- **TIME COMPLEXITY**

RECURRENCE RELATIONSHIP

Lets see it

A recurrence relation is an equation that recursively defines a sequence.

Fibonacci series:

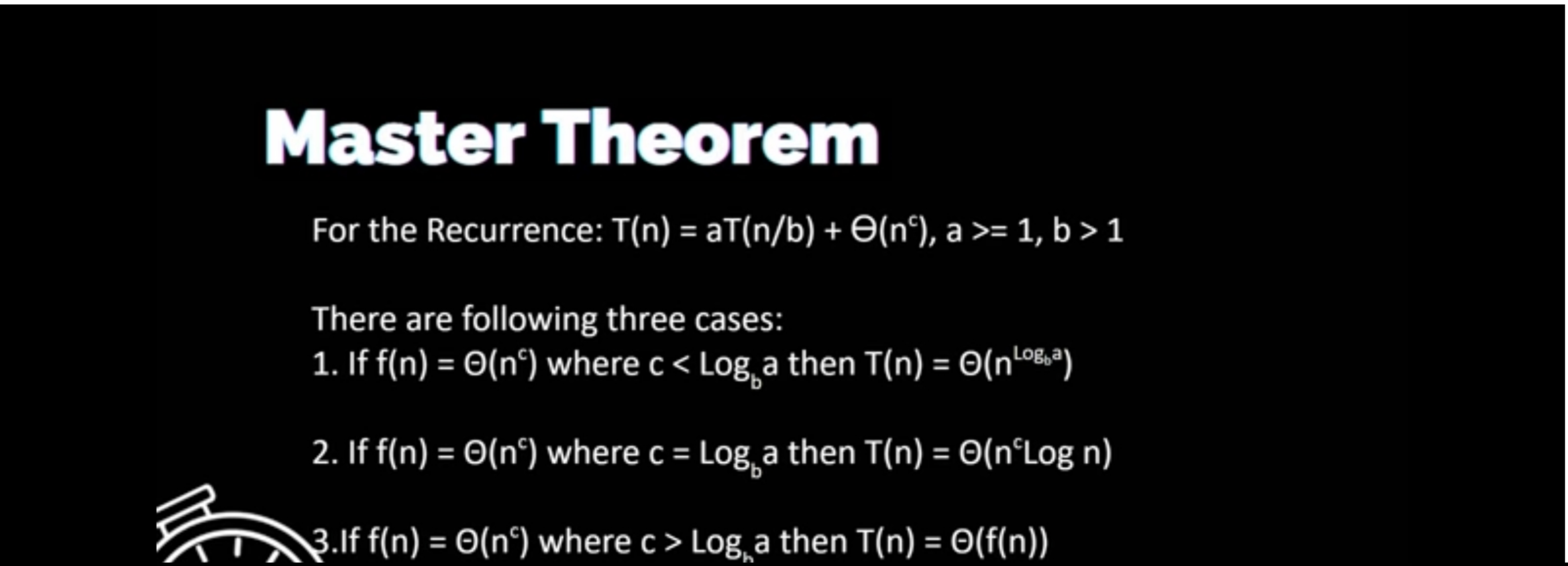
F(n) =f(n-1) + f(n-2)

MASTER THEOREM

Gives the time complexity for the recurrence relation:

T(N) =aT(N/b) + f(N)

For recurrence: T(n) =aT(n/b)+O(n^c)



Recurrence Tree Method:

1. T (n) = T (n-1) + n

$T(n) = T(n-1) + n$
$T(n-1) = T(n-2) + n-1$
$T(n-2) = T(n-3) + n-2$
.
.
$T(1) = 1$

Recurrence Tree Method:

1. T (n) = T (n-1) + n

$T(n) = T(n-1) + n$
$T(n-1) = T(n-2) + n-1$
$T(n-2) = T(n-3) + n-2$
.
.
$T(1) = 1$

Adding all the terms, we get

Recurrence Tree Method:

$T(n) = n + (n-1) + (n-2) + (n-3) + \dots + 1$
$T(n) = (n * (n+1))/2$

$T(n) = \Theta(n^2)$

```
/*
recurrence: in best case O(nlogn)
        in worst case              hence time complexity = O(n^2)
        T(n) = T(n-1) + n
        T(n-1) = T(n-2) +n-1
        .
        .
*/
/*
recurrence :
T(n) = 2T(n/2) + n                    level: n/2^k =1
T(n/2) = 2T(n/4) + n/2                       n = 2^k
T(n/4) = 2T(n/8) + n/4                       k =log n
T(n/8) = 2T(n/16) + n/8
.
.                                  T(n) = n+n+n+n+n+n .....log n
.                                  T(n) = nlogn
T(1) = 1
*/
```