# Maximum Product of Three Numbers: Approach Comparison

I'll compare the three approaches for finding the maximum product of three numbers in an array, focusing on their time and space complexity.

## Comparison of Approaches

| Approach | Time Complexity | Space Complexity | Description |
| --- | --- | --- | --- |
| Brute Force | O(n³) | O(1) | Checks all possible triplets with three nested loops |
| Sorting-based | O(n log n) | O(1) | Sorts array and checks two potential maximum products |
| Recursive | O(n³) | O(n) | Uses recursion to generate all combinations of three elements |

## Analysis

1. **Brute Force Approach**:
   * Examines every possible triplet combination using three nested loops
   * Very inefficient for large arrays (166M operations for 1000 elements)
   * Simple implementation but poor scalability
2. **Sorting-based Approach**:
   * Most efficient solution in terms of time complexity
   * Intelligently reduces the problem by recognizing that the maximum product will be either:
     + The product of the three largest numbers, or
     + The product of the two smallest numbers (potentially large negatives) and the largest number
   * Sorting dominates the time complexity
3. **Recursive Approach**:
   * Same time complexity as brute force (O(n³))
   * Uses additional space for the recursive call stack
   * No advantage over the other approaches

## Conclusion

The **sorting-based approach** is clearly superior with O(n log n) time complexity and O(1) space complexity. It's significantly faster than both the brute force and recursive approaches for large inputs and uses minimal extra space.