Fibonacci Series

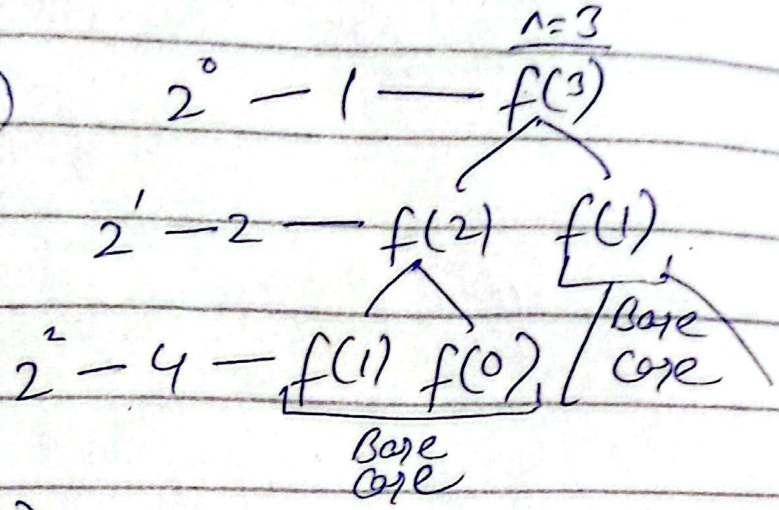
0, 1, 1, 2, 3, 5, 8, 13 ———— n^{th} fib term

Base case — 0th term, 1st term

n = 6

$t_n = t_{n-1} + t_{n-2}$

?



```
int fib(int n){
if(n==0||n==1)
    return n
return fib(n-1) + fib(n-2);
}
```

$$Sum = \frac{a(r^n - 1)}{(r-1)}$$

T.C = total calls + W.D in each call

$[2^0 + 2^1 + 2^2 + 2^3 + \cdots 2^{n-1}]$ — G.P

$$\frac{2^0 * (2^n - 1)}{2-1} = 1 * (2^n - 1) = \boxed{(2^n - 1)}$$

$$(2^n - 1) * K$$

$$\boxed{T.C = O(2^n)}$$

S.C = Depth of R.T * Memory

$= n * 1$

$$\boxed{S.C = O(n)}$$

→ Check if Array is Sorted

0	1	2	3	4
1	2	3	4	5

→ Ascending order

We will check for idx i
arr[i] >= arr[i-1]
↓
comparison case

arr, n=5
Compare arr[n-1] >= a[n-2]
we will go from the end

1	2	3	4	5
			n-2	n-1

} solved — sorted

will go on until base case hit

arr, n=4
arr, n=3
arr, n=2
↓
arr, n=1
arr, n=0 } Base case → return True already sorted

```
IsSorted(arr, n){
    if(n==0 || n==1) return True
    return arr[n-1] >= arr[n-2]
        && IsSorted(arr, n-1)
}
```

f(4)
↓
f(3)
↓
f(2)
↓
f(1)
↓
f(0)

T.C = n * O(1)

$\boxed{T.C = O(n)}$

$\boxed{S.C = O(n)}$

→ Binary Search

$mid = st + \frac{(e-s)}{2}$

|-1|0|3|5|9|12|

st=0 mid end=5

```
int bs(arr, tar, start, end){
 if(st<=end){
   mid=(st+ e-s)/2
 ① if(a[mid]==tar){
        return mid }
 ② if(a[mid]<=tar){
        2nd Half
   return bs(arr, tar, mid+1, end)
 ③ else 1st Half
   return bs(arr, tar, start, mid-1)
   }
   return -1
 }
```

T.C = $\log_2 * O(1)$

T.C = $\log_2$

$\boxed{T.C = O(\log n)}$

$\boxed{S.C = O(\log n)}$

$\frac{n}{2^k} = 1$

$n = 2^k$

$\log_2 n = k$

f(6) ↓ f(3) ↓ f(1)

n ↓ $n/2^1$ ↓ $n/2^2$ ↓ $n/2^3$ ↓ $1/\frac{n}{2^k}$