Task 03:

Queue Sorting with Limited Space

You have a queue of integers that you need to sort. You can only use additional space equivalent to one stack. Describe the steps you would take to sort the elements in the queue.

Steps:

1. **Iterative Processing:**
    - We'll process the queue elements one by one in multiple passes.
2. **Pass 1: Minimum Selection:**
    - In the first pass, we'll use the queue and the stack to find the minimum element.
        - **Dequeue:** Remove an element from the queue.
        - **Compare with Stack Top:**
            - If the dequeued element is less than or equal to the top element of the stack (if the stack is not empty), push the stack top back onto the queue.

        - If the dequeued element is greater than the stack top (or the stack is empty), push the dequeued element onto the stack. This ensures the stack always holds the current minimum element seen so far.
    - Repeat dequeuing and comparing until the queue is empty.

3. **Pass 2: Reassembly with Minimum:**
    - Now, the stack holds the minimum element(s) from the original queue.
    - **Dequeue from Stack:** Pop elements from the stack and enqueue them back into the queue. This effectively places the minimum elements at the front of the queue.
4. **Recursive Processing (Optional):**
    - The remaining elements in the queue might still be unsorted.
        - You can treat the remaining elements in the queue as a new sub-queue and repeat steps 1-3 recursively. This will ensure all elements are compared and placed in their correct positions.

        - **Recursive Base Case:** The recursion can stop when the queue size becomes 1 (already sorted).

5. **Final Queue:**
    - After all passes (either iterative or recursive), the queue will be sorted in ascending order.

Explanation:

- In the first pass, the stack acts as a temporary storage for potential minimum elements. We compare each dequeued element with the stack top to ensure the stack always holds the current minimum.
- In the second pass, we dequeue the minimum elements from the stack and place them at the front of the queue.
- The recursive approach (optional) further sorts the remaining elements by treating them as a sub-queue and applying the same steps iteratively.

Complexity:

- Time Complexity: O(n^2) in the worst case, where n is the number of elements in the queue. This is because each element might be compared with all other elements in the worst case scenario.

- Space Complexity: O(1), as we only use a single stack for temporary storage.