## Algorithm

- finite set of computational instructions
- set of steps to solve problem
- stepwise process to solve any problem

## Properties of Algorithm

1. Input / Output
   - should have 0 or more well defined inputs
   - must produce output

2. Unambigious / Definiteness
   - each step must be unambigious
   - Should be clear and must lead to only one meaning

3 Finiteness
   - must terminate after finite time

4. Correctness
   - Correct set of output must be provided from each set of inputs.

5. Feasibility
   - should work with available resources

6. Independent
   - Should be independent of any programming and code

Random Access Machine Model
- Base machine model for study of design and analysis of algorithms
- machine independent environment
- single processor (assumed)
- no concurrent operations



Assumptions made are
- Each basic operation (+, -) take 1 step
- Loops and subroutines are not basic operation
- No shortage of memory

## Time and Space Complexity

- There can be more than one solution of problem. We need to compare their performance and should choose best algorithm for solving any problem.
- Time complexity and space complexity is taken in consideration for this.
- Time and space complexity depends on various factors like
    - hardware
    - OS
    - processor etc



- Here, we consider only execution time of algorithm. and its operation involved

## Time Complexity

- represents amount of time that is required for algorithm to execute it
- quantifies the amount of time required for algorithm to run the function of some input.

Example,

For addition of two integers with n bit, N steps are taken.

Then,

total computational time $t(N) = c*n$

where,

$c =$ time consumed for addition of two bits

Here, $t(N)$ grows linearly with input size.

Space Complexity
represents memory needed for algorithm in its life

quantities the amount of space / memory taken by an algorithm to run function with some input.

## Factorial Algorithm

Factorial of non negative integer (n!) is product of all positive integers less than or equal to n.
It's product of all consecutive integers upto n.



Mathematically,

$n! = 1*2*3*4*\ldots*(n-1)*n$

For example,

$5! = 1\times2\times3\times4\times5 = 120$

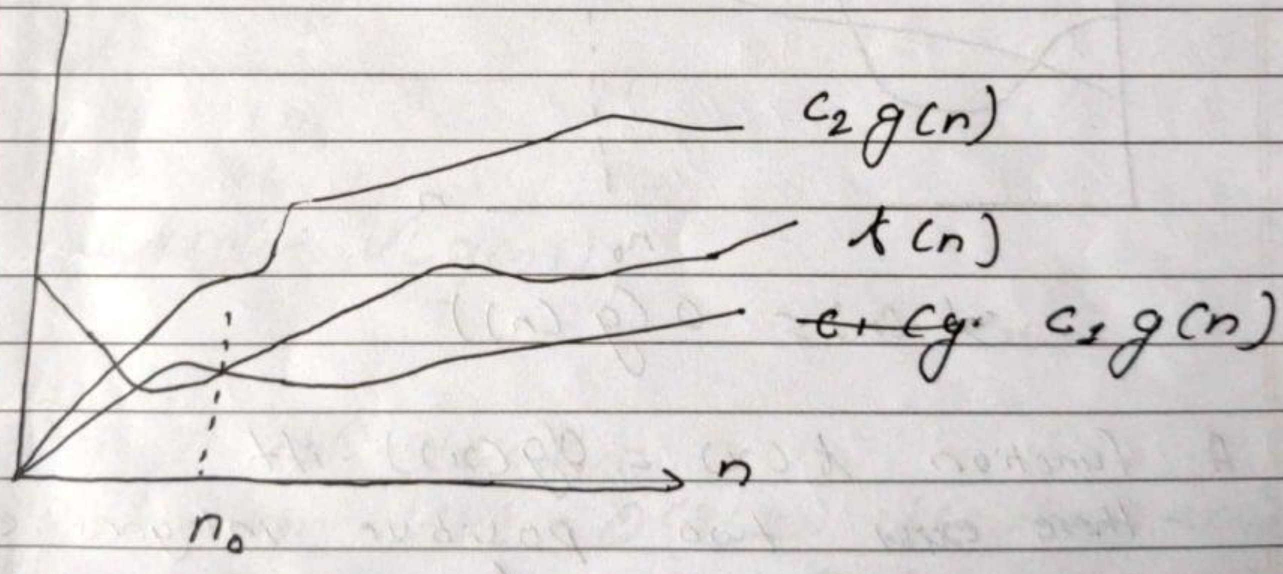
Pseudo Code.

```
factorial = 1
i = 1
while i <= number
    factorial = factorial * i
    i = i+1
end while
Display factorial
```

## Asymptotic Notation

- Asymptotic notations are mathematical notations which are used to describe running time of the algorithms.
- Complexity analysis of algorithm is hard if we try to analyse the exact. So, we take its nearby solution

- Complexity of an algorithm is mathematical function of size of input
- So, we analyse algorithm in term of upper and lower bound.
- We concentrate on worst case only



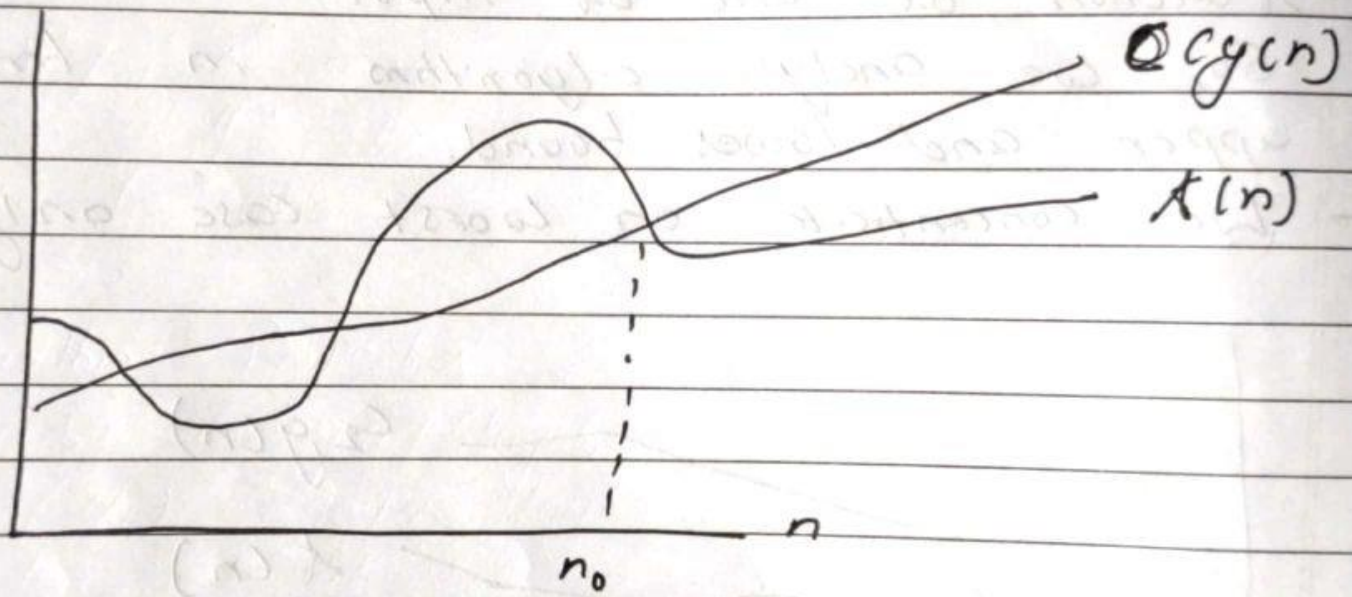

$f(n) = \Theta(g(n))$

Big Oh (O) Notation

defines upper bound of the algorithmic, it bounds a function only from above.
Big Oh notation is useful when we have only upper bound on time complexity of algorithm



Worst case complexity of an algorithm is found with big Oh (O)




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

A function $f(x) = O(g(x))$ iff
- there exist two positive integers c and $n_0$ such that for all $n >= n_0$, $0 <= c*g(n) \leq f(n)$

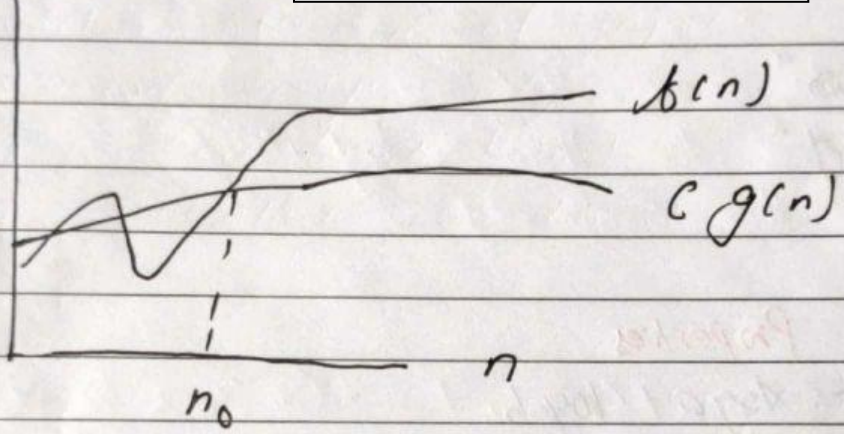
## Big Omega ($\Omega$) Notation

- A function $f(x) = \Omega(g(x))$ iff there exists two positive integers $c$ and $n_0$ such that for all

  $n > n_0$, $0 <= c * g(n) <= f(n)$

- represents lower bound of the algorithm
- gives best case of the algorithm.





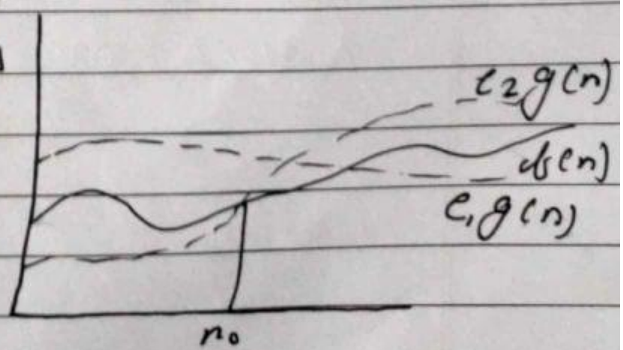

$f(n) = \Omega(g(n))$

## Big Theta ($\Theta$) Notation

- A function $f(x) = \Theta(g(x))$ iff there exists three positive constants $c_1$, $c_2$ and $n_0$ such that for all

  $n > n_0$, $0 \leq c_1 * g(n) \leq f(n) \leq c_2 * g(n)$

- it bounds function from above and below, so it gives exact asymptotic behavior



classmate

Date
Page



## Exponent Properties

① $x^a \cdot x^b = x^{a+b}$

② $\frac{x^a}{x^b} = x^{a-b}$

③ $(x^a)^b = x^{ab}$

④ $(x \cdot y)^a = x^a \cdot y^a$

⑤ $\left(\frac{x}{y}\right)^a = \frac{x^a}{y^a}$

## Logarithm Properties

① $\log(ab) = \log a + \log b$

② $\log(a/b) = \log a - \log b$

③ $\log(a^b) = b \log a$

④ $\log 1 = 0$ ; $\log 2 = 1$ ; $\log 1024 = 10$

Insertion Sort - Analysis [Example]

```
for (i=1; i<n; i++)
{
    x = A[i];
    j = i-1;
    while( j >= 0 && A[j] > x)
    {
        A[j+1] = A[j];
        j = j-1;
    }
    A[j-1] = x;
}
```

Analysis:

## Recursive Algorithm

- It can be solved in terms of itself
- Recurrance relation defines sequenced based on rule those next terms as function of previous terms
- Next term is function of previous term
- Defined by itself (in term of previous terms)
- can model the complexity of divide and conquer algorithm



## Solving Recurrence Relation

- To define the theorem by itself
- To find the complexity

## Theorem

Let $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$ be the recursive relation with all $c_i$ constants.

If the characteristic equation

$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \dots c_k = 0$ has $t$ distinct roots $r_1, r_2, r_3, \dots r_t$ with multiplicity $m_1, m_2, \dots m_t$

then it has a solution

$$a_n = (\alpha_{1,0} + \alpha_{1,1} n + \dots + \alpha_{1,m_1-1} n^{m_1-1}) r_1^n$$
$$+ (\alpha_{2,0} + \alpha_{2,1} n + \dots + \alpha_{2,m_2-1} n^{m_2-1}) r_2^n$$
$$+ \dots$$
$$+ (\alpha_{t,0} + \alpha_{t,1} n + \dots + \alpha_{t,m_t-1} n^{m_t-1}) r_t^n$$

for $n = 0, 1, 2, \dots$ where $\alpha_{i,j}$ are constants for $1 \le i \le t$ and $0 \le j \le m_i - 1$

# Recurrence Relation with Characteristic Equation

Solve the recurrence relⁿ.

$a_n = 2a_{n-1} - a_{n-2}$ with $a_0 = 3$, $a_1 = 6$



Solⁿ.

Characteristic equation is

$r^2 - 2r + 1 = 0$ $\quad // r^2 - \underset{2}{c_1} r - \underset{1}{c_2} = 0$

or, $(r-1)(r-1) = 0$

$\Rightarrow r_1 = 1, \; r_2 = 1$

i.e. $r_1 = 1 \quad m_1 = 2$ [multiplicity 2]

Hence, the solution is,

$a_n = (\alpha_{1,0} + \alpha_{1,1} n)$

which can be written as,

$a_n = x + yn$

Now,

when $a_0 = 3 \quad (n=0)$,

$a_0 = x + y \times 0$

i.e. $3 = x$

$a_1 = 6 \quad [n=1]$

$a_1 = x + y \times 1$

$6 = x + y$

$\therefore y = 3$

$\therefore x = 3, \; y = 3$

Solve $a_n = 5a_{n-1} - 7a_{n-2} + 3a_{n-3}$ with

$a_0 = 1$

$a_1 = 9$



$a_2 = 15$

→ Solution,

Characteristic equation is,

$$r^3 - 5r^2 + 7r - 3 = 0$$

Solving the characteristic equation,

$r = 1$ , $r = 1$, $r = 3$

i.e. $r_1 = 1$ ; $m_1 = 2$

$r_2 = 3$ ; $m_2 = 1$

Now,

Solve the recurrence

$a_n = a_{n-1} + 2a_{n-2}$

with $a_0 = 2$, $a_1 = 7$

→ Solⁿ.

Characteristic eqⁿ.



$r^2 - r - 2 = 0$

Solving,

$r_1 = 2$, $r_2 = -1$

Hence, the solution is,

$a_n = (\alpha_{1,0} + \alpha_{1,1} n)$

Solve the recurrence relation
$a_n = a_{n-1} + a_{n-2}$ with
$a_0 = 0$ $a_1 = 1$

→ Solution,

Characteristic eq$^n$ is
$r^2 - r - 1 = 0$

Solving for $r$,
with comparing $an^2 + bn + c = 0$

$$r = \frac{1 \pm \sqrt{1-4(-1)}}{2}$$



$$= \frac{1 \pm \sqrt{5}}{2}$$

$$\therefore r_1 = \frac{1+\sqrt{5}}{2}, \quad r_2 = \frac{1-\sqrt{5}}{2}$$

Now,
$$a_n = (x)\left(\frac{1+\sqrt{5}}{2}\right)^n + (y)\left(\frac{1-\sqrt{5}}{2}\right)$$

Put $n = 0$
$a_0 = x + y$
or, $0 = x + y$
or, $x = -y$

When $n = 1$
$$a_1 = x\left(\frac{1+\sqrt{5}}{2}\right) + y\left(\frac{1-\sqrt{5}}{2}\right)$$

$$a_1 = -y\left(\frac{1+\sqrt{5}}{2}\right) + y\left(\frac{1-\sqrt{5}}{2}\right)$$

$$= \frac{\cancel{-y} - \sqrt{5}y + \cancel{y} - \sqrt{5}y}{2}$$

$$= -y\sqrt{5}$$

classmate

Date

Page

i.e. $1 = -y\sqrt{5}$

or $y\sqrt{5} = -1$

$y = -1/\sqrt{5}$ and $x = 1/\sqrt{5}$



Sol$^n$ is.

$$\left(-1/\sqrt{5}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(1/\sqrt{5}\right)\left(\frac{1-\sqrt{5}}{2}\right)^n$$

## Solving Recurrence Relation to find Complexity

1. Iteration Method
2. Substitution Method
3. Recursion Tree Method
4. Master Method



### Iteration Method

- Expand the relation so that summation dependent on n is obtained
- bound the summation.

Example.

Example $T(n) = 2T(n/2) + 1$ ---> recursive condition

$T(1) = 1$ ---> base condition

→ Solution.

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

$= 2\left[\underbrace{2T(n/2^2) + 1}_{T(n/2)}\right] + 1$

$= 2^2 T\left[\frac{n}{2^2}\right] + 2 + 1$

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

$= 2^3 T\left[\frac{n}{2^3}\right] + 4 + 2 + 1$

$= 2^3 T[n/2^3] + 2^2 + 2^1 + 2^0$

$= 2^k T[n/2^k] + 2^{k-1} + 2^{k-2} + \dots 2^1 + 2^0$

$= 2^k T[n/2^k] + \sum_{i=0}^{k-1} 2^i$

We already know,

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$



So

$$T(n) = 2^k T[n/2^k] + \sum_{i=0}^{k-1} 2^i$$

$$= 2^k T[n/2^k] + 2^{k-1+1} - 1$$

$$= 2^k T[n/2^k] + 2^k - 1$$

Now,

assume $\cancel{n} = 2^k$

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

$$= 2^k T(1) + 2^k - 1$$

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

$$= n + n - 1$$

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

Example

Solve

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

$T(0) = 1$

Sol$^n$.

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

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

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

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

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

$$= 2^x T(n-x) + 2^{x-1} k + 2^{x-2} k + \dots + 2^0 k$$

$$= 2^n T(n-x) + \sum_{i=0}^{x-1} 2_i$$



where,

$$\sum_{i=0}^{n} 2_i = 2^{n+1} - 1$$

So,

$$T(n) = 2^x T(n-x) + 2^{x-1+1} - 1$$

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

Assume $n = x$,

$$T(n) = 2^x (T_0) + 2^n - 1$$

$$= 2^x + 2^x - 1$$

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

Example

Solve.

$T(n) = 1$ if $n = 1$

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

Sol$^n$.

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

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

$$= T(n/2^3) + 1 + 1 + 1$$

$$\ldots\ldots$$

$$T(n/2^k) + k$$

Put $n = 2^k$

$T(n) = T(n/2^k) + k$

$= $ [illegible]



$\log n = \log 2^k$

or, $\log n = k \log 2$

$= k$

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

$\therefore T(n) = O(\log n)$

Example

Solve.

$T(n) = 1$ if $n = 0$

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

Sol$^n$.

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

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

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

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

$\therefore T(n) = T(n-k) + 3k$

Put $n = k$,

$T(n) = T(0) + n$

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

## Substitution Method

Follow two steps



① Guess the solution

② Prove that the solution to be true by mathematical induction

Solve the recurrence relation

$T(n) = 1$ when $n = 1$

$T(n) = 2T(n/2) + n$ when $n > 1$

Sol^n.

Guess,

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

i.e. $T(n) \le c * n\log n$

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

$\le 2\left[c \frac{n}{2} \log \frac{n}{2}\right] + n$

$\le c * n \log \frac{n}{2} + n$

$\le c * n[\log n - \log 2] + n$

$\le c * n\log n - n\log 2 + n$

$\le c\, n\log n - n + n$

$\le c\, n\log n$

$\therefore T(n) \le c * n\log n$

Now, we've to show this is true for boundry condition

$T(1) \le c * 1\log 1 = 0$ ; which is false.

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

$T(2) \le c * 2\log 2$

$4 \le c * 2$

Example Solve by Iteration Method

$T(n) = 1$ if $n = 1$

$= 2T(n-1)$ if $n > 1$

Soln.

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

$= 2[2T(n-2)] = 2^2 T[n-2]$

$= 4[2T(n-3)] = 2^3 T(n-3)$

$= 8[2T(n-4)] = 2^4 T(n-4)$

$\cdots\cdots$

$2^k T(n-k)$



So $T(n) = 2^k T(n-k)$

Put $n-k = 1 \Rightarrow k = n-1$

$T(n) = 2^{n-1} T(1)$

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

$= 2^{n-1}$

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

Example $T(n) = T(n-1) + 1$ and $T(1) = \Theta(1)$

Soln.

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

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

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

$= (T(n-4) + 4$

$\cdots\cdots$

$T(n-k) + k$

Put $k = n-1$ [ Place $n-k = 1$ ]

$T(n-k) = T(1) = \Theta(1)$

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

Example Solve.

$T(n) = 2T(n-1) + k \quad ; \quad T(0) = 1$



Sol$^n$.

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

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

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

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

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

. . . . . .

$= 2^m T(n-m) + 2^{m-1} k + 2^{m-2} k + 2^0 k$

$= 2^m T(n-m) + k[2^{m-1} + 2^{m-2} + \ldots 2^0]$

$= 2^m T(n-m) + k[2^0 + 2^1 + \ldots 2^{m-1}]$

$= 2^m T(n-m) + k \sum_{i=0}^{m-1} 2^i$

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

Put $n = m$

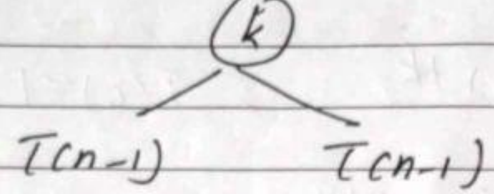
$T(n) = 2^m T(0) + k(2^n - 1)$

$= \cancel{2^n (T0)}$

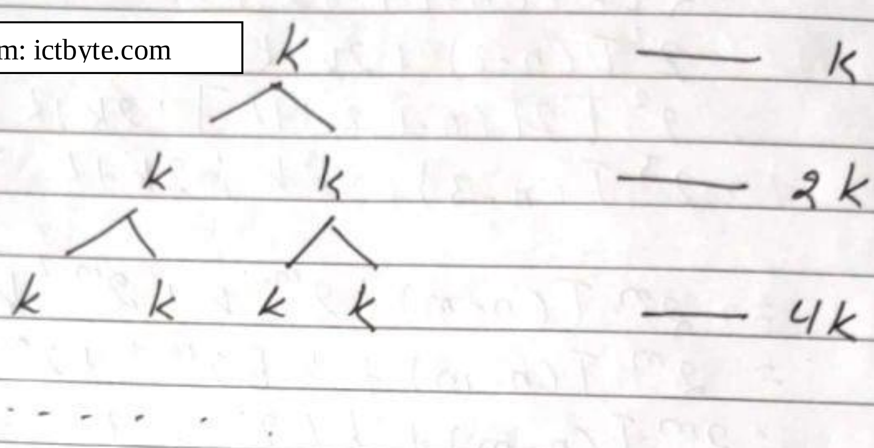
$= 2^n T(0) + k(2^n - 1)$

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

Solving same with recursion tree






$T(n) = k + 2k + 4k + 8k + \dots 2^P - 1)$

$= (k + 2^k + 2^k k + 2^3 k + 2^P k) \cdot T(0)$

$T(0) \quad \dots \quad 2^P$

Put $P = n$

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

Example

Solve

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

$T(1) = 1$

Sol$^n$.

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

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

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

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

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

-------

After $k^{th}$ term,

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



Put $n = 2^k$

$T(n) = n\,T(1) + n*k$

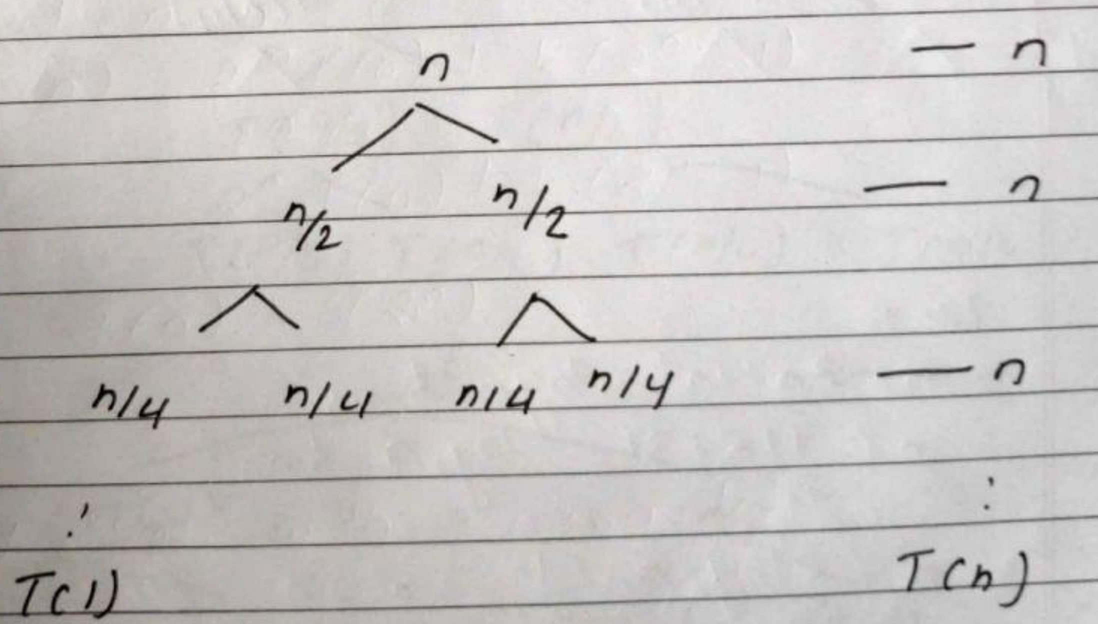
$= n + n*k$

$= n + n\log n$

$\therefore T(n) = O(n\log n)$

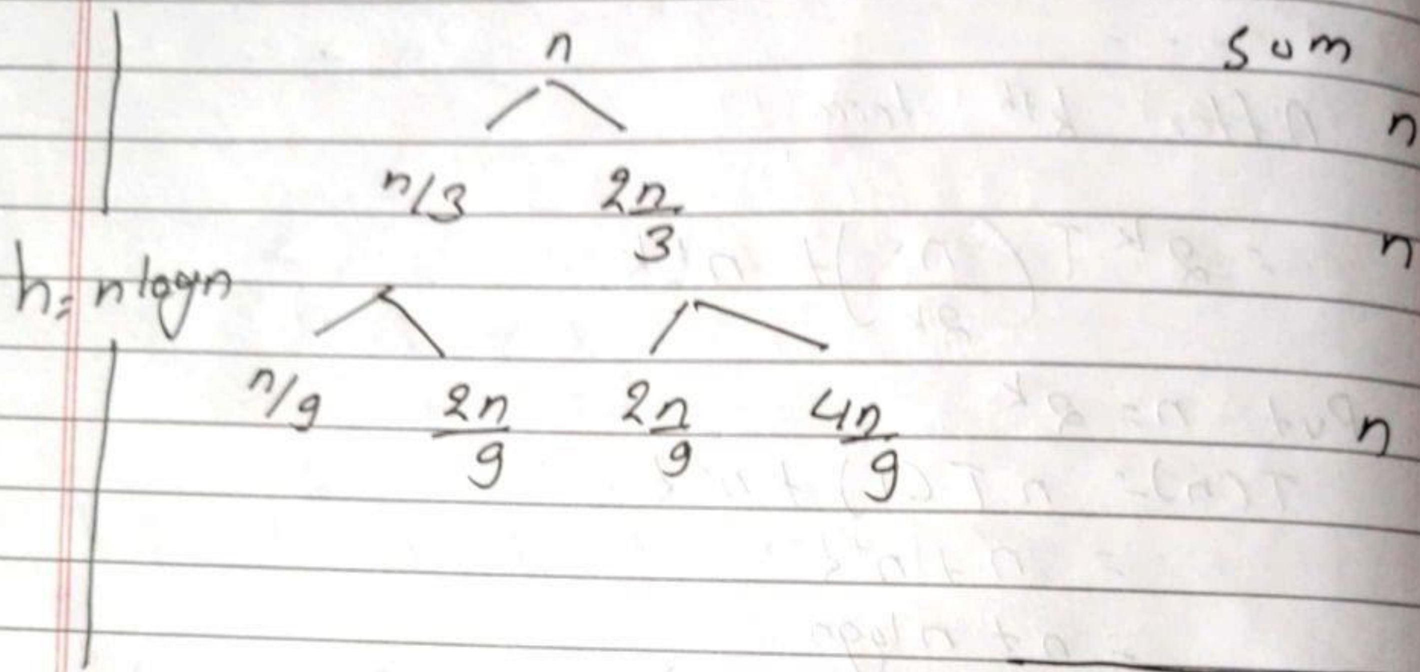
Examp Recursion Tree.

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




Total $= O(n\log n)$

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




Total $\Theta(n\log n)$

* Height of binary tree is $n\log n$.
  $\therefore T(n) = O(n\log n)$

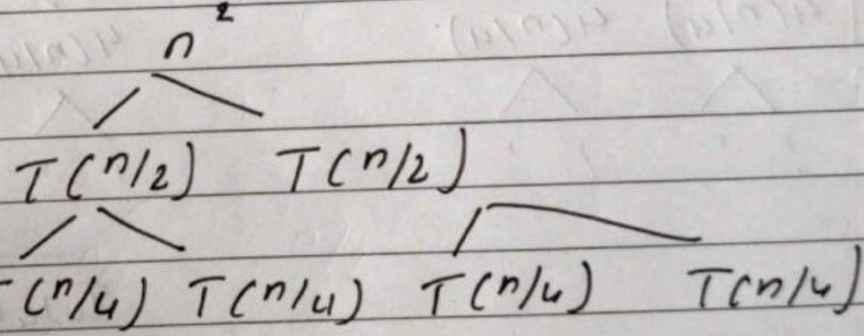
## Recursion Tree Method

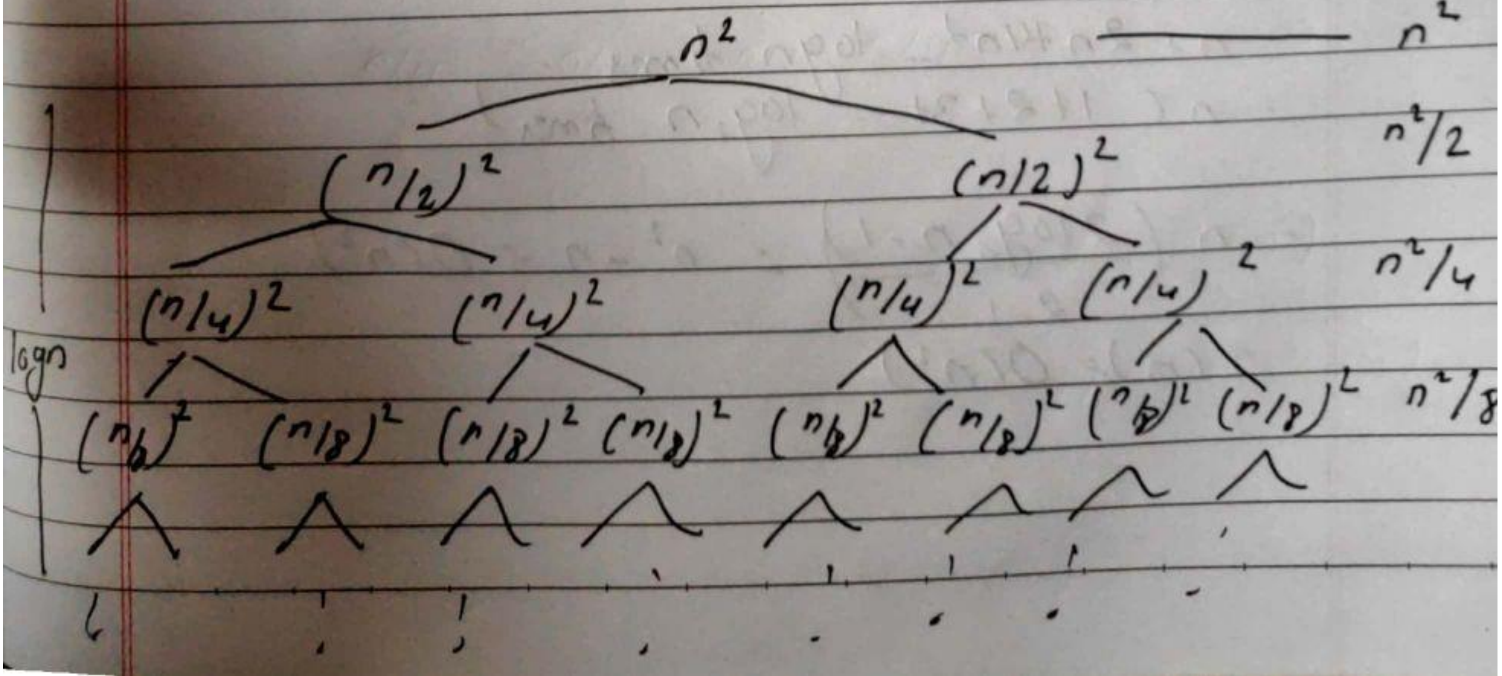
- Pictorial repesentation of iteration method. It's in form of tree, each level node are expanded
- Generally, second term of recurrence is considered as root
- When divide and conquer algorithm is used, it is important.
- Every root and child represents cost of single sub problem
- To determine cost of all levels of recursion, we add cost within each level and all levels are summed up.



Consider

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



$$T(n) = n^2 + n^2/2 + n^2/4 + \dots \text{ logn times}$$

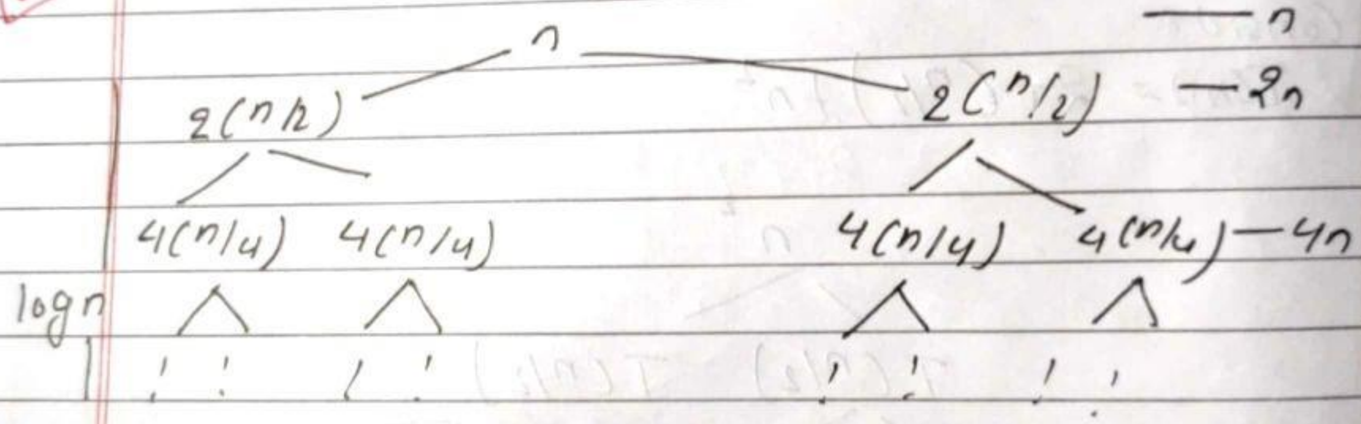
$$\le n^2 \sum_{i=0}^{\infty} \frac{1}{2^i}$$

$$\le n^2 \left( \frac{1}{1-1/2} \right) \le 2n^2$$

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



Example $T(n) = 4T(n/2) + n$




Total $\theta(n^2)$

We've,

$n + 2n + 4n + \dots$ logn times

$= n(1 + 2 + 3 + \dots \log_2 n \text{ times})$

$$= n\left(\frac{2^{\log_2 n} - 1}{2-1}\right) = n^2 - n = O(n^2)$$

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

## Master Method

Master method is technique to solve recurrence relation of form

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



where,

$n$ = size of problem / ~~sm~~ input

$a$ = number of sub problem

$n/b$ = size of each sub problem / same size assumed

$f(n)$ = cost of work outside recursion

Here,

$a \geq 1$ , $b > 1$ are constants

$f(n)$ is asymptotically positive function

* Asymptotically positive function means, for sufficiently large value of $n$, we've $f(n) > 0$.