A **stack** and a **queue** are essential data structures in computer science, each having various goals and following different rules for handling parts. Their structure and functionality substantially affect how they are implemented and used in software development.

A **stack** follows the Last-In-First-Out (LIFO) principle, meaning the last piece added to the stack is the first one to be removed. Imagine a stack of plates: a new plate is placed on top, and when one is needed, it is also taken off the top. In technical terms, stacks essentially use two operations: push, which puts an item to the top, and pop, which removes the top item. Additionally, the peek or top function permits the inspection of the top object without removing it. This structure facilitates recursive algorithms, function call management, and undo functionality in programs since it keeps track of the most recent activity first (Cormen et al., 2009).

A **queue**, on the other hand, follows the First-In-First-Out (FIFO) concept. The first element added to the queue is the first one to be deleted, comparable to people standing in line. A queue employs enqueue to add components to the back and dequeue to remove elements from the front. Queues are necessary in circumstances where order and fairness are critical, such as managing print tasks, handling task scheduling in operating systems, or buffering data in communication systems (Goodrich et al., 2014).

Although both stack and queue are linear data structures, the way they process data makes them ideal for distinct sorts of issues. A stack is appropriate for circumstances when activities need to be reversed or for tracking nested operations. For instance, during the execution of recursive functions, each function call is placed onto the call stack. When a function returns, it

is popped off the stack, ensuring the most recently called function completes first. This behavior is critical for ensuring the correct execution flow and memory management during recursion.

Conversely, queues are appropriate for cases that demand processing pieces in the exact order they come. For example, in real-time systems like customer service ticketing or network packet management, it is vital to ensure that the initial request is addressed first. Queues ensure order and fairness, making them useful in multitasking contexts and breadth-first search methods in graphs.

The operational variations also influence how these structures are executed. Stacks are commonly implemented using arrays or linked lists with insertion and removal occurring at the same end. In contrast, queues can be constructed with arrays or linked lists but require two pointers—one for the front and another for the rear—to manage the ends efficiently. Additionally, modifications like circular queues, priority queues, and double-ended queues (deques) offer extra functionality, but the core FIFO principle still applies.

While both structures organize data linearly, their access policies differentiate their utilization. A stack restricts access to only one end, making it more ideal for reverse-order processes, but a queue permits access at both ends in a regulated manner, enabling it to handle work in a timed or sequential sequence. Choosing between the two depends on the unique requirements of the situation, notably in terms of how data should be kept and accessed.

In summary, stacks and queues perform various tasks in computing. The stack's LIFO nature supports jobs that require last-in processing, while the queue's FIFO approach provides orderly processing of tasks as they come. Understanding their structure and operation facilitates efficient and effective problem-solving in diverse computer contexts.

Wordcount: 576

## References:

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press. https://mitpress.mit.edu/9780262533058/introduction-to-algorithms/

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). *Data Structures and Algorithms in Java* (6th ed.). Wiley. https://www.wiley.com/en-us/Data+Structures+and+Algorithms+in+Java%2C+6th+Edition-p-9781118771334