

# ***Recurrence Relation***

**–2**

***The recurrence can also be represented by sequence and terms as we see : 0, 2, 4, 6, 8 .... is a sequence of even numbers.***

***if we go through the sequence like  $t_0, t_1, t_2, t_3, \dots, t_n$ .***

***Where,***

$$t_0 = 0$$

$$t_1 = t_0 + 2 = 2$$

$$t_2 = t_1 + 2 = 4$$

***... ....***

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

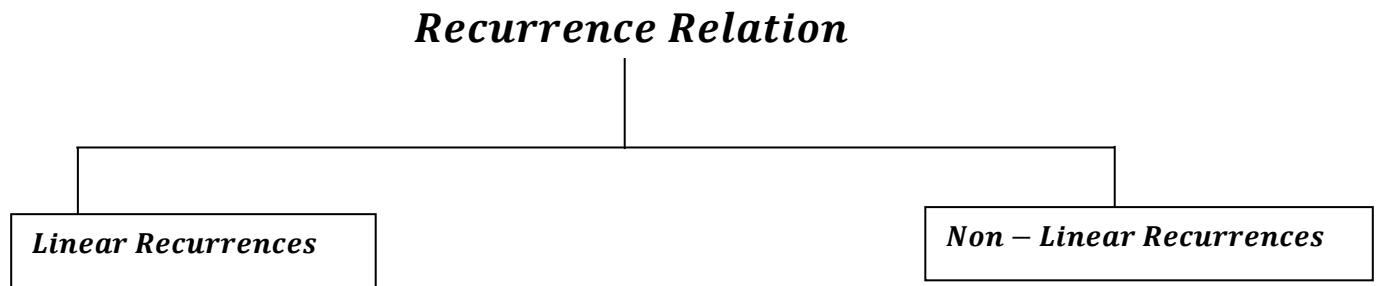
***if  $T(n)$  denotes the time complexity of the algorithm .***

$$T(n) = \begin{cases} T(n-1) + 2 & \text{when } n \geq 1 \\ 0 & \text{when } n = 0 \end{cases}$$

***There are two types of recurrence relation:***

***→ Linear Recurrences.***

***→ Non – Linear Recurrences.***



## ***Linear Recurrences***

***Linear recurrence equation for a sequence  $\{t_0, t_1, \dots, t_n\}$  expresses the final term  $t_n$  as a linear combination of its previous terms in a polynomial form.***

***Linear recurrence looks like :***

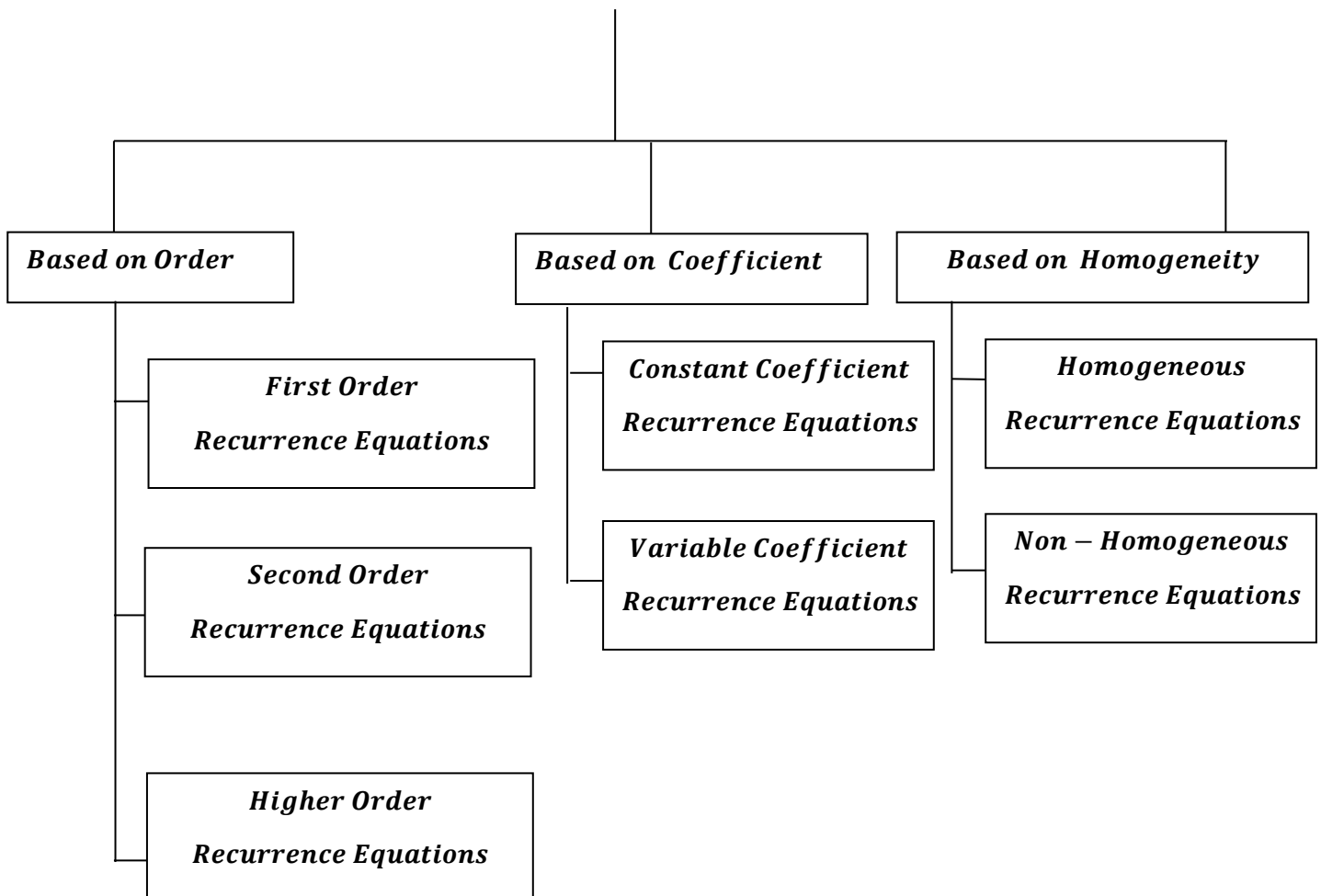
$$a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \dots + a_k t_{n-k} = f(n)$$

$$f(n) = \sum_{i=0}^k a_i \times \sum_{i=n}^{n-k} t_i$$

where,  $k$  and  $a_i$  terms are constant and  $a_0, a_k$  are non – zero and  $k$  being the order of the recurrence equation.

## ***Division of Linear Recurrences***

### ***Linear Recurrences***



## ***A. Based On Order***

***The number of preceding terms used for computing the present term of a sequence is called the order of a recurrence equation.***

### **1. First Order Recurrence Equation:**

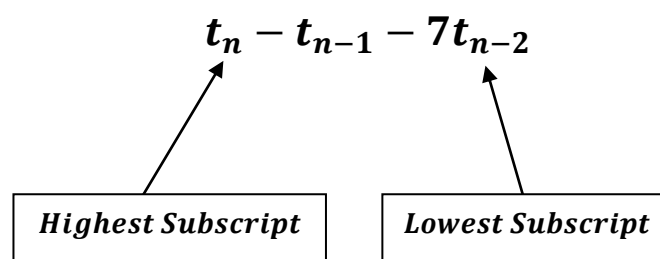
***The general form of a first – order recurrence equation is as follows:***

$$a_0t_n + a_1t_{n-1} = f(n)$$

***If  $t_n$  is computed using only one previous term, then the recurrence equation is called a first – order linear equation.***

***How find whether an equation is first order or not?***

***Lets take an equation:  $t_n - t_{n-1} - 7t_{n-2} = 0$ .***



*Hence order will be  $= n - (n - 2) = 2$  i. e.,*

*Difference between the highest and lowest subscripts of the dependent variable in a recurrence equation.*

*if we see the factorial,*

$$t_0 = 1$$

$$t_1 = t_0 \times 1 = 1$$

$$t_2 = t_1 \times 2 = 2$$

....

$$t_n = t_{n-1} \times n$$

*By above we take the above to compute, we get:*

$$t_n - nt_{n-1} = 0$$

*Hence,  $n - (n - 1) = 1$  i. e. first order.*

*Also if we take the time complexity  $T(n)$ , hence per  $T(n - 1)$  it takes constant time  $c$  i. e.  $T(1)$  in stack.*

*Therefore,  $T(n) = T(n - 1) + T(1)$  or*

*$T(n) = T(n - 1) + C$ , where  $C$  is constant.*

*and  $C$  represents 1 unit of time and  $T(1) = 1$ ,*

*hence  $C = T(1)$ .*

*And it is not  $n \times T(n - 1)$  as the multiplication takes in stack for a constant time , hence only it matters is  $T(n - 1)$  , i. e. the recursive use of stack's push and pop.*

*Hence it goes like:*

$$\begin{aligned}T(n) &= \\ \Rightarrow T(n - 1) + T(1) \\ \Rightarrow T(n - 2) + T(1) + T(1) \\ \Rightarrow T(n - 3) + T(1) + T(1) + T(1)\end{aligned}$$

*We can further represent the above as : –*

$$t_n = t_{n-1} + 1 \text{ (For single sequence `t`)}$$

$$\text{or, } t_n - t_{n-1} = 1.$$

*Hence here also we get :  $n - (n - 1) = 1$ , i. e. first – order recurrence equation.*

## **2. Second Order Recurrence Equation:**

*The generic form of a second – order recurrence equation is given as follows:*

$$a_0t_n + a_1t_{n-1} + a_2t_{n-2} = f(n)$$

***Fibonacci series which is invented by Leonardo Fibonacci, also known as Leonardo Bonacci an italian mathematician. The series states:***

***The base or starting numbers are 0 and 1. That is:***

$$\mathbf{fib(0) = 0}$$

$$\mathbf{fib(1) = 1}$$

$$\mathbf{fib(2) = fib(0) + fib(1) = 0 + 1 = 1}$$

$$\mathbf{fib(3) = fib(2) + fib(1) = 1 + 1 = 2}$$

$$\mathbf{fib(4) = fib(3) + fib(2) = 2 + 1 = 3}$$

... ..

***We can relate this with Recurrence Equation:***

$$\mathbf{fib(n) = fib(n - 1) + fib(n - 2).}$$

$$\mathbf{or, T(n) = T(n - 1) + T(n - 2).}$$

***Hence for single sequence `t` :***

$$\mathbf{t_n = t_{n-1} + t_{n-2} .}$$

*And fibonacci series is `second order of recurrence`.*

$$i. e. t_n - t_{n-1} - t_{n-2} = 0$$

$\Rightarrow n - (n - 2) = 2$ , hence second order of recurrence.

### **3. Higher Order Recurrence Equation:**

*Higher – order linear recurrence equations of order  $k$  can be formulated as follows:*

$$a_0 t_n + a_1 t_{n-1} + \cdots + a_k t_{n-k} = f(n)$$

*i. e.,*

$$f(n) = \sum_{i=0}^k a_i \times \sum_{i=n}^{n-k} t_i$$

*Here  $a_i$  and  $k$  are constants.*



## ***B. Based on Homogeneity***

### ***A. Homogeneous Recurrence Equation:***

***Suppose we have a recurrence equation:***

***$a_0t_n + a_1t_{n-1} + \dots + a_kt_{n-k} = f(n)$  and if  $f(n) = 0$ ,  
it is called a homogeneous equation.***

***Homogeneity test: Substitute  $t_n$  and all its factors  $t_{n-1}$ ,  
 $t_{n-2}, \dots, t_{n-k}$  with zero.***

***Eg: Fibonacci Series :***

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

***Substituting  $t_n, t_{n-1}, t_{n-2}$  with zero we get:***

$$0 = 0 + 0$$

***Therefore ,  $t_n = t_{n-1} + t_{n-2}$  is a homogeneous equation.***

### **B. Non – Homogeneous Recurrence Equation:**

*Suppose we have a recurrence equation:*

*$a_0t_n + a_1t_{n-1} + \dots + a_kt_{n-k} = f(n)$  and if  $f(n) \neq 0$ ,  
it is called a non – homogeneous equation.*

*Lets take factorial :*

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

*Applying the homogeneity test:*

$$\begin{aligned} 0 &= 0 + 1 \\ &= 1 \end{aligned}$$

*Hence, for factorial it is non – homogeneous in nature.*

## ***C. Based on Coefficient***

*In the generic linear recurrence equation :*

*$a_0t_n + a_1t_{n-1} + \dots + a_kt_{n-k} = f(n)$ , the terms  $a_i$  can be constants or variables .*

*Based on this scenario , we can classify linear recurrence equations into two types:*

- 1)linear recurrence equations with constants coefficients.*
- 2)linear recurrence equations with variable coefficients.*

*Example:*

$$t_n = n \times t_{n-2}$$

*This recurrence equation is dependent on the variable  $n$  and doesnot have constant coefficients. However, in algorithm study , these kinds of equations are rare and mostly constant coefficient linear recurrence equations are encountered.*

# ***Non – Linear Recurrences***

*The non – linear recurrence equation of a sequence  $\{t_0, t_1, \dots, t_n\}$  expresses  $t_n$  as a non – linear combination of its previous terms. In algorithm study, a unique form of non – linear recurrence equations, called divide – and – conquer recurrences is often encountered.*

*The divide – and – conquer recurrences are of the following form:*

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

*Here  $a$  is the number of subproblems,  $n$  is the size of the problem,  $\frac{n}{b}$  is the size of the subproblem, and  $f(n)$  is the cost of work done for non – recursive calls, which accounts for the division of a problem into sub problems and combination of the results of those sub problems.*

***Some of Divide and Conquer recurrences are:***

***Merge Sort, Quick Sort , Binary Search etc.***

\*\*\*\*\*