$i: 0 \longrightarrow n-1$

$j: 0 \longrightarrow n-1$ $\quad n/2$

$i=0$: $\quad j: 0 \to n-1 \to n$

$i=1$: $\quad j: 1 \to n-1 \to n-1$

$i=2$: $\quad j: 2 \to n-1 \to n-2$

$\vdots$

$i=n-1$: $\quad j: n-1 \to n-1 \to 1$

$1+2+3 \ldots n$

$= \frac{n(n+1)}{2} = O(n^2)$

$i: 0 \longrightarrow n-1 \longrightarrow n$

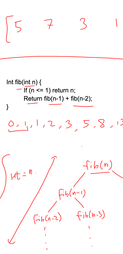
$j: 0 \to n/2 \longrightarrow n/2$

$\Rightarrow O(n \cdot \frac{n}{2}) = O(n^2)$

[5 7 3 1 6 2 8 0]

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

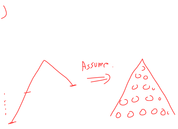
0, 1, 1, 2, 3, 5, 8, 13, 21 . . . . . .



1) Find the work done by a single func. call ignoring the recursive calls — x

2) Total no of func calls → ie, the no. of nodes in the rec. tree. — y

3) $TC = O(xy)$



Total no of nodes =

$\underbrace{1+2+4+8+ \ldots}_{n\text{-terms}}$

$a=1 \quad r=2 \quad n=n$

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

$\text{sum} = 1 \cdot \frac{(2^n-1)}{2-1} = O(2^n)$

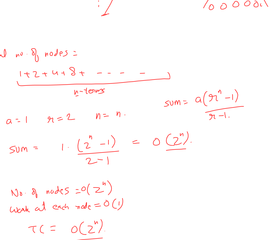
No of nodes $= O(2^n)$

work at each node $= O(1)$

$TC = O(2^n)$

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

$n = 5$



$AS = O(n)$

Step-1 : Find the space need by a single func call $\to x$

Step-2 : find the ht. $\to h$

Step-3 : $AS = O(x \cdot h)$.

```
int fun(int n) {
    if (n <= 1) return n;
    For (int i = 0; i < n; i++) { ... do something O(1) ...}
    Return fun(n-1) + fun(n-2);
}
```

No of nodes $= O(2^n)$.

work done at each node $= O(n)$.

$TC = O(n \cdot 2^n)$.

$AS = O(n)$.

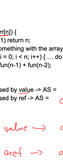
```
int fun(int n, int[n]) {
    If (n <= 1) return n;
    ... do something with the array ...
    For (int i = 0; i < n; i++) { ... do something O(1) ...}
    Return fun(n-1) + fun(n-2);
}

If array is passed by value -> AS =
If array is passed by ref -> AS =
```

$O(n^2)$

$O(n)$

Pass by value → a copy of the arr. is passed

Pass by ref → actual arr. / reference is passed



$AS = O(n^2)$ if arr[] is passed by value.

$AS = O(n + n) = O(n)$

to store the rec. stack

to store a single copy of the array.

No. of func calls $= O(2^n)$

work done in each func call $\xrightarrow{\text{arr is pass by value.}} O(n+n) = O(n)$

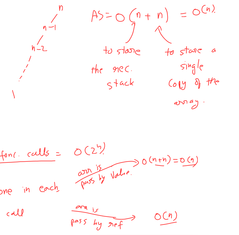
$\xrightarrow{\text{arr is pass by ref}} O(n)$

```
void fun(int n, bool b) {
    ...
    return fun(n/2, true) + fun(n/2, false);
}
```

$T(n) = T(\frac{n}{2}) + T(\frac{n}{2}) + c$

$T(n) = 2T(\frac{n}{2}) + c$

$n = \log n$



$n, \frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \ldots \ldots 1$

no. of terms = ht.

$1, 2, 4, 8 = \ldots \ldots, \frac{n}{2}, n$

$a \quad ar \quad ar^2 \quad ar^{n-1}$

$n = 1 \cdot 2^{n-1}$

$2^{n-1} = n$

$n-1 \to \log_2 n$

$n = O(\log_2 n) = O(\log n)$

Total no of nodes:

$\underbrace{1+2+4+8 \ldots \ldots}_{\log n \text{ terms}}$

$a = 1$

$r = 2$

$n = \log n$

$1 \cdot \frac{(2^{\log n}-1)}{2-1} = n-1 = O(n)$

$TC = O(n)$. $\quad AS = O(\log n)$.