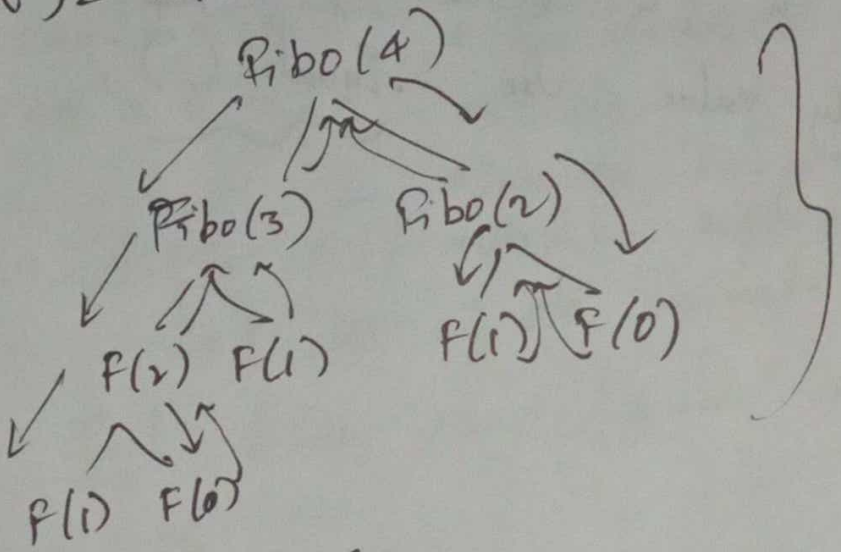
Recursion:-

* All function calls go into stack memory.
  → After execution of f^n is removed from stack and called to the place where called.
* Base condition is important to write.
* Without base condition — ∞ no of calls created in stack
  → at a point stack overflows. So program error!!!
* Recursion ⇌ iteration. (loops).
* Space complexity → not constant - due to calls.

Fibonacci:-

Fibo (N) = Fibo (N-1) + Fibo (N-2) → Recurrence relation




```
* class Fibonacci {
    psvm () {
        fiboc (n); }
  ps int fiboc (int n){
        if (n<2){
            return n; }
        return fiboc(n-1)+fiboch-2); } }
```

* If one rec f^n depend on the next f^n call then called "Tail recursion" not

Steps:

(1) Write into small problems.
(2) Write rec rel^n.
(3) Write rec tree
(4) Write the flow of function call.
  • Calls flow from left to right — tree calls.
(5) Observe value returning.

Variables:-

• Arguments, return type.

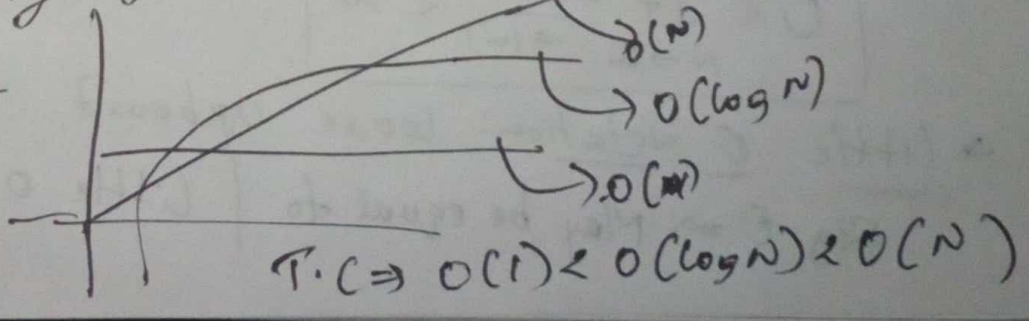
* Types of recurrences:-

• Linear rec rel^n — Fibonacci
• Divide and conquer - Search space & such - reduces.
  • Binary Search.

Time & Space Complexity:-

• Time complexity != time taken.
• Time complexity gives us rel^n abt time wrt input.

*



Revision:

* Factorial:

$f(5) = f(4) \times 5 = f(3) \times 4 \times 5 \ldots$

5
4 × 5
3 × 4 × 5
2 × 3 × 4 × 5
1 × 2 × 3 × 4 × 5

$\Rightarrow 1$

```
if (n == 1) {
    return 1;
}
return n * fact(n-1);
```

Step wise: $5 \times 4 \times 3 \times 2 \times 1$ ⇒ Main value or User given value.

(2) Sum of digits:

N = 342 ⇒ 1+3+4+2

342 + 2
13 + 4 + 2
1 + 3 + 4 + 2

⇒ 10 Stop;

(3) Reversing number

N = 12345 ⇒ 54321

5 + 1234
5 + 4 + 123