Zhu, Z., & Sun, W. (2021, September). Research on the Implementation of Recursive Algorithm Based on C Language. In *Journal of Physics: Conference Series* (Vol. 2023, No. 1, p. 012015). IOP Publishing.

Concise Summary of Recursion in Algorithms:

- **Types of Recursive Sorting Algorithms:** Recursive algorithms play a significant role in sorting algorithms, with two notable examples being merge sort and quick sort.
- **Implementation Methods:** Merge sort is typically defined using recursion, while quick sort is directly defined by recursion.
- **Steps in Writing Recursive Programs:**
    - Address common function calls in syntax.
    - Identify the recursive form and recursive boundaries at the algorithm level.
- **Common Types of Recursive Algorithms:**
    - Recursive algorithms based on the divide and conquer strategy.
    - Recursive algorithms based on the Backtracking strategy.
- **Advantages of Recursive Algorithms:**
    - Clear conceptual understanding.
    - Easy comprehension.
    - Simplified problem description.
    - Straightforward implementation.
    - Compact and concise code.
    - Ease of maintenance.

Recursive Algorithm Challenges:

- **Computational Complexity:** Recursive algorithms may exhibit high computational complexity in certain cases, leading to slower execution times.
- **Repeated Computation:** Improperly defined recursive functions can result in repeated computation, increasing inefficiency.
- **Increased Storage Space:** Deep recursive calls can consume significant storage space, potentially causing resource issues.
- **Function Call Overhead:** There is a cost associated with each function call in a recursive algorithm. In simpler calculations, this overhead can become a significant proportion of the overall execution time.
- **Challenges in Basic Thinking:** Understanding and implementing recursion can be challenging due to its distinctive thought process.

- **Expression of Recursion Concept:** Expressing recursive concepts in code can be intricate and require careful design.
- **Description of Recursion Process:** Describing how recursion unfolds and executes can be complex.
- **Execution Process of Recursion:** Managing the execution flow of recursive functions can be demanding.
- **Conditions and Environment:** Determining when and where to use recursion in a problem-solving context requires careful consideration.

Recursive Algorithm Process:

- **Basic Idea of Recursive Process:** Recursive algorithms break down a problem into a smaller-scale version of itself, solving it when the problem size reaches a certain limit. This process is described through self-reference, similar to mathematical induction.
- **Description Steps and Structure:** To describe a recursive process, you start by defining recursive parameters, termination conditions, and basic calculations. When the recursive parameter reaches a specific value, direct calculation occurs. Recursive calls are defined, where calculations include calls to the function itself, gradually reducing the complexity towards the termination condition.
- **Execution Process:** During the execution of a recursive function, focus on the self-calling process when the function is invoked. Track changes in function parameters and local variables. Each recursive call generates an independent instance with its own variables and parameters, akin to calling another function with the same name and code.
- **Design and Efficiency:** Recursive function design involves determining recursive parameters, defining lower parameter limits and calculation methods, and understanding the relationship between general and simple cases. Program design entails refining the recursive problem description, considering algorithm implementation factors, and selecting appropriate data structures. Efficiency considerations include the cost of function calls, parameter transfers, call environment saving, and instruction flow direction. Recursive functions often involve multiple calls, potentially leading to a deep level of nested function calls.