**Concept of Recursion:**

Recursion is a programming technique where a function calls itself to solve smaller instances of the same problem. This technique is often used for problems that can be broken down into simpler, similar sub-problems. Each recursive call processes a part of the problem, and the base case provides a simple answer that terminates the recursion.

**Advantages of Recursion:**

* Simplifies code for problems that have repetitive sub-problems.
* Helps in breaking down complex problems into more manageable parts.

**Disadvantages of Recursion:**

* Can lead to excessive memory usage if the recursion depth is too high.
* May result in performance issues if not properly managed, particularly with redundant calculations.

**Time Complexity:**

* The time complexity of the recursive algorithm is O(n), where nnn is the number of periods. This is because each recursive call reduces the number of periods by one until it reaches zero.

**Optimizing the Recursive Solution:**

To avoid excessive computation, especially with high numbers of periods, the recursion can be optimized by memoization or converting it to an iterative approach:

1. **Memoization:** Store the results of sub-problems so that they do not need to be recalculated multiple times. This can significantly reduce the number of calculations in problems with overlapping sub-problems.
2. **Iterative Approach:** An iterative method can often achieve the same result without the overhead of recursive calls. This is particularly useful in languages that do not optimize tail-recursive calls.