Exercise 7: Financial Forecasting

Analysis:

- Discuss the time complexity of your recursive algorithm.
- Explain how to optimize the recursive solution to avoid excessive computation.

Time Complexity of Recursive Algorithm

If the recursive algorithm calculates future values (e.g., profit, revenue, etc.) based on previous periods using a recurrence like:

```
double forecast(int n) {
   if (n == 0) return baseCase;
   return factor * forecast(n - 1);
}
```

Then:

- **Time Complexity:**
  $O(2^n)$ in the worst case if the recursive function recalculates overlapping subproblems multiple times (like in Fibonacci-style recursions).
  But if each call only makes one recursive call (as above), it becomes O(n).

Optimization to Avoid Excessive Computation

1.To optimize the recursive algorithm:

1. **Use Memoization (Top-Down DP)**

Store results of already-computed subproblems:

```
Map<Integer, Double> memo = new HashMap<>();
double forecast(int n) {
   if (n == 0) return baseCase;
   if (memo.containsKey(n)) return memo.get(n);
   double result = factor * forecast(n - 1);
   memo.put(n, result);
   return result;
}
```

- Time Complexity: O(n)
- Space Complexity: O(n) for memo table

2. **Use Iteration (Bottom-Up DP)**

Avoid recursion entirely by using a loop:

```
double forecastIter(int n) {
    double result = baseCase;
    for (int i = 1; i <= n; i++) {
        result = factor * result;
    }
    return result;
}
```

- Time Complexity: O(n)
- Space Complexity: O(1)

Let me know if your recursion uses a more complex relation (e.g., involving multiple previous steps like f(n) = f(n-1) + f(n-2)) so I can tailor the answer more specifically.