Recursion

By GC Jana

What is Recursion?

Recursion *is defined as a process which calls itself directly or indirectly and the corresponding function is called a recursive function.*

approach(1) – Simply adding one by one

$$f(n) = 1 + 2 + 3 + \ldots\ldots + n$$

approach(2) – Recursive adding

$$f(n) = 1 \qquad n=1$$

$$f(n) = n + f(n-1) \qquad n>1$$

Need of Recursive Function

1. Solving complex tasks
2. Divide and Conquer
3. Backtracking
4. Dynamic programming
5. Tree and graph structures

Properties of Recursion

Recursion has some important properties. Some of which are mentioned below:

- The primary property of recursion is the ability to solve a problem by breaking it down into smaller sub-problems, each of which can be solved in the same way.
- A recursive function must have a base case or stopping criteria to avoid infinite recursion.

Properties of Recursion

Recursion has some important properties. Some of which are mentioned below:

- Recursion involves calling the same function within itself, which leads to a call stack.
- Recursive functions may be less efficient than iterative solutions in terms of memory and performance.

Types of Recursion

1. **Direct recursion:** When a function is called within itself directly it is called direct recursion. This can be further categorized into four types:
    1. Tail recursion,
    2. Head recursion,
    3. Tree recursion and
    4. Nested recursion.

Types of Recursion

- **2.Indirect recursion:** Indirect recursion occurs when a function calls another function that eventually calls the original function and it forms a cycle.

Applications of Recursion

Recursion is used in many fields of computer science and mathematics, which includes:

- Searching and sorting algorithms: Recursive algorithms are used to search and sort data structures like trees and graphs.
- Mathematical calculations: Recursive algorithms are used to solve problems such as factorial, Fibonacci sequence, etc.
- Compiler design: Recursion is used in the design of compilers to parse and analyze programming languages.

Applications of Recursion

- Graphics: many computer graphics algorithms, such as fractals and the Mandelbrot set, use recursion to generate complex patterns.
- Artificial intelligence: recursive neural networks are used in natural language processing, computer vision, and other AI applications.

Advantages of Recursion:

- Recursion can simplify complex problems by breaking them down into smaller, more manageable pieces.
- Recursive code can be more readable and easier to understand than iterative code.
- Recursion is essential for some algorithms and data structures.
- Also with recursion, we can reduce the length of code and become more readable and understandable to the user/ programmer.

Disadvantages of Recursion

- Recursion can be less efficient than iterative solutions in terms of memory and performance.
- Recursive functions can be more challenging to debug and understand than iterative solutions.
- Recursion can lead to stack overflow errors if the recursion depth is too high.

How to write a Recursive Function

- **Components of a recursive function:**

Base case:

- Every recursive function must have a base case.
- The base case is the simplest scenario that does not require further recursion.
- This is a termination condition that prevents the function from calling itself indefinitely.
- Without a proper base case, a recursive function can lead to infinite recursion.

How to write a Recursive Function

- **Components of a recursive function:**

Recursive case:

- In the recursive case, the function calls itself with the modified arguments.
- This is the essence of recursion – solving a larger problem by breaking it down into smaller instances of the same problem.
- The recursive case should move closer to the base case with each iteration.

How to write a Recursive Function

- Let’s consider the example of factorial of number:

  In this example, the base case is when n is 0, and the function returns 1. The recursive case multiplies n with the result of the function called with parameter n – 1.
  The process continues until the base case is reached.

How to write a Recursive Function

```java
import java.util.Scanner;

public class Factorial {
    // Recursive Function to calculate the factorial of a number
    static int factorial(int n) {
        // Base case: If n is 0, the factorial is 1.
        if (n == 0) {
            return 1;
        }

        // Recursive case: Calculate the factorial by multiplyin
        return n * factorial(n - 1);
    }

    public static void main(String[] args) {
        int n = 4;

        // Calculate and print the factorial of n.
        int result = factorial(n);
        System.out.println("Factorial of " + n + " is: " + resul
    }
}
```

Output:

Factorial of 4 is:24

Types of Recursions

Direct Recursion

- These can be further categorized into **four types**:
    1. Tail recursion,
    2. Head recursion,
    3. Tree recursion and
    4. Nested recursion.

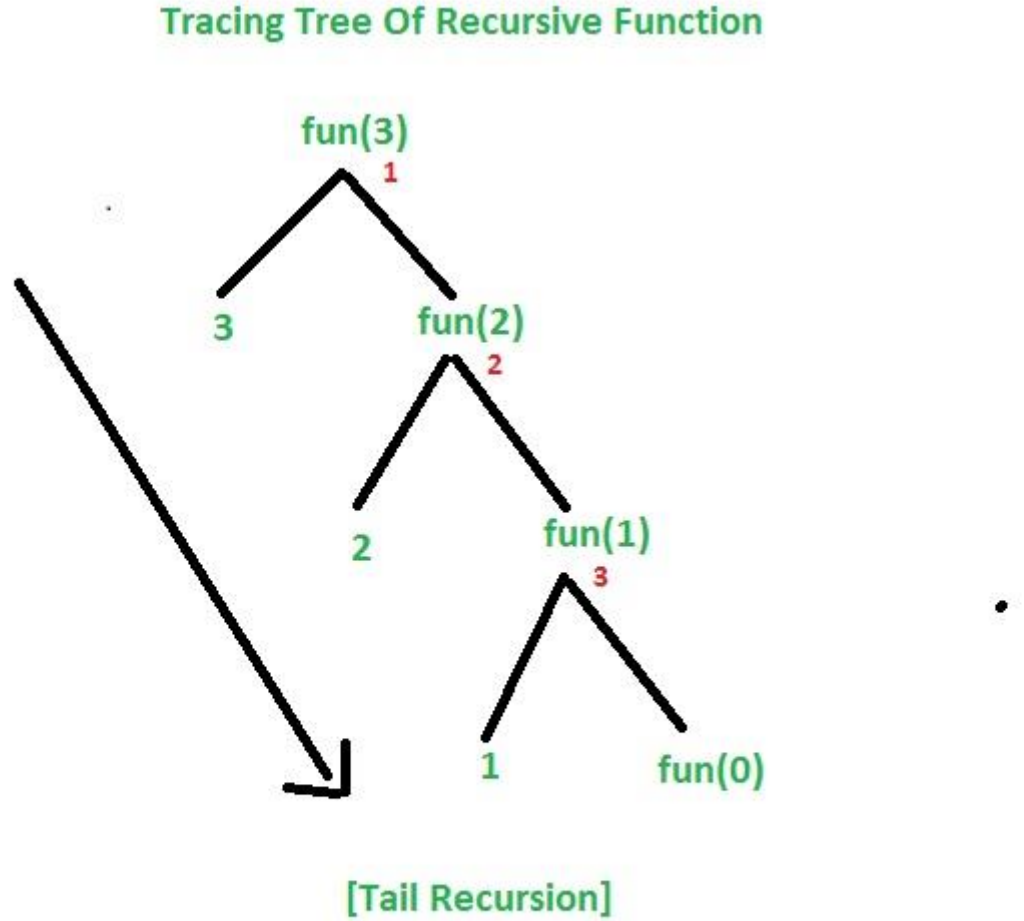
Direct Recursion: (1)Tail recursion

- **Tail Recursion**: If a recursive function calling itself and that recursive call is the last statement in the function then it's known as **Tail Recursion.**

After that call the recursive function performs nothing.

The function has to process or perform any operation at the time of calling and it does nothing at returning time.

Direct Recursion: (1)Tail recursion



Output: 3 2 1

*Digits in red showing that the order in which the calls are made and according to the order of calling the output are printed on the screen.Note that for fun(0) it gives nothing as output.

Direct Recursion: (1)Tail recursion

```java
// Java code Showing Tail Recursion
class GFG {

  // Recursion function
  static void fun(int n)
  {
    if (n > 0)
    {
      System.out.print(n + " ");

      // Last statement in the function
      fun(n - 1);
    }
  }

  // Driver Code
  public static void main(String[] args)
  {
    int x = 3;
    fun(x);
  }
}
```

```
Output:3 2 1
```

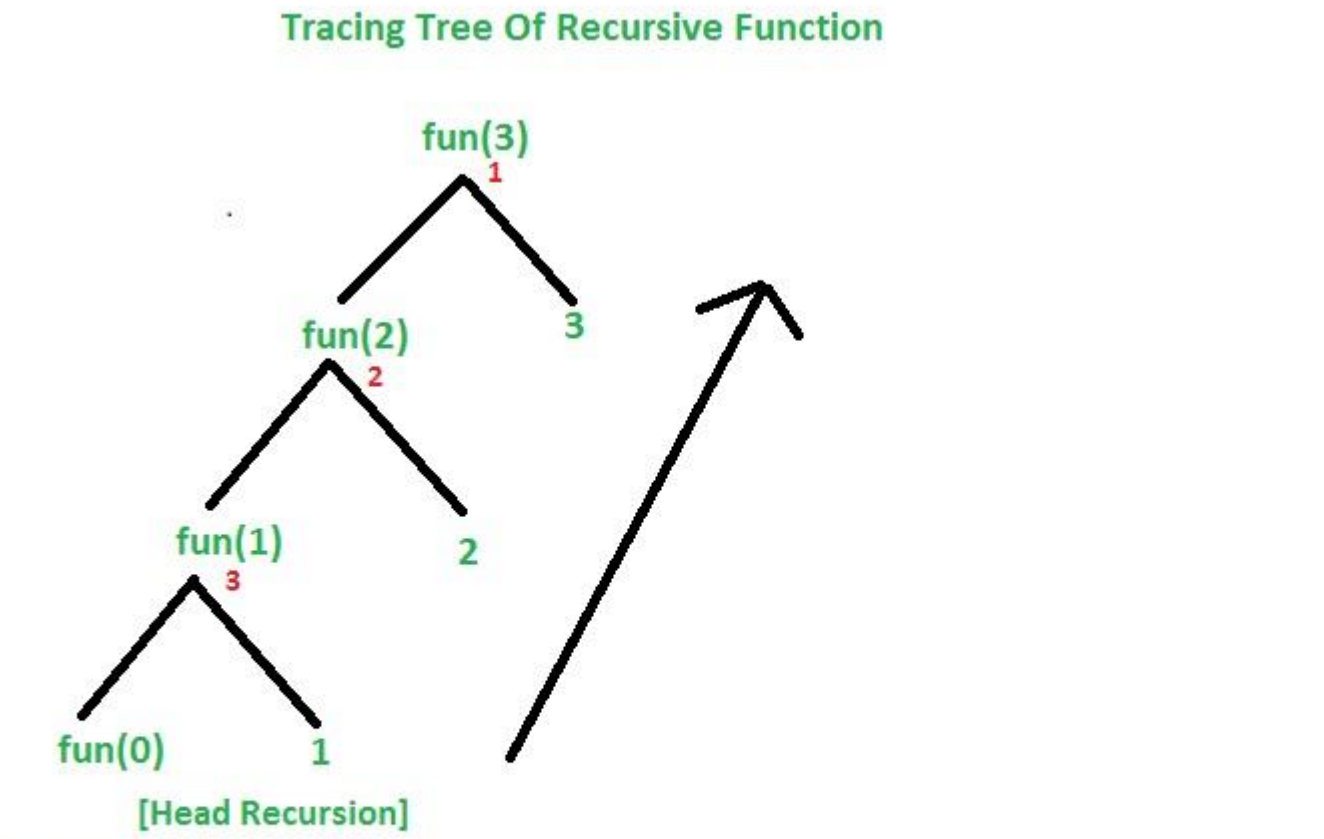
Direct Recursion: (2) Head recursion

- **Head Recursion**: If a recursive function calling itself and that recursive call is the first statement in the function then it's known as **Head Recursion.**

There's no statement, no operation before the call.

The function doesn't have to process or perform any operation at the time of calling and all operations are done at returning time.

Direct Recursion: (2) Head recursion



Output: 1 2 3

*Digits in red showing that the order in which the calls are made and note that printing done at returning time. And it does nothing at calling time.

Direct Recursion: (2) Head recursion

```java
// Java program showing Head Recursion
import java.io.*;

class GFG{

// Recursive function
static void fun(int n)
{
    if (n > 0) {

        // First statement in the function
        fun(n - 1);

        System.out.print(" "+ n);
    }
}

// Driver code
public static void main(String[] args)
{
    int x = 3;
    fun(x);

}
}
```

Output: 1 2 3

Direct Recursion: (3) Tree recursion

Direct Recursion: (4) Nested recursion