Peer Algorithm Analysis Report – Kadane's Algorithm

Student Name: Vanshika Jindal
Partner's Algorithm: Kadane's Algorithm (Maximum Subarray Problem)
My Algorithm: Boyer–Moore Majority Vote Algorithm
Course: Algorithmic Analysis and Peer Code Review
Language Used: Java

1. Algorithm Overview

Kadane's Algorithm is used to find the maximum sum of a contiguous subarray within a one-dimensional array of integers.
It works by keeping track of two variables — the current subarray sum and the maximum sum found so far.
If the current sum ever becomes negative, the algorithm resets it to zero, since a negative sum cannot contribute to a maximum subarray.

Kadane's Algorithm is known for its simplicity and speed, requiring only one pass through the array.
It is commonly applied in financial data analysis (profit/loss problems), AI, and data analytics.

2. Complexity Analysis

- Time Complexity: O(n) in all cases (best, average, and worst) because each element is processed exactly once.
- Space Complexity: O(1) since it only uses a few variables to store intermediate results.
- Recurrence Relation:
  MaxEndingHere(i) = max(arr[i], arr[i] + MaxEndingHere(i−1))

Comparison with Boyer–Moore Algorithm:

- Both Kadane's and Boyer–Moore algorithms have O(n) time complexity.
- Kadane's focuses on numerical subarray sums, while Boyer–Moore deals with finding the majority element in a list.

- Kadane's involves continuous value accumulation, while Boyer–Moore relies on cancellation logic (incrementing/decrementing counters).
- In practice, both are extremely efficient and suitable for large datasets.

3. Code Review & Optimization

Strengths:

- Clear and straightforward structure.
- Excellent time and space efficiency.
- Works correctly on both positive and negative arrays.

Weaknesses:

- Limited inline comments reduce readability.
- No input validation (e.g., handling empty or null arrays).
- Benchmarking code slightly increases overhead.

Suggestions:

- Add descriptive comments for clarity.
- Separate benchmarking logic from the main algorithm for cleaner results.
- Use helper functions or constants to improve code readability.

4. Empirical Validation (Benchmark Results)

Input Size (n)	Execution Time (ns)	Comparisons	Array Accesses
100	129,333	198	101
1,000	84,333	1,998	1,001
10,000	723,833	19,998	10,001
50,000	2,273,375	99,998	50,001
100,000	1,128,459	199,998	100,001

Observation:
The execution time grows roughly in proportion to the input size, confirming linear time complexity (O(n)).
Comparisons and array accesses also scale linearly, supporting the theoretical analysis.

5. Conclusion

Kadane's Algorithm is an efficient and elegant solution for the maximum subarray problem.
It achieves O(n) time and O(1) space complexity, making it one of the most optimized algorithms for this task.
Both theoretical and empirical results match perfectly, proving its effectiveness.
Compared to Boyer–Moore, Kadane's focuses on numerical optimization rather than element frequency, yet both share the same linear efficiency.

Minor improvements such as better commenting and cleaner benchmarking can make the implementation easier to read and maintain, but overall, it's a well-structured and optimal algorithm.