Cross-Review Summary

1. Overview of Algorithms

• Kadane's Algorithm (Almat's implementation):

o Purpose: Finds the maximum subarray sum in a sequence of integers.

o Approach: Uses dynamic programming, maintaining the maximum subarray ending at each index.

o Applications: Stock market analysis, signal processing, and optimization problems.

• Boyer–Moore Majority Vote Algorithm (Elkham's implementation):

o Purpose: Identifies the majority element in an array (appearing more than $\lfloor n/2 \rfloor$ times).

o Approach: Two-phase process: (1) candidate selection via voting, (2) verification by counting occurrences.

o Applications: Data stream analysis, voting systems, distributed consensus.

2. Design and Implementation Comparison

• Modularity:

o Kadane's code was structured with clear handling of edge cases (empty and single-element arrays).

o Boyer–Moore was separated into candidate selection and verification, with metrics collection for benchmarking.

• Metrics and Benchmarking:

o Kadane's implementation focused on correctness with JUnit tests.

o Boyer–Moore integrated a `PerformanceTracker` to measure comparisons, array accesses, memory usage, and execution time.

• **Edge Cases:**
  - Kadane: Explicitly handled arrays of size 0 and 1.
  - Boyer–Moore: Covered empty arrays, all-equal arrays, and no-majority cases.

3. Theoretical Complexity

• **Kadane's Algorithm:**
  - Time Complexity: O(n) (single pass).
  - Space Complexity: O(1).

• **Boyer–Moore Majority Vote:**
  - Time Complexity: O(n) (two passes).
  - Space Complexity: O(1).

Both algorithms are optimal in linear time and constant space, though applied to different problem domains.

4. Empirical Results

• **Kadane's Algorithm:**
  - Scales linearly with input size, confirming theoretical O(n).
  - Extremely fast in practice since it involves simple additions and comparisons.

• **Boyer–Moore Algorithm:**
- Benchmarks confirmed linear time across different input distributions.
- Constant factors (comparisons, array accesses) vary depending on distribution

(e.g., all-equal vs. random).
- Verification adds extra passes but does not impact asymptotic performance.

5. Strengths and Weaknesses

• **Kadane's Algorithm:**
- ✅ Efficient, simple, and widely applicable.
- ❌ Limited to maximum subarray sum; not useful for majority element detection.

• **Boyer–Moore Majority Vote:**
- ✅ Very memory-efficient; scales to very large arrays.
- ❌ Only works for majority element; must verify candidate in a second pass.

6. Conclusion

Both algorithms are efficient O(n) solutions with constant space, but optimized for different problem types. Kadane's focuses on **optimization of numeric sums**, while Boyer–Moore targets **frequency detection**. Together, they highlight how linear-time algorithms can solve distinct computational challenges effectively.