**Checkpoint Data Structures and Algorithms:**

**Data Structures:**

1. **Array**: This is one of the most basic data structures, where elements are stored in contiguous memory locations. It’s essential for understanding other structures.
2. **Linked List**: A linear data structure consisting of nodes where each node contains data and a reference (link) to the next node.
3. **Stack**: A data structure that follows the LIFO (Last In First Out) principle. You add and remove elements from one end (top).
4. **Queue**: A data structure that follows the FIFO (First In First Out) principle. Elements are added at the back and removed from the front.
5. **Hash Tables**: Alose called a dictionary, a data structure used to store key-value pairs. It offers fast retrieval using a hash function.

**Sorting Algorithms:**

1. **Merge Sort**: A divide-and-conquer algorithm that divides the input into smaller sub-arrays, sorts them, and then merges them back together.
2. **Quick Sort**: A divide-and-conquer algorithm that selects a pivot element, partitions the array into two sub-arrays, and sorts them recursively.
3. **Bubble Sort**: A simple sorting algorithm that repeatedly steps through the list, compares adjacent items, and swaps them if they are in the wrong order.

**Data structures and sorting algorithms that I am not used to yet:**

* **Hash Tables**: While I have worked with dictionaries in Python or objects in JavaScript (which are based on hash tables), understanding the implementation of hash functions and collision resolution techniques is new for me.
* **Merge Sort & Quick Sort**: These are more advanced sorting algorithms, and I have not yet fully implemented them in certain programming languages or fully understood their recursive nature.
* **Stacks and Queues in More Complex Applications**: I have not yet had extensive experience applying stacks or queues in real-world applications (like in graph algorithms or system design).

**Why is that the case**?

* Lack of practical experience in implementing these structures or algorithms from scratch.
* Limited exposure to certain data structures or algorithms beyond basic sorting tasks or small-scale applications.
* Not yet practicing enough problems involving these concepts, such as coding challenges or project work.