**Time Complexity**

**Recurrence Relation**

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

Let's see it with an example.

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 the Recurrence: T(n)=aT(n/b) +(nc), a>=1,b>1

There are following three cases:

1. If f(n)= (nc) where c<Logba then T(n)= (nLogba­)
2. If f(n)= (nc) where c=Logba then T(n)= (nc­Logn)
3. If f(n)= (nc) where c>Logba then T(n)= (f(n))

Problems:

1. T(n)=2T(n/2) + ϴ(n)

a=2, b=2, c=1

* C=logba

Time Complexity: (nc­Log2n)

Merge sort time complexity example

1. T(n)=3T(n/2) + ϴ(n2)

a=3, b=2, c=2

* C>logba

Time Complexity: (n2)

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

T(n)=n+(n-1)+(n-2)+(n-3)+…..+1

T(n)=(n\*(n+1))/2

T(n)= (n2)