**Understand Recursive Algorithms**

**Concept of Recursion**

* **Definition**: Recursion is a programming technique where a function calls itself directly or indirectly to solve a problem. Each recursive call breaks the problem into smaller subproblems, which are easier to solve.
* **Base Case and Recursive Case**:
  + **Base Case**: The condition under which the recursion terminates.
  + **Recursive Case**: The part of the function where the function calls itself with a subset of the original problem.

**Example**

Recursion simplifies problems like factorial calculation, Fibonacci series, and tree traversals by breaking them into smaller instances of the same problem.

**Analysis**

**Time Complexity**

* **Time Complexity**: O(n)O(n)O(n), where n is the number of years. The function makes n recursive calls, one for each year.
* **Space Complexity**: O(n)O(n)O(n) due to the call stack space used by the recursive calls.

**Optimizing the Recursive Solution**

* **Memorization**: Store the results of previous computations to avoid redundant calculations.
* **Iterative Approach**: Convert the recursive solution to an iterative one to avoid the overhead of recursive calls and reduce space complexity.