Design and Analysis of Algorithms

UNIT → 1

INTRODUCTION

* ALGORITHM

- An algorithm is a well defined list of steps for solving a particular problem.
- It is a tool for solving a well specified computational problem that takes one or set of values as input and produces some value or set of values as output.

eg. Sorting Problem:-

Input: A sequence of n nos $n_1, n_2, n_3 \ldots n_n$

Output: Ordering of the inputted sequence $n_1, n_2, n_3 \ldots n_n$ such that $n_1 \leq n_2 \leq n_3 \ldots \leq n_n$

* Applications of algorithm:-

- Internet
- Ecommerce
- finding shortest root in a road map.
- Product of matrices.

* algorithm is independent from any programming languages.

MARCH 2014

M	T	W	T	F	S	S
31						
3					1	2
10	4	5	6	7	8	9
17	11	12	13	14	15	16
24	18	19	20	21	22	23
	25	26	27	28	29	30

* ANALYSIS OF ALGORITHM :-

- It depends upon the time and space taken by the algorithm.

Time complexity :-
The amount of time required by the algorithm is called time complexity.

Space Complexity :-
The amount of space/memory required by the algorithm is called as space complexity.

Growth functions :-

	2^n	$n!$
$n=1$	2	1
$n=2$	4	2
$n=3$	8	6
$n=4$	16	24
$n=5$	32	120

$$n! > 2^n \quad \forall\, n \geq 4$$

$n!$ has higher growth rate than 2^n.

FEBRUARY 2014
M T W T F S S
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28

Growth rate :-

constant $\leq$ logarithm $\leq$ polynomial $\leq$ exponential $\leq$ factorial

$1, n, n \log n, n^2, n^3, n^5, 2^n, n!$

* Types of Analysis:-

① Worst case : max no. of steps taken on any instance of size a

② Best case : min no. of steps taken on any instance of size a

③ Average case : an average no. of steps taken on any instance of size a.

Complexity of algorithms:-

It is the function $f(n)$ which gives the running time and / or storage space requirements of the algorithm in term of the size n of the input data.

* Time and Space Complexity Comparison Table

Sorting algo.	Time complexity			Space complexity
	Best case	Avg case	Worst case	Worst case.
① Bubble Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
② Selection sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
③ Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
④ Merge sort	$\Omega(n\log n)$	$\Theta(n\log n)$	$O(n\log n)$	$O(n)$
⑤ Heap sort	$\Omega(n\log n)$	$\Theta(n\log n)$	$O(n\log n)$	$O(1)$

⑥ Quick Sort	$\Omega(n\log n)$	$\Theta(n\log n)$	$O(n^2)$	$O(n\log n)$
⑦ Radix Sort	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$

* ASYMPTOTIC NOTATIONS

Execution time of an algo depends on the inst^n set, processor speed, disk I/O speed etc. Hence, we estimate the efficiency of an algo asymptotically.

Time function of an algo is represented by $T(n)$ where n is the output size.

Following asymptotic notations are used to calculate the running time complexity of an algorithm.

O - Big Oh
Ω - Big omega
θ — Big theta
o - little oh
ω - little omega.

(1) Big Oh (O)

- It is commonly used notation.
- A function $f(n)$ can be represented is the order of $g(n)$ that is $O(g(n))$, if there exists a value of positive integer n as n_0 and a positive constant c, such that :-

MARCH 2014
M T W T F S S
31 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

$f(n) \leq c \cdot g(n)$ for $n > n_0$ in all case

Hence function $g(n)$ is an upper bound for function $f(n)$, as $g(n)$ grows faster than $f(n)$

- Big O notation defines an upper bound function.

② OMEGA NOTATION (Ω)

It is used when the function $g(n)$ defines a lower bound for the function $f(n)$

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

iff there exists a +ve integer n_0 and a +ve number c, such that

$|f(n)| \geq c|g(n)|$ for all $n \geq n_0$

③ Theta Notation (θ) :-

It is used when the function $f(n)$ is bounded both from above and below by the function $g(n)$.

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

iff there exist two +ve constants c_1 and c_2 and a +ve integer n_0 such that

$$c_1 |g(n)| \le |f(n)| \le c_2 |g(n)|$$
for all $n \ge n_0$

Examples:-

① Big oh notation (upper bound)

① $f(n) = 5n^3 + 7n^2 + 2n + 3$

for $n \ge 3$

$5n^3 + 7n^2 + 2n + 3n$

$= 5n^3 + 7n^2 + 3n$

for $n^2 \ge 3n$

$= 5n^3 + 7n^2 + n^2$

$= 5n^3 + 8n^2$

for $n^3 \ge 8n^2$

$= 5n^3 + n^3 = 6n^3$

$c = 6$

$g(n) = n^3$

② $f(n) = 4n^3 + 10n^2 + 5n + 1$

$g(n) = n^3$

$f(n) \le 5 \cdot g(n)$ for all value of $n > 2$

MARCH 2014

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

$f(n) = 4n^3 + 10n^2 + 5n + 1$

for $n \geq 1$

$= 4n^3 + 10n^2 + 5n + n$

$= 4n^3 + 10n^2 + 6n$

for $n^2 \geq 6n$

$= 4n^3 + 10n^2 + n^2$

$4n^3 + 11n^2$

for $n^3 \geq 11n^2$

$= 4n^3 + n^3$

$= 5n^3$

$c = 5$

(3) Prove that $n^3 - 3n^2 - 8 = O(n^3)$

as per defination

$n^3 - 3n^2 - 8 \leq cn^3$

$\Rightarrow 1 - \frac{3}{n} - \frac{8}{n^3} \leq c$

$n_0 = 1$

so, $c = 1$

(4) Prove that $2^{n+1} = O(2^n)$

$$2^{n+1} \leq c \cdot 2^n$$

$$\Rightarrow 2 \leq c$$

so, $2^{n+1} = O(2^n)$

(5) Prove that $5n+3 = O(n)$

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

$$n \geq 3$$

$$5n + n$$

$$= 6n$$

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

$$c = 6$$

$$g(n) = n$$

(6) Prove that $6n^2 + 2n + 3 = O(n^2)$

$$6n^2 + 2n + 3 \leq 7n^2$$

$$= O(n^2)$$

$$6n^2 + 2n + 3$$

$$n \geq 1 \quad \Rightarrow c = 7$$

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

$$6n^2 + 2n + n$$

$$6n^2 + 3n$$

$$n^2 \geq 3n$$

$$6n^2 + n^2 = 7n^2$$

MARCH 2014

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

(7) $f(n) = n^2 + n \qquad n > n_0 \geq 0$

$n^2 + n \leq c * g(n)$

$n^2 + n \leq c * n^2 \qquad n = 0$

$1 + 1 \leq 2 * 1 \qquad n = 1$

$2 \leq 2$

$2^2 + 2 \leq 2 * 4 \qquad n = 2$

$6 \leq 8$

for $n = 3 \qquad 3^2 + 3 \leq 2 * 9$

$12 \leq 18$

$c = 2, \; n_0 = 1$

for $c = 2, \; n_0 = 1$ such that $n \geq 0$

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

$n^2 + n \leq 2 * n^2$

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

(8) $f(n) = 6n^2 + 5$

$6n^2 + 5 \leq c * n^2, \; n = 1$

$6 + 5 \leq c * 1$

$11 \leq 11 * 1$

$c = 11, \; n = 1$

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

MARCH 2014

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

$f(n) = 7n^2 + 8n + 39$

$7n^2 + 8n + 39 \leq c * g(n)$

$7n^2 + 8n + 39 \leq c * n^2$

$n = 1$

$7 + 8 + 39 \leq c * 1$

$54 \leq 54$

$n = 1, \quad c = 54$

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

$7n^2 + 8n + 39 \leq 54 * n^2$

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

(9) $f(n) = n^3 + 35n^2 + 1$

$n^3 + 35n^2 + 1 \leq c * g(n)$

$n^3 + 35n^2 + 1 \leq c * n^3$

$1 + 35 + 1 \leq 37 * 1$

for $n = 1$

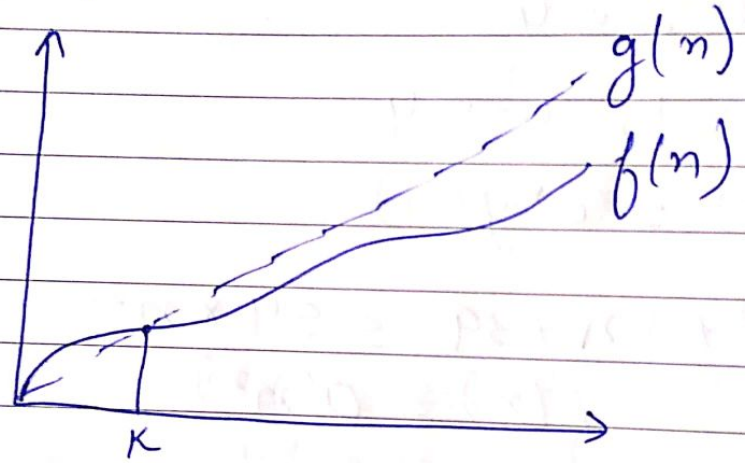
$37 \leq 37$

$c = 37, \quad n = 1$

$f(n) = O(n^3)$

* Big Oh Notation O:

The notation $O(n)$ is the formal way to express the upper bound of an algorithm's running time. It measures the worst case time complexity or the longest amount of time an algorithm can possibly take to complete.



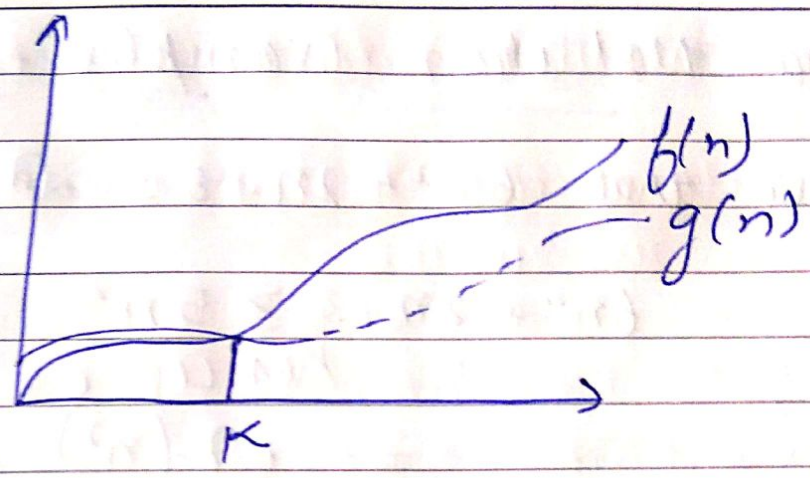
$f(n) \leq c \cdot g(n)$ for all $n \geq n_0$

there exists $c > 0$

* Omega Notation (Ω):-

The notation $\Omega(n)$ is the formal way to express the lower bound of an algorithm's running time. It measures the best case time complexity or the best amount of time an algorithm can possibly take to complete.

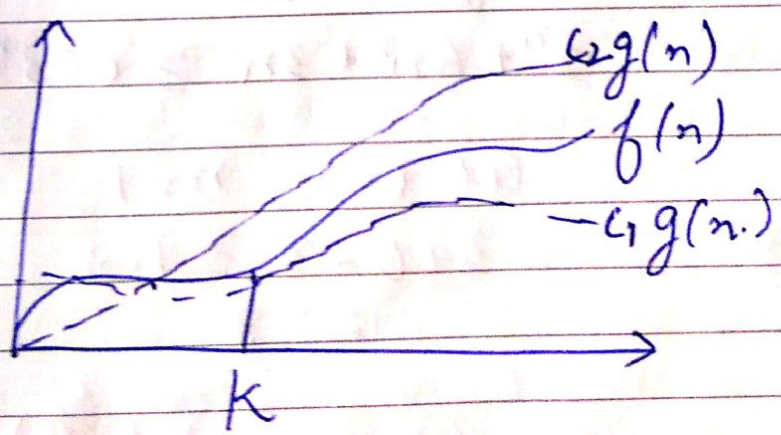
MARCH 2014
M T W T F S S
31 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
22 23



$g(n) \leq c \cdot f(n)$ for all $n \geq n_0$
there exists $c > 0$

Theta Notation (Θ) :-

The notation $\Theta(n)$ is the formal way to express both the lower bound and the upper bound of an algorithm's running time. It is represented as follows:



Omega Notation examples:-

① Prove that $6n^2 + 2n + 3 = \Omega(n^3)$

$$6n^2 + 2n + 3 \geq 5n^2$$

hence $c = 5$

$$= \Omega(n^2)$$

② Prove that $5n+3 = \Omega(n)$

$$5n+3 \geq 4(n)$$

hence $c = 4$

$$= \Omega(n)$$

③ $f(n) = 3^n + 6n^2 + 3n$

let $g(n) = 3^n$

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

$$3^n + 6n^2 + 3n \geq c \cdot 3^n$$

let $c = 1$, $n = 1$

$$3 + 6 + 3 \geq 1 \cdot 3$$

$$12 \geq 3$$

$$f(n) = \Omega(3^n)$$

for $c = 1$, $n_0 = 1$

MARCH 2014
M T W T F S S
31 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

* Characteristics of an Algorithm:-

① Input / Output:- whenever any input is given then it should produce an output.

② Correctness:- Every step must be logically correct.

③ Definiteness:- It should give the same (fix) result at any instance of time

eg = 3+5=8 (all time)

④ Effectiveness:- The same algorithm can be executed on any pen and paper. It should be executable on all machines.

⑤ Generality:- It should be applicable to classes of problems eg - $\frac{1}{2} \times b \times h$

(for all right angled triangle)

⑥ Finiteness:- No. of steps should be finite and result should be get in finite time.

* HOW to calculate time complexity :-

① W.A. Algorithm which stores maximum of n nos in an array.

		Time
	ARRAY MAX (A, n)	
1.	current max ← A[0]	1
2.	for i ← 1 to n-1 do	n
3.	if A[i] > current max	1 × (n-1)
4.	Then current max ← A[i]	1 × (n-1)
5.	Return current max	1

Time complexity ⇒ $1 + n + (n-1) + (n-1) + 1$

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

$= \cancel{2} + n + 2n - \cancel{2}$

$= 3n$

② $x = 3 * y + 2;$

$z = z + 1;$

Calculate running time of a program :-

If x, y, z are scalars, this piece of code takes a constant amount of time which we write as O(1)

O(1) means some constant, it might be 5 or 1 or 10

MARCH 2014
M T W T F S S
31 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

④ omega notation

$$f(n) = 6 * 2^n + n^2$$
$$g(n) = 2^n$$

$$f(n) \geq c \cdot g(n)$$
$$6 * 2^n + n^2 \geq c \cdot 2^n$$
let $c = 1$, $n = 1$
$$6 * 2 + 1 \geq 1 \cdot 2$$
$$\Rightarrow 13 \geq 2$$
$$f(n) = \Omega(2^n)$$
for $c = 1$, $n_0 = 1$

* Theta Notation examples :-

① $f(n) = 3n+2$
let $g(n) = n$

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$$c_1 g(n) \leq f(n)$$
or it can be written as
$$f(n) \geq c_1 g(n)$$

$$3n+2 \geq c_1 g(n)$$
$$3n+2 \geq c_1 \cdot n$$

taking $c_1 = 1$, $n = 1$

$\Rightarrow 3+2 \geq 1 \cdot 1$

$\Rightarrow 5 \geq 1$

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

$3n+2 \leq c_2 \cdot n$

$3 \cdot 1+2 \nleq 4 \cdot 1$ (taking $c_2 = 4, n=1$)

$3 \cdot 2+2 \leq 4 \cdot 2$ (taking $c_2 = 4, n=2$)

$6+2 \leq 8$

So $f(n) = \Theta(n)$

for $c_1 = 1$

$c_2 = 4$

$n_0 = 2$

(2) $f(n) = 3 * 2^n + 4n^2 + 5n + 2$

let $g(n) = 2^n$

$c_1 g(n) \leq f(n) \leq c_2 g(n)$

$c_1 g(n) \leq f(n)$

or $f(n) \geq c_1 g(n)$

$3 * 2^n + 4n^2 + 5n + 2 \geq c_1 2^n$

take $c_1 = 1, n = 1$

$\Rightarrow 3 * 2 + 4 + 5 + 2 \geq 1 \cdot 2$

$\Rightarrow 17 \geq 2$

MARCH 2014

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

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

$3*2^n + 4n^2 + 5n + 2 \le c_2 \cdot 2^n$

let $c = 9$, $n = 1$

$3*2 + 4 + 5 + 2 \le 9*2$

$17 \le 18$

Hence $f(n) = \theta(2^n)$ for $c_1 = 1$, $c_2 = 9$ and $n_0 = 1$

(3) $f(n) = 3n+3$

$g(n) = n^2$ (given)

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

$f(n) \ge c_1 g(n)$

$3n+3 \ge c_1 n^2$ $(c_1 = 1, n_0 = 1)$

$3*1 + 3 \ge 1 \cdot 1$

$6 \ge 1$

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

$3n+3 \le c_2 \cdot n^2$ $(c_2 = 1, n = 1)$

$3*1+3 \nleq 1 \cdot 1$

$3*2+3 \nleq 1 \cdot 2$ $(c_2 = 1, n = 2)$

So $f(n) \ne \theta(n^2)$

APRIL 2014

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

* Big O notation examples:-

① $f(n) = 6n^2 + 3n + 2^n$

let $g(n) = 2^n$

$$f(n) \le g(n)$$

$$6n^2 + 3n + 2^n \le c \cdot 2^n$$

take $c = 6, n = 1$

$$6 + 3 + 2 \le 12$$

$$f(n) = O(2^n) \text{ for } c = 6, n_0 = 1$$

* Little Oh notation (o):-

o notation is used to denote an upper bound.

$O(g(n)) = \{ f(n) :$ for any +ve const. $c > 0$, if a const $n_0 > 0$ such that $0 \le f(n) < c\,g(n)$ $\forall\, n \ge n_0$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

eg ① $f(n) = n^4 + 35n^2 + 84$

$g(n) = n^4$

MARCH 2014

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23

$$\Rightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^4+35n^2+84}{n^4}$$

$$\Rightarrow \lim_{n\to\infty} \frac{n^4}{n^4} + \lim_{n\to\infty} \frac{35n^4}{n^4} + \lim_{n\to\infty} \frac{84}{n^4}$$

$$\rightarrow 1+0+0 = 1$$

$$\therefore f(n) \neq n^4$$

(2) $f(n) = 2n$

$g(n) = O(n^2)$ given

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2n}{n^2} = \lim_{n\to\infty} \frac{2}{n} = 0$$

for $n = \infty$

$$\Rightarrow f(n) = O(n^2)$$

(3) $f(n) = 2n^2$

$g(n) = n^2$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2\cancel{n^2}}{\cancel{n^2}} = 2$$

$$\therefore f(n) \neq O(n^2)$$

* <u>little omega Notation</u>:- It is denoted as ω.

The function $f(n) = \omega(g(n))$

iff $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$

or $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

(1) $f(n) = \frac{n^2}{2}$

$g(n) = \omega(n)$

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{n}{n^2/2} = \lim_{n\to\infty} \frac{2}{n} = \frac{2}{\infty}$

$= 0$

$\therefore f(n) = \omega(n)$

(2) $f(n) = n^2/2$

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

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{n^2}{n^2/2} = \lim_{n\to\infty} \frac{2\cancel{n^2}}{\cancel{n^2}} = 2$

$\therefore f(n) \neq \omega(n^2)$

MARCH 2014

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Analysis of simple for loop:-

```
for (i=1; i<=n; i++)
    v[i] = v[i] + 1;
```

This loop will run exactly n times, and because the inside of the loop takes constant time, the total running time is proportional to n.

$O(n)$.

Analysis for nested for loop :-

```
for(i=1; i<=n; i++)
  for (j=1 ; j<=n ; j++)
      a[i,j] = b[i,j] * x;
```

outer loop executes n times, while the inner loop executes n times for every execution of the outer loop.

$n \times n = n^2$ times.

$O(n^2)$

* Analysis of Bubble sort :-

```
for (i = n-1 ; i < 1 ; i--)
    for (j = 1 ; j <= i ; j++)
        if (a[j] > a[j+1])
            swap a[j] and a[j+1];
```

- The innermost statement takes $O(1)$ time
- It does not necessarily take the same time. when the condition is true as it does when it is false, but both times are bounded by a constant

- The outer loop executes n times, but inner loop executes a no. of times that depends on i.

first time inner loop executes $n-1$
second time $n-2$

and so on.

Thus, total no. of times, the inner statement executes is

$$(n-1) + (n-2) + \cdots + 3 + 2 + 1$$

This is the sum of an arithmetic series

$$\sum_{i=1}^{n-1} (n-i) = \frac{n(n-i)}{2} = \frac{n^2}{2} - \frac{n}{2}$$

$(i=1)$

MARCH 2014
M T W T F S S
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

Running time is $O\left(\frac{n(n-1)}{2}\right)$

$= O\left(\frac{n^2-n}{2}\right)$

ignore smaller term

Big Oh represents upper bound

Thus $O(n^2)$ by ignoring constant factor.

* Analysis of Binary Search :-

- It is little harder to analyse as it does not have a for loop.
- Search interval halves each time we iterate the search.

The sequence of search intervals looks something like this :-

$n, n/2, n/4, \ldots 8, 4, 2, 1$

It is not obvious how long this sequence is, but if we take logs, it is :-

$\log_2 n, \log_2 n-1, \log_2 n-2, \ldots 3, 2, 1, 0$

APRIL 2014

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

Second sequence decrements by 1 each time down to 0,
it length must be $\log_2 n + 1$.

It takes only constant time to do each test of binary search, so the total running time is just the no. of times that we iterate, which is $\log_2 n + 1$.

Since, Base of the log doesn't matter in an asymptotic bound
we can write that binary search is $O(\log n)$.

* POLYNOMIALS:-

Given a nonnegative integer d, a polynomial in n of degree d is a function $p(n)$ of the form

$$p(n) = \sum_{i=0}^{d} a_i n^i$$

where the constants $a_0, a_1, \ldots a_d$ are the co-efficients of the polynomial and $a_d \neq 0$.

A polynomial is asymptotically +ve if and only if $a_d > 0$. For an asymptotically +ve polynomial $p(n)$ of degree d,

MARCH 2014
M T W T F S S
31 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23

we have $p(n) = \Theta(n^d)$

For any real constant $a \geq 0$, the function n^a is monotonically increasing, and for any real constant $a \leq 0$, the function n^a is monotonically decreasing.

Thus a function $f(n)$ is polynomially bounded if $f(n) = O(n^k)$ for some constant k.

* RECURRENCES:-

A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.

When we state and solve recurrences, we often omits floors, ceiling and boundary condition.

Floors and ceilings:-

For any real number x, we denote the greatest integer less than or equal to x by $\lfloor x \rfloor$ → floor of x
and the least integer greater than or equal to x by $\lceil x \rceil$ → ceiling of x
for all real x.

$$x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$$

APRIL 2014

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

* Solutions of recurrence relations:-

There are three methods for solving a recurrence relation:

1. Substitution method
2. Recursion tree Method
3. Master Method.

(1) Substitution Method:-

- It involves guessing the form of the solution and thus using mathematical induction to find the constants and show the solution works.

- This method is powerful, but it can be applied only in cases when it is easy to guess the form of the answer.

eg (i) Consider the recurrence:-

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T\left(\lfloor n/2 \rfloor\right) + n & n > 1 \end{cases}$$

Find an asymptotic bound on T.

Ans:- We guess the solution is $O(n \log n)$

Thus for a constant c

MARCH 2014

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

$T(n) \leq c \cdot n \log n$

Substituting this in the given recurrence

$T(n) \leq 2c\left(\frac{n}{2}\right) \log\left(\frac{n}{2}\right) + n$

$\leq cn \log\left(\frac{n}{2}\right) + n$

$\leq cn \log n - cn \log 2 + n$

$T(n) \leq cn \log n - n(c \log 2 - 1)$

$\leq cn \log n - n(c1 - 1) \Rightarrow cn \log n - nc + n$

$\leq c \cdot n \log n$

$\lfloor 3.14 \rfloor = 3$

$\lceil 3.14 \rceil = 4$

By mathemical induction

$T(1) \leq c \cdot 1 \log 1$ where $n = 1$

$\log 1 = 0$

$1 \nleq 0$

$T(2) \leq c \cdot 2 \log 2$ (where $n = 2$)

$\leq c \cdot 2$ ($\log 2 = 1$)

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

$= 2T(1) + 2$

$2 \cdot 1 + 2 = 4$

Hence $4 \leq 2c$ if $c \geq 2$

$T(3) \leq c \cdot 3 \log_2 3$

$\leq 3c$

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

$= 2T(3/2) + 3$

$2T(1) + 3 = 5$

Hence $5 \leq 3c$ if $c \geq 2$

$T(n) = O(n \log n)$ for $c = 2$, $n_0 = 2$

$\therefore$ $T(n) = O(n \log n)$ is true.

(2) $T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 1$

$= \left(T\left(\frac{n}{4}\right) + 1\right) + 1$ $\quad \left[T(n/2) = T(n/4) + 1\right]$

$= T\left(\frac{n}{4}\right) + 2$

$\Rightarrow \left(T\left(\frac{n}{8}\right) + 1\right) + 2$ $\quad \left[T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) + 1\right]$

$\Rightarrow T\left(n/8\right) + 3$

$\vdots$

$\Rightarrow T\left(\frac{n}{2^i}\right) + i$

MARCH 2014

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

The series terminates when $\frac{n}{2^i} = 1$

$$\Rightarrow n = 2^i$$

$$\Rightarrow i = \log_2 n$$

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

$$= T\left(\frac{n}{n}\right) + \log_2 n$$

$$= T(1) + \log_2 n$$

$$= \log_2 n$$

$$T(n) = O(\log_2 n)$$

(3) $T(n) = T\left(\frac{n}{2}\right) + a$ if $n > 1$

$= b$ if $n = 1$

Ans-1 $T(n) = \left(T\left\lfloor\frac{n}{4}\right\rfloor) + a\right) + a = T\left(\frac{n}{4}\right) + 2a$

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

$= \left(T\left(\frac{n}{8}\right) + a\right) + 2a = T\left(\frac{n}{8}\right) + 3a$

$\left(n = \frac{n}{4}\right.$

$\vdots$

$$\Rightarrow \left(T\left(\frac{n}{2^i}\right) + ia\right)$$

APRIL 2014

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

This series terminates when $\frac{n}{2^i} = 1$

$\Rightarrow n = 2^i$ or $i = \log_2 n$

$\Rightarrow T\left(\frac{2^i}{2^i}\right) + a \log_2 n$

$\Rightarrow T(1) + a \log_2 n$

$\Rightarrow b + a \log_2 n$

$T(n) = O(\log_2 n)$

(4) $T(n) = 2T\left(\frac{n}{2}\right) + n$

Ans $2\left(2T\left(\frac{n}{4}\right) + n/2\right) + n$ $\left(\because T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{4}\right) + \frac{n}{2}\right)$

$= 4T\left(\frac{n}{4}\right) + \cancel{2} \cdot \frac{n}{\cancel{2}} + n$

$= 4T\left(\frac{n}{4}\right) + 2n$

$= 4T\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$

$\left(\because T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + \frac{n}{4}\right)$

$\Rightarrow 8T\left(\frac{n}{8}\right) + \cancel{4}\frac{n}{\cancel{4}} + 2n$

MARCH 2014

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

$$= 8T\left(\frac{n}{8}\right) + 3n$$

$$\Rightarrow 2^3 T\left(\frac{n}{2^3}\right) + 3n$$

$$\Rightarrow 2^i T\left(\frac{n}{2^i}\right) + in$$

The series will complete when $\frac{n}{2^i} = 1$

$$n = 2^i \quad \text{or} \quad i = \log_2 n$$

$$T(n) \leq \cancel{n}\, T\left(\frac{n}{\cancel{n}}\right) + n\log_2 n$$

$$= nT + n\log_2 n \qquad (\because T(1) = 1)$$

$T(n) = O(n\log_2 n)$ is true.

(5) $T(n) = 2T\left(\left\lfloor\frac{n}{2}\right\rfloor + 16\right) + n$

Find an asymptotic bound on T

<u>Sol^n</u>:- $T(n) = 2T\left(\frac{n}{2} + 16\right) + n$

$$= 2\left[2T\left(\frac{n}{4} + 16\right) + \frac{n}{2}\right] + n$$

$$= 4T\left(\frac{n}{4}\right) + 32 + \cancel{2}\cdot\frac{n}{\cancel{2}} + n$$

$$= 4T\left(\frac{n}{4}\right) + 32 + 2n$$

APRIL 2014

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

$$= 4T\left[2T\left(\frac{n}{8}\right)+16\right) + \frac{n}{4}\right] + 32+2n$$

$$= 8T\left(\frac{n}{8}\right) + 64 + \cancel{4}\cdot\frac{n}{\cancel{4}} + 32 + 2n$$

$$= 8T\left(\frac{n}{8}\right) + 3n + 96$$

$$\Rightarrow 2^3T\left(\frac{n}{2^3}\right) + 3n + 96$$

$$\Rightarrow 2^iT\left(\frac{n}{2^i}\right) + in + x$$

The series terminates when $\frac{n}{2^i} = 1$

$$n = 2^i, \quad i = \log_2 n$$

$$T(n) = \cancel{2^i}T\left(\frac{n}{\cancel{2^i}}\right) + in$$

$$= nT\left(\frac{n}{n}\right) + n\log_2 n$$

$$= nT(1) + n\log_2 n$$

$$T(n) = n\log_2 n$$

(6) Consider the recurrence: -

$$T(n) = T(n-1) + 1 \quad \text{and} \quad T(1) = \Theta(1)$$

solve it.

Sol^n: $T(n) = T(n-1) + 1$

$\Rightarrow (T(n-2)+1) + 1$ $\quad (\because T(n-1) = T(n-2)+1)$

$\Rightarrow (T(n-3)+1) + 2$

$\Rightarrow T(n-4)+1)+3$

$\vdots$

$= T(n-i)+i$

when $i = n-1$ $\because$ series will complete when $n-i=1$

i.e. $T(n-i) = T(1) = \Theta(1)$

i.e. $T(n) = \Theta(1) + (n-1) = 1 + n - 1$
$= \Theta(n)$.

② Recursion Tree Method :-

In this method, we convert the recurrence into a tree whose nodes represent the costs incurred at each level in the tree and sum of the costs at each level helps to determine the total cost of all levels of the recursion.

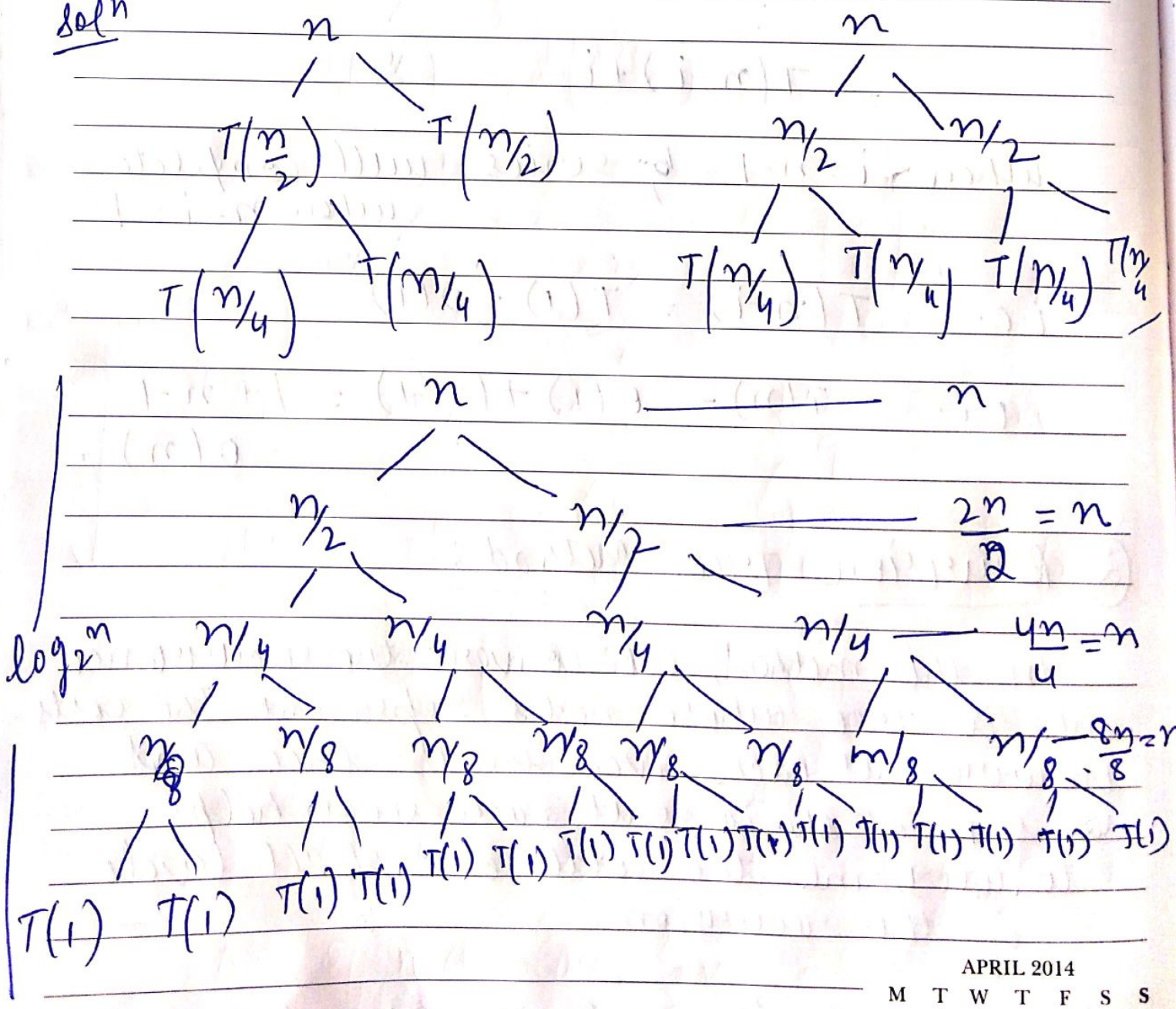
It is useful when the algo is based on divide and conquer algorithm.

(1) Consider the recurrence.

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

Obtain the asymptotic bound using recursion tree method.

Sol^n



APRIL 2014

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

$$\frac{n}{2^i} = 1$$

$$\Rightarrow n = 2^i$$

$$\Rightarrow i = \log_2 n$$

$\therefore$ No. of levels $= i+1 = \log_2 n + 1$

cost for individual node $= \frac{n}{2^i}$

No. of nodes in each level $= 2^i$

$= i = 0, 1, 2, \ldots$

$\therefore$ Total cost of each level $= \frac{n}{2^i} \times 2^i = n$

No. of nodes at depth $\log_2 n$ has 2^i

$= 2^{\log_2 n}$

$= n \log_2 2 =$

$n = \Theta(n)$

Total cost of tree = total cost of tree upto last level + cost of last levels

$$T(n) = n + n + n \ldots \log_2 n - 1 + \Theta(n)$$

$$= n(\log_2 n - 1) + \Theta(n)$$

$$= n \log_2 n - n + \Theta(n)$$

$$= \Theta(n \log_2 n)$$

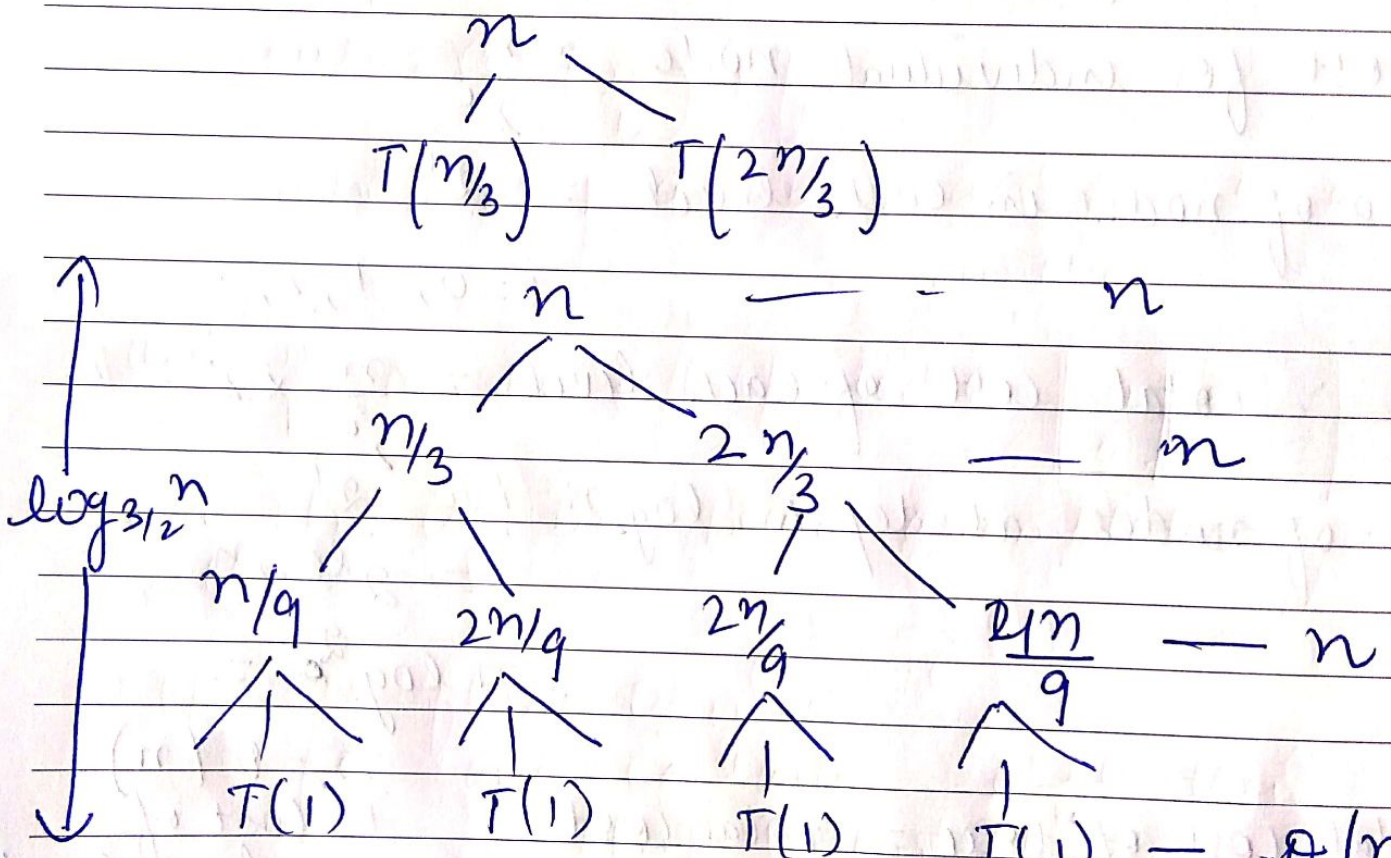
MAY 2014

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

(2) Consider the following recurrence :-

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$$

Obtain the asymptotic bound using recursion tree method.



$\Theta(n \log n)$

Since $\left(\frac{2}{3}\right)^i n = 1$

so $i = \log_{3/2} n$

No. of levels $= i + 1 = \log_{3/2} n + 1$

No. of nodes each level $= 2^i$

APRIL 2014

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

Cost of each node $= \left(\frac{2}{3}\right)^i n = \frac{n}{3^i}$

Total cost of each level $= \frac{n}{3^i} \times 2^i = n\left(\frac{2}{3}\right)^i$

The longest path from the root to a leaf is:-

$$n \rightarrow \frac{2}{3}n \rightarrow \left(\frac{2}{3}\right)^2 n - - 1$$

So $\left(\frac{2}{3}\right)^i n = 1$

$$\Rightarrow i = \log_{3/2} n$$

No. of nodes at height $\log_{3/2} n$ has 2^i

$$= 2^{\log_{3/2} n}$$

$= n^{\log_{3/2} 2}$ nodes or ~~leaves~~ leaves

Each contributing cost $T(1)$

Total cost of all leaves $= n^{\log_{3/2} 2} \cdot T(1)$

$$= \Theta\left(n^{\log_{3/2} 2}\right)$$

(3) MASTER METHOD:-

The master method is used for solving the following type of recurrences

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

MAY 2014

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

with $a \geq 1$ and $b > 1$

Master Theorem:-

Let $T(n)$ be defined on the non-negative integers by the recurrence.

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

where $a \geq 1$ and $b > 1$ and $f(n)$ be a function.

and $\frac{n}{b}$ can be interpreted as $\left\lceil \frac{n}{b} \right\rceil$ or $\left\lfloor \frac{n}{b} \right\rfloor$

Then $T(n)$ can be bound asymptotically as follows

① if $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$ then $T(n) = \Theta(n^{\log_b a})$

② If $f(n) = \Theta(n^{\log_b a})$

then $T(n) = \Theta(n^{\log_b a} \log n)$

③ If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$

and if $\boxed{\text{if } \left(\frac{n}{b}\right) \leq c f(n)}$

for some $c < 1$

APRIL 2014

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

$T(n) = \theta(f(n))$, $\boxed{\text{if } a\left(\frac{n}{b}\right) \le c f(n)}$

is called the negative condition.

Example:-

① $T(n) = 9(T)\left(\frac{n}{3}\right) + n$

Compare the above recurrence with the

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

we get $a = 9$, $b = 3$ and $f(n) = n$

Now $n^{\log_b a} = n^{\log_3 9} = n^2 = \theta(n^2)$

$n^{\log_b a} > f(n)$

$n^2 > n$

So $\epsilon = 1$

$f(n) = O\left(n^{\log_b a - \epsilon}\right)$

$O\left(n^{2-1}\right) = O(n)$

Hence we can apply case 1 and the solution is $\theta\left(n^{\log_b a}\right)$

$= \theta(n^2)$

(2) $T(n) = 7T\left(\frac{n}{2}\right) + n^2$

$a = 7, \; b = 2, \; f(n) = n^2$

Now $n^{\log_b a}$ $n^{\log_2 7}$ $n^{2.8}$

$n^{\log_b a} > f(n)$

i.e $n^{2.8} > n^2$

Hence $\epsilon = 2.8 - 2 = 0.8$

$f(n) = O\left(n^{\log_b a - \epsilon}\right)$

$= O\left(n^{2.8 - 0.8}\right) = O(n^2)$

$T(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^{\log_2 7}\right)$

<u>Note</u>: In this example $f(n) < n^{\log_b a}$ and for that case sol^n is case 1

$\log_2 7 = \frac{\log 7}{\log 2}$

(3) $T(n) = T\left(\frac{2n}{3}\right) + 1$

Here $a = 1, \; b = 3/2$

$f(n) = 1$

$n^{\log_b a} = n^{\log_{3/2} 1} = n^0$

APRIL 2014

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

$T(n) = \Theta(n^{\log_b a} \log n)$

$= \Theta(1 \log n) = \Theta(\log n)$

<u>Note</u>:- In this example $f(n) = n^{\log_b a}$ are of same size and solⁿ is case 2

(4) $T(n) = 16T\left(\frac{n}{4}\right) + n^2$

$a = 16, \quad b = 4, \quad f(n) = n^2$

$n^{\log_b a} = n^{\log_4 16} = n^2$

$n^{\log_b a} = f(n) = n^2$

$T(n) = \Theta(n^{\log_b a} \log n)$

$\Theta(n^2 \log n)$

(5) $T(n) = 3T\left(\frac{n}{4}\right) + n \log n$

$a = 3, \quad b = 4, \quad f(n) = n \log n$

$n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$

$f(n) > n^{\log_b a}$

$f(n) \cancel{n^{\log_b a}} > n^{0.793}$

$\epsilon = 1 - 0.793 = 0.2$

MAY 2014

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

$f(n) = \Omega(n^{\log_4 3 + \epsilon})$

Now for regularity condition

$a f\left(\frac{n}{b}\right) \leq c f(n)$

i.e. $a * \left(\frac{n \log n}{4}\right) \leq c \cdot n \log n$

$3 * \left(\frac{n}{4}\right) \log\left(\frac{n}{4}\right) \leq \frac{3}{4} n \log n$

$\therefore c = 3/4$

Therefore the solution is

$T(n) = \Theta(n \log n)$

(6) $T(n) = 16T\left(\frac{n}{4}\right) + n^3$

By master theorem.

$a = 16, \quad b = 4 \quad f(n) = n^3$

$n^{\log_b a} = n^{\log_4 16} \quad n^2$

$f(n) \gtrsim n^{\log_b a}$

$n^3 > n^2$

$\epsilon = 3 - 2 = 1$

APRIL 2014

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29					

$f(n) = \Omega(n^{\log_4 16 + \epsilon})$

$= n^{2+1}$

Now for regularity condition

$af\left(\frac{n}{3}\right) \leq cf(n)$

$= 16 f\left(\frac{n^3}{4^3}\right) \leq cn^3$

$16 \cdot \frac{n^3}{64} \leq cn^3$

for $c = \frac{1}{4}$

By case 3 the solution

is $T(n) = \Theta(f(n))$

$= \Theta(n^3)$.