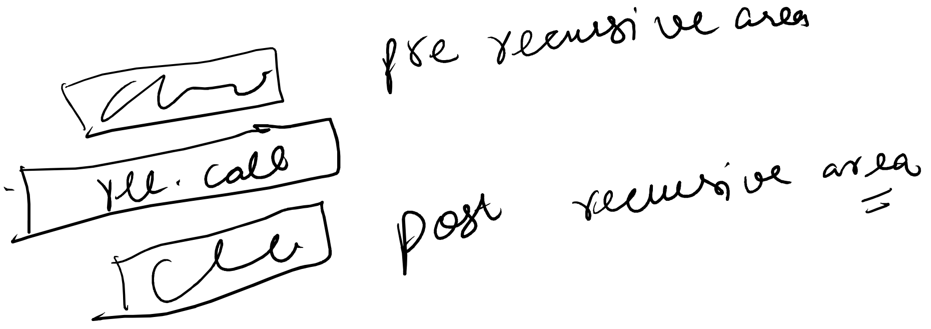
2.2 - Recursion - II

Saturday, October 4, 2025 3:08 PM

#

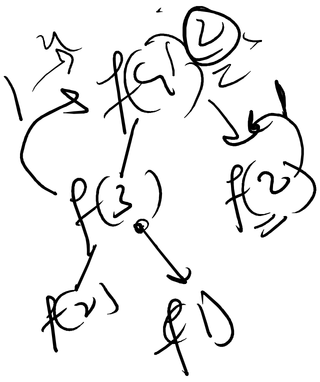


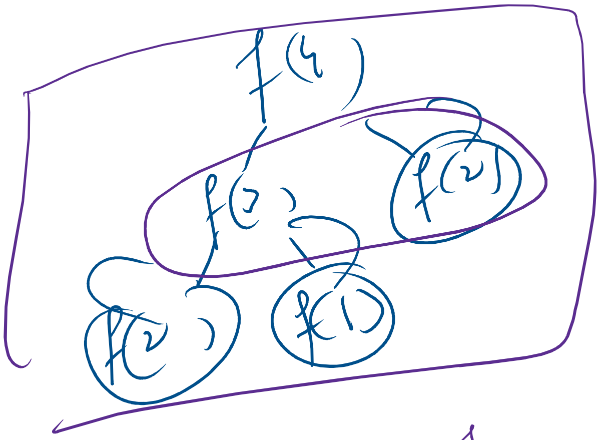
fibonacci

```
fib(n) {
1  if (n==1 || n==2) ret n-1
2  int x = fib(n-1);
3  int y = fib(n-2);
4  ret x+y
}
```

0 1 1 2

f(2)
f(3)
f(4)
m(f(4)
1 2 3 4

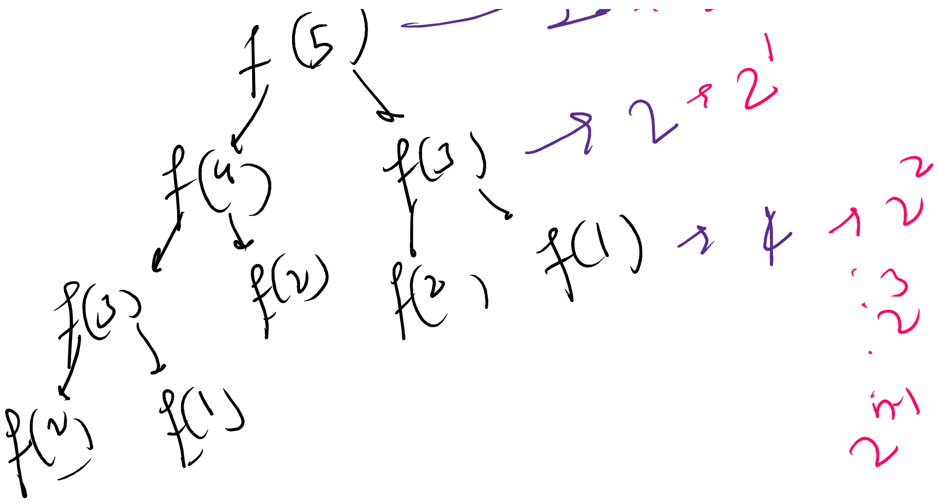




$f(5)$ — 1×2^0

2×2^1

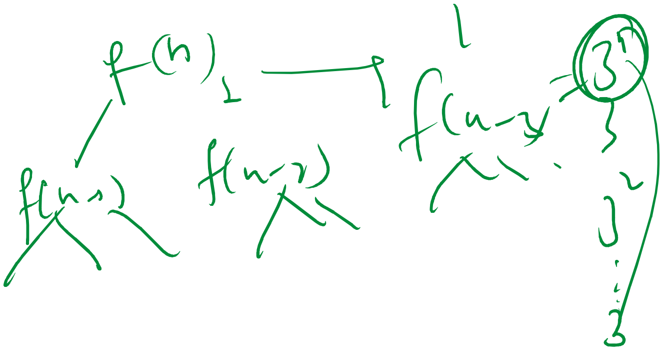
$\boxed{2^n}$,



⇒ Whenever there is branching in Recursion

$$T.C = (\text{No of branches})^{\text{depth of recursion}}$$

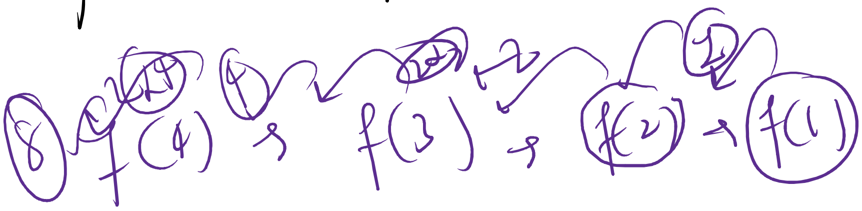
$$\left[f(n) = f(n-1) + f(n-2) + f(n-3) \right]$$



$$f(n) = 2 * f(n-1)$$

$$f(n) = f(n-1) + f(n-1)$$

$n = 20$



⊛ Math.pow(x,n) → O(n)

⊛ Base case is always written on changing variables

$2^{10} \to \boxed{2^5}$ e exe x^n $x^{n/2}$ → Exe

$2^{11} \to 2 \times 2^{10} \to 2^5$ exe

Recursion on Arrays

[int arr[]] → for(i=0; i<n; i++){
sout(arr[i]);
}

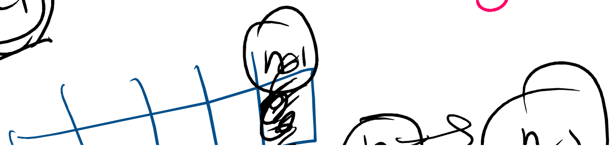
input, current index, length

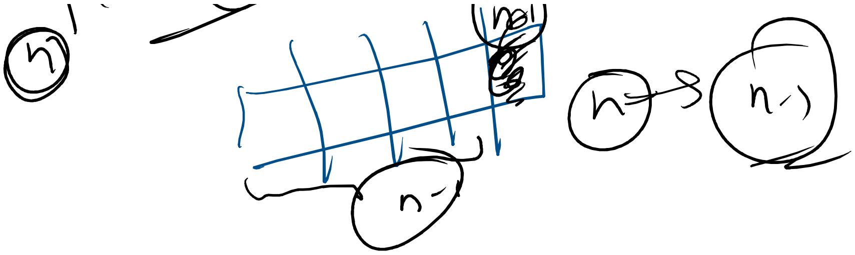
⇒ f(arr, i, n) → Template I

⇒ f(arr, n) → length of array remaining → Template II

f(x, n-1)

n




n
n-1
n-1
n
n-1