perplexity

explain the type of sorting algorithm written here

The sorting algorithm described in the provided pseudo-code is **Bubble Sort** applied to a linked list. Here's an explanation of how the algorithm works:

Key Characteristics:

1. **Comparison-based Sorting**: The algorithm compares adjacent elements (node data) and swaps them if they are in the wrong order.
2. **Iterative Process**: It repeatedly traverses the list until all elements are sorted in ascending order.

Steps in the Algorithm:

1. Two pointers, `pptr` and `fptr`, are used to traverse the linked list:
   - `pptr` starts at the first node.
   - `fptr` iterates through the subsequent nodes relative to `pptr`.
2. For each node pointed to by `pptr`, the algorithm compares its data with the data of nodes pointed to by `fptr`:
   - If `pptr->data > fptr->data`, their values are swapped.
3. The process continues until all nodes are sorted:
   - The outer loop (`while pptr != NULL`) ensures that every node is considered as a starting point for comparison.
   - The inner loop (`while fptr != NULL`) ensures that all subsequent nodes are compared with the current node (`pptr`).
4. The swapping mechanism ensures that smaller values "bubble up" toward the beginning of the list, while larger values move toward the end.

Why It Resembles Bubble Sort:

- Bubble Sort involves repeatedly swapping adjacent elements if they are in the wrong order, and this algorithm follows a similar logic, albeit adapted for linked lists where direct indexing is not possible.
- The use of nested loops and comparisons aligns with Bubble Sort's approach to sorting.

Efficiency:

- **Time Complexity**: $O(n^2)$, where n is the number of nodes in the linked list. This is because every pair of nodes is compared in a nested loop structure.
- **Space Complexity**: $O(1)$, as it only uses pointers for traversal and swapping without requiring extra memory.

This method is straightforward but not optimal for large datasets due to its quadratic time complexity.

⁂