**Step 1**

**Recursion** is when a method calls itself to solve a smaller version of the same problem.

It typically has two parts:

1. **Base Case** – defines when the recursion stops.
2. **Recursive Case** – defines how the function reduces the problem.

**Recursion** simplifies problems that have a repeating or self-similar structure, like Fibonacci series.

We will forecast future value based on:

* An initial amount
* A fixed annual growth rate
* A number of years

Formula(compound growth)

FV(n) = FV(n-1)\*(1+ growthRate)

**Step 4** Analysis

**Time complexity**

T(n) = T(n-1) +O(1)

Time complexity: O(n)

**Space complexity**

Each call adds a layers to the call stack

Space complexity: O(n)

Optimization can be done by using Iteration we can convert recursion to loop to reduce call stack usage. With this space complexity turns out to be O(1) and time complexity remains same.