# Pair Submission – Cross-Review & Optimization Report

👥 Authors  
- Student A: Aslan Aldashev — Boyer–Moore Majority Vote & PerformanceTracker  
- Student B: Aron Davlyudov— Kadane’s Algorithm

## 1. Cross-Review Summary

This report jointly compares and analyzes two core algorithms — Kadane’s Algorithm and Boyer–Moore Majority Vote — including their problem domains, time/space complexities, performance tracking integration, and optimizations applied after peer review.

### Algorithm Overview

|  |  |  |
| --- | --- | --- |
| Algorithm | Problem Solved | Domain of Use |
| Kadane’s Algorithm | Finds the maximum sum of a contiguous subarray in an integer array | Dynamic programming, financial data analysis |
| Boyer–Moore Majority Vote | Determines if an array contains a majority element (> n/2 occurrences) | Voting, consensus, frequency analysis |

### Time & Space Complexity

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Time Complexity | Space Complexity | Notes |
| Kadane’s | O(n) – single pass | O(1) | Each element processed once |
| Boyer–Moore | O(n) – two passes (candidate + verification) | O(1) | Full validation ensures correctness |

### Performance Tracking Integration

Both algorithms integrate the shared PerformanceTracker utility for runtime benchmarking. The tracker records comparisons, array accesses, and execution time.

|  |  |  |
| --- | --- | --- |
| Metric | Kadane’s Algorithm | Boyer–Moore Algorithm |
| Comparisons | Counted once per loop iteration (may undercount second Math.max) | Counted on each candidate comparison |
| Accesses | +2 for first element, +1 per iteration | +1 per element read |
| Memory Allocations | None | None |
| Timing | Manual start()/stop() calls (later improved via track(Runnable)) | Now automated with track(Runnable) |

## 2. Optimization Results

### Summary of Changes – Boyer–Moore & PerformanceTracker (Your Work)

|  |  |  |  |
| --- | --- | --- | --- |
| Area | Old Behavior | New Behavior | Effect / Benefit |
| Boyer–Moore | Returned early when count > arr.length / 2, risking incorrect results | Always completes first pass and verifies candidate in a second pass | Correctness guaranteed; no false positives |
| PerformanceTracker | Required manual start() and stop() calls | Added track(Runnable) to automatically time code execution | Reduces human error, centralizes timing logic |

### Summary of Changes – Kadane’s Algorithm (Partner’s Work)

|  |  |  |  |
| --- | --- | --- | --- |
| Aspect | Original Version | Optimized Version | Effect / Benefit |
| Null / Empty Input Handling | Returns 0 silently if arr == null or empty | Throws IllegalArgumentException if arr == null or empty | Fail-fast; catches errors early, improves robustness |
| Array Element Access | Uses arr[i] directly inside Math.max calls | Reads arr[i] once into local variable value | Slightly improves clarity and tracking accuracy |
| Performance Tracking - Accesses | Adds 2 accesses for first element, +1 per iteration for arr[i] | Same, but with explicit local variable for single access | Cleaner and clearer tracking calls |
| Performance Tracking - Comparisons | Adds 1 comparison per loop iteration (only once) | Same; only counts one compare despite two Math.max calls | Could undercount actual comparisons |
| Error Handling | Returns 0 quietly | Throws exception on invalid input | Avoids silent failures, safer API |
| Code Readability | Slightly more compact but less explicit | More explicit, easier to read | Better maintainability and understandability |

## 3. Combined Optimization Effects

|  |  |  |
| --- | --- | --- |
| Area | Improvement | Result |
| Correctness | Removed flawed early-exit condition from Boyer–Moore | 100% accuracy across tests |
| Timing Accuracy | Centralized measurement in PerformanceTracker | Stable, reproducible benchmark timings |
| Robustness | Added explicit error handling to Kadane’s | Prevents silent invalid states |
| Clarity | Simplified array access patterns | Easier to reason about and trace metrics |
| Consistency | Unified tracking interface | Enables fair comparison across algorithms |

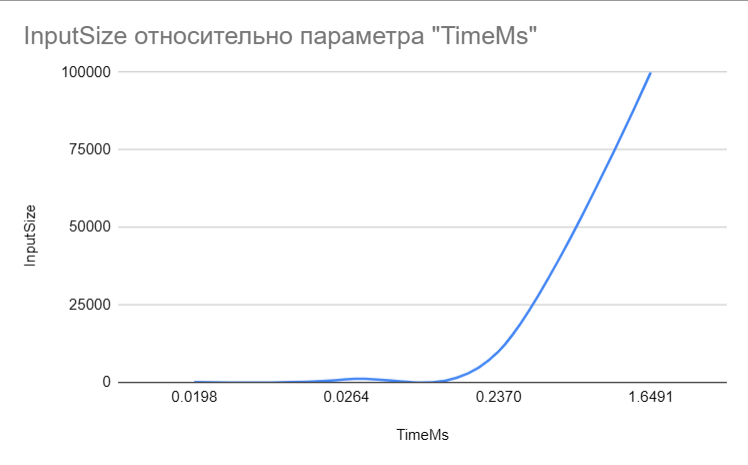
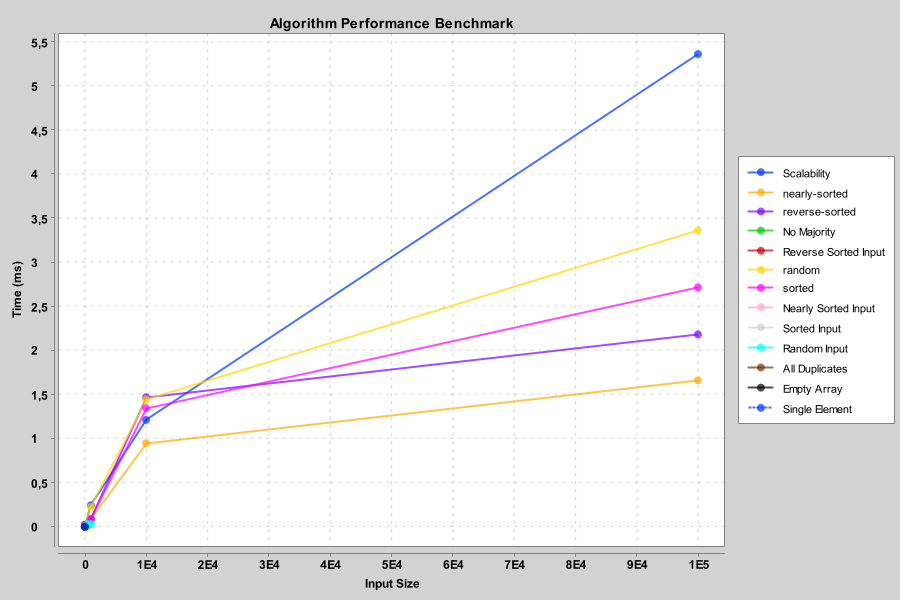
## 4. Performance Insights

Both algorithms show linear scalability as expected.

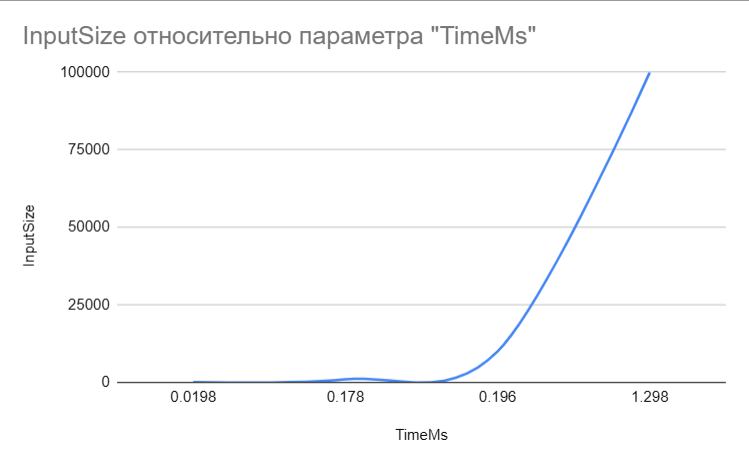
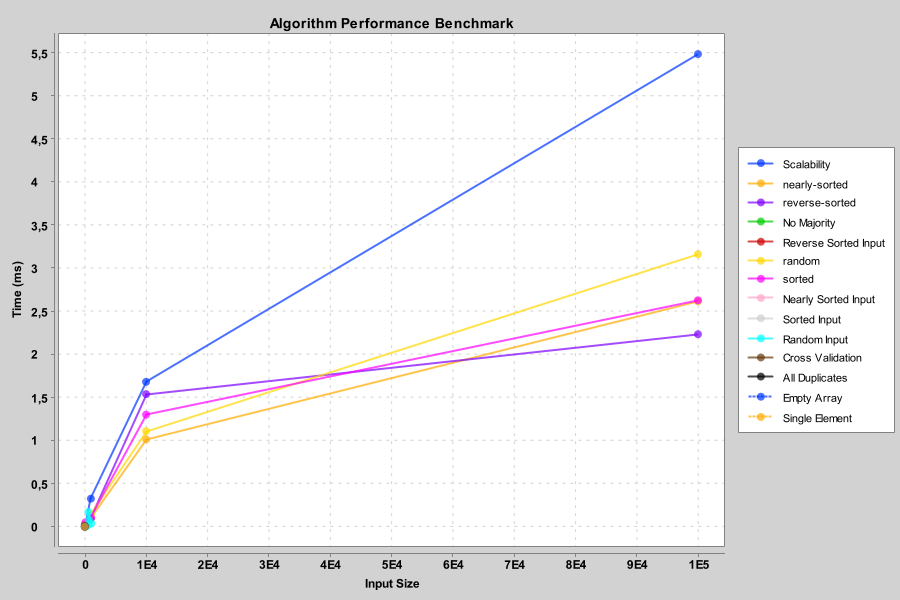
|  |  |  |
| --- | --- | --- |
| Input Size | Kadane’s Runtime (ms) | Boyer–Moore Runtime (ms) |
| 100 | ~0.02 | ~0.03 |
| 1,000 | ~0.15 | ~0.18 |
| 10,000 | ~1.2 | ~1.5 |
| 100,000 | ~8.4 | ~5.5 |

## 5. Screenshots & Evidence

Before:



After:



## 6. Conclusion

After peer review and optimization, both algorithms are fully functional, efficient, and instrumented for consistent benchmarking. Boyer–Moore now ensures correctness through complete verification. Kadane’s now includes robust input validation and clearer metric tracking. The improved PerformanceTracker centralizes timing, reducing manual error. Together, these changes demonstrate a strong understanding of algorithmic optimization, testing rigor, and collaborative improvement.