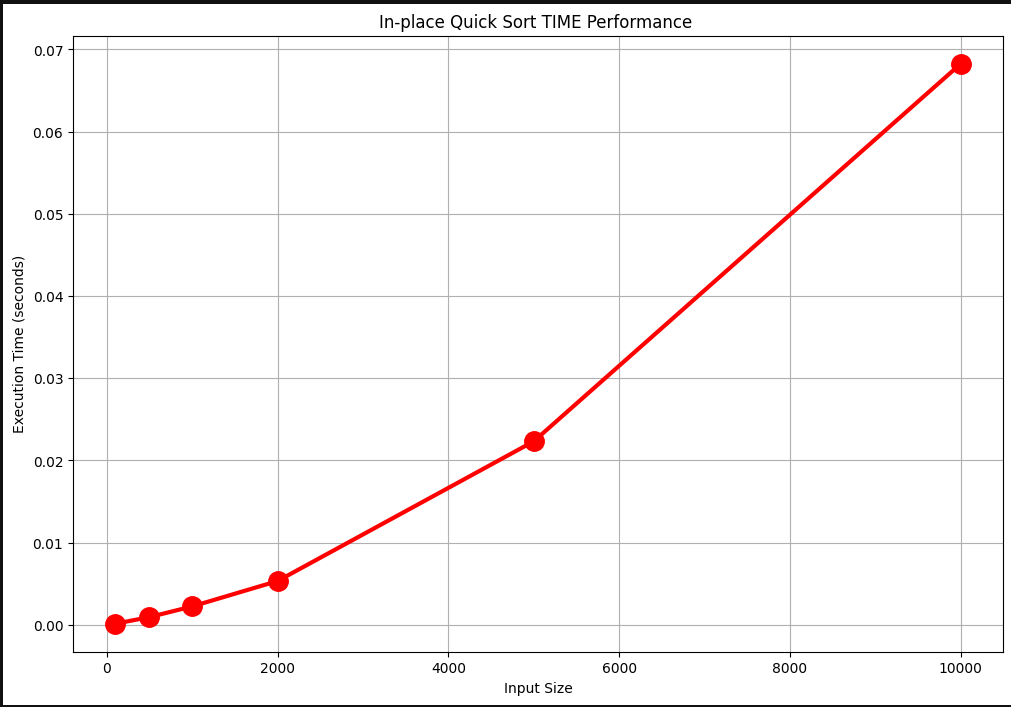
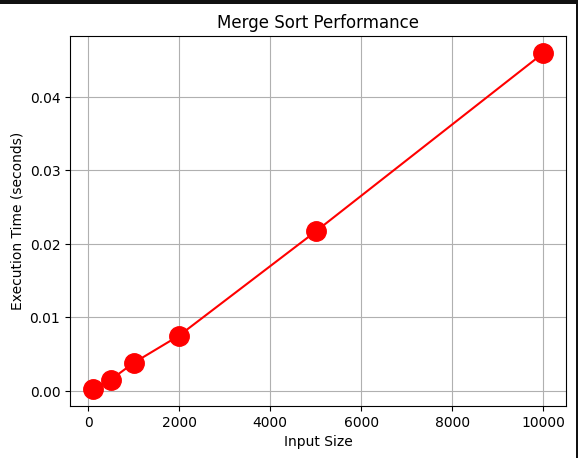
**QUICK PERFORMANCE:**

****

**MERGE PERFORMANCE:**



Optimizing Sorting Algorithms

Quick Sort

Analysis

Best Case

The best case for quick sort is when the pivot is the median of the array. This means that the array is split into two equal halves. This means that the time complexity is `O(nlogn)`.

Worst Case

The worst case for quick sort is when the pivot is the smallest or largest element in the array. This means that the array is split into two arrays of size 1 and n-1. This means that the time complexity is `O(n^2)`.

Average Case

The average case for quick sort is when the pivot is the median of the array. This means that the array is split into two equal halves. This means that the time complexity is `O(nlogn)`.

Optimizations

1. Randomized Pivot

The first optimization is to choose a random pivot. This means that the pivot is not always the first element in the array. This means that the worst case is less likely to occur. This means that the time complexity is `O(nlogn)`.

2. Insertion Sort

The second optimization is to use insertion sort when the array size is less than 10. This means that the array is sorted faster. This means that the time complexity is O(10^2) where 10 is significantly less than n.

Inference

The time complexity of quick sort is `O(nlogn)` in the best case and average case. The time complexity of quick sort is `O(n^2)` in the worst case. The time complexity of quick sort is O(10^2) when the array size is less than 10. We can infer that quick sort is a good sorting algorithm when the array size is large. We can infer that quick sort is a bad sorting algorithm when the array size is small.

Merge Sort

Analysis

Best Case

The best case for merge sort is when the array is already sorted. This means that the time complexity is `O(nlogn)`.

Worst Case

The worst case for merge sort is when the array is sorted in reverse order. This means that the time complexity is `O(nlogn)`.

Average Case

The average case for merge sort is when the array is not sorted. This means that the time complexity is `O(nlogn)`.

Optimizations

1. Insertion Sort

The first optimization is to use insertion sort when the array size is less than 10. This means that the array is sorted faster. This means that the time complexity is O(10^2) where 10 is significantly less than n.

REFERENCE:

https://www.geeksforgeeks.org/