Design and Analysis of Algorithms

Assignment 2: Algorithmic Analysis and Peer Code Review

Student A: Tastemir Akerke

Student B: Mnaidarova Nargiza

Algorithm of student A: Boyer–Moore Majority Vote

---

1. Algorithm Overview

The Boyer–Moore Majority Vote Algorithm is a linear-time, constant-space method for finding a majority element in an array (an element that appears more than n/2 times).

It works in two phases:

1. Candidate selection — a potential majority is chosen.

2. Verification — the algorithm confirms if the candidate truly occurs more than n/2 times.

This approach requires only one pass (O(n)) and O(1) additional memory.

---

2. Asymptotic Complexity

| Case | Time Complexity | Space Complexity |

|---|---|---|

| Best | Θ(n) | Θ(1) |

| Average | Θ(n) | Θ(1) |

| Worst | Θ(n) | Θ(1) |

The algorithm always performs a single linear scan.

It’s extremely efficient for large datasets and suitable for streaming or memory-limited environments.

---

3. Code Review and Optimizations

- ✅ Clear and readable structure
- ✅ Handles edge cases (empty arrays, single element, no majority)
- ✅ Integrated performance tracking (comparisons, assignments, array accesses)
- ⚙️ Potential optimization: early exit during verification when majority confirmed

---

4. Empirical Results

Benchmarks were performed on random arrays with sizes n = 100, 1000, 10000, and 100000.

Execution times were recorded using the integrated PerformanceTracker and written to CSV.

Example results:

n	hasMajority	Time (ms)	Comparisons	Array Accesses	Assignments
100	false	0.06	200	200	125
1000	false	0.56	2000	2000	1241
10000	false	1.92	20000	20000	12551
100000	false	8.25	200000	200000	125065

The results confirm linear scalability (time ∝ n).

---

5. Conclusion

The Boyer–Moore Majority Vote algorithm achieves optimal O(n) performance with constant space.

The implementation meets all assignment requirements: correctness, metric tracking, CLI benchmarking, and clean Git workflow.

Future work includes comparing performance with **Kadane’s Algorithm** from the peer implementation.

End of Report