Summary

1. **In your own words, why do you think your computer runs algorithms differently from other computers?**

   Computers run algorithms differently due to variations in hardware, operating systems, and system configurations. The CPU architecture, available memory, and disk speed can all impact performance. Additionally, compiler optimizations and system-specific libraries can cause differences in execution. These factors combined mean that even if two computers are running the same algorithm, the performance and behavior might be different.

2. **In quicksort, what does "partitioning" do?**

   In quicksort, partitioning is the process of rearranging the elements of the array so that all elements less than a chosen pivot element come before the pivot, and all elements greater than the pivot come after it. The pivot element is then in its final sorted position. This partitioning step is crucial as it divides the array into subarrays which can be recursively sorted.

3. **Why is O(n2) the worst-case time complexity for quicksort? Give an example of an array that adversely affects quicksort this way.**

   The worst-case time complexity of quicksort is O(n2). This occurs when the pivot selection is poor, such as always picking the smallest or largest element as the pivot. An example of an array that causes this behavior is a sorted or reverse-sorted array. For instance, given the array ([1, 2, 3, 4, 5]), if we always pick the first element as the pivot, each partitioning step results in one subarray being empty and the other containing all remaining elements, leading to O(n2) performance.

4. **In the benchmarking code, we had to create copies of the data_list. Why is that?**

   In benchmarking code, creating copies of the data list is necessary to ensure that each algorithm operates on the same initial data set. If we don't create copies, the first algorithm to run would sort the list, and subsequent algorithms would operate on an already sorted list, which could skew the benchmarking results and make comparisons inaccurate.

5. **Why does "log-linear" grow faster than "linear" time?**

   "Log-linear" time, denoted as O(n log_n), grows faster than linear time, O(n), because the logarithmic component adds a multiplicative factor to the linear growth. As the input size n increases, the log_n term, although growing slower than n, still increases the overall growth rate of the function more than a purely linear growth would.

6. **Describe how the log function behaves when plotted. What does this tell you about binary search?**

   When plotted, the logarithm function y = log(x) rises quickly at first and then gradually flattens out, increasing at a slower rate as x increases. This behavior indicates that the logarithmic growth is much slower compared to linear growth. For binary search, this means that the number of comparisons needed to find an element in a sorted array grows very slowly compared to the size of the array. Specifically, binary search reduces the problem size by half with each step, leading to a time complexity of O(log_n), which is highly efficient for large datasets.

Self-reflection part

1. **What Did I Learn?**

This week, I learnt the complexities of algorithms, focusing on sorting techniques like quicksort, understanding time complexities, and the importance of benchmarking. I also explored how system differences impact algorithm performance and the mathematical underpinnings of algorithm analysis.

2. **What Went Smoothly?**

Understanding the conceptual framework of sorting algorithms and time complexities was relatively smooth. The practical exercises on partitioning in quicksort and the significance of logarithmic functions in binary search were clear and reinforced through hands-on coding tasks.

3. **What Was Difficult About the Content This Week?**

The most challenging aspect was ensuring accurate benchmarking.

4. **How Will You Approach Things Differently Next Time?**

Break Down Complex Topics: Approach complex topics in smaller, more manageable segments to avoid feeling overwhelmed.

5. **Do You Have Any Feedback About the Content for This Week?**

Good!