Recursive vs Iterative Fibonacci Computation

Objective
To compare recursive and iterative approaches for computing Fibonacci numbers and analyze their performance and scalability using time complexity.

Problem Overview
The Fibonacci sequence is a classic example used to demonstrate algorithmic efficiency. While the recursive solution is intuitive and closely follows the mathematical definition, it performs redundant computations. The iterative solution avoids repetition and computes results efficiently.

Recursive Approach
The recursive method recalculates the same Fibonacci values multiple times, leading to exponential time complexity O(2■). As N increases, execution time and call stack usage grow rapidly, making this approach impractical for large inputs.

Iterative Approach
The iterative method builds the Fibonacci sequence step by step using a loop. It runs in linear time O(N) and uses constant extra space, making it highly efficient and scalable even for large values of N.

Performance Analysis
Benchmark results clearly show that the recursive approach becomes infeasible beyond moderate input sizes, while the iterative solution completes almost instantly even for large N values.

Conclusion
Recursive Fibonacci implementations are useful for learning recursion but should be avoided in production code. The iterative approach is significantly faster, more memory-efficient, and suitable for real-world applications.