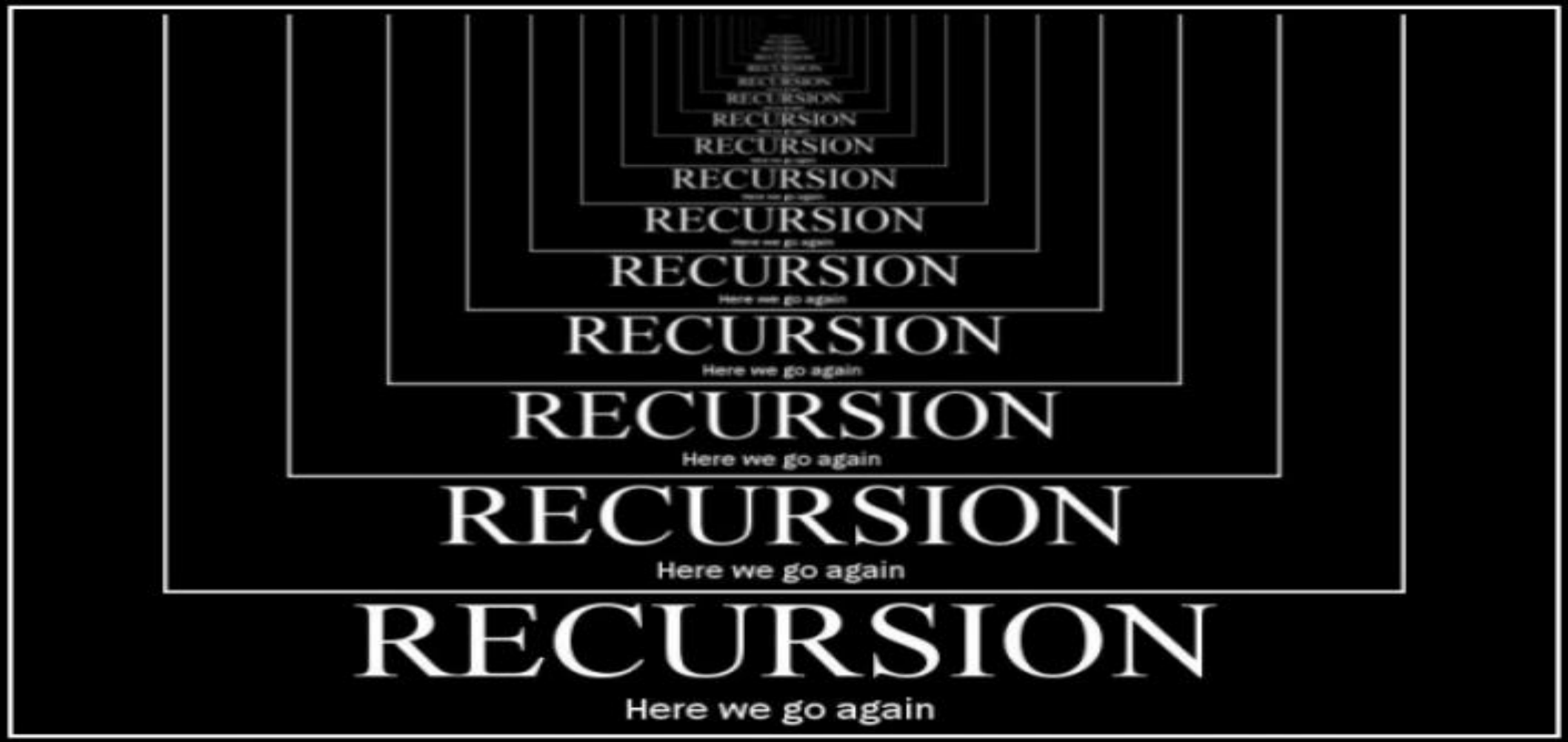

RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again

Notable Points

- Every recursive program can also be written without recursion
- Recursion is used for programming convenience, not for performance enhancement
- Sometimes, if the function being computed has a nice recurrence form, then a recursive code may be more readable

Common Errors in Writing Recursive Functions

- No base case
- The base case is never reached
- Mixing up loops and recursion

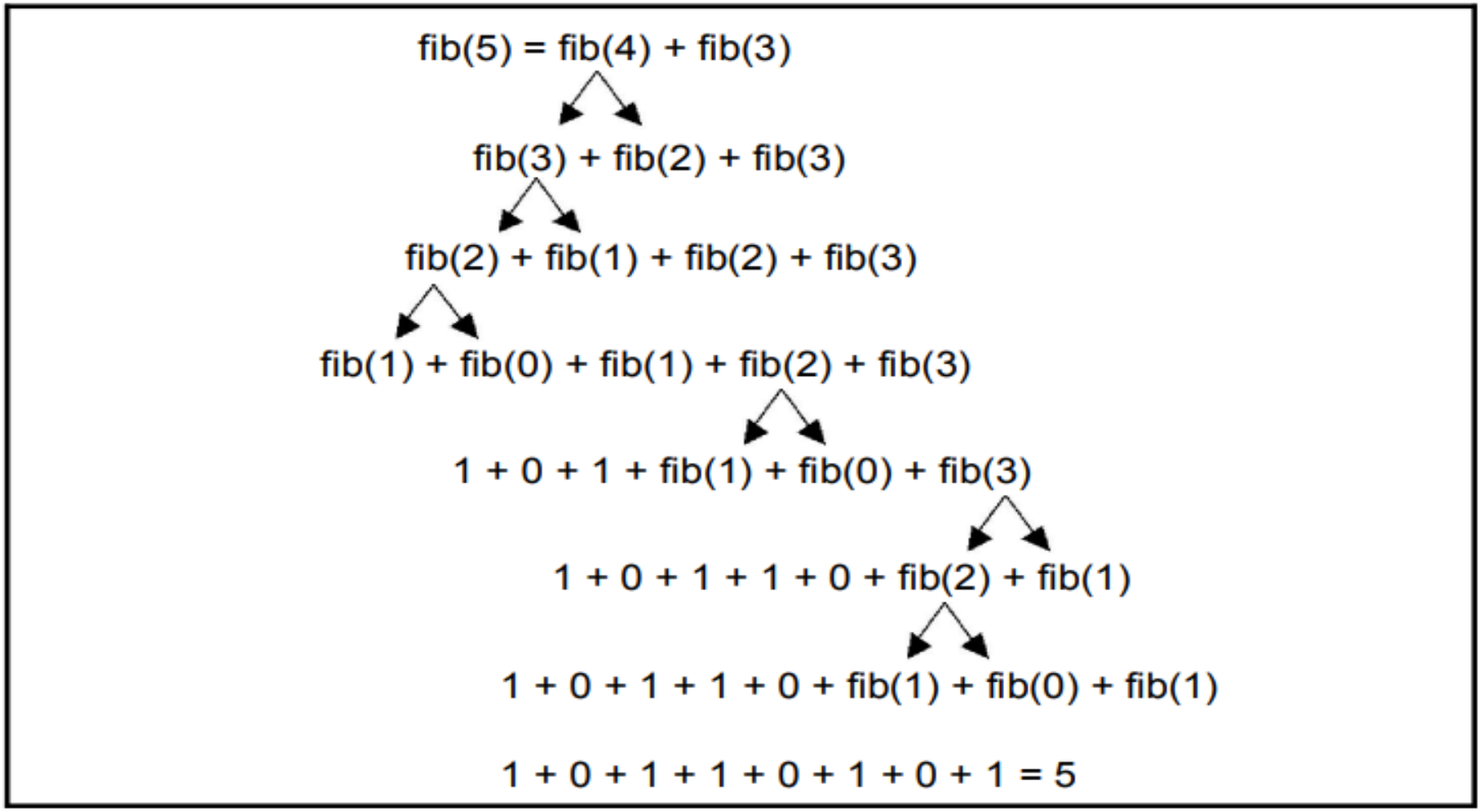
Difference b/w Recursion & loop

- Both involve repetition.
- Both involve a termination test.
- Both can occur infinitely.

Iteration	Recursion
Iteration explicitly user a repetition structure.	Recursion achieves repetition through repeated function calls.
Iteration terminates when the loop continuation.	Recursion terminates when a base case is recognized.
Iteration keeps modifying the counter until the loop continuation condition fails.	Recursion keeps producing simple versions of the original problem until the base case is reached.
Iteration normally occurs within a loop so the extra memory assigned is omitted.	Recursion causes another copy of the function and hence a considerable memory space's occupied.
It reduces the processor's operating time.	It increases the processor's operating time.

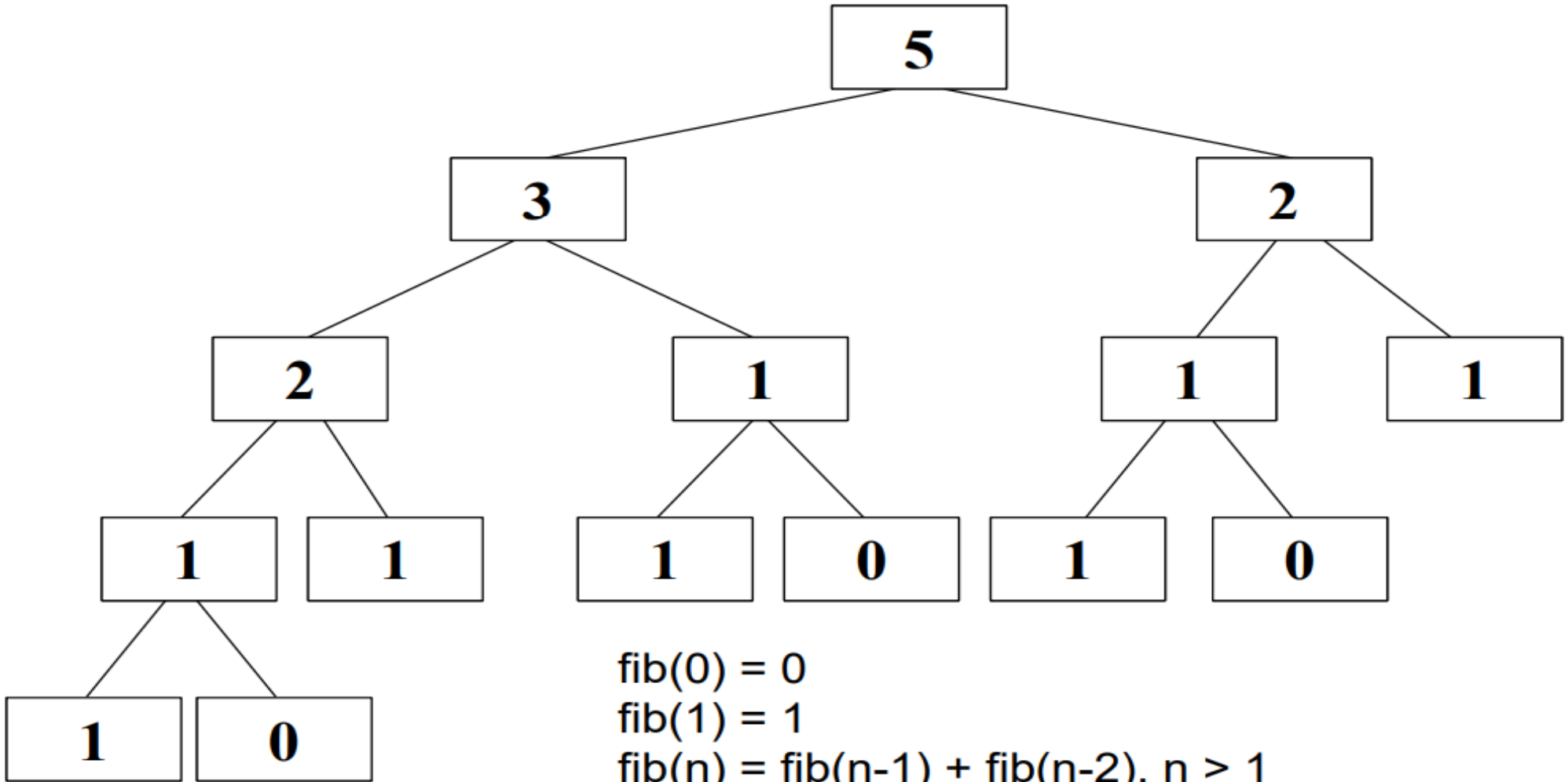
Fibonacci

We will now use the definition to compute fib(5):



We see that fib(2) is computed 3 times, and fib(3), 2 times in the above calculations.
We save the values of fib(2) or fib(3) and reuse them whenever needed.

Recursive Call Tree



fib(0) = 0
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2), n > 1