Design and Analysis of Algorithms.

UNIT-1

Properties of Algorithm:

1. Efficient Output
2. Finiteness.

Analysing of Algorithms:

→ It can be done in two ways.

Time Complexity T(n).

→ Machine indepedent

→ Counting no of instructions
(or)
no. of assignments
(or)
no. of operations.

Space complexity.

→ Instruction space

→ Data space

→ Environment space.

Constant time complexity.

→ not depends on input size.

```
int sum(a,b) {
    return a+b;      — 1
}
```

$T(n) = 1$

Linear time complexity.

```
int sum(A, n) {
    Sum=0;                      1
    for(i=0; i<n; i++)
        1    n+1  n
    Sum=sum+A[i];               — n
    return sum;                 — 1
}
```

Step Count function $T(n) = 3n + 4 = O(n)$

cost	Repetition
1	1
1	1
1	n+1
1	n
1	n
1	1

$T(n) = 3n+4.$

⇒ Linear time complexity depends on input size. (i.e n).

Asymptotic Notations:

1. Big Oh (O)
2. Omega (Ω)
3. Theta (Θ)
4. Small oh (o) $\Rightarrow f(n) < c.g(n)$
5. small Omega (ω) $\Rightarrow f(n) > c.g(n)$.

1 Big Oh (O):

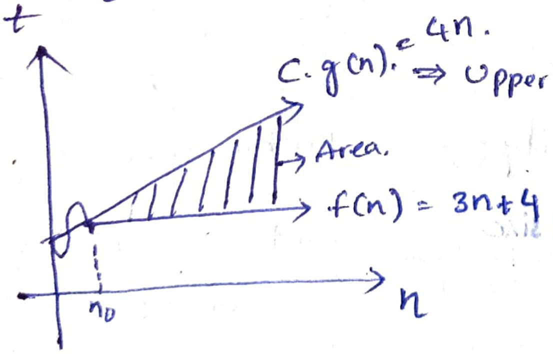
→ Considering two functions.

step count function
$f(n)$ lesser than $g(n) \rightarrow$ simple function.

$f(n) = O\,g(n)$ iff

$f(n) \leq c.g(n)$, $c > 0$, $n \geq n_0 \geq 1$

n_0 is the value from which value the inequality is satisfying.



Eg:-

$T(n) = f(n) = 3n+4$ $\qquad$ $g(n) = n$ (taking n because it is a simple linear function).

$f(n) = O\,g(n)$.

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

$3n+4 \leq c.n$ $\qquad$ Assume $c = 4$, $n_0 = 4$.

$3n+4 \leq 4n$.

$4 \leq n. \Rightarrow n \geq 4$

$\boxed{n_0 = 4}$

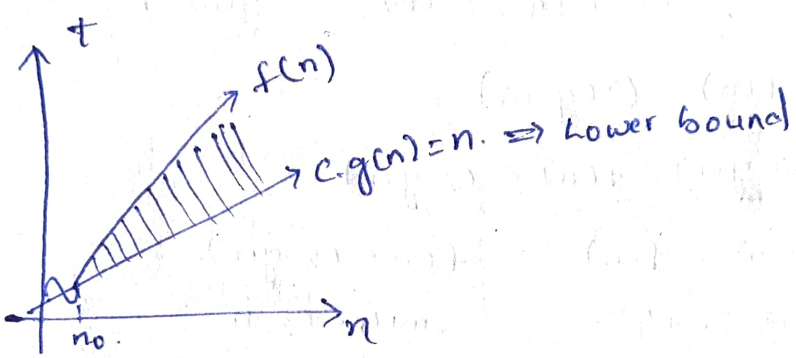
2. Omega (Ω):

→ Considering two functions.

$\underline{f(n)}$ greater than. $g(n) \rightarrow$ Simple function

$f(n) = \Omega(g(n))$ iff

$f(n) \geq c \cdot g(n)$, $c>0$, $n \geq n_0 \geq 1$



eg:-

$f(n) = T(n) = 3n+4 = O(n)$, $g(n) = n$.

$f(n) = \Omega(g(n))$

$f(n) \geq c \cdot g(n)$

$3n+4 \geq c \cdot n$.

$3n+4 \geq n$. $c = 1$, $n_0 = 1$

$2n+4 \geq 0$

$n+2 \geq 0$

$n \geq -2 \geq 1$.

$\boxed{n_0 = 1}$

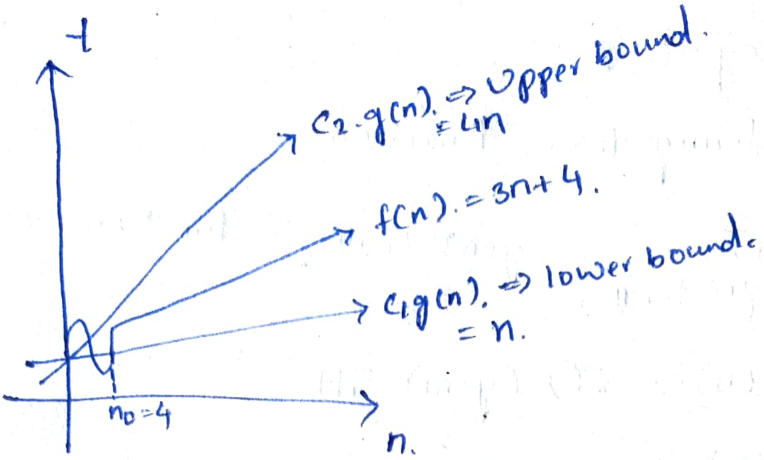
3. Theta (Θ):

→ Considering two functions.

$\underline{f(n)}$ equal $g(n) \rightarrow$ simple function,

$f(n) = \Theta(g(n))$ iff

$C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)$, $C_1, C_2 > 0$, $n \geq n_0 \geq 1$



eg: $f(n) = T(n) = 3n+4 = O(n)$, $g(n) = n$.

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

$c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$

$c_1 \cdot g(n) \le f(n)$. $f(n) \le c_2 \cdot g(n)$.

$c_1 \cdot n \le 3n+4$ $3n+4 \le c_2 \cdot n$.

$n \le 3n+4$ $3n+4 \le 4n$.

$c_1 = 1,\ n_0 = 1$ $n_0 = 4,\ c_2 = 4$

$T(n) = 3n+4 = \Theta(n)$.

$T(n) = 3n+4 = \Omega(n)$.

eg:

① $T(n) = 2n+5$, $g(n) = n$.

$f(n) = T(n) = 2n+5$.

$2n+5 \le c \cdot g(n)$

$2n+5 \le c \cdot n$.

$2n+5 \le c \cdot n$

$2n+5 \le 3n. \Rightarrow c = 3$.

$\boxed{\cancel{n_0 = 2}}$ $5 \le n$

$n \ge 5$

$\Rightarrow \boxed{n_0 = 5}$

$\therefore T(n) = 2n+5 = O(n)$.

(2) $f(n) = T(n) = 2n+5$

$f(n) \geq c \cdot g(n)$

$2n+5 \geq c \cdot n. \Rightarrow c = 1.$

$2n+5 \geq n.$

$n+5 \geq 0.$

$n \geq -5$

$n \geq -5 \geq 1. \Rightarrow \boxed{n_0 = 1}$

$\therefore T(n) = 2n+5 = \Omega(n).$

(3) $f(n) = T(n) = 2n+5$

$f(n) = 2n+5$ $\qquad g(n) = n.$

$\underline{c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n).}$

$c_1 \cdot n \leq 2n+5$

$n \leq 2n+5$

$c_1 = 1,$

$n+5 \geq 0$

$n \geq -5 \geq 1$

$\boxed{n_0 = 1}$

$f(n) \leq c_2 \cdot g(n).$

$2n+5 \leq c_2 n.$

$\boxed{c_2 = 3.}$

$2n+5 \leq 3n.$

$5 \leq n.$

$n \geq 5$

$\boxed{n_0 = 5}$

$\therefore T(n) = 2n+5 = \Theta(n).$

```
            T(n).
RSum(A, n) {
    if (n==0) return 0;              1
    return RSum(A, n-1) + A[n]       T(n-1)+1
}                                    T(n-1)+2
```

$$T(n) = \begin{cases} T(n-1)+2 & n>0 \\ 2 & n=0. \end{cases}$$

Recurrsive Relation.

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

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

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

$\vdots$

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

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

1) Fibonacci Series:

	Cost	Repetition
Fib(n) {		
fib1 = 0;	1	1
fib2 = 1;	1	1
for(i=2 to n) do {	1	1
	1	n
fib3 = fib1+fib2	1	$n-2$
fib1 = fib2;	1	$n-2$
fib2 = fib3;	1	$n-2$
}	1	$n-2$
}		$5n-5 = O(n)$

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

2) Addition of two matrices:

```
Add (A, B, C, m, n) {
  for (i=0 to m) {
    for(j=0 to n) {
      C (i, j) = A(i, j)+B(i, j);
    }
  }
}
```

$i=0$ — 1

$i<m$ — $m+1$

$i++$ — m

$j=0$ — m

$j<n$ — $m(n+1)$

$j++$ — mn

statement — mn

$3mn+4m+2$

$3mn+4m+2 = O(mn)$

$T(n) = O(mn).$

3) Linear Search

```
Linear search (A, n, key) {
  for (i = 0 to n) do {
    if (Key == A[i])
      return i;
  }
  return -1;
}
```

→ Best case: $T(n) = O(1)$.

→ Average case: $T(n) = \frac{n}{2} = O(n)$
middle for this case. A.N

→ Worst case: $T(n) = O(n)$

4). Binary Search.

```
Binary search ( A, start, end, key) {
  if (start > end) return -1;
  mid = (start + end) / 2;
  if (key == A[mid]) return mid;
  else if (key < A[mid]) {
     return Binarysearch (A, start, mid-1, key);
  }
  return Binary search (A, mid+1, end, key);
}
```

$$T(n) = \begin{cases} 2, & n=0 \\ 4, & \text{found at mid} \\ 5+T(n/2) & n>0 \end{cases}$$

$$T(n) = \begin{cases} 2 & n=0 \\ 4 & n=1 \\ 5+T(n/2) & n>0 \end{cases}$$

$T(n) = 5 + T(n/2)$

$= {}^{2}T(n/4) + 5 + 5$

$= {}^{3}T(n/8) + \overset{3}{5+5+5}$

$\vdots$

$= {}^{k}T(n/2^k) + k(5)$.

$2^{\frac{n}{k}} = 1$.

$n = 2^k$

$k = \log n$.

$\log n$ times.

$T(n) = T(1) + \log n (5)$

$= 5 \cdot \log n + 4$

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

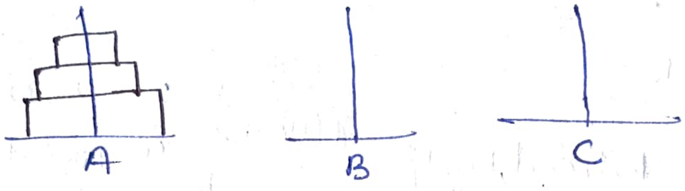
Q:- log n, n, n^2, 1, 2^n, n^n, n^3

A:- $1 \le \log n \le n \le n^2 \le n^3 \le 2^n \le n^n$. [Ascending order].

$1 = O(n)$.

Problem

Towers of Hanoi :



⇒ If the tower having n disks it requires $2^n - 1$ steps to move from A to C.

Algorithm:

```
TOH (n, A, C, B) {
    if (n==1) mov (A, C);
    if (n>1) {
        TOH (n-1, A, B, C);
        TOH (1, A, C, B);
        TOH (n-1, B, C, A);
    }
}
```

Eg: N=3.

```
TOH (3, A, C, B) {
    TOH (2, A, B, C) {
        TOH (1, A, C, B)
        TOH (1, A, B, C)
        TOH (1, C, B, A)
    }
TOH (1, A, C, B)
TOH (2, B, C, A)
TOH (1, B, A, C)
TOH (1, B, C, A)
TOH (1, A, C, B)
```

1. m(A, C).
2. A→B
3. C→B
4. A→C
5. B→A
6. B→C
7. A→C

Time complexity

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

$T(n) = 2\,\underline{T(n-1)}+1$ $\quad T(n-1)$

$= 2\,[\underline{2.T(n-2)+1}]+1$

$= 2^2\,T(n-2)+\underline{2+1} \rightarrow 2^2-1.$

$= 2^2\,[2.T(n-3)+1]\,2+1.$

$T(n) = 2^3\,T(n-3)+\underline{2^2+2+1} \rightarrow 2^3-1.$

After k steps,

$T(n) = 2^k\,T(n-k)+2^k-1$

$n-k=1$

$k=n-1$

$\Rightarrow T(n) = 2^{n-1}.T(1)+2^{n-1}-1$

$= 2.2^{n-1}-1 = 2^n-1$

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

Master's Theorem:- [dividing recurrence Relation].

$T(n) = a.T(n/b)+\theta(n^k \log^p n)$ $\quad a>0,\ b>0,\ k\geq 0.$

i) $a>b^k$: $T(n) = \theta(n^{\log_b a})$

ii) $a=b^k$: a] $p>-1$: $T(n) = \theta(n^{\log_b a}.\log^{p+1} n)$

b] $p=-1$: $T(n) = \theta(n^{\log_b a}.\log\log n).$

c] $p<-1$: $T(n) = \theta(n^{\log_b a}).$

iii) $a<b^k$: a] $p>=0$: $T(n) = \theta(n^k \log^p n)$

b] $p<0$: $T(n) = \theta(n^k)$

Eg:

1) $T(n) = a.T(n/2) + 2$

$a = 1,\ b = 2,\ k = 0,\ p = 0.$

$1 = 2^0 = 1 \Rightarrow a = b^k.$

$T(n) = n^{\log_2^1} . \log^1 n.$

$= n^{\log_2^{2^0}} . \log^1 n.$

$= n^0 . \log n$

$= \Theta(\log n)$

$= O(\log n).$

2) $T(n) = \cancel{2(} 2T(n/2) + n$

3) $T(n) = 2T(n/2) + n\log n.$

$a = 2,\ b = 2,\ k = 1,\ p = 1$

ii) $2 = 2^1$

$T(n) = \Theta(n^{\log_2^2}$

For decreasing recurrence Relation:

$T(n) = aT(n-b) + f(n)$

$f(n) = O(n^k)$; $a > 0$, $b > 0$, $k \geq 0$.

1. $a < 1$: $T(n) = f(n)$
2. $a = 1$: $T(n) = O(n \cdot f(n))$
3. $a > 1$: $T(n) = O(a^{n/b} \cdot f(n))$.

Eg: ① $T(n) = 2T(n-1) + 1$

$a = 2$, $b = 1$, $f(n) = 1 = O(n^0)$.

$T(n) = O(2^{n/1} \cdot 1)$

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

② $T(n) = T(n-1) + n$.

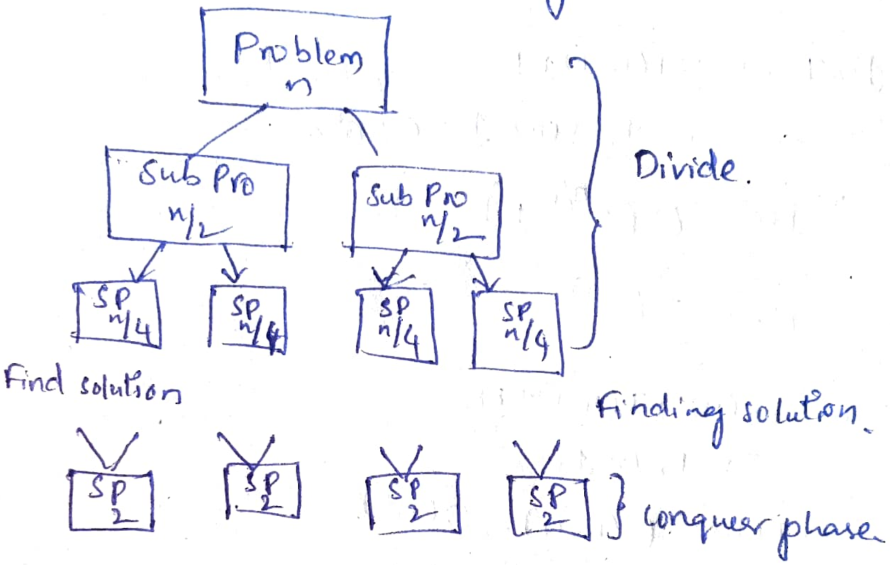
$a = 1$, $b = 1$,

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

Divide and Conquer

Eg: 1) Merge sort 2) Quick sort 3) Binary search
4) Tree traversals 5) Strassen's matrix multiplication

steps.
1. Dividing the problem into subparts.
2. Finding the solution to subparts.
3. Combining all these solutions to get overall solution.



1) Merge sort

Recurrence relation : $T(n) = \begin{cases} 1 & n=1 \\ 2T(n/2)+n & n>1 \end{cases}$

Time complexity : $O(n \log n)$.

2) Quick sort

Recurrence relation for worst case [because of choosing fixed position of pivot].

i.e Pivot = A[start]
Pivot = A[end]

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

Best case: $T(n) = \begin{cases} 2T(n/2)+n & n>1 \\ 1 & n=1 \end{cases}$

Time complexity for worst case : $O(n^2)$

" " " Best case : $O(n \log n)$.

Merge	Quick.
1. $O(n \log n)$	1. $O(n^2)$.
2. Extra space is required	2. No extra space is required.
	3. Spread sheets, programming languages.

Randomization.

→ Optimization technique for avoiding the worst case in Quick sort

⇒ Choose pivot randomly. with uniform distribution.

3) Strassen's Matrix Multiplication.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}_{2\times 2} \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}_{2\times 2}$$

$$A \times B = \begin{bmatrix} A_{11}\times B_{11} + A_{11}\times B_{21} & A_{11}\times B_{12} + A_{21}\times B_{22} \\ A_{21}\times B_{11} + A_{21}\times B_{21} & A_{21}\times B_{12} + A_{22}\times B_{22} \end{bmatrix}_{2\times 2}$$

⇒

Size	no. of * operators
2x2	8
4x4	64
8x8	512

$$T(n) = \begin{cases} 8T(n/2)+1 & n>1 \\ 1 & n=1 \end{cases}$$ → Recurrence Relation

$T(n) = O(n^3)$ → time complexity.

→ Add (same row)
Multiply (same column).

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}.$$

$P = (A_{11}+A_{22})\cdot(B_{21}+B_{22})$

$Q = (A_{21}+A_{22})\cdot B_{11}$

$R = A_{11}\cdot(B_{12}-B_{22})$

$S = A_{22}\cdot(B_{21}-B_{11})$

$T = (A_{11}+A_{12})\cdot B_{22}$

$U = (A_{21}-A_{11})\cdot(B_{11}+B_{12})$

$V = (A_{12}-A_{22})\cdot(B_{21}+B_{22})$

$C_{11} = P+S-T+V$

$C_{12} = R+T$

$C_{21} = Q+S$

$C_{22} = P+R-Q+U$

$$T(n) = \begin{cases} 7\cdot T(n/2)+1 \\ 1 & ;\ n=1 \end{cases}$$

$T(n) = O(n^{2.81}) = O(n^{\log 7})$

for 4x4

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}_{4\times 4} \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}_{4\times 4}$$

SMM

$2\times 2 = 7$

$4\times 4 = 49$

$8\times 8 = 343.$

NMM.

$2\times 2 = 8$

$4\times 4 = 64$

$8\times 8 = 512$

Pseudo Code:

```
Strassens (A,B,n) {
    if(n==1) {
        C <- new matrix nxn.
        C[0][0] = A[0][0]* B[0][0];
    } else {
        C <- new matrix nxn.

    A11, A12, A21, A22 <- A.split_matrix();

    B11, B12, B21, B22 <- B.split_matrix();
```

$P =$ strassens $(A11 + A22,\ B_{11} + B_{22},\ n/2)$;

$Q =$ strassens $(A_{21} + A_{22},\ B_{11},\ n/2)$;

$R =$ strassens $(A_{11},\ B_{12} - B_{22},\ n/2)$;

$S =$ strassens $(A_{22},\ B_{21} - B_{11},\ n/2)$;

$T =$ strassens $(A_{11} + A_{12},\ B_{22},\ n/2)$;

$U =$ strassens $(A_{21} - A_{11},\ B_{11} + B_{12},\ n/2)$;

$V =$ strassens $(A_{12} - A_{22},\ B_{21} + B_{22},\ n/2)$;

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{22} = P + R - Q + U$

$C =$ combine_quardants $(C_{11}, C_{12}, C_{21}, C_{22})$;

}

return C;

}

$1\times1+2\times3 \quad 1\times2+2\times4$

$3\times1+4\times3 \quad 3\times2+4\times4$

$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$

Eg:-

$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}_{2\times2}$, $B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}_{2\times2}$

$P = (1+4).(1+4) = 25$

$Q = (3+4).1 = 7$

$R = 1.(2-4) = -2$

$S = 4.(3-1) = 8$

$T = (1+2).4 = 12$

$U = (3-1).(1+2) = 6$

$V = (2-4).(3+4) = -14$

$C_{11} = 25+8-12-14 = 7$

$C_{12} = -2+12 = 10$

$C_{21} = 7+8 = 15$

$C_{22} = 25+(-2)-7+6 = 22$

$A \times B = \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$.