Recurrence Relations:-

In mathematics, a recurrence r/n is an equation that expresses the n^{th} term of a sequence as a function of the k preceding terms, for some fixed k (independent from n), which is called the order of relation.

* An equation which represents a based on some rule.

* It helps in finding the subsequent term (next term) dependent upon the preceding term (previous term).

* If the previous term is a given inc. series, then we can easily determine the next term.

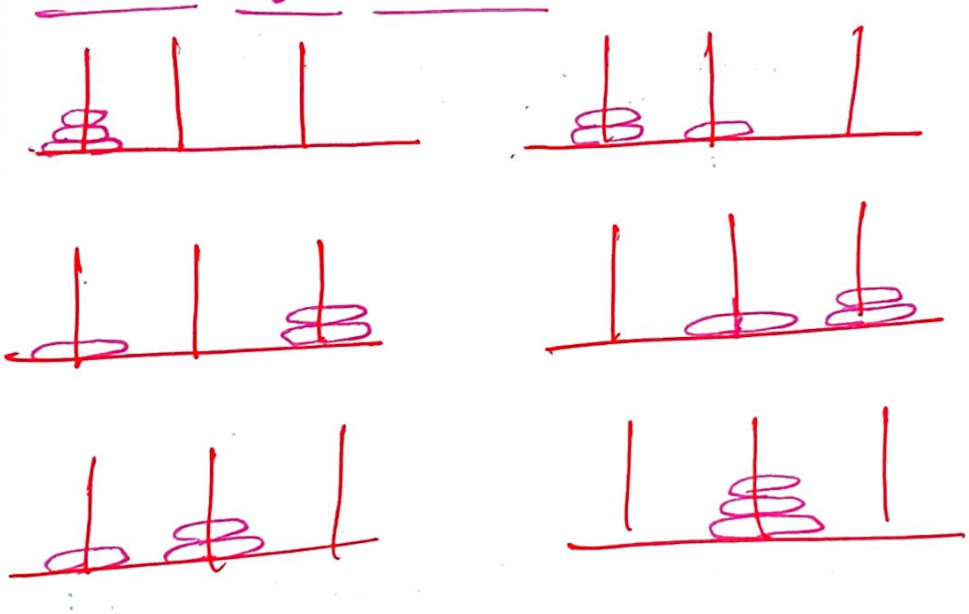
* Recurrence relations are used to reduce complicated problems to an iterative process based on simpler versions of the problem.

Ex: Tower of Hanoi puzzle.

A recurrence or recurrence relation defines an infinite sequence by describing how to calculate the n^{th} element of the sequence given the values of smaller elements, as in:

$$\boxed{T(n) = T(n/2) + n \quad T(0) = T(1) = 1}$$

Tower of Hanoi



RECURRENCE RELATION :-

★ Recurrence relations often rise in calculating the time and space complexity of algorithms.

★ Recursive algorithm is one which makes a recursive call to itself with a smaller i/p's.

Recurrences & running time :-

An equation that describe a function in terms of its values or smaller i/p's.

$$\boxed{T(n) = T(n-1) + n}$$

To solve the recurrence :

i) Find an explicit formula of the expression.

ii) Bound the recurrence by an exp. that involves n

Ex :-

$$\boxed{S(n) = \begin{cases} 0 & n=0 \\ c+S(n-1) & n>0 \end{cases}}$$

$$S(n) = \begin{cases} 0 & n=0 \\ n+S(n-1) & n>0 \end{cases}$$

$$T(n) = \begin{cases} c & n=1 \\ 2T(n/2)+c & n>1 \end{cases} \qquad T(n) = \begin{cases} c & n=1 \\ aT(n/b)+cn & n>1 \end{cases}$$

Examples of Recurrences :-

- $T(n) = T(n-1)+n$ $\quad \Theta(n)^2$

⇒ Recursive algorithm that loops through the i/p to eliminate one item.

- $T(n) = T(n/2)+c$ $\quad \Theta(\log n)$

⇒ Recursive algorithm that halves the i/p in one step.

- $T(n) = T(n/2)+n$

⇒ Recursive algorithm that haves the i/p but must examine every item in the i/p.

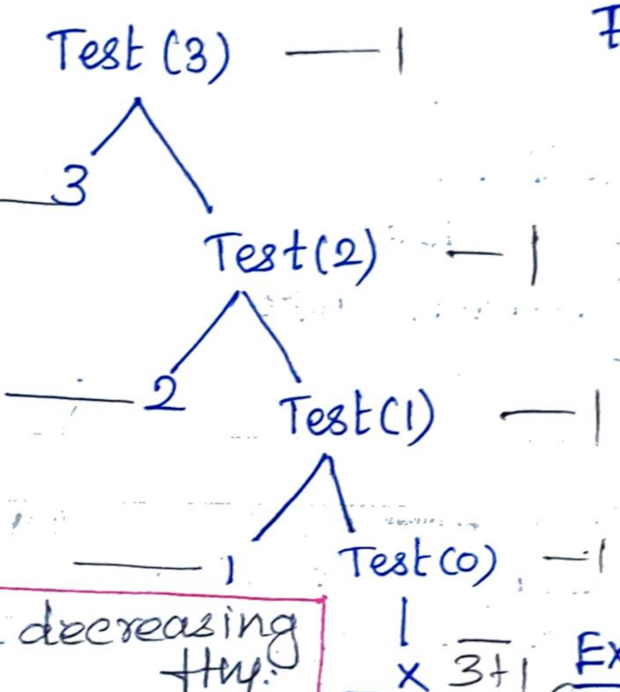
- $T(n) = 2T(n/2)+1$

⇒ Recursive algorithm that splits the i/p into 2 halves and does a constant amount of other work.

Recurrence Relations:-

$\boxed{T(n) = T(n-1) + n}$ //

Ex:1 Tracing tree / Recursive tree.



$f(n) = n+1$

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

T(n)
```
Void test (int n)
{
   if (n>0)
   {
1 —   printf("%d",n)
      Test(n-1);
   }
}
```

Test(3) printf n times.

Ex:1.

(n+1) calls

```
T(n) —— Void test(int n)
         {
1 ———      if(n>0)
           {
1 ———        printf("%d", n)
T(n-1) ———   Test (n-1);
           }
         }
```

$T(n) = T(n-1) + 1$ // → c/1/k/a/... anything

└→ (a.)   O(n)

prepare a recq. r/n)

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+1 & n>0 \end{cases}$$

(1.) Ref → (1a) → (1b)

Ex:2 - decreasing fun.

```
T(n) —— Void Test(int n)
         {
1 ———      if(n>0)
           {
n+1 ———      for(i=0; i<n; i++)
             {
n ———          printf("%d",n)
             }
T(n-1) ———   Test(n-1);
           }
         }
```

$T(n) = T(n-1) + 2n+2$ O(n)

↑ recurrence r/n. (1b) - O(n)

$T(n-1) + n$.

any constant c/a/k/1 Assumption

Ex:3 Factorial.

```
int factorial (unsigned int n)
{
  if (n==0)
  return 1
  return n * factorial (n-1);
}
```

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

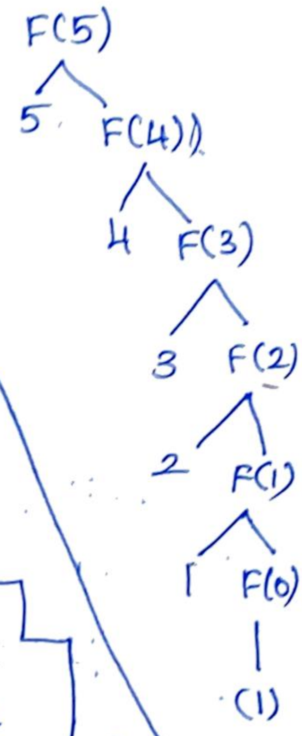
$= n - 1 + 3$

$= n - 2$

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

Recurrence relation is

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1)+1 & n > 0 \end{cases}$$



Solutions of Recurrence r/n's :-

Four Mthds/. for solving recurrence r/n's.

- Substitution Mthd.
- Iteration Mthd.
- Recursion Tree Mthd.
- Master Mthd.

$= 1+1+(n-1)+3$

-2

$\boxed{O(n)}$

Substitution Method: (1.a)

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+1 & n>0 \end{cases}$$

$T(n) = T(n-1)+1$ → (a)

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

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

$= T(n-2)+1$

$\Rightarrow T(n-2-1)+1$

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

Sub. $T(n-1)$ in (a)

$T(n) = [T(n-2)+1]+1$

$T(n) = T(n-2)+2$

$T(n) = [T(n-3)+1]+2 = [T(n-3)+1]+2$

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

⋮

continue for k times.

$$\boxed{T(n) = T(n-k)+k}$$

Assume $n-k=0$

$\therefore n = k$

$\therefore T(n) = T(n-n)+n$

$= T(0)+n$

$$\boxed{T(n) = 1+n}$$ → linear class

$O(n)$ $\Theta(n)$.

pg No: 1 (b)

Ex: 2. $T(n) = T(n-1)+n$

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+n & n>0 \end{cases}$$

Tree method:- - preparing a tree & tracing it.

$T(n)$ ——— n

n $T(n-1)$ ——— $n-1$

$(n-1)$ $T(n-2)$ ——— $n-2$

$(n-2)$ $T(n-3)$ ——— $n-3$

⋮

$T(2)$ ——— 2

——— 1

2 $T(1)$ — $T(0)$ — 0

1

$0+1+2+3+\dots+(n-1)+n$

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

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

(c)

Solns: to Recurrence Relations:- definition

↳. Soln. to Recursive Tree Method & Substitution Mthd.

Ex: 1.

Ex: Recursion Tree / Trav

eqn.:

Substi.

Substitution Mthd

Fwd — Bkwd.

$\underbrace{1 \text{ to } n}$. $\underbrace{n, n-1, n-2, \ldots 3, 2, 1, 0.}$

Advs:-

* Easy to prove
* prone to mistakes.

Ex: 2 Substitution Method:

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+n & n>0 \end{cases}$$

$T(n) = T(n-1)+n \rightarrow$ (a)

$\therefore T(n) = T(n-1)+n$

Sub $n = n-1 \Rightarrow \therefore T(n-1) = T(n-1-1)+n-1$

$\boxed{T(n-1) = T(n-2)+n-1} \rightarrow$ (1)

Sub (1) in (a)

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

$n-1 \quad T(n-1) = [T(n-2)+n-1]+n$

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

$\boxed{T(n-2) = T(n-3)+n-2} \rightarrow$ (2)

$= T(n-2)+(n-1)+n$ — (2)

$\rightarrow$ avoiding adding terms to prepare a seq.

$n-2 \quad T(n-2) = T(n-3)+(n-2)+(n-1)+n$ — (3)

Method followed for induction

after 2, continue for k steps

$T(n) = T(n-k)$

$$\boxed{T(n-k) = T(n-k)+(n-(k-1))+(n-k-2)+\cdots+(n-1)+n}$$

eqn: eq.. Assume $n-k$ has became 0

$\therefore n-k=0$

$n=k.$

$T(n) = T(n-n)+(n-n+1)+(n-n+2)+\cdots (n-1)+n$

$= T(0)+1+2+3+\cdots+(n-1)+n \rightarrow$ (4)

$= 1+\dfrac{n(n+1)}{2}$

$T(n) = O(n)^2.$

Ex:-

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1) + \log n & n > 0 \end{cases}$$

```
Void Test(int n)
{
  if(n>0)
  { for(i=1; i<n; i=i*2)
     { printf("%d", i);          log n
     }
     Test(n-1);                  T(n-1)
  }
}.
```

$T(n) = T(n-1) + \log(n)$

$O(n \log n)$

Properties of Asymptotic Notations:-

① General properties.

if $f(n)$ is $O g(n)$ then $a * f(n)$ is $O g(n)$

eg: $f(n) = 2n^2+5$ is $O(n^2)$

then $= 7 \cdot f(n) = 7(2n^2+5)$

$= 14n^2 + 35$ is $O(n^2)$

Also true for Ω & Θ. i.e. $\boxed{\forall \text{ all three notations}}$

$f(n) = \Omega g(n) \rightarrow a * f(n) \rightarrow \Omega g(n)$

② Reflexive property:-

if $f(n)$ is given then $f(n)$ is $O f(n)$

eg:- $f(n) = n^2 \Rightarrow O(\underset{\substack{\uparrow \\ f(n)}}{n^2})$

③ Transitive property:

if $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$

then $f(n) = O(h(n))$

eg:- $f(n) = n \xrightarrow{\text{upper bound}} g(n) = n^2 \xrightarrow{\text{upper bound.}} h(n) = n^3$

n is $O(n^2)$ & n^2 is $O(n^3)$

then n is $O(n^3)$.

$\boxed{\forall \text{ three notations } O, \Omega \,\&\, \Theta.}$

(4) Symmetric property: (Theta only)

If $f(n)$ is $\Theta(g(n))$ then $g(n)$ is $\Theta(f(n))$,

eg:- $f(n) = n^2 \rightarrow g(n^2) = n^2$

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

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

When both the fns are same, that they are symmetric.

(5) Transpose symmetric:- (O & Ω) only

If $f(n) = O(g(n))$ then $g(n)$ is $\Omega(f(n))$
UP. LB.

eg:- $f(n) = n$ $g(n) = n^2$

then n is $O(n^2)$ and
n^2 is $\Omega(n)$

If one fn. forms an upper bound for other fn then the other fn. will form a lower bound for the other fn.

If $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

When same fn. is both upper & lower bound

$g(n) \leq f(n) \leq g(n)$

$\boxed{f(n) = \Theta(g(n))}$ //

Properties of Asymptotic Notations :- #2.

If $f(n) = O(g(n))$

and $d(n) = O(e(n))$

then $f(n) + d(n) = O(\max(g(n), e(n)))$

$$\boxed{\begin{aligned} &\text{eg: } f(n) = n = O(n) \\ &d(n) = n^2 = O(n^2) \\ &f(n) + d(n) = n + n^2 = O(n^2) \end{aligned}}$$

If $f(n) = O(g(n))$

$d(n) = O(e(n))$

then $f(n) * d(n) = O(g(n) * e(n))$

$n \quad n^2 = n^3$

—x—