Assignment 2: Algorithmic Analysis and Peer Code Review

Pair 3 — Linear Array Algorithms

Student B: Prince Sharma, Kartik Jindal
Algorithm Analyzed: Boyer–Moore Majority Vote
Algorithm Implemented: Kadane's Algorithm
Course: Design and Analysis of Algorithms

Algorithm Overview — Boyer–Moore Majority Vote

Goal: To find the majority element (appears more than n/2 times) in a list.
Type: Linear time algorithm — O(n), O(1) space.

Working Principle:
1. Initialize a candidate and counter = 0.
2. Traverse the array:
• If counter = 0, set current element as candidate.
• If current element = candidate → increment counter.
• Else → decrement counter.
3. The final candidate is the majority element.

Example: Array = [2, 2, 1, 2, 3, 2, 2] → Output = 2 (appears 5 times out of 7)

Complexity Analysis

Case	Explanation	Complexity
Best Case	All elements same → one traversal	Θ(n)
Average Case	Random data → one traversal	Θ(n)
Worst Case	Alternating values → one traversal	O(n)

Space Complexity: Uses only a few variables → O(1) auxiliary space. In-place and memory-efficient.

Comparison with Kadane's Algorithm

Metric	Boyer-Moore	Kadane's
Goal	Find majority element	Find max subarray sum
Time	O(n)	O(n)
Space	O(1)	O(1)
Nature	Counting	Summation

Code Review and Optimization Suggestions

Observations:
• Code is clear and readable.
• Variables named meaningfully (candidate, count).
• Efficient single-pass structure.

Inefficiencies Found:
• No second pass verification to confirm majority element.
• Input validation missing (null/empty array not handled).

Suggested Improvements:
1. Add a second pass to confirm that the candidate truly occurs > n/2 times.
2. Add error handling for edge cases (empty or null input).
3. Include metrics like comparisons and array accesses for empirical testing.

Empirical Results — Kadane's Algorithm

Input Size (n)	Execution Time (ns)
100	129,333
1,000	84,333
10,000	723,833
50,000	2,273,375
100,000	1,128,459

Observation: Execution time grows linearly with input size, confirming O(n) complexity. Minor variations are due to CPU scheduling and randomness in inputs.

Conclusion

• Boyer-Moore and Kadane's algorithms both run in linear time O(n).
• Both are memory-efficient with O(1) auxiliary space.
• Partner's code was correct but could improve input validation and readability.
• Empirical results confirmed theoretical linear complexity for Kadane's Algorithm.
• Recommendation: Add input checks and integrate metric tracking in future improvements.