Iterative vs Recursive – Summary

1. Introduction

In programming, many problems can be solved using **iteration** (loops) or **recursion** (function calling itself). Understanding their differences helps in choosing the right approach.

2. Iteration

- **Definition**: Repeatedly executing a block of code using loops (`for`, `while`).
- **Usage**: Suitable for problems where the number of repetitions is known or can be determined.
- **Memory**: Uses **stack implicitly**, no extra function calls.
- **Performance**: Generally faster than recursion due to no function call overhead.
- **Example**:

```
int sum = 0;
for(int i = 1; i <= 5; i++) {
    sum += i;
}
```

3. Recursion

- **Definition**: Solving a problem by **function calling itself**.
- **Usage**: Suitable for problems that can be divided into **similar subproblems**.
- **Memory**: Uses **stack memory** for each function call.
- **Performance**: Slightly slower due to function call overhead.
- **Example**:

```
int sum(int n) {
    if(n == 0) return 0;
    return n + sum(n-1);
}
```

4. Key Differences

Feature	Iteration	Recursion
Method	Loops (for, while)	Function calls itself
Memory Usage	Low, uses same stack frame	Higher, uses stack per call
Speed	Faster	Slower due to call overhead

Feature	Iteration	Recursion
Readability	Can be longer/complex	Clearer for hierarchical problems
Termination	Loop condition	Base case
Use Case	Simple repetitive tasks	Problems with natural recursion

5. Advantages & Disadvantages

Iteration: - Advantages: Efficient, low memory usage. - Disadvantages: Code may be less intuitive for recursive structures.

Recursion: - Advantages: Elegant, easy to read for divide-and-conquer problems. - Disadvantages: More memory usage, slower, risk of stack overflow.

6. Tips for Choosing

- Use **iteration** for simple repetitive tasks.
- Use **recursion** for problems that naturally fit recursive approach (trees, graphs, divide-and-conquer).
- Consider **tail recursion** optimization if recursion is preferred but deep.

7. Summary

- Iterative = loops, efficient, uses little memory.
- Recursive = function calls itself, elegant, uses more memory.
- Choosing depends on problem nature, readability, and efficiency requirements.