STACK VS QUEUE COMPARISON

1. IMPLEMENTATION

Stack (LIFO - Last In First Out)

```
class Stack:
    items = []

    push(x):
        items.append(x)

    pop():
        return items.pop()
```

Queue (FIFO - First In First Out)

```
class Queue:
    items = []

    enqueue(x):
        items.append(x)

    dequeue():
        return items.remove_first()
```

Comparison Algorithm

Input: [1, 2, 3, 4, 5]

Stack operations:
→ push(1), push(2), push(3), push(4), push(5)
→ pop() → 5, pop() → 4, pop() → 3, pop() → 2, pop() → 1
→ Output: [5, 4, 3, 2, 1] (REVERSED)

Queue operations:
→ enqueue(1), enqueue(2), enqueue(3), enqueue(4), enqueue(5)
→ dequeue() → 1, dequeue() → 2, dequeue() → 3, dequeue() → 4, dequeue() → 5
→ Output: [1, 2, 3, 4, 5] (PRESERVED)

2. TEST CASES

Input	Stack Output	Queue Output
[1, 2, 3]	[3, 2, 1]	[1, 2, 3]
[A, B, C]	[C, B, A]	[A, B, C]
[X]	[X]	[X]
[]	[]	[]

Key Observation: Stack reverses order, Queue maintains order

3. COMPLEXITY ANALYSIS

Stack
- push(): O(1) - add to end
- pop(): O(1) - remove from end
- Space: O(n)

Queue (array-based)
- enqueue(): O(1) - add to end
- dequeue(): O(n) - remove first, shift all elements
- Space: O(n)

Summary
Both use linear space O(n). Stack has O(1) for all operations.
Queue has O(n) dequeue due to array shifting.

Optimization: Use linked list or deque for O(1) queue operations.