

Computer Notes

DESIGN
ANALYSIS
&
ALGORITHM

PART-I

(1)

Asymptotic Notation

1. O - Notation.

2. Ω - Notation.

3. Θ - Notation

Q) Let $f(n)$ & $g(n)$, be two tye functions.

Big O Notation (Upper bound.)

$$f(n) = O(g(n)) \text{ iff } [f(n) \in g(n)]$$

$f(n)$ is order of $g(n)$

there exists 2. constant 3. $c > 0$ & $n_0 > 0$

such that

$$\boxed{f(n) \leq c.g(n), \forall n, n \geq n_0}$$

$$\boxed{0 < f(n) \leq g(n)}$$

Q.

$$f(n) = n$$

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

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

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

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

$$n \leq c.n^2, \forall n, n \geq 1$$

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

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

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

$$n^2 = O(n^4)$$

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

$$n^2 \leq c \cdot n^4$$

$g(n)$ is greater than c times of $f(n)$.

Ω -Notation (lower bound)

$$f(n) = \Omega(g(n)) \text{ iff}$$

there exist 2-constant $c > 0$ & $n_0 > 0$

such that

$$f(n) \geq c \cdot g(n), \forall n \geq n_0.$$

$$O < f(n) \geq g(n)$$

Θ -Notation ($L.B = U.B$)

$$f(n) = \Theta(g(n)) \text{ iff}$$

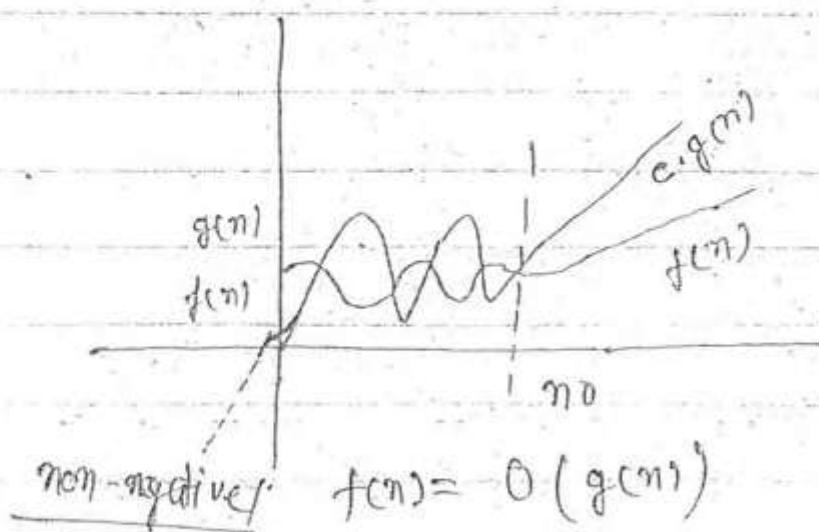
(1) $f(n) = O(g(n))$

&

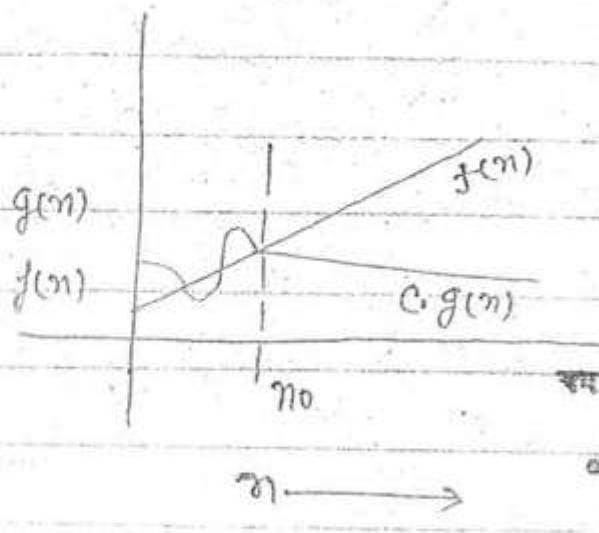
(2) $f(n) = \Omega(g(n))$

$$\boxed{c_2 g(n) \leq f(n) \leq c_1 g(n)} \quad \forall n, n \geq n_0$$

Big - O- Notation

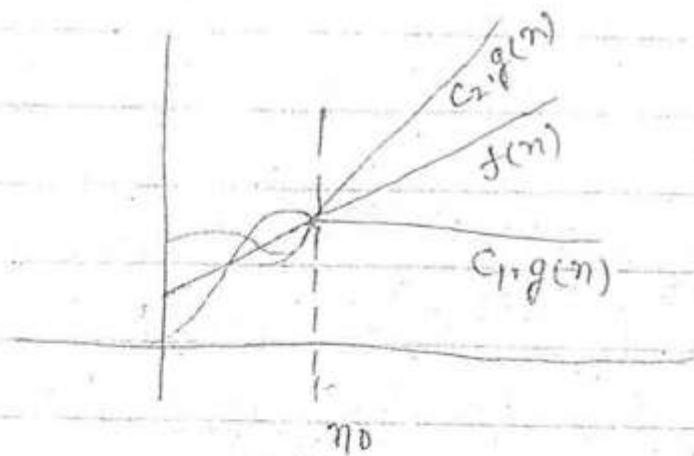


Omega - Notation



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

Theta - Notation



$$n \rightarrow$$

$$\left. \begin{array}{l} f(n) = o(g(n)) \\ f(n) = \Omega(g(n)) \end{array} \right\} \quad f(n) = \Theta(g(n))$$

$A = n^2$

$$B = n^3$$

$$\boxed{A = O(B)}$$

$A = n^2$

$$B = \underline{n^2}$$

$$\boxed{A = \Theta(B)}$$

$A = n^2$

$$B = n$$

$$\boxed{A = \Omega(B)}$$

?

P



A



Best case $n(\Omega(n))$

Worst case $n^2 (O(n^2))$

Best $\Rightarrow n^2 \Rightarrow O(n^2)$
Worst $\Rightarrow n^2$

Types of Time Complexity

- 1) Constant Time complexity $\Rightarrow O(1)$
- 2) Logarithmic " $\Rightarrow O(\log n) \Rightarrow \overline{\log(n)}$ "
- 3) Linear " $\Rightarrow O(n)$ "
- 4) Quadratic " $\Rightarrow O(n^2)$ "
- 5) Cubic " $\Rightarrow O(n^3)$ "
- 6) Polynomial " $\Rightarrow O(n^k), k > 3$ "
- 7) Exponential " $\Rightarrow O(a^n) a > 1$ "

Consider the following two functions

$$f(n) = n.$$

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

Which one of the following are true?

- a) $f(n) = O(g(n))$
- b) $f(n) = \Omega(g(n))$
- c) $f(n) = \Theta(g(n))$
- d) We can't compare

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

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

↓

$$n \leq c \cdot n^2 \quad \forall n, n \geq 1$$

↓

1

#

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

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

then which one of the following is true?

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

Solⁿ

n	n^2	2^n
1	1	2
2	4	4
3	9	8

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

↓

$$n^2 \leq c \cdot 2^n, \forall n, n \geq 4$$

4	16	16
5	25	32
6	36	64
7	49	128
8	64	256

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

$$g(n) = n (\log n)^{10}$$

Which one of the following is true ?

a) $f(n) = o(g(n))$

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

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

d) We can't compare

$$n^2 \log n$$

$$n (\log n)^{10}$$

$$n$$

$$(\log n)^9$$

$$\log n$$

>

$$9 \log(\log n)$$

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

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

Which one of the following is true ?

a)

b)

c)

d)

Solⁿ

$$\begin{array}{c} 2^n \\ n \log 2 \\ n \end{array} < \begin{array}{c} n^n \\ n \log n \\ n \log n \end{array}$$

n	2^n	n^n
1	2	1
2	4	4
3	8	27
4	16	256
5	32	3125

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

$$f(n) \leq c(g(n)), \forall n, n \geq 2$$

$$\frac{4}{2^n} \leq \frac{4}{1} \cdot n^n$$

Which one of the following is true or false?

a) $100n \log n = O\left(\frac{n \log n}{100}\right)$ // In any problem constant will not be considered

b) $\sqrt{\log n} = O(\log(\log n))$ // In any problem constant will not be considered

c) $0 < \alpha < \gamma$ then

$$n^\alpha \text{ is } O(n^\gamma)$$

d) $2^n \neq O(n^k), k > 0$

$$\underline{1^m} \quad (a) \quad 100 \cdot n \log n = O \frac{n \log n}{100}$$

$$n \log n \leq c \cdot n \log n$$

$$\Downarrow \quad 10000$$

$$1000000$$

Identify True/False for the following Stmt.

a) $(n+k)^m = O(n^m)$

where k is constant.

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

if both for " are sum
don't apply log

c) $2^{2^n} = O(2^n)$

which one of the following Stmt is false. ?

$n^2 \cdot 2^{3\log_2 n}$ is $O(n^5)$

$\frac{4^n}{2^n}$ is $O(2^n)$

X c) $2^{\log_2 n}$ is $\Theta(n^2)$

d) none

formulas

1) $\log_c(ab) = \log_c a + \log_c b$

2) $\log_b \frac{1}{a} = \log_b a^{-1} \Rightarrow -\log_b a$

3) $a^{\log_b n} = n^{\log_b a}$ ✓ n.v.

4) $\log_c a + \log_c b = \log_c(ab)$

5) $\log_a b = \frac{\log_c b}{\log_c a}$

6) $\log_b a = \frac{1}{\log_a b}$

7) $\log^K n = (\log n)^K$

Q) n

a) $n^2 \cdot n^3 = n^5$

b) $\frac{4^n}{2^n} \Rightarrow \left(\frac{2^2}{2}\right)^n \Rightarrow \frac{2^{2n}}{2^n} \Rightarrow 2^{2n-n} \Rightarrow 2^n$

c) n is $O(n^2)$ ✓
 n is \sqrt{n} ✗

? Which one of the following options provides the increasing order of asymptotic complexity of $f_i(n)$ f_1, f_2, f_3, f_4 & f_4 .

(4) $f_1(n) = 2^n$

(2) $f_2(n) = n^3/2$

(1) $f_3(n) = n \log_2(n)$

(3) $f_4(n) = n^{\log_2 3}$

a) $f_3 f_2 f_4 f_1$

b) $f_2 f_3 f_1 f_4$

c) $f_3 f_2 f_1 f_4$

d) $f_2 f_3 f_4 f_1$ (◎)

Solⁿ

2^n	$n \log n$	n^n	2^n
n	$(\log n)^2$	n^n	2^n
$n^{3/2}$	$n \log n$		
n^m	$n \log n$		

O., Ω , Θ

$n^2 = O(n^2)$ — Tight upper bound

$n^2 = O(n^3)$ — not T.U.B.

#

O

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

$n^2 \neq n^2$ TUB

$n^2 < n^3$ NTUB

#

O

$n^2 < n^3$

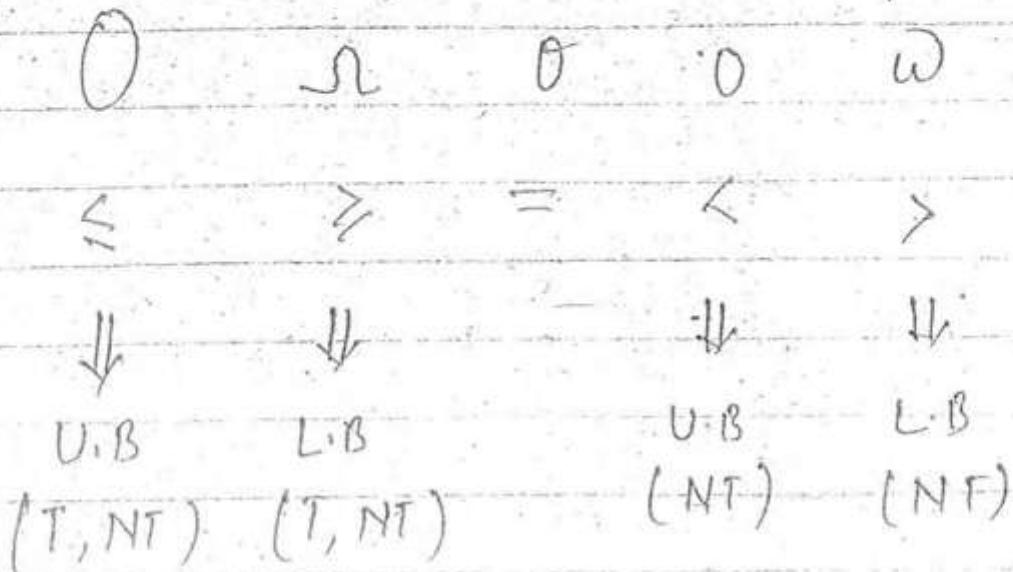
$$f(n) = \underline{n}(g(n))$$

#

$n^2 > n$ — NTUB

$n^2 = n^2$ — T.L.B.

$$n^2 \geq n \rightarrow NT LB$$



$f(n) = n^1$ [not comparable]

$$g(n) = n^{\sin \theta}$$

$f(n) = \lceil n \rceil$ } not comparable.

$$g(n) = n^{\sin \theta}$$

$$\lceil n \rceil > n^{\sin \theta} \quad (\sin \theta = 0)$$

$$\lceil n \rceil < n^{\sin \theta} \quad (\sin \theta = 1)$$

$O(1)$

$\log(\log n)$

$\log n$

$\log n$

\sqrt{n}

n

n^2

n^3

n^K

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

Properties of Asymptotic Notation

1) Reflexive \Rightarrow

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

Ex

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

↓

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

$$A = \{1, 2, 3\}$$

$$B = \{a\}$$

2 Symmetric

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

then

$$g(n) \neq O(f(n))$$

$$A = \{1, 2, 3\}$$

$$R = \{(x, y) | y, x\}$$

$$[M = M^T]$$

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

then

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

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

then

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

transitivs

$$2 < 3 \text{ & } 3 < 4$$

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

$$2 < 4$$

$g(n) = O(h(n))$

then

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

$$2 < 4 < 3$$

) ~~If~~ Similarly n, θ also satisfied transitives.

Properties

L

$$f(n) = O(f(n/2)) \quad \underline{\text{eq}} \quad n = O(n/2)$$

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

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

then

i) $\log(f(n)) = O(\log(g(n)))$

$$(ii) 2^{f(n)} = O(2^{g(n)})$$

$$(iii) h(n) \cdot f(n) = O(h(n) \cdot g(n))$$

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

$$g(n) \text{ & } \\ g(n) = O(h(n))$$

then

$$(i) f(n) + g(n) = O(d(n) + h(n))$$

$$(ii) f(n) \cdot g(n) = O(d(n) \cdot h(n))$$

Let $f(n)$, $g(n)$ & $h(n)$ be three non-negative functions which are defined as follows.

$$(f(n) = o(g(n))) \text{ & } g(n) \neq o(f(n)))$$

$$g(n) = o(h(n)) \text{ & } h(n) = o(g(n))$$

then which one of the

false?

a) $f(n) + g(n) = o(h(n))$

b) $f(n) = o(h(n))$

c) $h(n) \neq o(f(n))$

d) $f(n) \cdot h(n) \neq o(g(n) \cdot h(n))$

$$\left. \begin{array}{l} a \leq b \\ b > a \end{array} \right\} a = b.$$

\Leftarrow

$$f(n) = o(g(n)) \text{ & } (g(n) \neq o(f(n)))$$

$$\Rightarrow f(n) < g(n)$$

$$g(n) = O(h(n)) \quad \& \quad h(n) = O(g(n))$$

$$\begin{array}{c} g(n) = h(n), \quad f(n) < g(n) \\ f(n) < g(n) = h(n) \end{array}$$

Q Suppose $T_1(n) = O(f(n))$

and

$$T_2(n) = O(f(n))$$

then which one of the
following is true ?

~~a)~~ $T_1(n) + T_2(n) = O(f(n))$

b) $T_1(n) = O(T_2(n))$

c) $T_2(n) = O(T_1(n))$

d) $T_1(n) / T_2(n) = O(1)$

SE)
b

$$T_1(n) = \Theta(f(n))$$

$$T_2(n) = \Theta(f(n))$$

R

$$\begin{aligned} T(n) &= \sum_{i=1}^n i \\ &= 1 + 2 + 3 + \dots + n \\ &= \frac{n(n+1)}{2} \\ &= \Theta(n^2) \end{aligned}$$

ii)

$$\begin{aligned} T(n) &= \sum_{i=1}^n i^2 \\ &= 1^2 + 2^2 + 3^2 + \dots + n^2 \\ &= \frac{n(n+1)(2n+1)}{6} = \Theta(n^3) \end{aligned}$$

$$(III) T(n) = n!$$

$$= n(n-1)(n-2)(n-3) \dots (n-(n-1))$$

$$= n \times n \times n \times \dots \times n$$

$$= O(n^n)$$

(IV)

$$T(n) = \sum_{i=1}^n u^i \quad \left| \frac{u(x^{n-1})}{x-1} \right.$$

$$= u^1 + u^2 + u^3 + \dots + u^n$$

$$= \frac{u(u^{n-1})}{u-1}$$

$$= O(u^n)$$

(V)

$$T(n) = 2 + \frac{1}{n} \quad \left| \frac{5 + \frac{1}{n^2} = O(1)}{n + \frac{1}{n^2} = O(n)} \right.$$

$$T(n) = 2 + 0$$

$$= 2$$

$$= O(1)$$

$$2 + \frac{1}{n} = O(1)$$

$$(vi) \quad 2n^2 + n \log n = O(n^2)$$

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

Hence

$$n \cdot n + n = O(n^2)$$

$$(vii) \quad n^3 + 16n^2 = O(n^3)$$

$$(viii) \quad 5n^2 - 6n = O(n^2)$$

Q Consider the following two functions. The functions are given like as

$$f_1(n) = \begin{cases} n^3 & 0 \leq n \leq 10,000 \\ n^2 & n > 10,000 \end{cases}$$

$$f_2(n) = \begin{cases} n & 0 \leq n \leq 100 \\ n^3 & n > 100 \end{cases}$$

In $f_1(n)$ and $f_2(n)$ what is the relation?

$$\therefore f_1(n) = f_2(n)$$

$f(n)$

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



$$f(n) \leq c g(n), \forall n, n \geq n_0$$

In big-O notation only a starting is available.

$$f(n) = n^3 \quad 0 \leq n \leq 10,000$$

$$= n^2 \quad n \geq 10,000$$

$$g_2(n) = n \quad 0 \leq n \leq 100$$

$$n^3 \quad n \geq 100$$

$$g_1(n) \leq c \cdot g_2(n)$$

$$n^2 \leq c \cdot n^3, \quad n \geq 10,000$$

$$g_1(n) = O(g_2(n))$$

Tuesday

17/06/11

Q Explain why running time algo A is at least $\Omega(n^2)$ is meaningless.

Best case n^2 [$\Omega(n^2)$]

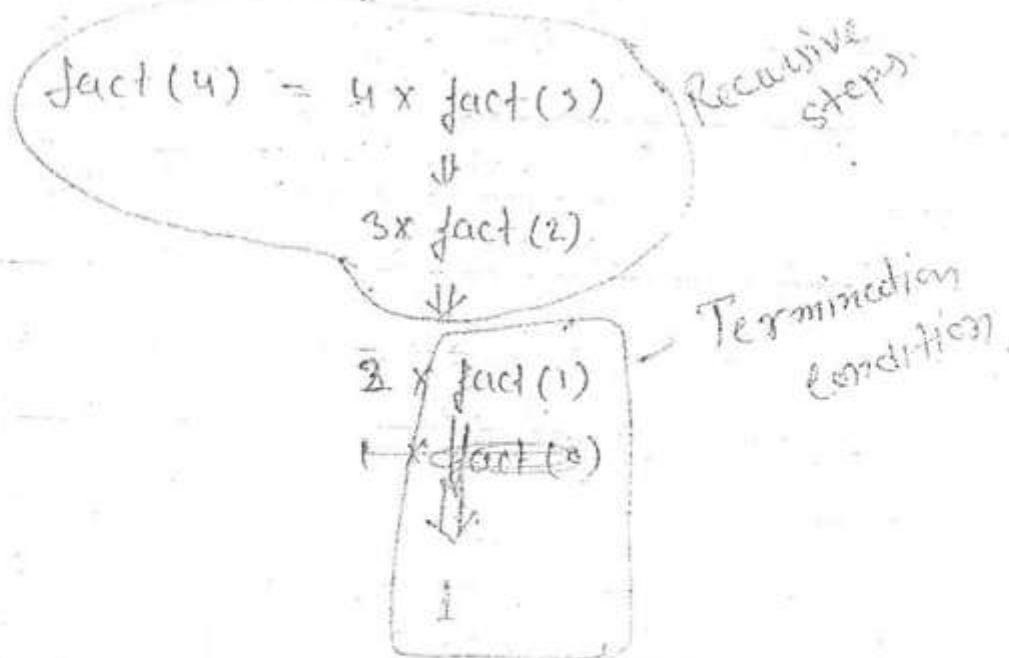
worst n n^2 [$\Omega(n^2)$]

All [$\Omega(n^2)$]

Recursion

A function call that itself to solve a particular problem is called recursion.

e.g.



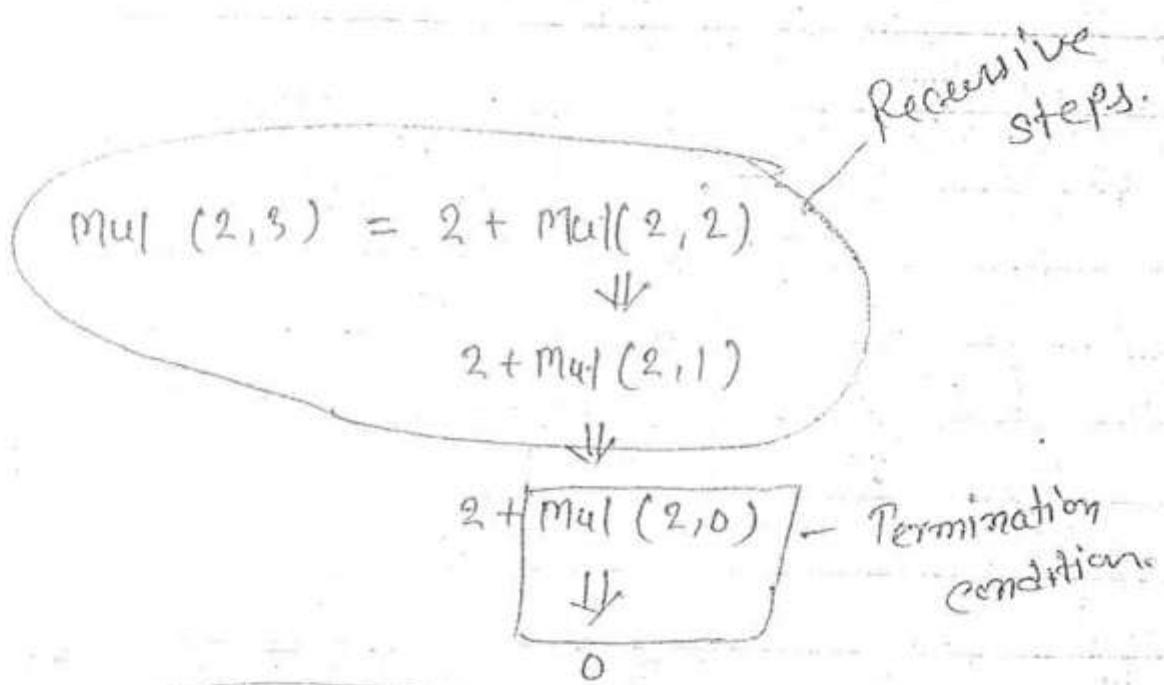
In this "fun" to "fun" but parameter is change.

Properties of Recursion

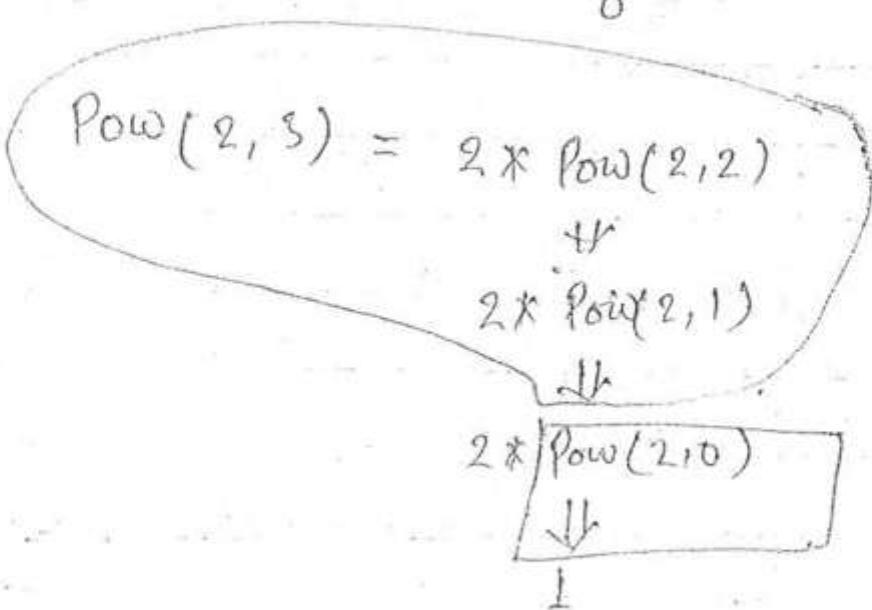
- (1) Should have Termination condition.
- (2) Should have recursive steps.
- (3) For every fun-call there should be change in the parameter values.

(Q3)

eg



eg



Recurrence Relⁿ

$$\text{MUL}(a, b) = \begin{cases} 0 & \text{if } a=0 \text{ (or) } b=0 \\ a + \text{mul}(a, b-1) & \text{otherwise} \end{cases}$$

$$\text{POW}(a, b) = \begin{cases} 0 & \text{if } a=0 \\ 1 & \text{if } b=0 \\ a * \text{pow}(a, b-1) & \text{otherwise} \end{cases}$$

$$\text{FACT}(n) = \begin{cases} 1 & \text{if } n \leq 1 \text{ or } n=0 \\ n * \text{fact}(n-1) & \text{otherwise} \end{cases}$$

Q Write a recurrence Relⁿ to find G.C.D of (m, n)

$$\text{GCD}(m, n) = \begin{cases} 1 & \text{if } m=0 \text{ & } n=0 \\ m & \text{if } n=0 \\ n & \text{if } m=0 \\ \text{GCD}(n, m \% n) & \text{otherwise} \end{cases}$$

eg. $\checkmark \text{GCD}(5, 50) = 5$

$$\checkmark \text{GCD}(5, 7) = 1$$

$$\checkmark \text{GCD}(0, 5) = 5$$

$$\checkmark \text{GCD}(5, 0) = 5$$

$$\checkmark \text{GCD}(0, 0) = 1$$

$\text{GCD}(5, 50)$

↓

$$\begin{array}{r} 5) 50 (10 \\ \underline{-50} \\ 0 \end{array}$$

$$\text{GCD}(5, 0) \Rightarrow 5$$

$\text{GCD}(5, 7)$

↓

$$\begin{array}{r} 5) 7 (1 \\ \underline{-5} \\ 2 \end{array}$$

$$\text{GCD}(5, 2)$$

↓

$$\begin{array}{r} 5) 2 (0 \\ \underline{-0} \\ 2 \end{array}$$

$\text{GCD}(5, 7)$

↓

$$7) 5 (0$$

$$\text{GCD}(7, 5)$$

↓

$$\begin{array}{r} 7) 5 (1 \\ \underline{-5} \\ 0 \end{array}$$

$$\text{GCD}(5, 2)$$

$$2) 5 (2$$

$$\begin{array}{r} 2) 5 (2 \\ \underline{-4} \\ 1 \end{array}$$

$$\text{GCD}(2, 1)$$

$$\begin{array}{r} 2) 2 (2 \\ \underline{-2} \\ 0 \end{array}$$

$$\text{GCD}(1, 0) = 1$$

$$\# \quad \text{GCD}(5, 100)$$

$$\begin{array}{r} \downarrow \\ 100) 5 \longdiv{10} \\ \quad \quad \quad \underline{0} \\ \quad \quad \quad 5 \end{array}$$

$$\text{GCD}(100, 5)$$

$$\begin{array}{r} \downarrow \\ 5) 100 \longdiv{20} \\ \quad \quad \quad \underline{100} \\ \quad \quad \quad 0 \end{array}$$

$$\text{GCD}(5, 0)$$

$$\begin{array}{r} \downarrow \\ 5 \end{array}$$

$$\# \quad \text{GCD}(100, 200)$$

$$\begin{array}{r} \downarrow \\ 200) 100 \longdiv{0} \\ \quad \quad \quad \underline{0} \\ \quad \quad \quad 100 \end{array}$$

$$\text{GCD}(200, 100)$$

$$\begin{array}{r} \downarrow \\ 100) 200 \longdiv{2} \\ \quad \quad \quad \underline{200} \\ \quad \quad \quad 0 \end{array}$$

$$\text{GCD}(100, 0)$$

$$\begin{array}{r} \downarrow \\ 100 \end{array}$$

$$\# \quad \text{GCD}(a, b)$$

$$\begin{array}{r} \downarrow \\ b \end{array}$$

$$b) \quad a \mid c$$

$$\begin{array}{r} \downarrow \\ \frac{a}{0} \end{array}$$

#

Write a recursive C-Program to find factorial of n .

fact (int n)

{

 int f ; if ($n == 0$ (or) $n == 1$)

return (1)

else

{

 $f = n * \text{fact}(n-1)$ return (f)

}

}

Termination Condition another name is Initial Condition

P Write a recursive c program to find n^{th} fibonacci number. ?.

n	0	1	2	3	4	5	6	7	8	9	10
$\text{fib}(n)$	0	1	1	2	3	5	8	13	21	34	55

Recurrence Reln

$$\text{fib}(n) = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ \text{fib}(n-1) + \text{fib}(n-2) & \text{otherwise} \end{cases}$$

✓ $\text{fib}(\text{int } n)$

```
{
    int f;
    if (n==0) return (0);
    if (n==1) return (1);
    else
    {
        f = fib(n-1) + fib(n-2);
        return (f);
    }
}
```

P Write a recursive c program
to find GCD of m, n, q.

GCD (int m , int n)

{

int f ,

if (m == 0 && n == 0)

return (1)

if (m == 0)

return (n)

if (n == 0)

return (m)

else

{

f = GCD(n , m % n)

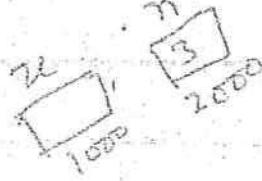
return (f);

}

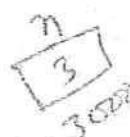
}

Main ()

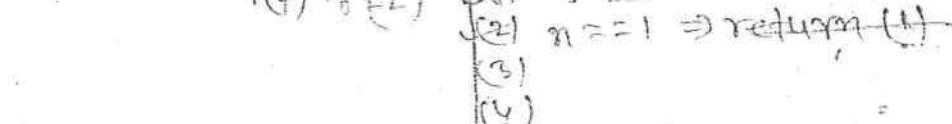
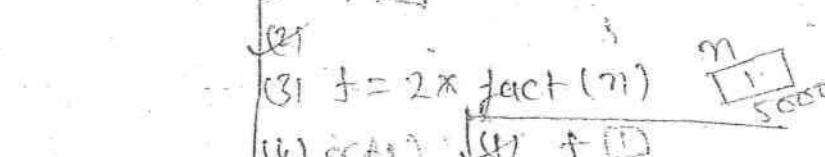
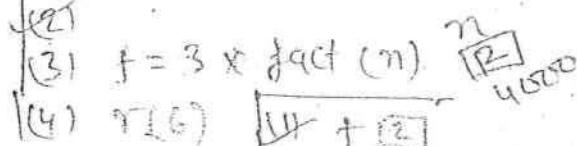
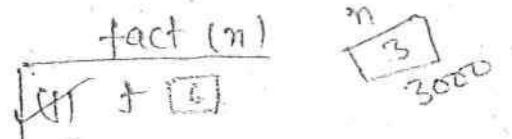
```
{  
    int n, n=3  
    n = fact(n);  
    printf("%d", n);  
}
```



fact (int n)



```
{  
    (1) int f;  
    (2) if (n==0 || n==1)  
        return (1);  
    else  
        {  
            (3) f = n * fact(n-1),  
            (4) return (f);  
        }  
}
```



(3)
(4)

More space is Drawback of the Recursive Program

Simple C Program of factorial n.

fact (int n)

{

 int f=1, i; i

 for (i=1, i<=n, i++)

 f = f * i;

 return (f);

}

Solving Recurrence Relation

(1) Substitution Method.

(2) Recursive Tree Method.

(3) Master Method.

I Substitution Method

Substitution the given fun again and again until the given fun is remove.

eg

$$\begin{aligned} T(n) &= T(n-1) + n && \text{if } n > 1 \\ &= L && \text{if } n = 1 \end{aligned}$$

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

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

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

(iv) $T(50) = T(49) + 50$

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

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

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

1
|
4
50

$$= T(n-50) + n-49 + n-48 + n-47 + \dots - n-1 + n$$

1
|
n-1
V

$$= T(n-(n-1)) + n-(n-2) + n-(n-3) + n(n-4) + \dots + n-1 + n$$

$$= T(1) + 2 + 3 + 4 + \dots + n-1 + n$$

$$= 1 + 2 + 3 + \dots + n$$

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

$$= \underline{\underline{O(n^2)}}$$

Solve the following recursion Relation

$$\begin{aligned} * T(n) &= n * T(n-1) && \text{if } n \geq 1 \\ &= 1 && \text{if } n = 1 \end{aligned}$$

Solⁿ

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

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

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

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

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

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

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

$$= (n-0) * (n-1) * (n-2) * T(n-3)$$

\downarrow
 $n-1$

$$= (n-0) * (n-1) * \dots * (n-(n-3)) * (n-(n-2))$$

$$* T(n-(n-1))$$

$$= n * (n-1) * (n-2) * \dots * 3 * 2 * T(1)$$

$$= 1 * 2 * 3 * \dots * n$$

$$= n!$$

$$= O(n^n)$$

$$* T(n) = T(n/2) + c \text{ if } n \geq 1$$

$$= 1 \text{ if } n = 1$$

Solⁿ

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

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

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

$$= \boxed{T(n/2^3) + c} + 2c$$

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

↓ K times

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

$$= T(1) + KC$$

$$= 1 + c * \log_2 n$$

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

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

Soln

$$\begin{aligned} T(n) &= 2T(n/2) + c & \left| \begin{array}{l} \frac{n}{2^K} = 1 \\ n = 2^K \end{array} \right. \\ &= 2[2T(n/2^2) + c] + c \\ &= 2^2 T(n/2^2) + 2^1 c + 2^0 c \\ &= 4 \\ &= 2^2 [2T(n/2^3) + c] + 2^1 c + 2^0 c \\ &= 2^3 T(n/2^3) + 2^2 c + 2^1 c + 2^0 c \\ &\quad \downarrow K \end{aligned}$$

$$T(n) = 2^K T(n/2^K) + c[2^0 + 2^1 + 2^2 + \dots + 2^{K-1}]$$

$$= n \cdot T(1) + c[2^0 + 2^1 + 2^2 + \dots + 2^{K-1}]$$

$$= n + c \left[\frac{1(2^K - 1)}{(2 - 1)} \right] = n + c[2^K - 1]$$

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

$$= \underline{\underline{O(n)}}$$

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

Solve

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2[2T(n/2^2) + n/2] + n \\ &= 2^2 T(n/2^2) + 2n \end{aligned}$$

\downarrow
K

$$\begin{aligned} T(n) &= 2^K T(n/2^K) + K \cdot n \\ &= nT(1) + n \cdot K \log_2 n \\ &= n + n \log_2 n \\ &= \underline{\underline{\mathcal{O}(n \log n)}} \end{aligned}$$

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

Soh

$$T(n) = T\left(\frac{n}{2}\right) + n \\ = \left[T\left(\frac{n}{2^2}\right) + \frac{n}{2} \right] + n \\ = T\left(\frac{n}{2^3}\right) + \frac{n}{2^1} + \frac{n}{2^0} \\ = T\left(\frac{n}{2^4}\right) + \frac{n}{2^2} + \frac{n}{2^1} + \frac{n}{2^0}$$

$$T(n) = T\left(\frac{n}{2^K}\right) + kn \left[\frac{1}{2^0} + \frac{1}{2^1} + \dots + \frac{1}{2^{K-1}} \right] \\ = T(1) + n \left[\frac{1 - (1/2)^K}{1 - 1/2} \right] \\ = 1 + n \left[1 - \frac{1}{n} \right] \\ = n \left[\frac{n-1}{n} \right] \\ \Rightarrow O(n)$$

$$\begin{aligned} T(n) &= \Theta(T(n/2) + n^2) \quad \text{if } n > 2 \\ &= c \quad \text{if } n \leq 2 \end{aligned}$$

Soln

$$\begin{aligned} T(n) &= \Theta(T(n/2) + n^2) \\ &= \Theta\left[T\left(\frac{n}{2}\right) + \left(\frac{n^3}{8}\right)\right] + n^2 \\ &= 2^3 T\left(\frac{n}{2^3}\right) + n^3 + n^2 \end{aligned}$$

||

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

$$T(n) = \begin{cases} \Theta T\left(\frac{n}{2}\right) + n^2 & \text{if } n > 2 \\ C & \text{if } n \leq 2 \end{cases}$$

$n=2$
 $K = \log_2 n$

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

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

$$= \Theta^2\left[\Theta T\left(\frac{n}{2}\right) + 2^1 n^2 + 2^0 n^2\right]$$

$$= \Theta^2\left[\Theta T\left(\frac{n}{2}\right) + \left(\frac{n}{2}\right)^2\right] + 2n^2 + 2^0 n^2$$

$$= \Theta^3\left[\Theta T\left(\frac{n}{2}\right) + 2^2 n^2 + 2^1 n^2 + 2^0 n^2\right]$$

$$T(n) = \Theta^{K-1} \uparrow T\left(\frac{n}{2^{K-1}}\right) + n^2 \left[2^0 + 2^1 + 2^2 + \dots + 2^{K-2} \right]$$

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

$$= n^3 \times C + n^2 \left[\frac{2^K - 1}{2^1 - 1} \right]$$

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

$$= n^3 + n^3 - n^2$$

$$\Rightarrow \underline{\underline{O(n^3)}}$$

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

$$= c \quad \text{if } n \leq 2$$

Solⁿ

$$\begin{aligned} T(n) &= 7T\left(\frac{n}{2}\right) + n^2 \\ &= 7 \left[T\left(\frac{n}{2^2}\right) + \left(\frac{n^2}{2}\right)^2 \right] + n^2 \\ &= 7T\left(\frac{n}{2^2}\right) + 7\left(\frac{n^2}{4}\right)^2 + \left(n^2\right)^2 \\ &= 7T\left(\frac{n}{2^2}\right) + \left(\frac{7}{4}\right)^1 n^2 + \left(\frac{7}{4}\right)^0 n^2 \\ &= 7T\left(\frac{n}{2^3}\right) + \left(\frac{7}{4}\right)^2 n^2 + \left(\frac{7}{4}\right)^1 n^2 + \left(\frac{7}{4}\right)^0 n^2 \end{aligned}$$

\downarrow
 $K-1$

$$\begin{aligned} T(n) &= 7^{K-1} T(2) + n^2 \left[\underbrace{1 + \left(\frac{7}{4}\right)^1 + \cdots + \left(\frac{7}{4}\right)^{K-2}}_{\frac{\left(\frac{7}{4}\right)^K - 1}{\frac{7}{4} - 1}} \right] \\ &= \frac{7^K}{7} T(2) + n^2 \left[\left(\frac{7}{4}\right)^K - 1 \right] \\ &= n^{\log_2 7} + n^2 \left[\left(\frac{7}{4}\right)^K - 1 \right] \\ &= n^{\log_2 7} + n^2 \times \left(\frac{7}{4}\right)^K - n^2 \end{aligned}$$

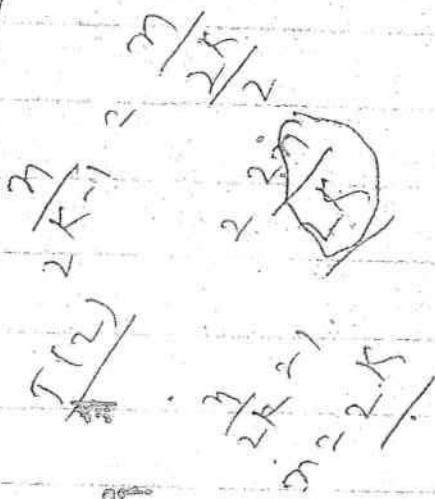
$$= n^{\log_2 7} + n^2 \times \frac{n^{\log_2 7}}{n^2}$$

$$\Rightarrow n^{\log_2 7} + n^{\log_2 7} - n^2$$

$$\Rightarrow 1 n^{2.81} - n^2$$

$$\Rightarrow O(n^{2.81})$$

$$\Rightarrow O(n^{\log_2 7})$$



$$T\left(\frac{2}{2^{K-1}}\right)$$

$$T\left(\frac{2}{2^K}\right)$$

$$T\left(\frac{2}{2^K}\right)$$

$$T\left(\frac{2}{2^K}\right)$$

$$\log_2 \frac{n}{2^K}$$

$$\log_2 \frac{n}{2^K}$$

$$2^K$$

$$n^{\log_2 7}$$

~~20/01/11~~

to

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

Ans

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

$$= [T(n-2) + \log(n-1)] + \log(n-0)$$

$$= T(n-3) + \log(n-2) + \log(n-1) + \log(n-0)$$

↓

$$T(n-3) + \log(n-2) + \log(n-1) + \log(n-0)$$

↓
n-1 times

$$T(n) = T(n-(n-1)) + \log(n-(n-2)) + (n-(n-3)) \dots \log(n)$$

$$= 1 + \log 2 + \log 3 + \log 4 \dots \log(n-1) + \log n$$

$$= 1 + \log(2, 3, 4, \dots, n)$$

$$1 + \log n!$$

$$= 1 + \log n^n$$

$$= 1 + n \log n$$

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

$$\begin{aligned} T(n) &= T(n-1) + \frac{1}{n} && \text{if } n \geq 1 \\ &= 1 && \text{if } n = 1 \end{aligned}$$

Simplifying

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

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

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

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

↓ $n-1$ times

$$\Rightarrow T(n-(n-1)) + \frac{1}{(n-(n-1))} + \frac{1}{(n-(n-1))} + \frac{1}{(n-(n))}$$

$$\Rightarrow T(1) + \frac{1}{2} + \frac{1}{1}$$

$\log n$

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + n \log n & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

$\log_2 n = k$

$$\begin{aligned} T(n) &= 2 \left[2T\left(\frac{n}{2^2}\right) + \frac{n}{2} \log \frac{n}{2} \right] + n \log n \\ &= 2^2 \left[2T\left(\frac{n}{2^3}\right) + \frac{n}{2^2} \log \frac{n}{2^2} \right] + n \log \frac{n}{2} + n \log n \end{aligned}$$

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

| k time

|

$$= 2^k T\left(\frac{n}{2^k}\right) + n \left[\log \frac{n}{2^0} + \log \frac{n}{2^1} + \log \frac{n}{2^2} + \dots + \log \frac{n}{2^{k-1}} \right]$$

$$f(n) = n T(1) + n \left[\log \frac{n}{2^0} + \log \frac{n}{2^1} + \log \frac{n}{2^2} + \dots + \log \frac{n}{2^{k-1}} \right]$$

$$\begin{aligned} &= n + n \left[(\log_2 n - 0) + (\log_2 n - 1) + (\log_2 n - 2) + \dots + (\log_2 n - (k-1)) \right] \end{aligned}$$

$$= n + n \left[K * \log_2 n - \left[0 + 1 + 2 + \dots + (K-1) \right] \right]$$

$$= n + n \left[(\log_2 n)^2 - \frac{(K-1)(K)}{2} \right]$$

$$= n + n \left[(\log_2 n)^2 - \frac{(\log_2 n)^2}{2} \right]$$

$$= O(n(\log_2 n)^2)$$

$$T(n) = \begin{cases} 5T\left(\frac{n}{5}\right) + \frac{n}{\log_5 n} & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

$n = 5^K$
 $\log_5 n = K$

S_1^n

$$S_1^n \left[ST\left(\frac{n}{5^2}\right) + \frac{n/5}{\log_5 n/5} \right] + \frac{n}{\log_5 n}$$

$$= S_1^2 \left[ST\left(\frac{n}{5^2}\right) + \frac{n}{\log_5 n/5} + \frac{n}{\log_5 n} \right]$$

$$= S_1^2 \left[ST\left(\frac{n}{5^3}\right) + \frac{n}{5^2} \log_5 \frac{n}{5^2} \right] + \frac{n}{\log_5 n/5} + \frac{n}{\log_5 n}$$

$$= S_1^3 \left[T\left(\frac{n}{5^3}\right) + n \log_5 \frac{n}{5^2} + \frac{n}{\log_5 n/5} + \frac{n}{\log_5 n} \right]$$

K times.

$$= S_1^K T\left(\frac{n}{5^K}\right) + n \left[\frac{1}{\log \frac{n}{5^2}} + \frac{1}{\log \frac{n}{5^1}} + \dots + \frac{1}{\log \frac{n}{5^K}} \right]$$

$$= n T(1) + n \left[\frac{1}{\left(\log \frac{n}{5^0}\right)} + \frac{1}{\left(\log \frac{n}{5^1}\right)} + \frac{1}{\left(\log \frac{n}{5^2}\right)} + \dots \right]$$

$$= \left[\frac{1}{\left(\log_5 n - (K-1)\right)} \right]$$

$$= n + n \left[\frac{1}{k} + \frac{1}{k+1} + \frac{1}{k+2} + \dots + \frac{1}{n} \right]$$

$$\approx n + n \left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{k} \right]$$

$$= n + n [\log k] = n + n \log(\log_5 n)$$

$$= O(n \log(\log_5 n))$$



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

$$\left| \begin{array}{l} n-2k=0 \\ n=2k \\ k=n/2 \end{array} \right.$$

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

$$\begin{aligned} \cancel{\text{SD}}^n &= T(n-2^1) + 2^1 \log(n-2) + 2\log n \\ &= T(n-2^2) + 2^2 \log(n-2) + 2\log(2^1 \cdot 0) \\ &= T(n-2^3) + 2^3 \log(n-2) + 2\log(n-2) + 2\log(n-0) \end{aligned}$$

↓
K times

$$\begin{aligned} T(n) &= T(n-2 \times K) + 2\log(n(2^{K-2})) + 2\log(n-(2^K-4)) \\ &\quad + \dots + 2\log(n-0) \end{aligned}$$

$$= T(0) + 2\log 2 + 2\log 4 + 2\log 8 + \dots + 2\log 2^K$$

$$= 1 + 2 \left[\log 2 + \log 4 + \log 8 + \dots + \log 2^K \right]$$

$$= \left[\log(2 \times 1) + \log(2 \times 2) + \log(2 \times 3) + \dots + \log(2 \times K) \right]$$

$$= \left(\log_2 2 + \log_2 1 \right) + \left(\log_2 2 + \log_2 2 \right) + \left(\log_2 2 + \log_2 3 \right)$$

$$+ \dots + \left(\log_2 2 + \log_2 \frac{K}{2} \right)$$

$$= K \times \log_2^2 + [\log_2 1 + \log_2 2 + \log_2 3 - \log_2 K]$$

$$= K + \log K!$$

$$= K + K \log K$$

$$= n/2 + n/2 \log n/2$$

$$= O(\log n)$$

ii)

$$\begin{aligned} T(n) &= T(n-2) + n^2 && \text{if } n > 0 \\ &= 1 && \text{if } n = 0 \end{aligned}$$

Solⁿ $T(n) = T(n-2) + n^2$

$$= T(n-4) + (n-2)^2 + n^2$$

$$= T(n-4) + (n-2)^2 + n^2$$

$$= T(n-6) + (n-4)^2 + (n-2)^2 + n^2$$

$$= T(n-8) + (n-6)^2 + (n-4)^2 + (n-2)^2 + n^2$$

$$= T(n-10) + (n-8)^2 + (n-6)^2 + (n-4)^2 + (n-2)^2 + n^2$$

$$\begin{aligned}
 &= T(n - 2k) + (n - (2k-2))^2 + (n - (2k-4))^2 + \dots \\
 &= T(0) + 2^2 + 4^2 + 6^2 + \dots = \frac{(n-0)^2}{(2k)^2} \\
 &= 1 + [4 + 16 + 36 + 64 + \dots (2k)^2] \\
 &= 1 + 2^2 [1^2 + 2^2 + 3^2 + 4^2 + \dots k^2] \\
 &= 1 + 2^2 \left[\frac{k(k+1)(2k+1)}{6} \right] \\
 &= \frac{k^3}{6} = k^3 \\
 &= \left(\frac{n}{2}\right)^3 \\
 &= O(n^3)
 \end{aligned}$$

$$T(n) = 2T(n-1) + n \quad \text{if } n > 1$$

$$= 1 \quad \text{if } n = 1$$

$$\underline{8d^n} \quad T(n) = 2T(n-1) + n$$

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

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

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

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

=

1

⋮

⋮

$n-1$ times.

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

$$2^{n-1} T(1) + 2^0(n-0) + 2^1(n-1) + \dots + 2^{n-3}(n-(n-3))$$

$$+ 2^{n-2}(n-(n-2))$$

$$= 2^{n-1} + \underbrace{2^0(n-0) + 2^1(n-1) + \dots + 2^{n-3}}_{\text{Sum of first } (n-3) \text{ terms of GP}} + 2^{n-2}(2)$$

$$S = 2^0(n-0) + 2^1(n-1) + 2^2(n-2) + \dots - 2^{n-3}(3) + \dots - 2^{n-2}(2)$$

$$2S = \frac{1}{2}(n-0) + \frac{1}{2^2}(n-1) + \dots$$

$$\begin{aligned} &+ \cancel{2^{n-3}(3)} + \cancel{2^{n-2}(2)} \\ &+ 2^{n-3}(3) + 2^{n-2}(2) \end{aligned}$$

$$S - 2S = 2^0(n-0) - 2 - 2^2 - 2^3 - 2^4 - 2^5 - \dots - 2^{n-3} - 2^{n-2} - 2^n$$

$$fS = -n[2^1 + 2^2 + 2^3 + \dots + 2^{n-2}] + 2^n$$

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

$$\Rightarrow -n + 2^{n-1} - 2 + 2^n$$

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

$$T(n) = \sum_{i=1}^K n \left(\frac{1}{2}\right)^i \cdot i$$

$$n = 2^K$$

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

$$P(n) = \frac{n}{2^1} + \frac{n}{2^2} \cdot 2 + \frac{n}{2^3} \cdot 3 + \frac{n}{2^4} \cdot 4 - \frac{n}{2^K} \cdot K$$

$$= n \left[\frac{1}{2^1} \cdot 1 + \frac{1}{2^2} \cdot 2 + \frac{1}{2^3} \cdot 3 + \dots + \frac{1}{2^K} \cdot K \right]$$

$$S = \frac{1}{2^1} \cdot 1 + \frac{1}{2^2} \cdot 2 + \frac{1}{2^3} \cdot 3 + \dots + \frac{1}{2^K} \cdot K$$

$$\frac{1}{2} \cdot S = + \frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} + \dots + \frac{K+1}{2^K} + \frac{K}{2^{K+1}}$$

$$S - \frac{1}{2}S = \frac{1}{2^1} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} + \dots + \frac{1}{2^K} - \frac{K}{2^{K+1}}$$

$$S = 2 \left[\frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \dots + \frac{1}{2^K} \right] - \frac{K}{2^K}$$

$$= 2 \left[\frac{1}{2} \left(\frac{1 - \left(\frac{1}{2}\right)^K}{1 - \frac{1}{2}} \right) \right] - \frac{K}{2^K}$$

$$S = 2 \left[1 - \frac{1}{2^k} \right] - \frac{k}{2^k}$$

$$= 2 \left[\frac{n-1}{n} \right] - \frac{\log n}{n}$$

$$\begin{aligned} T(n) &= n \left[2 \left[\frac{n-1}{n} \right] - \frac{\log n}{n} \right] \\ &= O(n) \end{aligned}$$

$$\begin{cases} T(n) = \sqrt{n} T(\sqrt{n}) + n & \text{if } n \geq 2 \\ = 2 & \text{if } n = 2 \end{cases}$$

$$\begin{aligned} T(n) &= n^{1/2} T(n^{1/2}) + n \\ &\quad + \frac{1}{n^{1/2}} = 2 \end{aligned}$$

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

$$= n^{3/4} * T(n^{1/2}) + 2n$$

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

$$T(n) = n^{\frac{1}{2^k}} \left(n^{Y_2} \right) + 3n.$$

$$\begin{aligned} & \downarrow \\ &= n^{1 - \frac{1}{2^k}} T(n^{\frac{1}{2^k}}) + kn. \\ &= \frac{n}{n^{Y_2 k}} T(n^{Y_2 k}) + kn. \end{aligned}$$

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

$$\frac{n}{2} \times 2 + kn$$

$$= n + n \cdot \log(\log n)$$

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

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

$$\frac{1}{2^k} \log_2 = 1$$

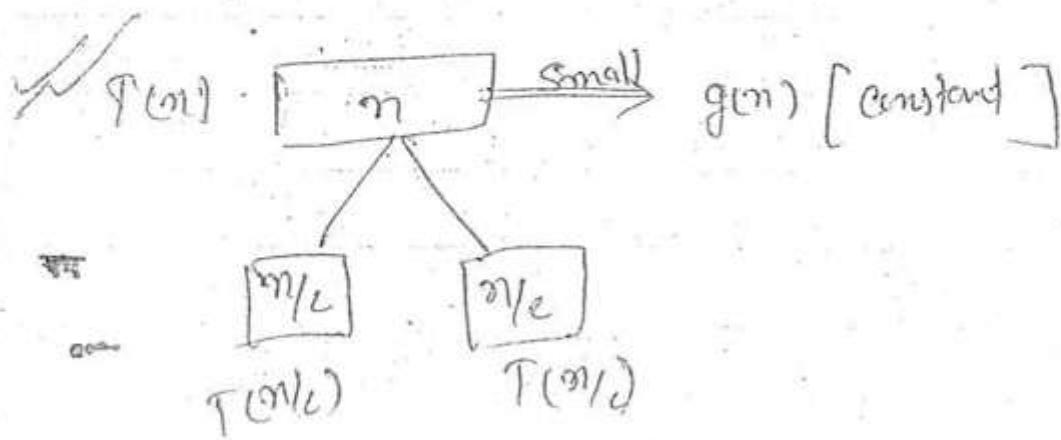
$$\log_2 = 2^k$$

$$\log(\log n) = k + \log_2^2$$

$$\log(\log n) = k.$$

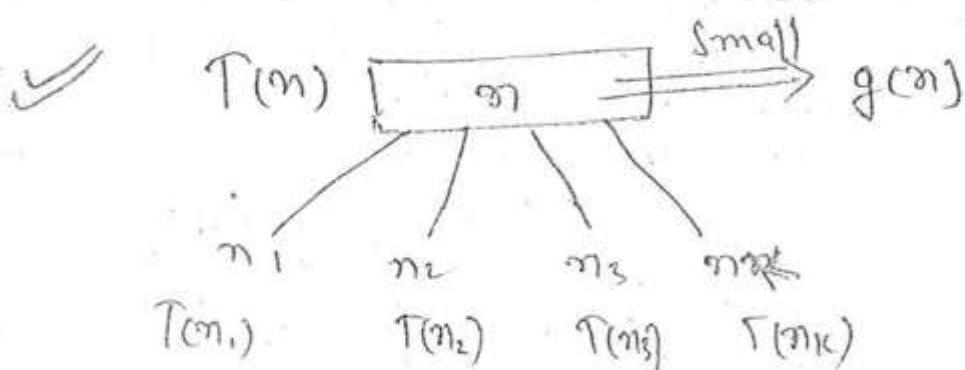
Divide and Conquer (DAC)

- (I) Divide
- (II) Conquer
- (III) Combine



✗ $T(n) = \begin{cases} g(n) & \text{if } n \text{ is small} \\ T(n/2) + f(n) + f_n \\ 2T(n/2) + f_n & \text{if } n \text{ is big} \end{cases}$.

$f(n)$ is time required
Divide / Combine



$$T(n) = \begin{cases} g(n) & \text{if } n \text{ is small} \\ T(n_1) + T(n_2) + T(n_3) + \dots + T(n_k) + f(n) & \end{cases}$$

Divide

Divide the given problem into small sub problem.

Conquer

Conquer the subproblem to get solution recursively,

If the subproblem is small then return solution recursively.

Combine

Combine the subproblem solution to get original problem soln.

Control abstraction of DAC

```
DAC(p,q)
{
    if (small(p,q))
        return (solution(p,q))
```

E/M

```

    {
        m = Divide (p, q)
    }
    Return
    Comb. Inv (DAC (p, m) , DAC (m+1, q))
}
}

```

Applications of DAC

- Appn (1) Power of an element
- (2) Finding max & min in given array.
- (3) Binary search.
- (4) Merge sort.
- (5) Quick sort.
- (6) Selection procedure.
- (7) Strassen's Matrix multiplication.

① Power of an Element

i/p: element $a \geq 1$ & integer $n \geq 0$

o/p : Find a^n

without DAC

Power(a, n)

{

int f=1, i;

for (i=1, i<=n, i++) Time Complexity = $\Theta(n)$

f = f*a;

return (f);

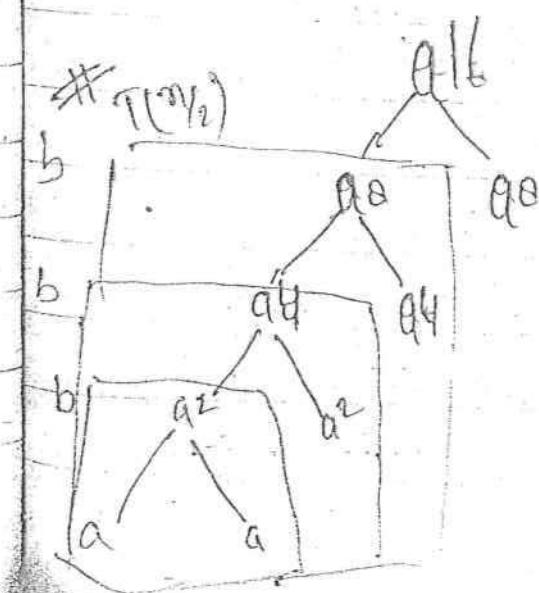
}

#

$$a^n = a^{n/2} \cdot a^{n/2}$$

↓

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



Using DAC

Power(a, n)

```
{  
    int f ; mid ;  
    if( n == 1 ) return a ;  
    else  
    {  
        mid = n / 2  
        f = Power(a, mid )  
        f = f * f  
        return (f)  
    }  
}
```

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

$$\begin{aligned} T(n) &= T(n/2) + c \\ &= T(n/2^2) + c + c \\ &= T(n/2^3) + c + c + c \\ &\vdots \\ &= T(n/2^k) + k \cdot c \end{aligned}$$

$$= T(1) + C \cdot \log n$$

$$\Rightarrow O(\log n)$$

KC



$C \cdot \log n$



$O(\log n)$

L Finding maximum and minimum in the given array
of n elements

i/p An Array of n -elements.

o/p Find max
min.

eg:-

i/p A: [10, 20, 30, 1, 2, 3, 11, 21, 31]
1 2 3 4 5 6 7 8 9.

o/p Max = 31

min = 1.

Without Divide and Conquer.

Straight maximum (a, n, max, min)

{
int i;
Max = min = a[1];

```
for (i=2 ; i<=n ; i++)
```

```
{  
    if (a[i] > max)
```

```
        max = a[i];
```

```
    else
```

```
        if (a[i] < min)
```

```
            min = a[i]
```

```
}
```

```
return [max, min]
```

Comparisons

Best Case : $\boxed{1(n-1)}$

Time Complexity

$\overline{\Theta(n)}$

Worst case : $\boxed{2(n-1)}$

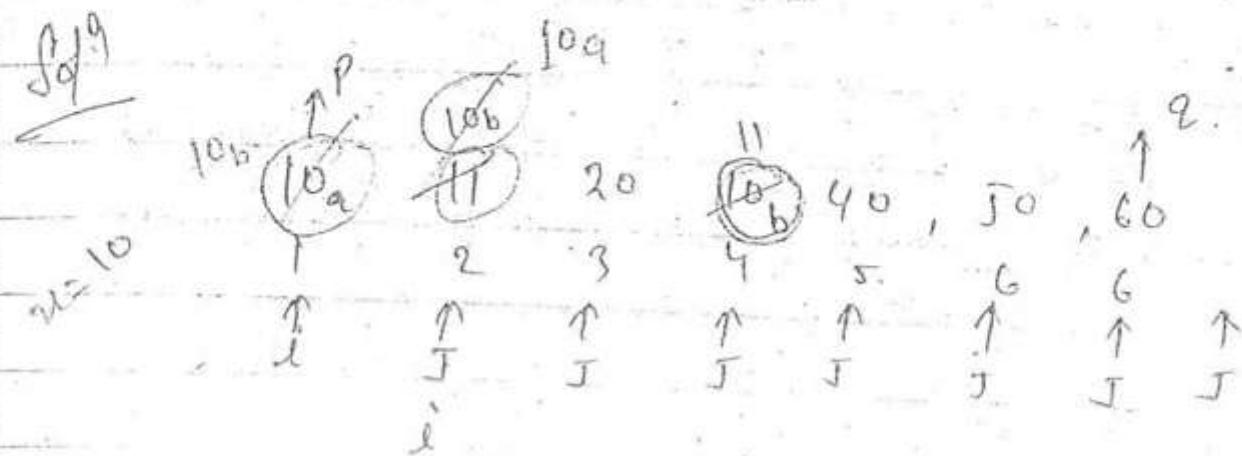
Avg Case : $\frac{n-1}{2} + \left(\frac{n-1}{2}\right) * 2$

$$= \frac{n-1}{2} + n - 1 \Rightarrow \frac{3}{2}(n-1)$$

$$= 1.5(n-1)$$

Q Apply partition algo on following algo.

10, 11, 20, 10, 40, 50, 60



Answer

come

(10b) 10a (20, 11, 40, 50, 60)

stable sorting Tech

The relative ordering of repeated elements not changed not changed after sorting the that sorting is called.

stable sorting technique.

Q.S is not stable sorting.

Selection procedure

i/p: An array of n elements and K

O/P: Find K^{th} smallest element

eg

i/p: A $[10, 20, 30, 1, 2, 3, 11, 21, 31]$ $K=4$

O/P = 10 $K=4$ | 11 $K=5$ | 31 $K=9$

A. ① 1, 2, 3, (10), 11, 20, 21, 30, 31 $\Rightarrow O(n \log n)$
1 2 3 4 5 6 7 8 9

② Go to location $a[K] \Rightarrow O(1)$

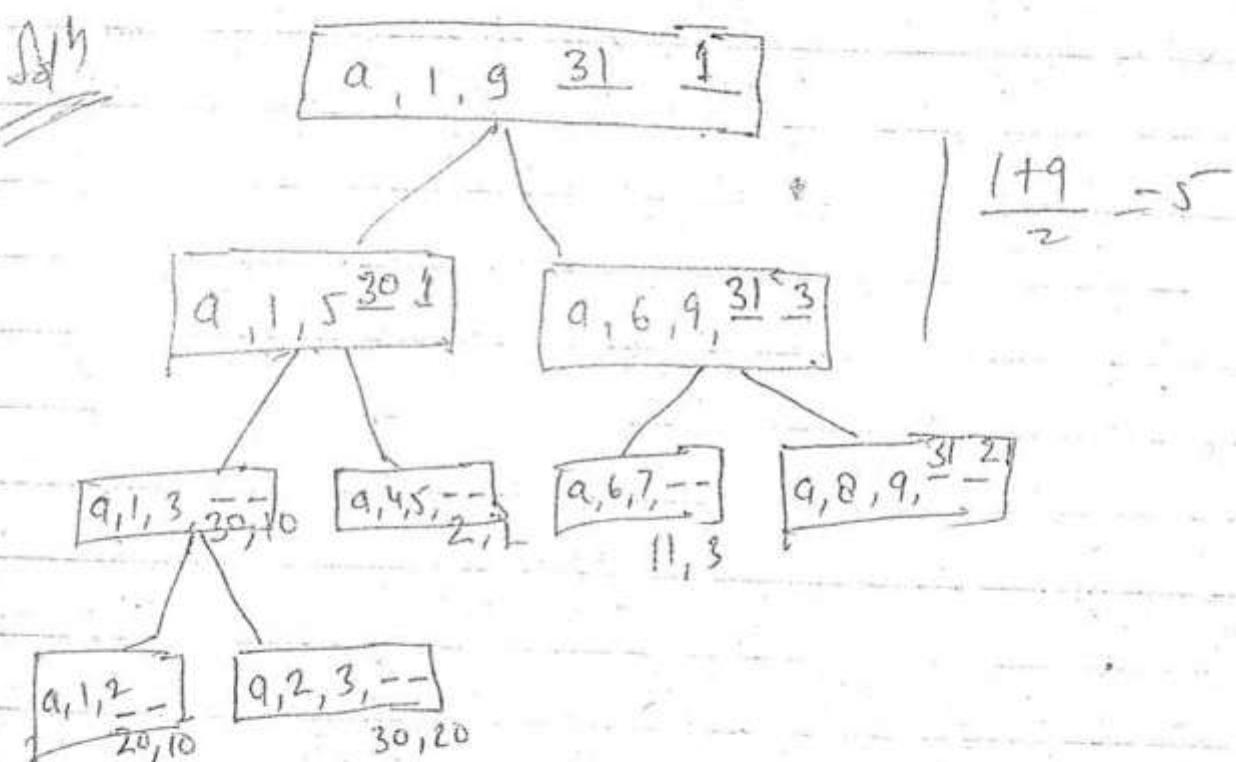
$O(n \log n)$

Using DAC Max & min.

VP : A [10, 20, 30, 1, 2, 3, 11, 21, 51]
1 2 3 4 5 6 7 8 9

Max : 51

min : 1



DACmaxmin(a, i, J, max, min)

Small

```
{ int mid ,  
  if ( i == J )  
    { max = min = a[ i ] ;  
     return ( max , min ) ;  
    }  
  
  if ( i == J - 1 )  
  {  
    if ( a[ i ] < a[ J ] ) max = a[ J ]  
      min = a[ i ]  
    else  
      max = a[ i ] , min = a[ J ] ;  
  }  
}
```

else
{ mid = (i + J) / 2
 DACmaxmin (a, i, mid, max, min)
 DACmaxmin (a, mid + 1, J, max, min)
 }

C-constant

$T(n/2)$

$T(n/2)$

if ($\max_1 < \max_2$)

$\max = \max_2$

else

$\max = \max_1$

if ($\min_1 > \min_2$)

$\min = \min_2$

else

$\min = \min_1$

return (\max, \min)

3

Let $T(n)$ be the number of comparisons required to find max & min.

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

\Downarrow

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

$$\# \Gamma(n) = 2\Gamma(n/2) + 2$$

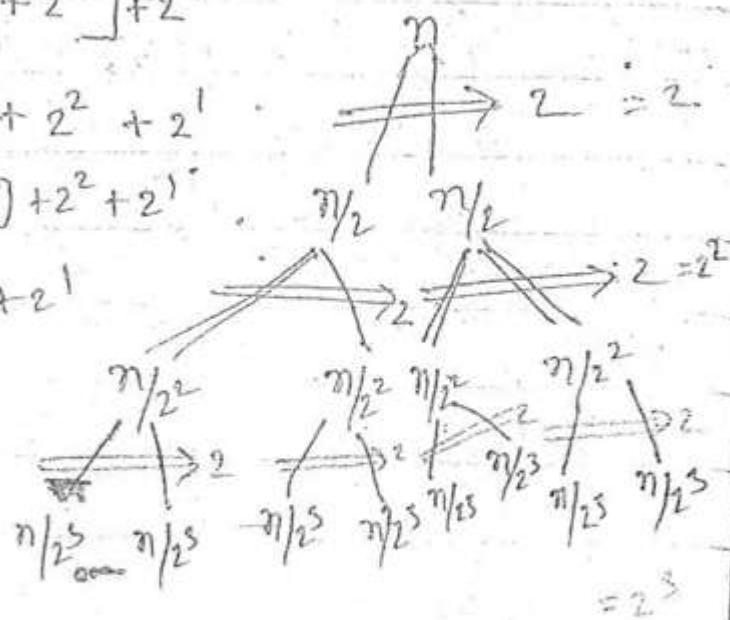
$$= 2[2\Gamma(n/2) + 2] + 2^1$$

$$= 2^2 \Gamma(n/2) + 2^2 + 2^1$$

$$= 2^2 [2\Gamma(n/2) + 2] + 2^2 + 2^1$$

$$= 2^3 \Gamma(n/2) + 2^3 + 2^2 + 2^1$$

1
1
1
↓



$$= 2^{k-1} \Gamma(n/2^{k-1}) + [2^1 + 2^2 + 2^3 + \dots + 2^{k-1}]$$

$$= \frac{n}{2} \times 1 + \left[\underbrace{2(2^{k-1}-1)}_1 \right]$$

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

$$= \frac{3n}{2} - 2$$

$$= 1.5n - 2$$

$$= O(n)$$

\Downarrow It is best, worst & Avg case.

Note - Using DAC and without DAC Taking same time . So we have to consider space more, ~~$T(n) = n + 1$~~

Now using DAC recursive program take more space compare to non-recursive program.

Binary Search

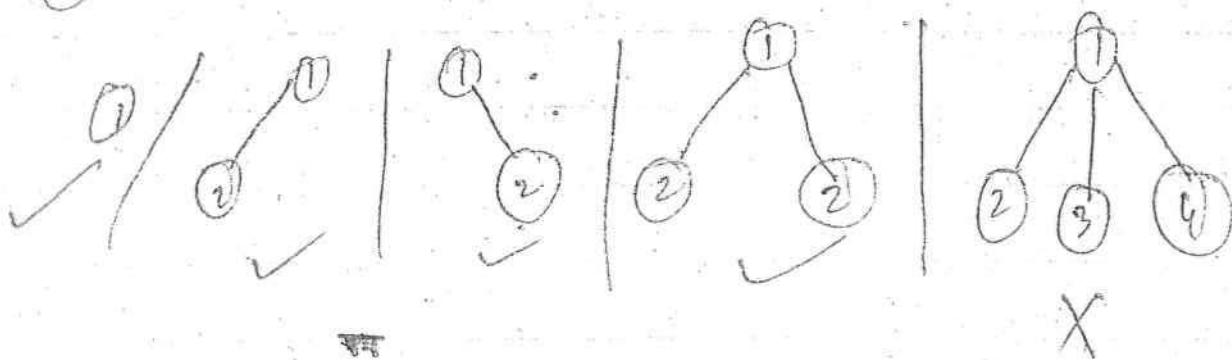
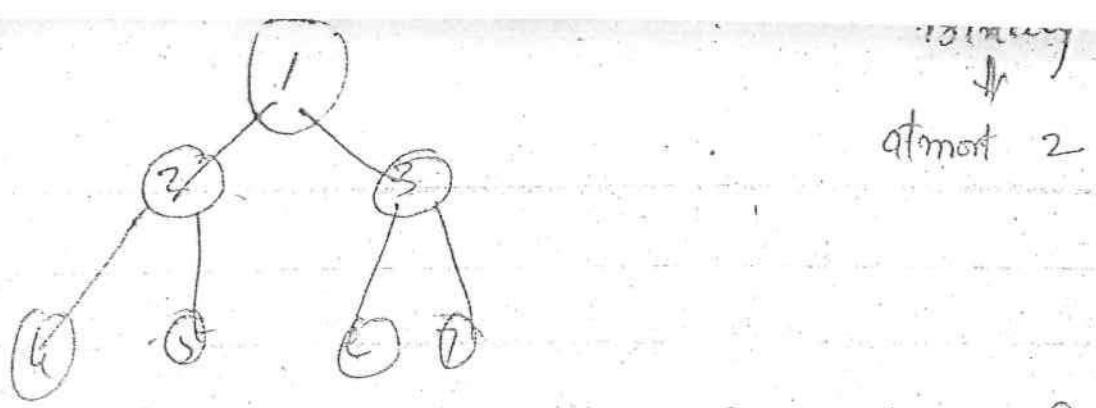
Graph

- 1 - no-root
- 2 - graph may be directed / undirected
- 3 - may contain cycle / non cycle
- 4 - May be connected / disconnected

Tree (DAH) Directed acyclic graphs

- (i) root
- (ii) Directed
(Top \rightarrow Bottom)
- (iii) no cycle

- (iv) Connected

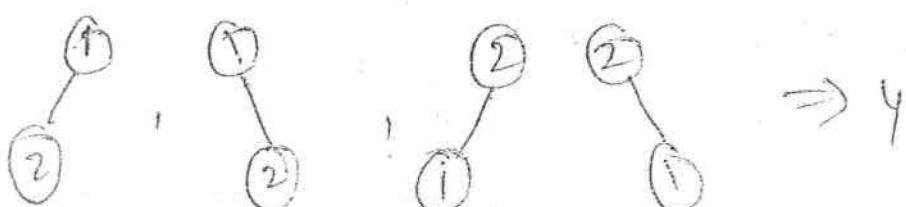


A Tree is a set of binary tree iff at every node max^m 2 children is present.

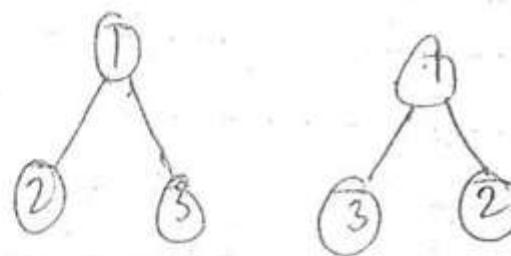
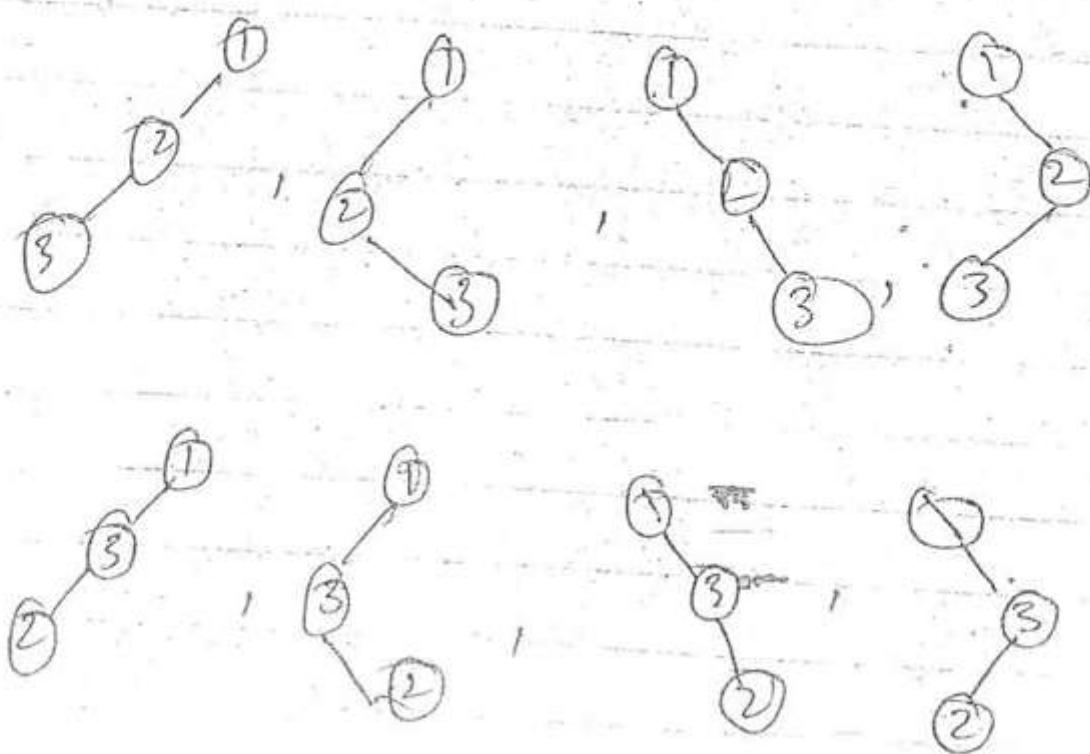
Q. Find no of binary trees with n=3 (1, 2, 5)

$$\text{Sol} \quad \begin{array}{c} \underline{n=1} \\ 1 \Rightarrow 1 \end{array}$$

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



$$n=3 \quad (1, 2, 3)$$



$$\begin{aligned} 1 &- 1^0 \\ 2 &- 1^0 \\ 3 &- 3^0 \end{aligned}$$

$$\Rightarrow \underline{\underline{3^0}}$$

the no of binary trees with n nodes

$$n \Rightarrow n! \frac{(2^n)}{n+1}$$

By formula

if $n=3$

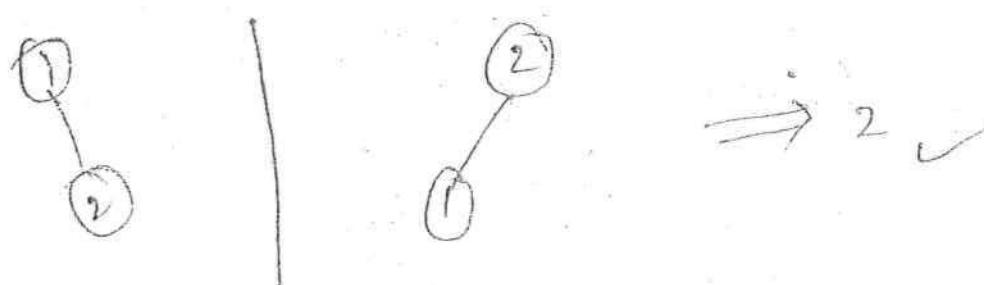
$$\frac{3! \times 6c_3}{4} = \frac{6 \times 6 \times 5 \times 4}{1 \times 2 \times 3 \times 4} \\ = 30$$

Binary Search Tree

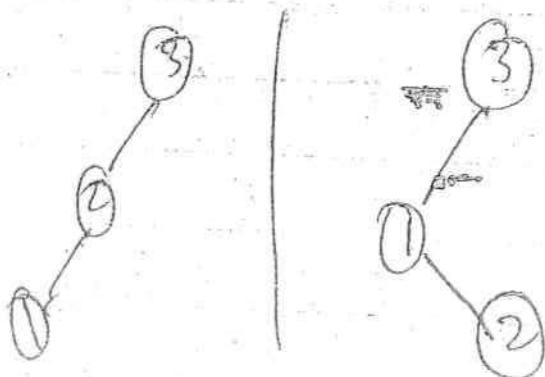
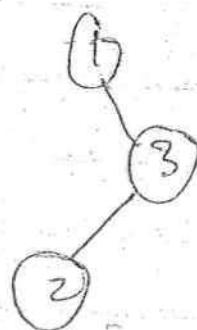
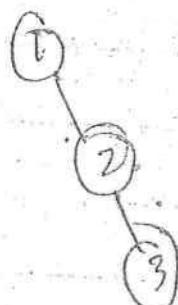
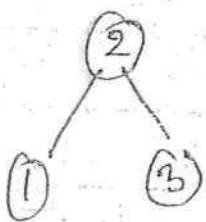
(Repetition not allowed)

- (a) At every node Root is greater than all the elements present in left sub tree (LST)
- (b) At every node Root is smaller than all the elements present in RST.

e.g. $n=2 (1, 2)$



$$n = 3 \quad (1, 2, 3)$$



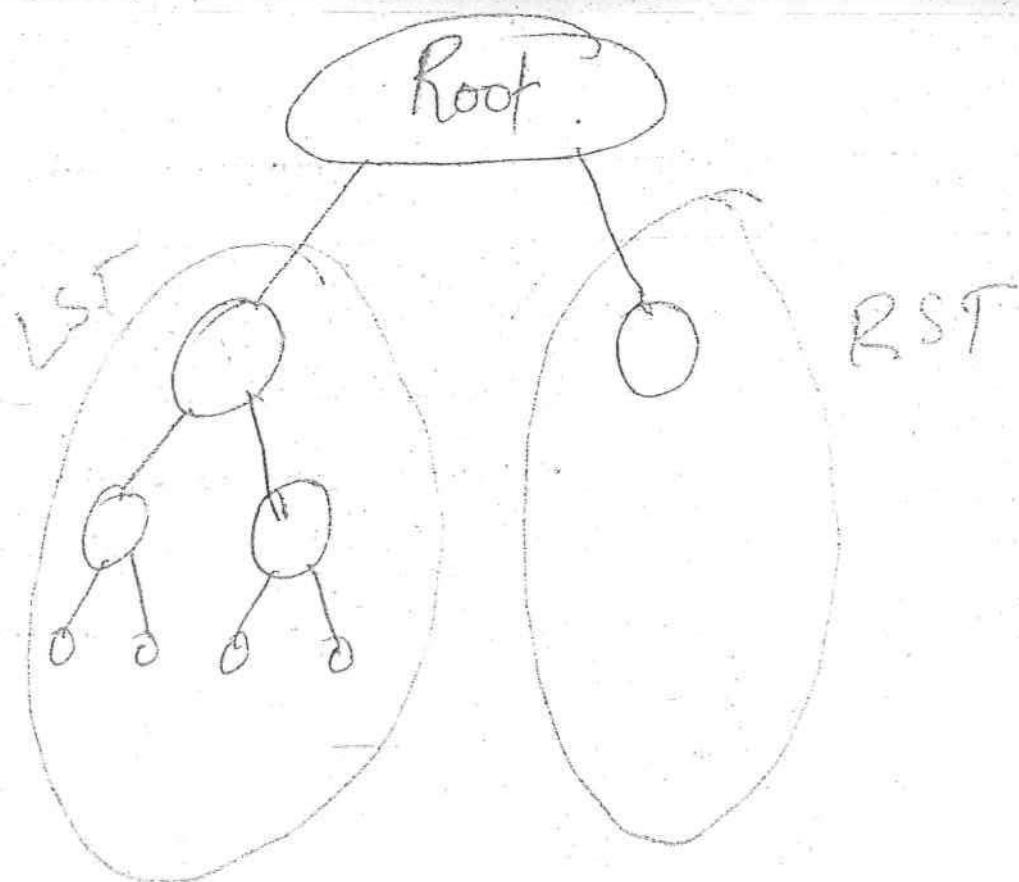
$$\text{binary Search Tree} = \frac{(2^n)c_n}{n+1}$$

$$n=1 \Rightarrow 1$$

$$n=2 \Rightarrow 2$$

$$n=3 \Rightarrow 5$$

$$n=n \Rightarrow \frac{(2^n)c_n}{n+1}$$



Searching

Without Divide and Conquer

I/P An array n -elements, element x .

O/P Find position of x , if x is present
otherwise return (-1)

Soln

Ex
I/P $A[10, 20, 30, 1, 4, 3, 11, 21, 31]$ $x=11$

O/P : 7 | $x=11$ ✓

9 | $x=31$ ✓

1 | $x=10$ ✓

-1 | $x=90$ ✓

Without DAC

Linear Search

linearSearch(a, n, x)

```
{  
    int i;  
    for (i=1, i<=n, i++)  
    {  
        if ( $a[i] == x$ )  
            return (i);  
    }  
    return (-1);  
}
```

Best case - $\mathcal{O}(1)$

Worst case $\rightarrow \mathcal{O}(n)$

Avg. case $\rightarrow \frac{n+1}{2} = \mathcal{O}(n)$

Successful search

Unsuccessful Search

Best case - $\Omega(n)$

Worst - $O(n)$

Avg. - $O(n)$

If
 $O(n)$

Binary Search (using DAC)

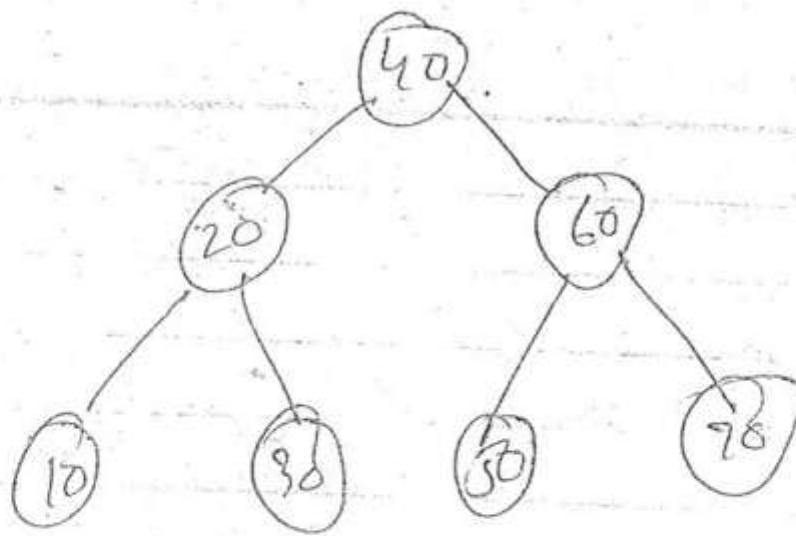
i/p = A sorted array of n-elements, elements n.

o/p return position of x if x is present else
return (-1)

eg

i/p: A[10, 20, 30, 40, 50, 60, 70] x=40
1 2 3 4 5 6 7

Soln.



$$\log_2 7 = 12.83$$

= 3 Comparisons.

Binary Search (a, i, j, u)

```

    int mid;
    if (i == -1)
        {
            if (a[i] == u)
                return(i);
            else
                return (-1);
        }
    }
```

Binary

C1

```

else
{
    mid = (i+j)/2
    if (a[mid] == x)
        return (mid)
    else
    {
        if (a[mid] > x)
            BinarySearch(a, i, mid-1, x)
        else
            BinarySearch(a, mid+1, j, x)
    }
}

```

6

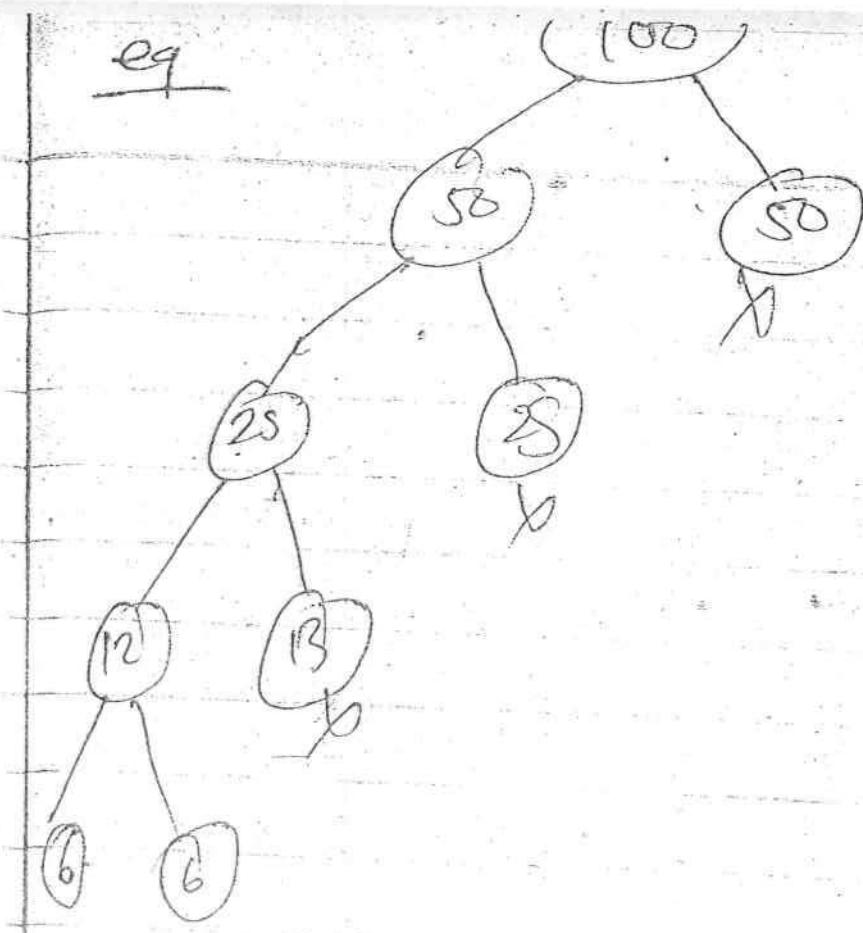
Recurrence Relation

Let $T(n)$ be the time complexity to do binary search with DAC

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

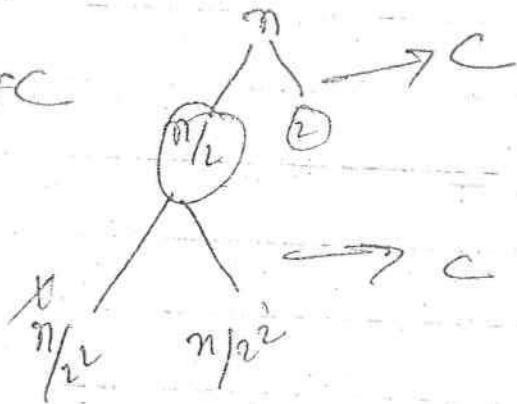
\downarrow

$O(\log n)$

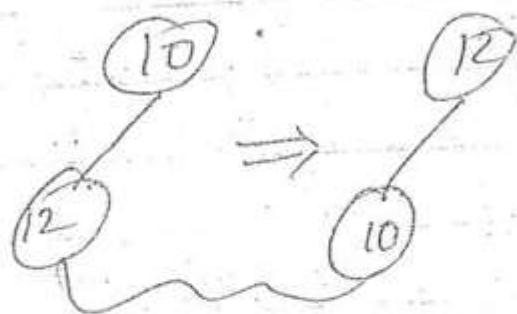


$$T(n) = T(n/2) + C$$

$$= T(n/2) + C \quad FC$$

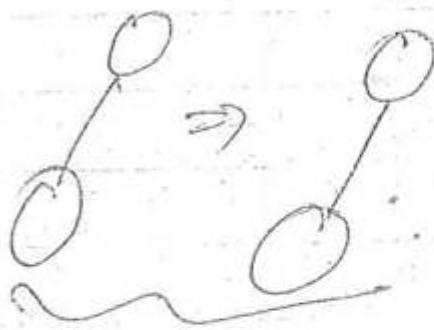


Unlabeled Binary tree



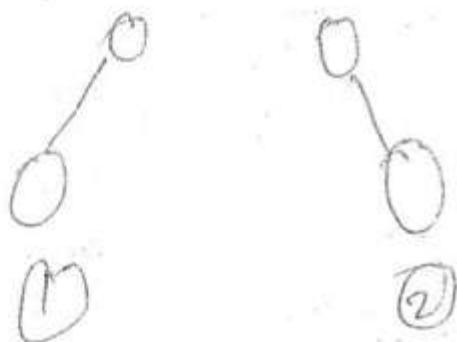
2 - different unlabeled tree

Labeled B.P



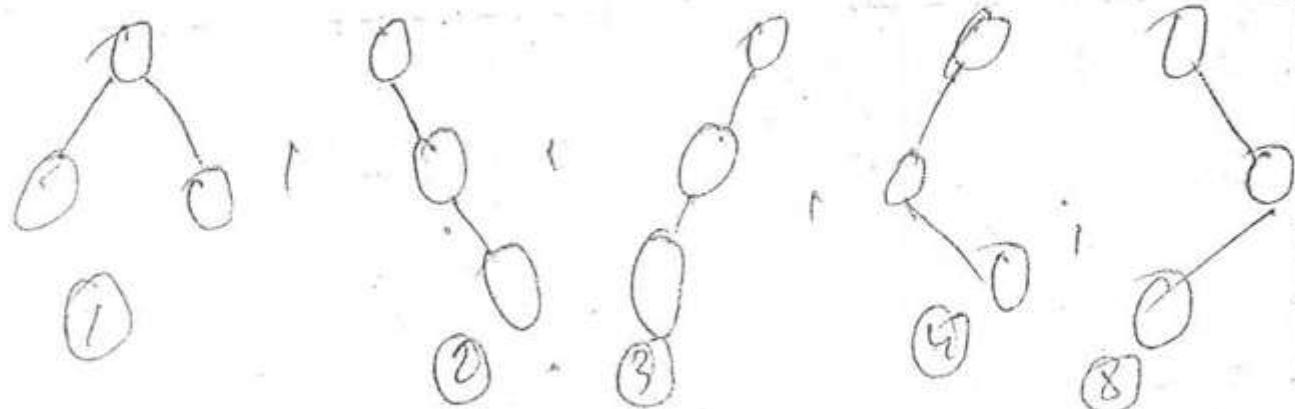
both are same.

P/ $n=2$ how many different unlabeled B.P.



2

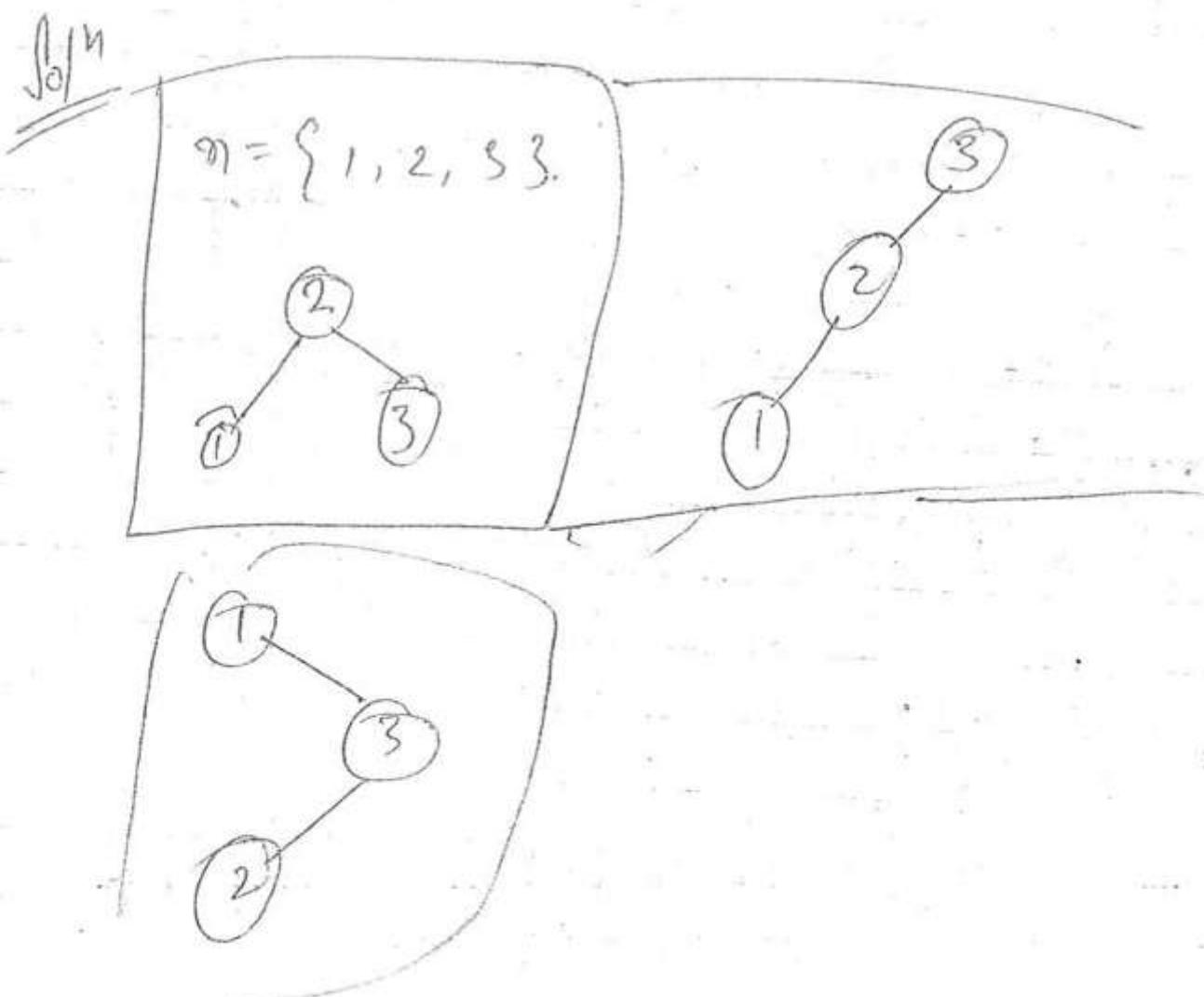
$n=3$



= 5

Q. Here given set of n distinct elements and an unlabeled binary tree with n nodes. In how many ways can we populate the tree with the given set. So that it becomes a BST.

- a) 0
- b) 1
- c) $n!$
- d) $\frac{(2n)!}{n+1}$



Q
A binary search tree is given to locate no 43, which one of the following sequences are possible?

a) 61, 52, 14, 17, 40, 43

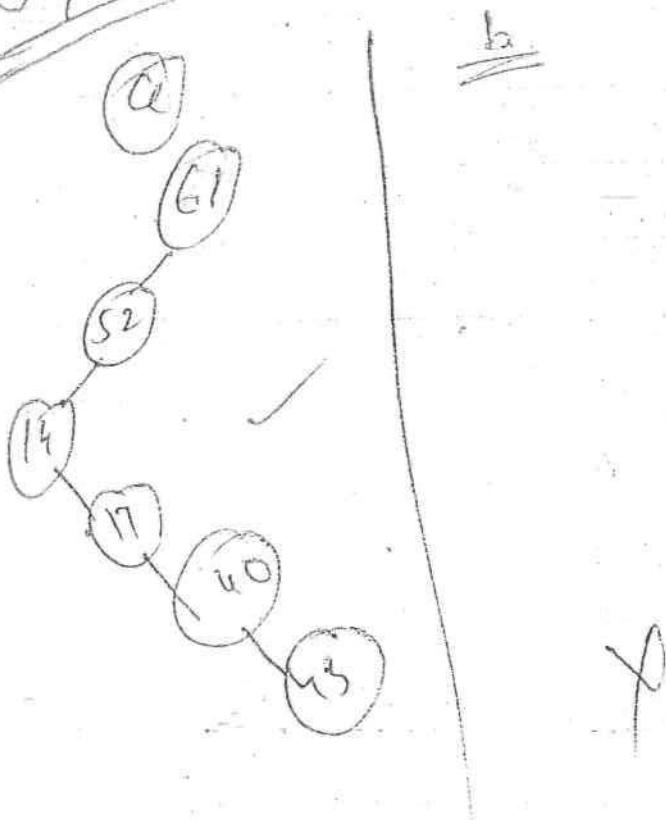
b) 2, 3, 50, 40, 60, 43

c) 10, 65, 31, 47, 8, 37, 45

d) 91, 61, 52, 14, 40, 43

e) 17, 20, 27, 66, 18, 43

Sol:



(Linear Binary Search)

$\Omega(1)$ - Best case

$O(n)$ - Worst case

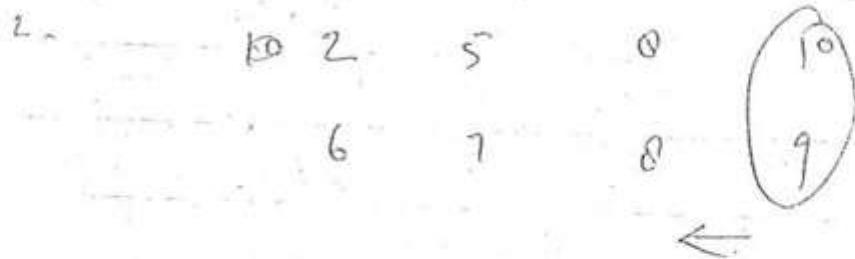
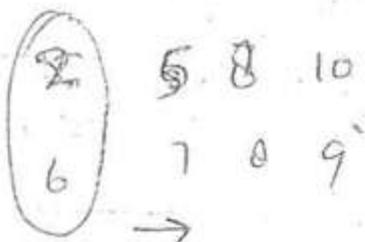
$\Theta(n)$ - Avg case

Binary Search

means always goes left or right.

If Postion is greater than element we go right side.
and if " " is smaller " " then we go left side.

eg -



ele < Postion \Rightarrow right

ele > Postion \Rightarrow left

$\Omega(n)$ — Best Case

$\Theta(\log n)$ — Worst "

$\mathcal{O}(n \log n)$ — Avg Case

#

I/P : An array n -elements

O/P : Find repeated elements.

eg.

I/P. A : 10, 20, 30, 1, 2, 3, 10, 20, 30
1 2 3 4 5 6 7 8 9

O/P : 10, 20, 30

(A1)

(i) Apply Linear Search on every element n to find n
 $\Rightarrow \mathcal{O}(n^2)$

(A2)

(ii) Sort the given array by taking
 $\Rightarrow \mathcal{O}(n \log n)$

1, 2, 3, 10, 10, 20, 20, 30, 30

(ii) Scan the array to check

$$a[i] = a[i+1] \text{ for all } i$$

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

$$n = 1000$$

$$n^2 = 10,000,000$$

$$n = 1000$$

$$n \log n = 1000 \times 3$$

$$= 3000$$

A3

(1)

for ($i=1, i \leq n, i++$)

$$\text{flag}[a[i]] = 0 \rightarrow O(n)$$

flag:

0	0	0	...	0	-	-	0	/	0
1	2	3		10.			20		30

(2)

for ($i=1, i \leq n, i++$)

{ if ($\text{flag}[a[i]] = 0$) }

$\text{flag}[a(i)] = 1$
 else
 $\text{Point}(a(i)) \Rightarrow O(n)$
 }
 $\{$

$$\Rightarrow O(n) + O(n) = O(n)$$

#

i/p : An array N-bolts &
another array of M-nuts

o/p : find all matching bolt & nut pair.

eg

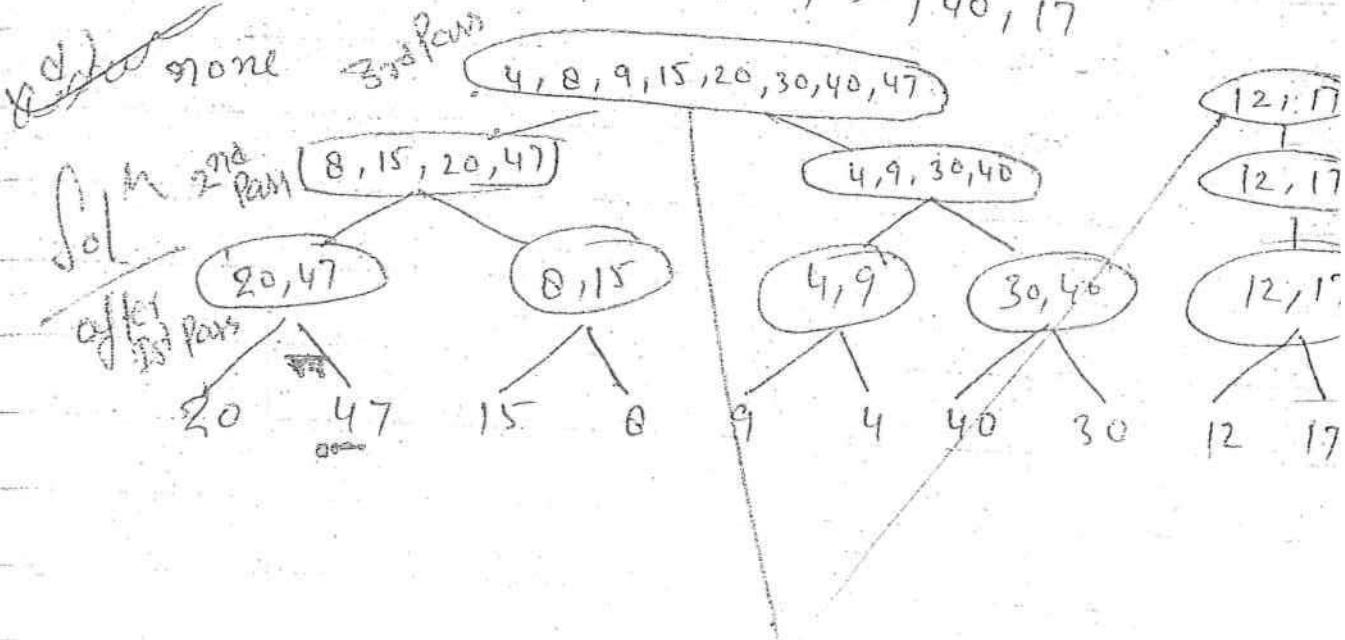
i/p B : $b_1, b_2, b_3, b_4, \dots, b_n$
 N : $n_1, n_2, n_3, n_4, \dots, n_m$

(A1) Apply linear search to find matching bolt
and nut pair on every bolt.

$$\Rightarrow O(mn)$$

b) ~~0, 15, 20, 47, 4, 9, 30, 40, 12, 17~~

c) 15, 20, 47, 4, 0, 9, 12, 30, 40, 17



offer forth pass

$$\text{order} = \log_2 12 = 4$$

Quick Sort (Using divide)

- (1) Divide and Conquer.
- (2) It is one of the In-place.
- (3) Very practical.

Algo :

- (1) Divide the given problem into subproblem using pivot element. \Rightarrow Partition Algo
- (2) Conquer the subproblem recursively to get solutions of subproblems.

Partial Divide and Conquer approach.

Partition (a, p, q)

```
{  
    u = a[p];  
    i = p;  
    for (j = p+1; j <= n, j++)  
        {  
            if (a[j] <= u)  
                i = i + 1  
                swap (a[i], a[j])  
        }  
}
```

A₂

i) Sort the nut array in ascending order.
 $\Rightarrow O(m \log m)$

ii) For every element of bolt array apply binary search on nut array to find matching pair.
 $\Rightarrow O(m \log n)$

$$O(m \log m) + O(m \log n) \Rightarrow O(m \log m) + O(m \log n)$$

*

I/P = An Array of n elements.

O/P = Find all leaders.

Note:- An element e is said to be leader iff it is greater than all other elements present on R.H.S of it.

eg

A : 10, 50, 25, 70, 15, 35, 44, 100, 5, 7
1 2 3 4 5 6 7 8 9 0

In this Leader = 7, 100

(A1) Apply linear search to check every element present on R.H.S. of it smaller or not
 $\Rightarrow O(n^2)$

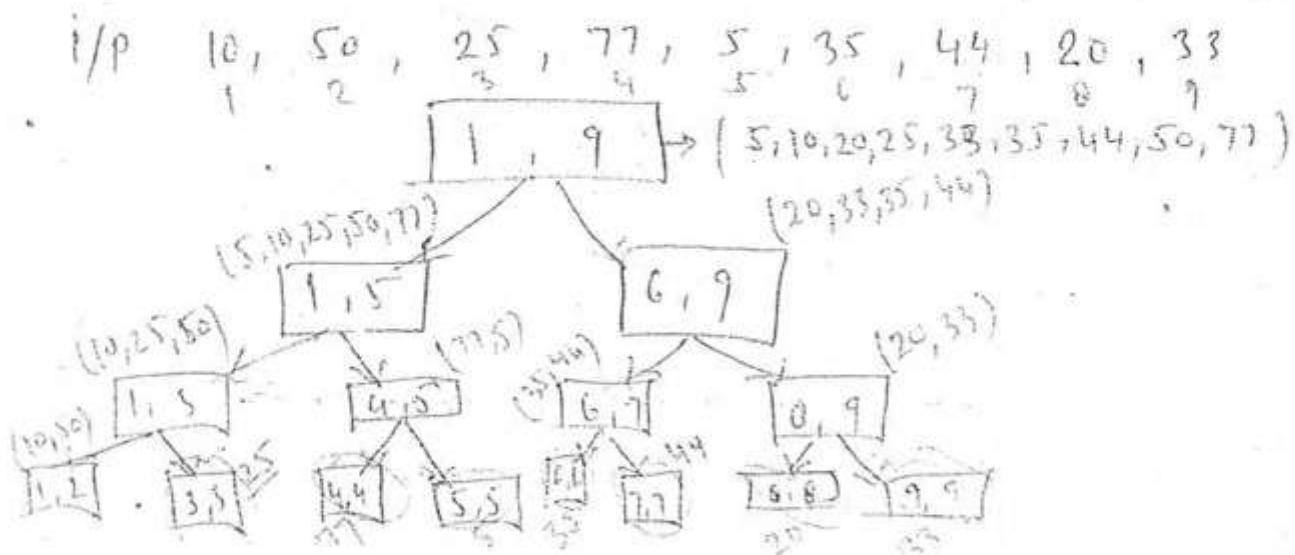
(A2) we can't apply binary search because order of the elements changing.

~~eg.~~ 100, 70, 50, 40, 30, 10

leader = 10, 30, 40, 50, 70, 100.

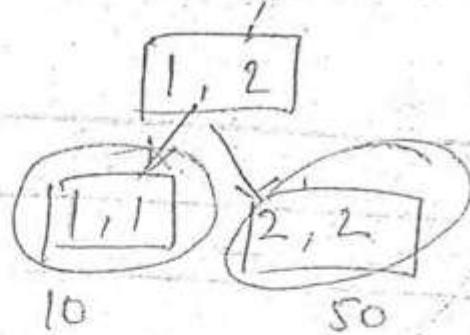
Merge sort

Divide and Conquer. Merge sort.

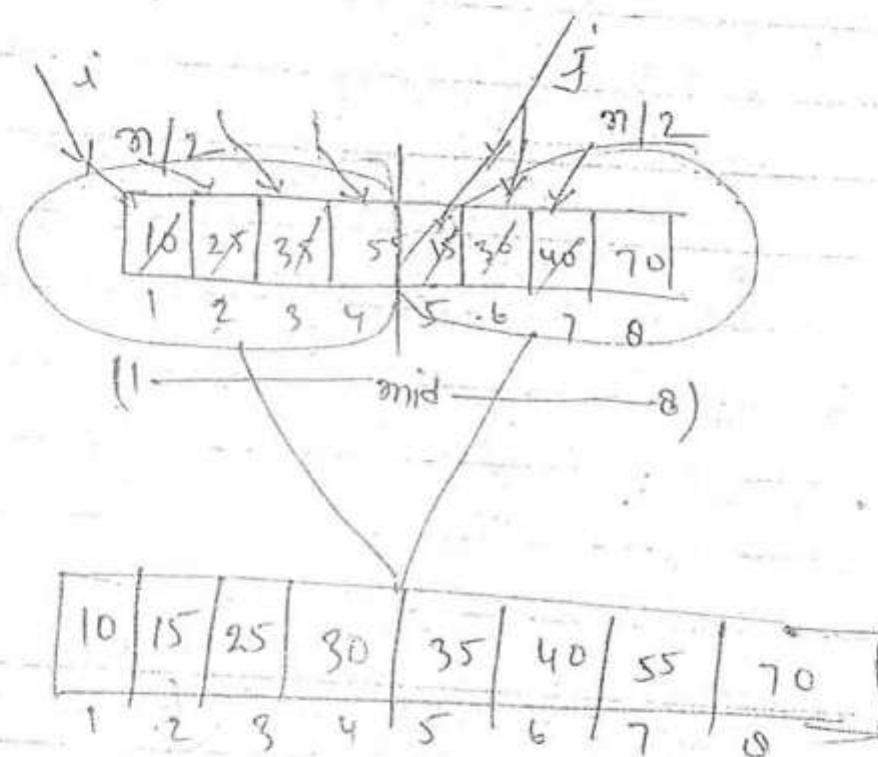


lmet

next



~~Agg~~ - eq



$$(10, 15) \Rightarrow 10$$

$$(25, 15) \Rightarrow 15$$

$$(25, 30) \Rightarrow 25$$

$$(35, 30) \Rightarrow 30$$

$$(35, 40) \Rightarrow 35$$

$$(55, 40) \Rightarrow 40$$

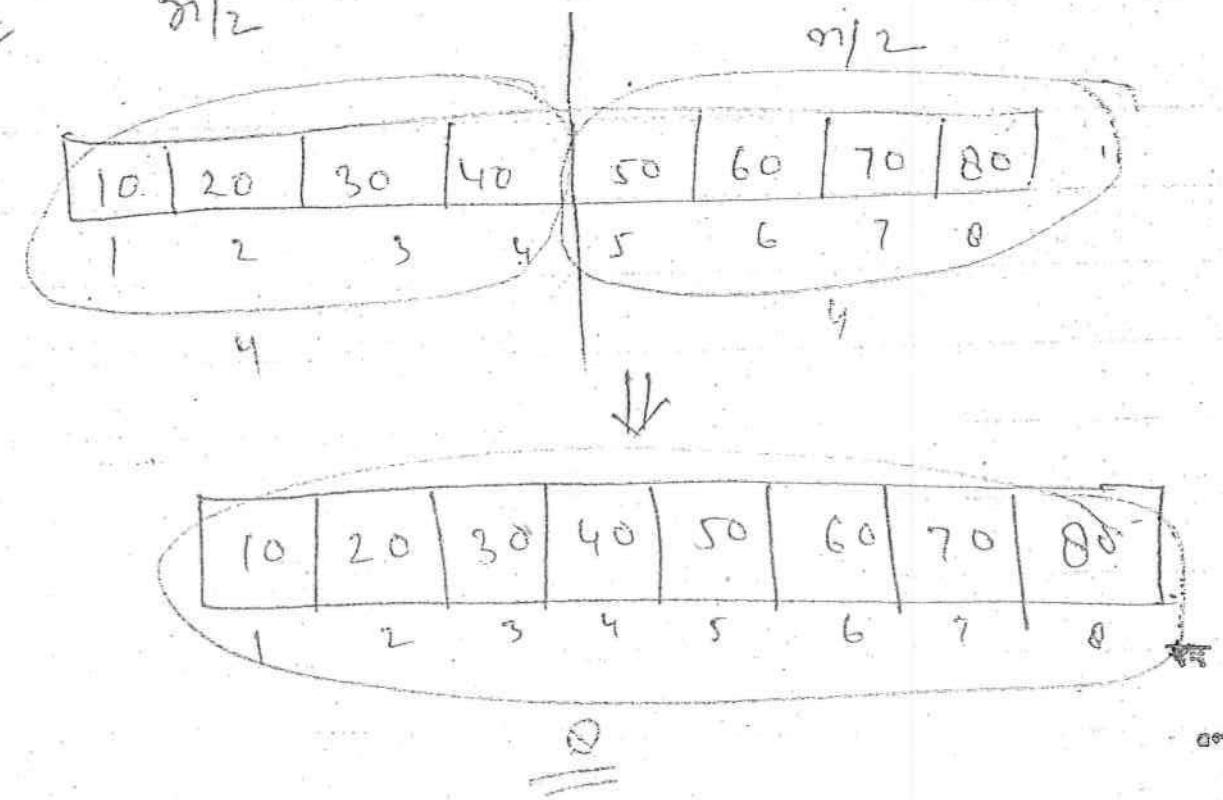
$$(55, 70) \Rightarrow 55$$

$$70 \Rightarrow 70$$

$$\text{total Comparison} = \frac{n+1}{2} = 7$$

$$= \frac{n}{2} + \frac{n}{2} - 1$$

$$= \underline{\underline{n-1}}$$



$$(10, 50) \Rightarrow 10$$

$$(20, 50) \Rightarrow 20$$

$$(30, 50) \Rightarrow 30$$

$$(40, 50) \Rightarrow 40$$

Best Case $\Rightarrow n/2 \Rightarrow O(n)$

Worst Case $\Rightarrow n/2 + n/2 - 1 \Rightarrow O(n)$

#

	$m/2$	$n/2$
A:	[10 20 30 40 9 19 29 39]	

1 2 3 4 | 5 6 7 8

Using Divide and Conquer

Mergesort(a, i, j)

{
int mid ;

if ($i == j$) return ($a[i]$)

else

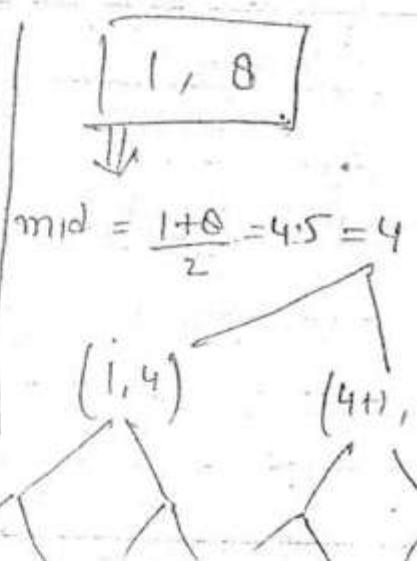
{
mid = $(i+j)/2 \Rightarrow C$

Mergesort(a, i, mid) $\rightarrow T^{(m/2)}$

Mergesort($a, mid+1, j$) $\rightarrow T^{(n/2)}$

Merge(a, i, mid, j)

return (a)



$$mid = \frac{i+j}{2} = \frac{1+8}{2} = 4.5 = 4$$

#

Let $T(n)$ be the time required to sort n elements.

$$T(n) = \begin{cases} c & \text{if } n=1 \\ O(1) + T(n/2) + T(n/2) + \theta(n) \\ \quad \downarrow \\ \quad 2T(n/2) + \theta(n) & \text{if } n>1 \\ \quad \downarrow \end{cases}$$

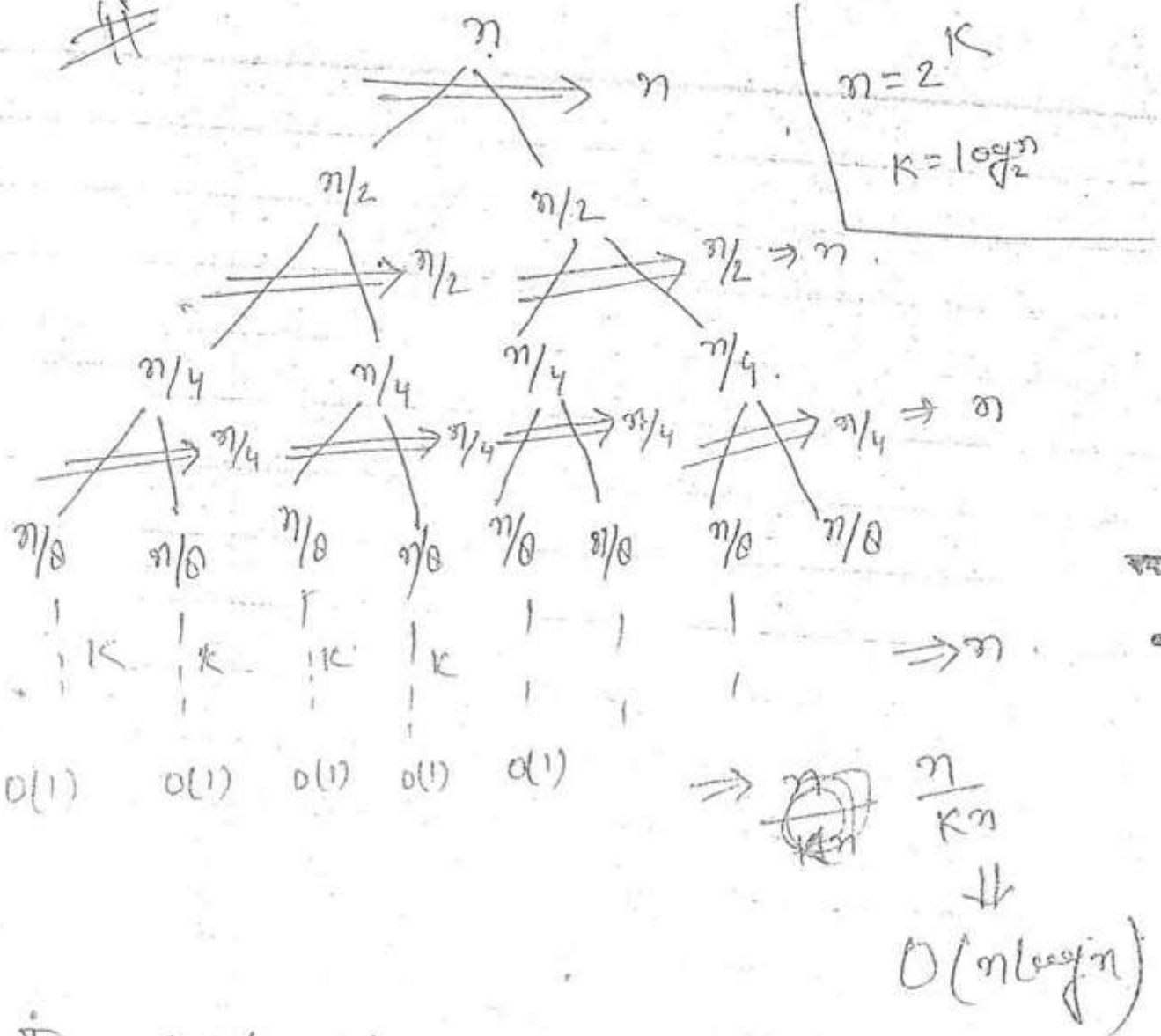
$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2[2T(n/2) + n/2] + n \\ &= 2^2T(n/2^2) + 2n \\ &= 2^3T(n/2^3) + 3n \end{aligned}$$

↓
K

$$= 2^K T(n/2^K) + Kn \Rightarrow n + \log_2 n \cdot n$$

$\Theta(n \log n)$

(1)



Time Complexity of MergeSort.

Best case = ~~O(n)~~ $O(n \log n)$

Worst case = $O(n \log n)$

i.e. Best case = Worst case.

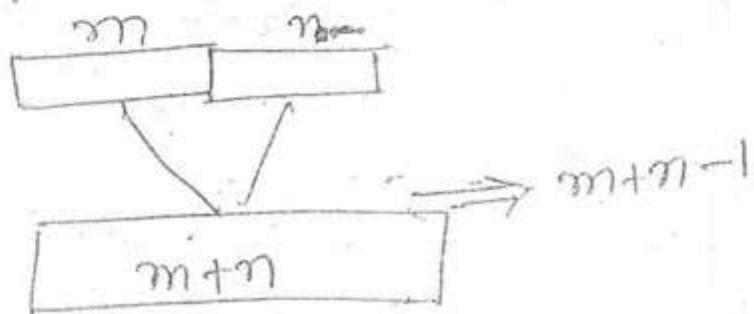
① Merge Sort is preferable for larger size array
for smaller size. Smaller size can be
preferable by insertion sort

②

Merge sort is not in-place sorting because in merge sort process it is taken an extra array of size m .

③

In order to merge 2 sorted subarray of m & n into sorted array $m+n$ elements required $O(m+n)$ times.



Q

If one uses 2-way mergesort algo to sort the following elements in ascending order

20, 47, 15, 0, 9, 4, 40, 30, 12, 17

then

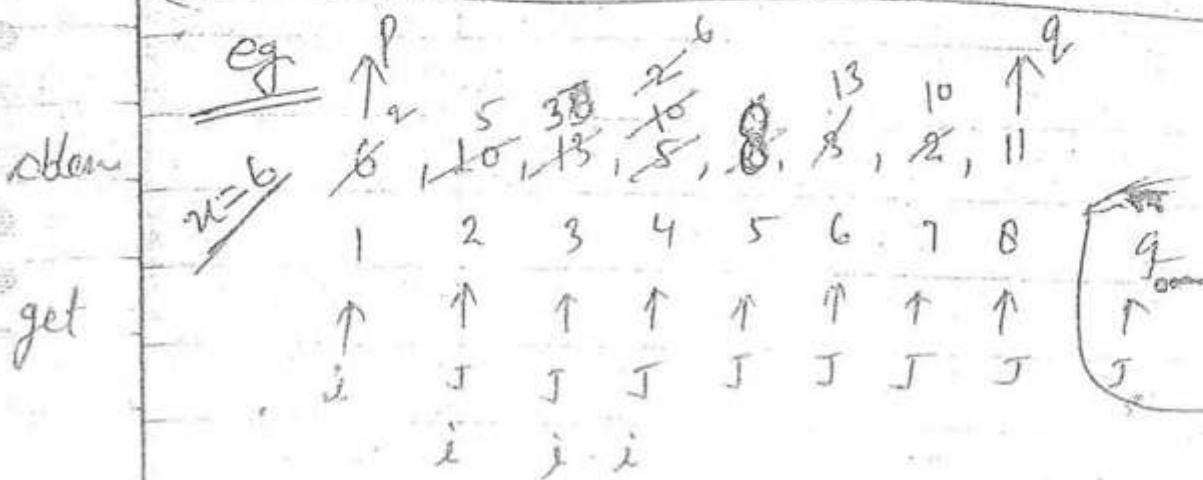
What will be the output after 2nd pass of the algo.

- (a) 8, 9, 15, 20, 47, 4, 12, 17, 30, 40

swap(a[i] , a[p])

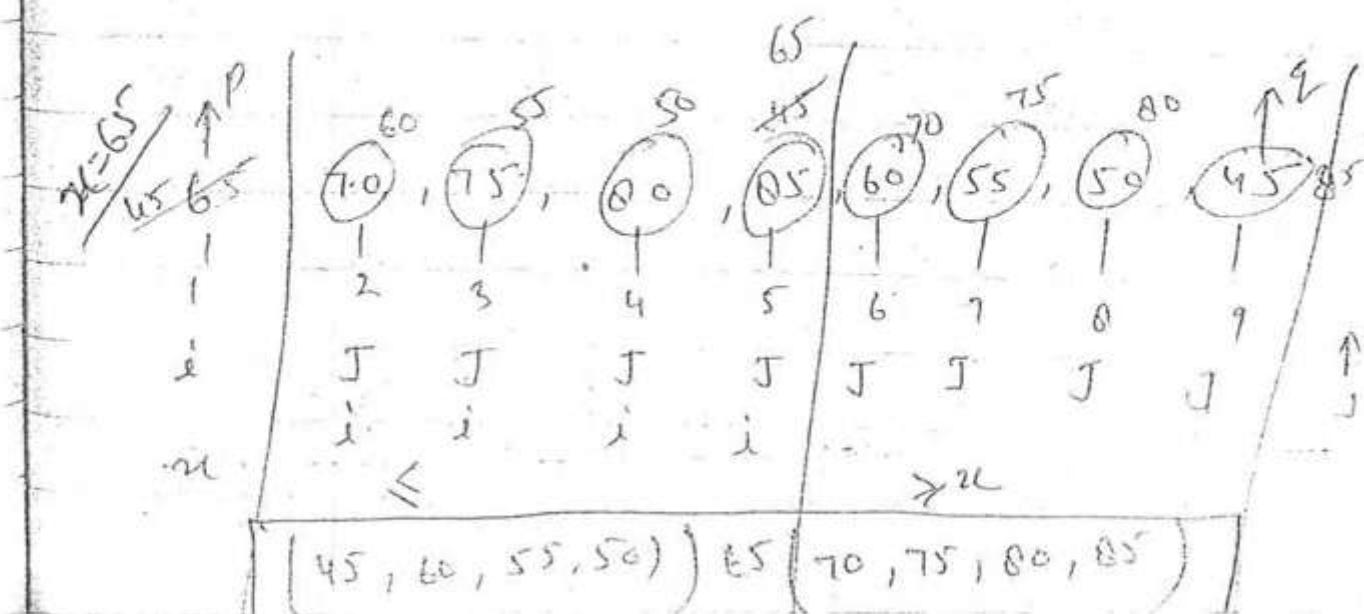
return(i);

}



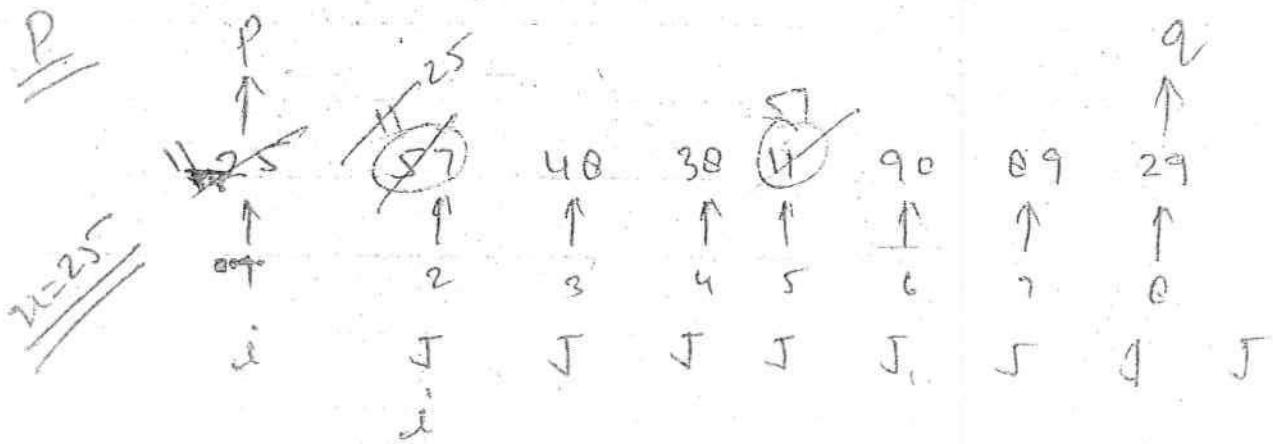
(2 5 3) 6 (8 13 10 11)

eg Apply partition algo to the following input



$\leq u$	$\geq u$	$?_u$	$?$
----------	----------	-------	-----

$i \quad J$



$$(ii) 25 (48, 38, 57, 90, 89, 29)$$

P(1)

10 20 30 40 50 60 70 80 90 100

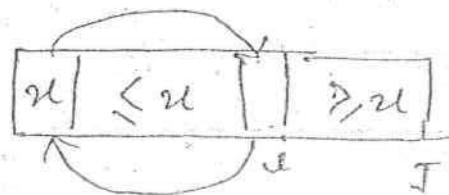
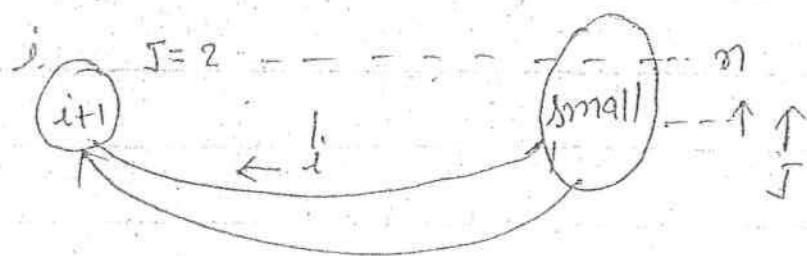
P(2) -

100 90 80 70 60 50 40 30 20 10

100

11/11/11

Partition Algo

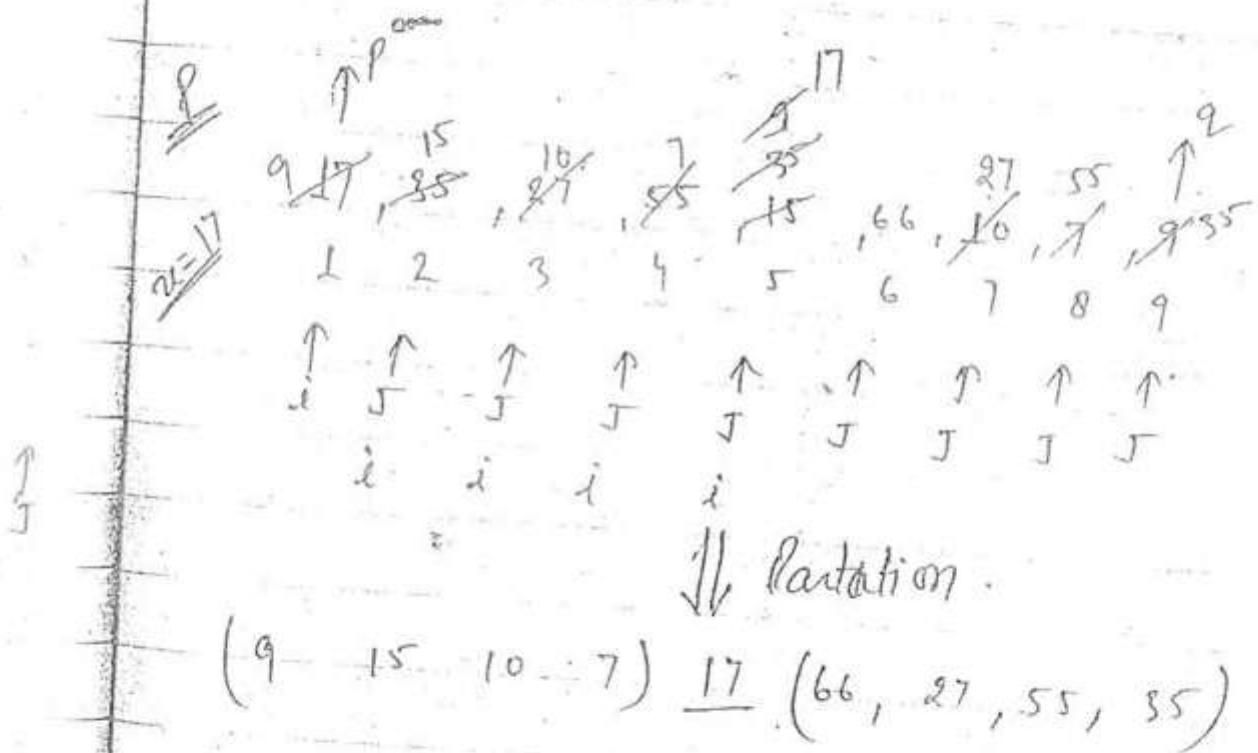
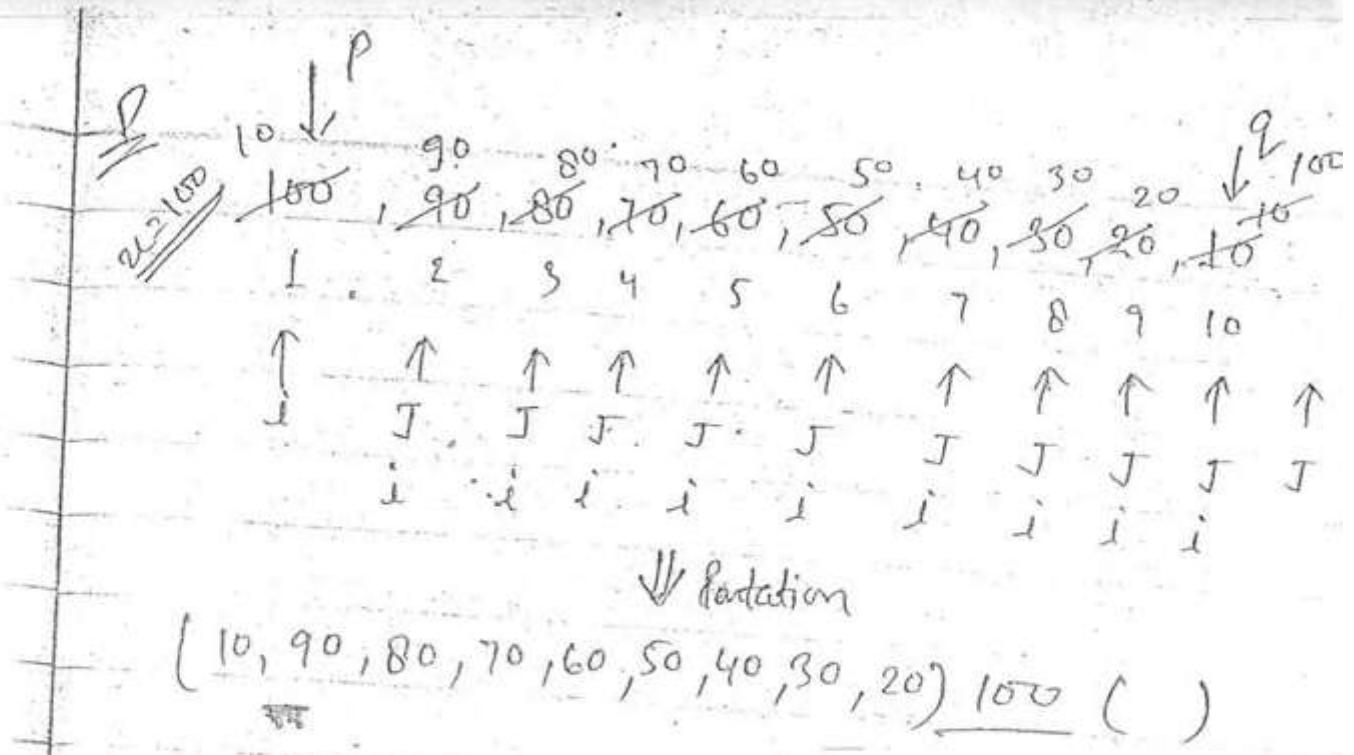


~~n=10~~

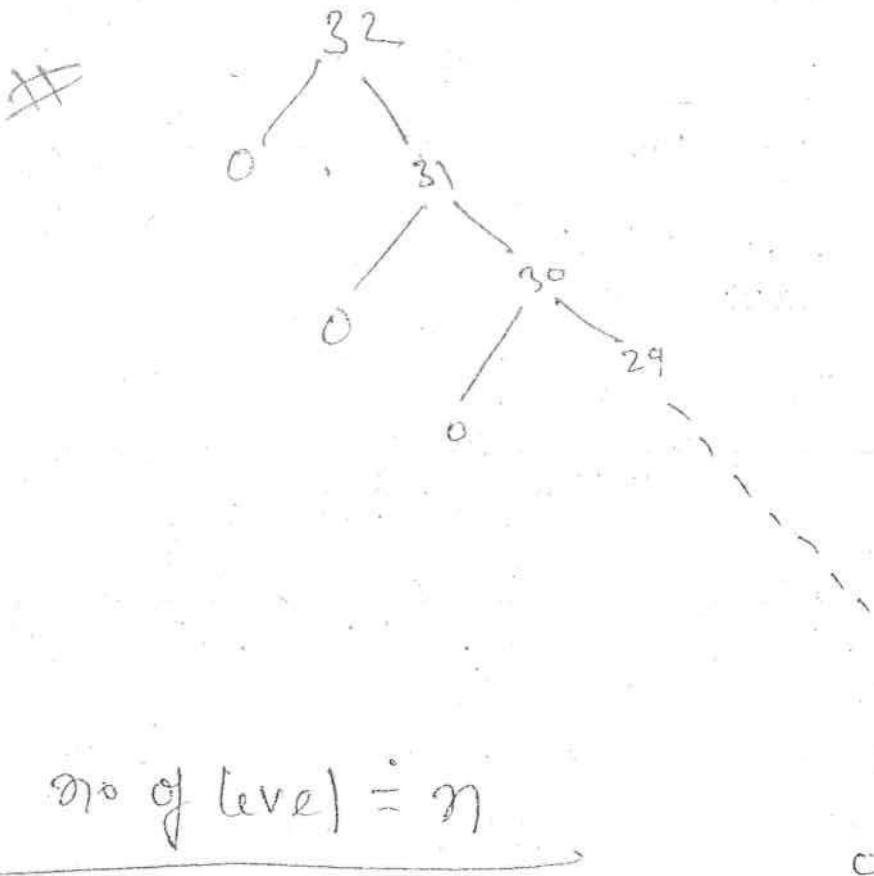
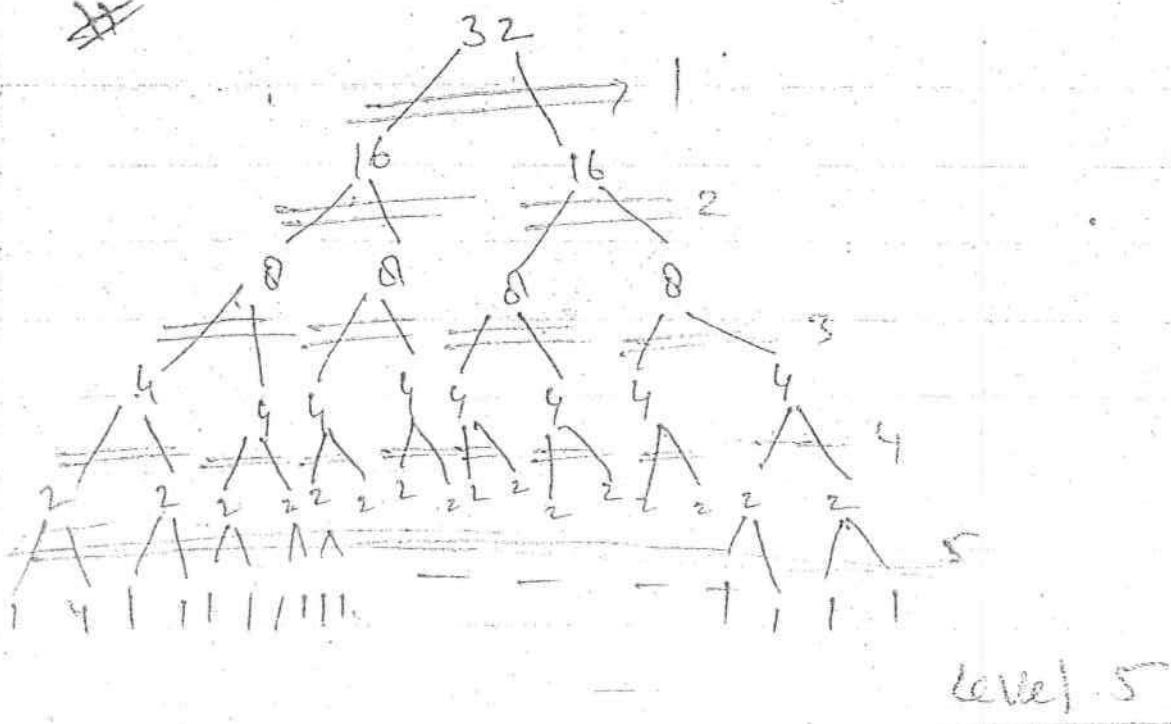
	↑ p									q ↑	
	10	20	30	40	50	60	70	80	90	100	
	1	2	3	4	5	6	7	8	9	10	
	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	
	i	j	j	j	j	j	j	j	j	j	

↓ Partition

() 10 (20, 30, 40, 50, 60, 70, 80, 90, 100)



No of Level is $= \log_2 n$



Quicksort

Quicksort(a, p, q)

{
If ($p == q$) return ($a[p]$)
else

{
 $m = \text{partition}(a, p, q)$; \nearrow^n
 $\text{Quicksort}(a, p, m-1)$; $\nearrow^{n/2}$
 $\text{Quicksort}(a, m+1, q)$; $\nearrow^{n/2}$

return (a)

}

}

Best Case T.C of Quicksort Algo.

n elements.

\Downarrow Partition $\Rightarrow n$.

If we are lucky.

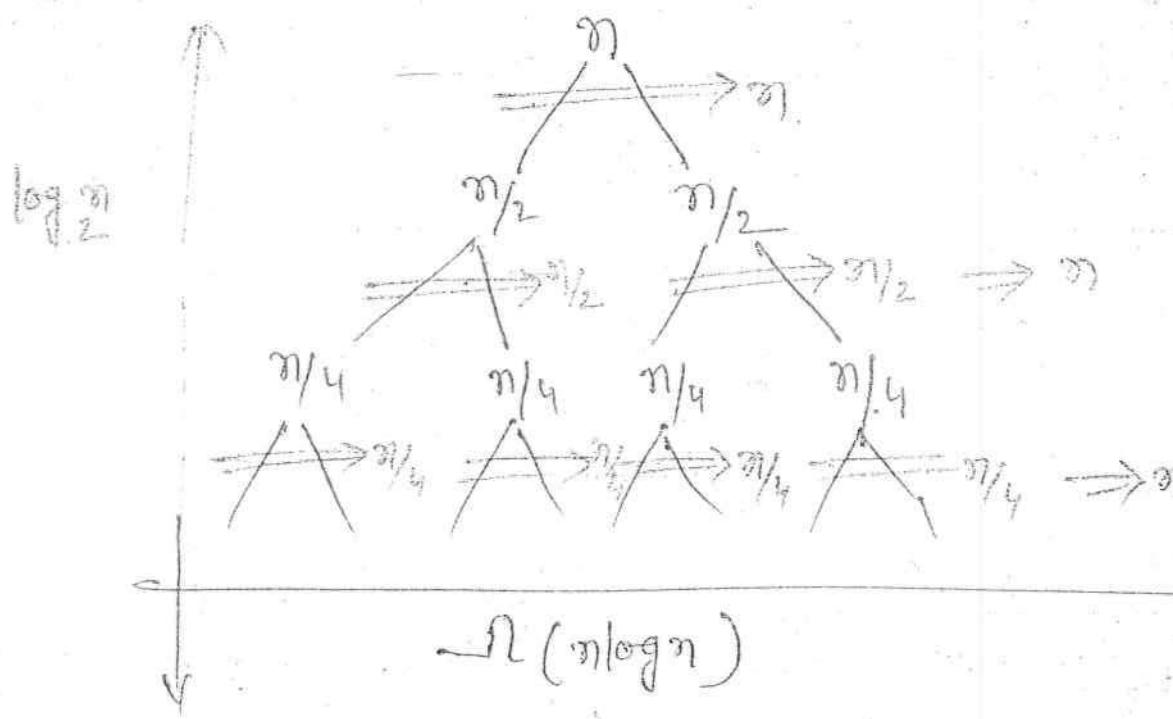
n_1^{-1} p_1 n_2

Recurrence Relation

$$T(n) = \begin{cases} c & \text{if } n=1 \\ n + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) & \text{if } n>1 \end{cases}$$

\Downarrow

$$2T\left(\frac{n}{2}\right) + n \Rightarrow O(n \log n)$$



Eg. 10, 20, 30, 40, 50

↓ Partition $\Rightarrow 5$

() 10 (20, 30, 40, 50)

↓ Partition $\Rightarrow 4$

() 20 (30, 40, 50)

↓ Partition $\Rightarrow 3$

() 30 (40, 50)

↓ Partition $\Rightarrow 2$

() 40 (50)

↓ Partition $\Rightarrow 1$

() 50 ()

#

50 40 30 20 10

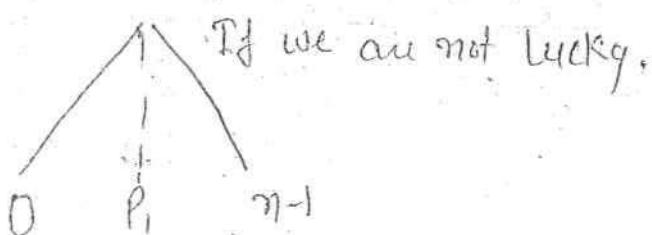
↓ Partition $\Rightarrow 5$

(40, 30, 20, 10) 50 ()

Worst Case

n = elements

\Downarrow Partition $\Rightarrow n$



$$T(n) = \begin{cases} 0 & \text{if } n=1 \\ n + T(0) + T(n-1) \\ \Downarrow \\ T(n-1) + n \Rightarrow O(n^2) \end{cases}$$

Average Case

LULULULU - - - - \Rightarrow Lucky,
 lucky unlucky.

$$\begin{aligned} L(n) &= V\left(\frac{n}{2}\right) + V\left(\frac{n}{2}\right) + n \\ &= 2V\left(\frac{n}{2}\right) + n \end{aligned}$$

$$U(n) = L(n-1) + L(0) + n$$
$$= L(n-1) + n.$$

$$L(n) = 2U(n/2) + n.$$

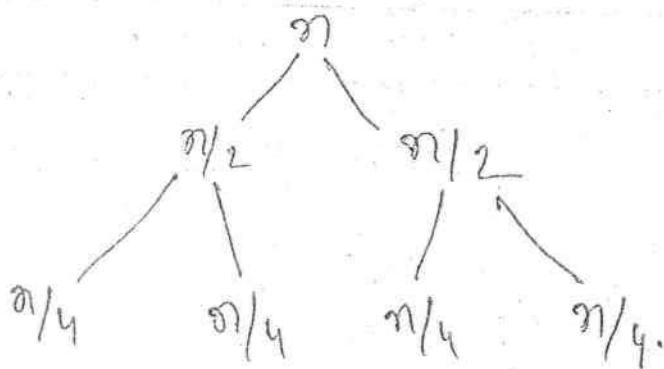
$$\Downarrow$$
$$= 2 \left[L(n/2-1) + n/2 \right] + n$$
$$= 2L(n/2) + O(n)$$

$$\Downarrow$$
$$O(n \log n)$$

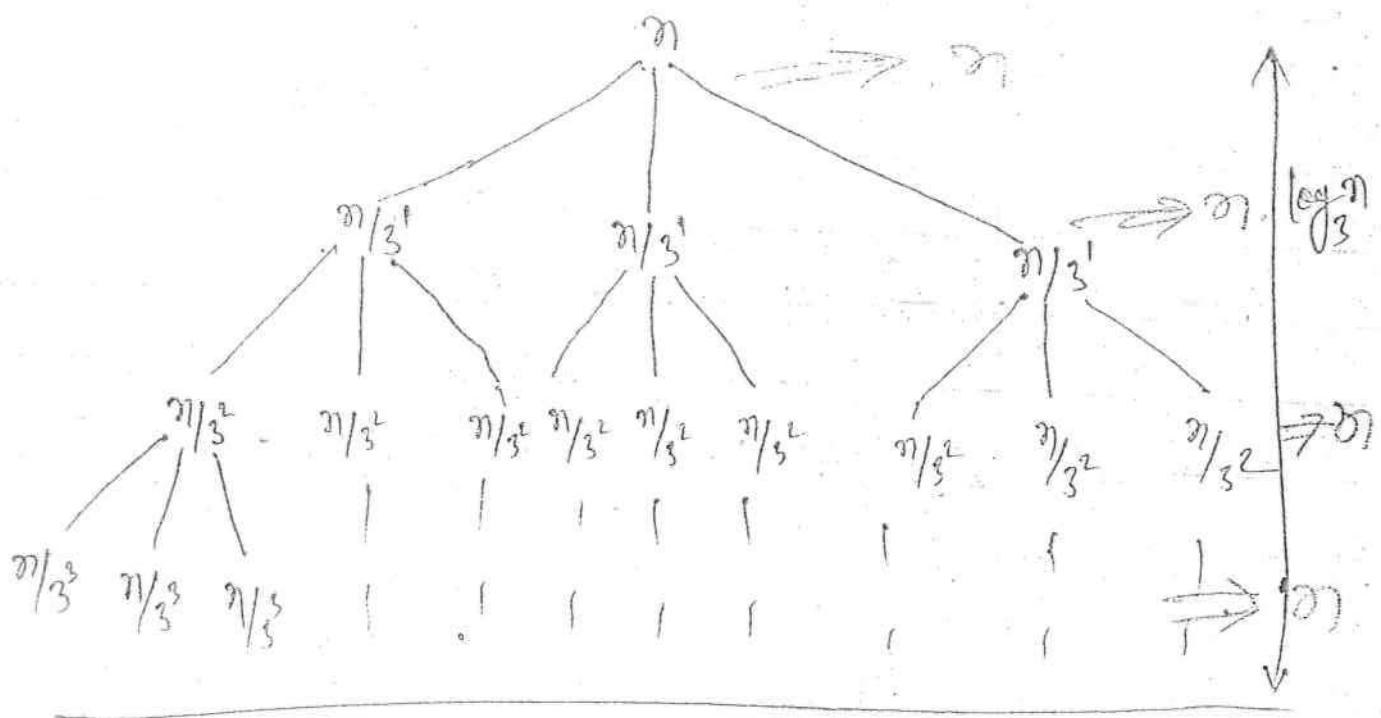
Average Case \rightarrow $O(n \log n)$

Recursive Tree Method

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

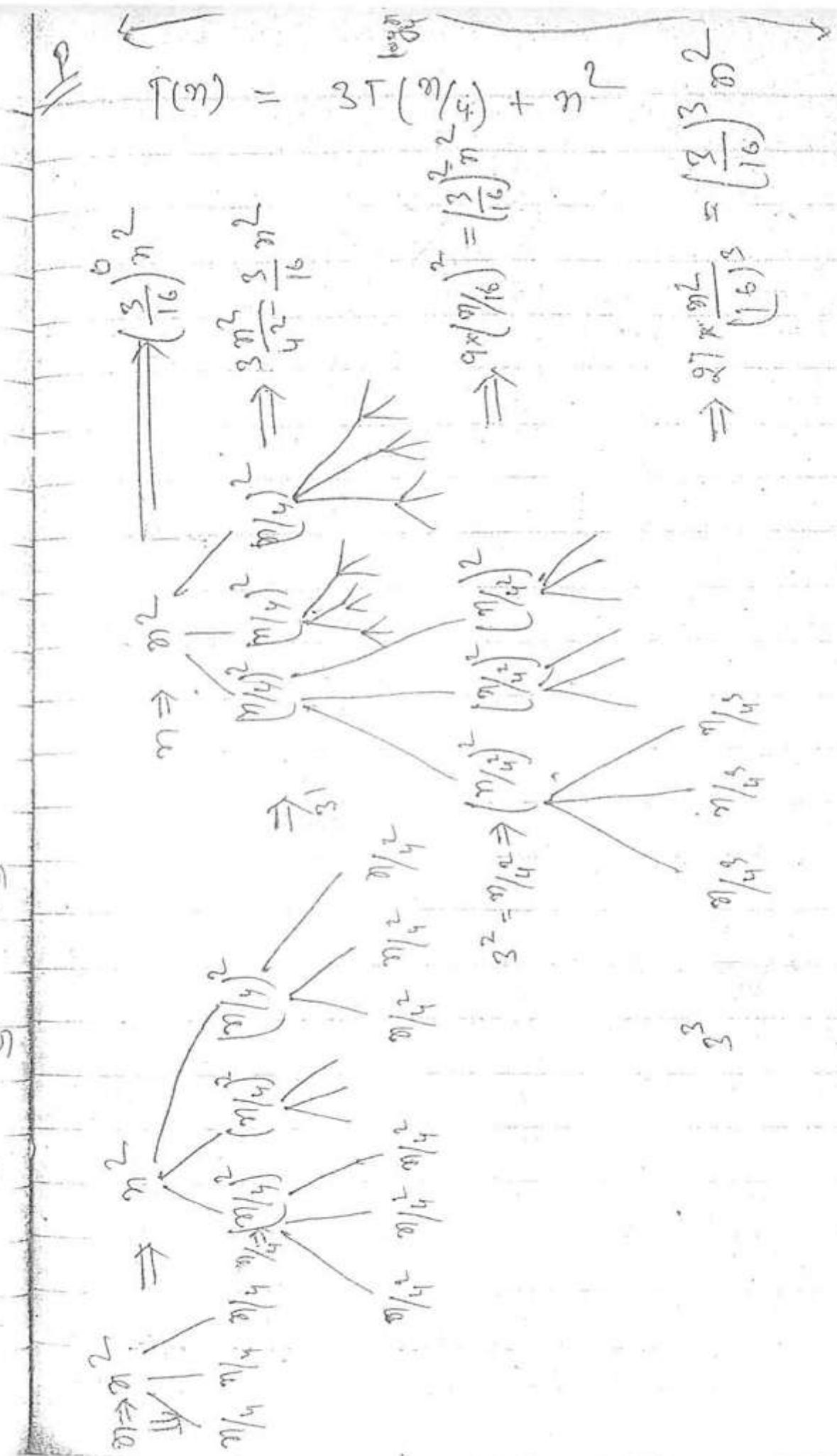


L $T(n) = 3T(n/3) + n$



$n \log n$

$\Rightarrow O(n \log n)$



$$\begin{aligned}
 T(n) &= 3T\left(\frac{n}{4}\right) + n^2 \\
 &= 3 \left[3T\left(\frac{n}{4^2}\right) + \left(\frac{n}{4}\right)^2 \right] + n^2 \\
 &= 3^2 T\left(\frac{n}{4^2}\right) + \frac{3}{16} n^2 \cdot \left(\frac{3}{16}\right)^0 n^2 \\
 &= \underbrace{\left(\frac{3}{16}\right)^2 n^2 + \left(\frac{3}{16}\right)^1 n^2}_{n + n + \dots + n} + \underbrace{\left(\frac{3}{16}\right)^0 n^2}_{n^2} \rightarrow n^2 \log_4 n \\
 \Rightarrow n^2 &\left[\left(\frac{3}{16}\right)^0 + \left(\frac{3}{16}\right)^1 + \dots + \left(\frac{3}{16}\right)^{\log_4 n} \right] \\
 \Rightarrow n^2 &[2] \\
 \Rightarrow O(n^2) &
 \end{aligned}$$

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

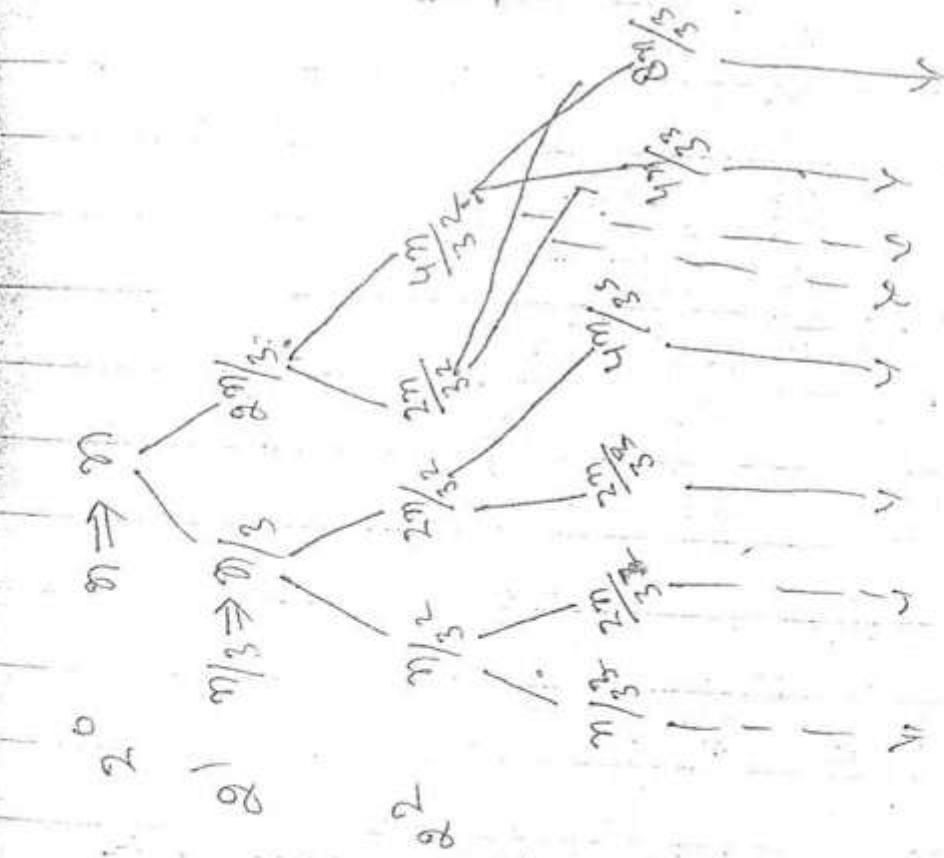
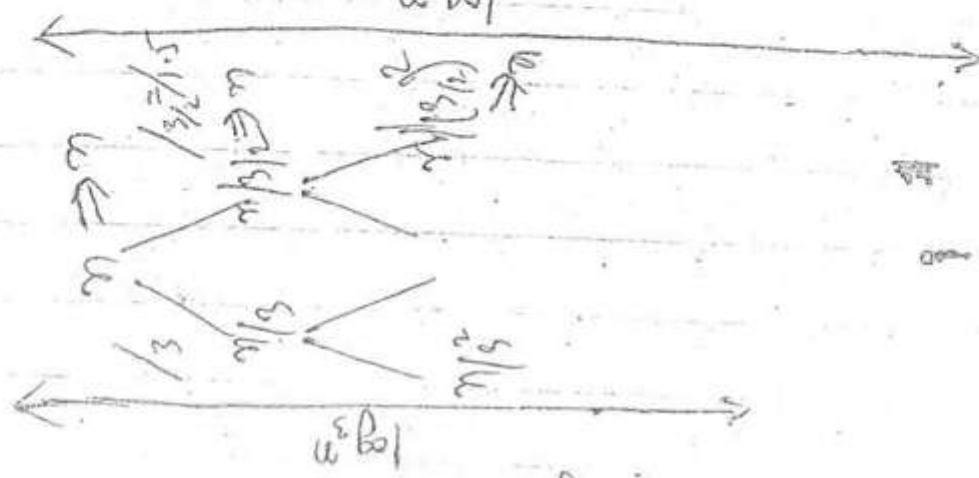
\Rightarrow

$$\begin{array}{c} 27 \\ \Rightarrow (1) \\ 9 \Rightarrow (2) \\ 3 \Rightarrow (3) \\ 1 \end{array}$$

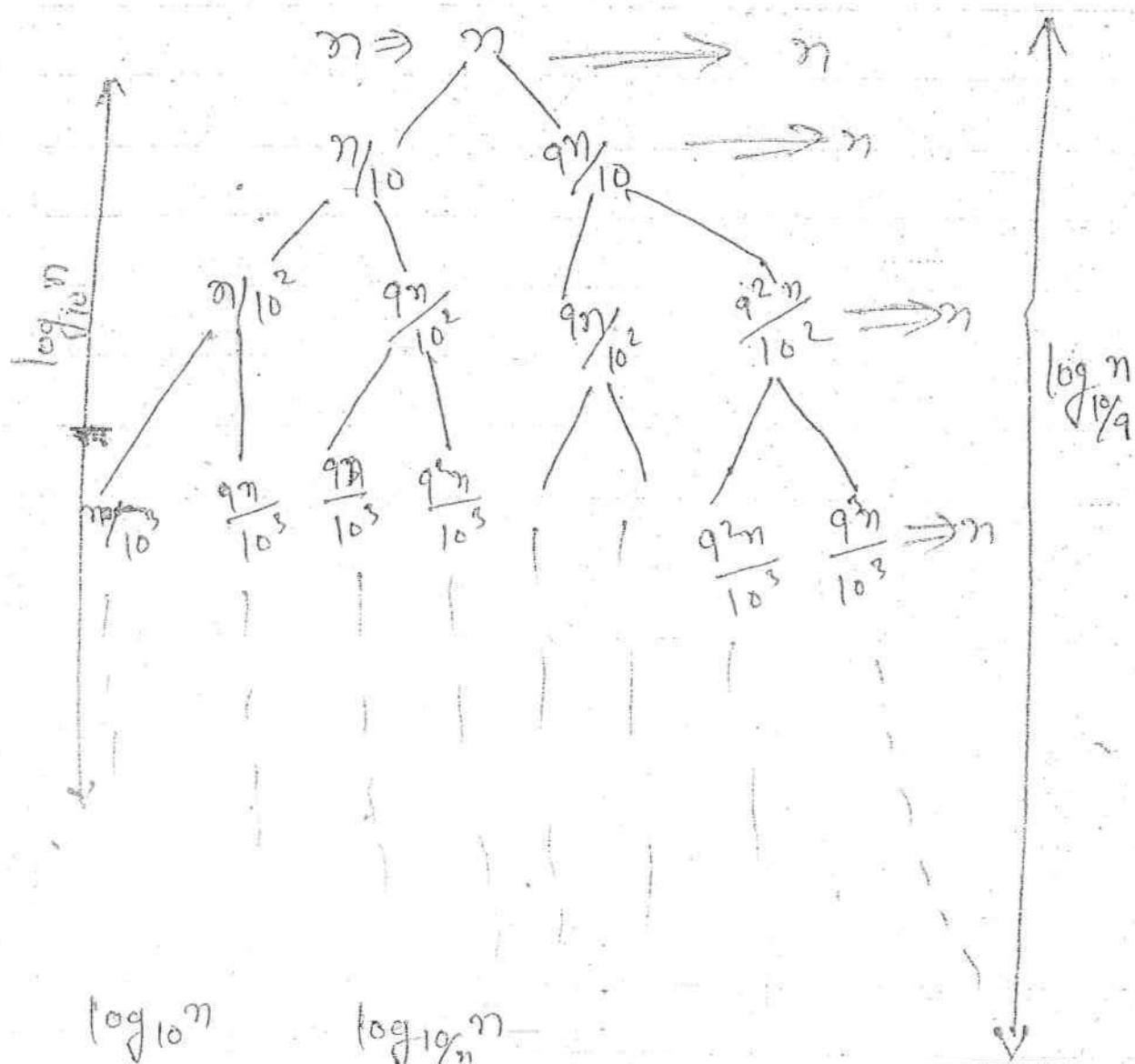
$$\begin{array}{c} 27 \\ \Rightarrow (1) \\ 9 \Rightarrow (2) \\ 3 \Rightarrow (3) \\ 1 \end{array}$$

$\begin{array}{c} 27 \\ \Rightarrow (1) \\ 9 \Rightarrow (2) \\ 3 \Rightarrow (3) \\ 1 \end{array}$

$O(n \log_{3/2} n)$



$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + n$$



$$\begin{array}{l} \log_{10} n \\ \Downarrow \\ 100 \xrightarrow{\quad} ① \\ 10 \xrightarrow{\quad} ② \\ 1 \end{array}$$

$$\begin{array}{l} \log_{10} n \\ \Downarrow \\ 100 \xrightarrow{\quad} (1) \\ 50 \xrightarrow{\quad} (2) \\ 25 \xrightarrow{\quad} (3) \\ 12 \xrightarrow{\quad} (4) \\ 6 \xrightarrow{\quad} (5) \\ 3 \xrightarrow{\quad} (6) \\ 1 \end{array}$$

$O(n \log_{10} n)$

$QSort(a, p, q)$:

```

    {
        if (p == q)
        else
            {
                m = Partition(a, p, q)
                Q.S(a, p, m-1)
                Q.S(a, m+1, q)
            }
    }

```

Best Case ↗ choosing pivot

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

Worst case = $2T(n/2) + n$.

$$T(n) = O(1) + O(n) + T(0) + T(n-1) \quad \left| \begin{array}{l} \text{eq} \\ \frac{n}{100,000} \\ \downarrow \\ T(n-1) + n \end{array} \right.$$

Average case \Rightarrow

$$\begin{aligned}
 T(n) &= O(1) + O(n) + T(n/5) + T(4n/5) \\
 &= T(n/5) + T(4n/5) + O(n) \Rightarrow O(n \log_{5/4} n) \\
 &= O(n \log n)
 \end{aligned}$$

Q After apply few passes of quick sort on the given array we got following output.

1, 10, 5, 8, 25, 44, 55, 36, 70

then

how many pivot elements are there in the above output.

Ans There are three Pivot element

1, 25, 70

Q In quick sort of n -element the $(\frac{n}{4})^{\text{th}}$ smallest element is selected as a pivot element using $O(n)$ time Algo. What will be the worst case Time complexity of quick sort.

(a) $O(n^2)$

(b) $O(n \log n)$

(c) $O(n)$

(d) none.

$$\begin{aligned} T(n) &= O(n) + O(n) + T(\frac{n}{4}) + n + O(\frac{3n}{4}) \\ &= T(\frac{n}{4}) + T(\frac{3n}{4}) + n \\ &= O(n \log n) \end{aligned}$$

The median of n -elements chosen ~~O(n)~~ by taking of $O(n)$ time Algo. Then what will be the T.C. of Q.S.

- a) $O(n^2)$
- b) $O(n \log n)$
- c) $O(n)$
- d) none.

Sol^{ans}

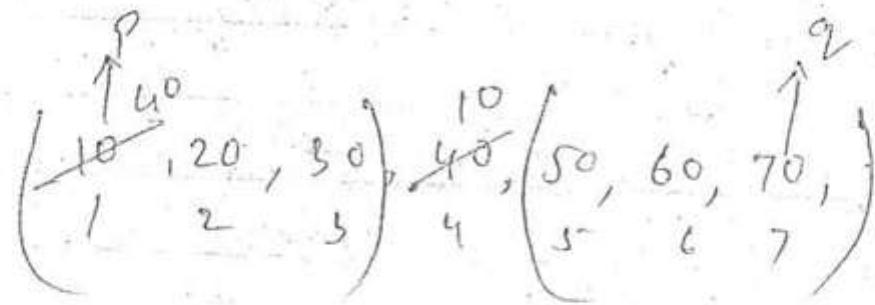
$$\begin{aligned}T(n) &= O(n) + O(n) + 2T(n/2) \\&= 2T(n/2) + n \\&= \underline{\underline{O(n \log n)}}\end{aligned}$$

Randomized Q.Sort:

- (1) It is independent of input order.
- (2) ~~Worst~~ For the sorted input array it will $O(n \log n)$ time worst case.
- (3) No specific input behaves as worst case.

Drawbacks of Q.S

eg



$p = \text{random generator } (1, 2, 3, 4, 5, 6, 7)$

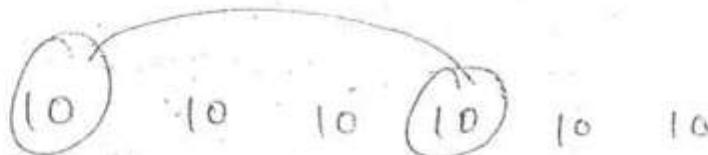
swap $[a[p], a[q])$

Q.S (a, p, q)

#

Randomized Q.S will give you worst case behavior of Q.S. $O(n!)$ if array contains same element

eg



$() 10 (10, 10 (10) 10, 10)$

$() 10 (10, 10, 10)$

R.Q.S worst case $\Rightarrow O(n^2)$

A2

(1) for ($i=1, i \leq K, i++$)

find i^{th} smallest element and delete

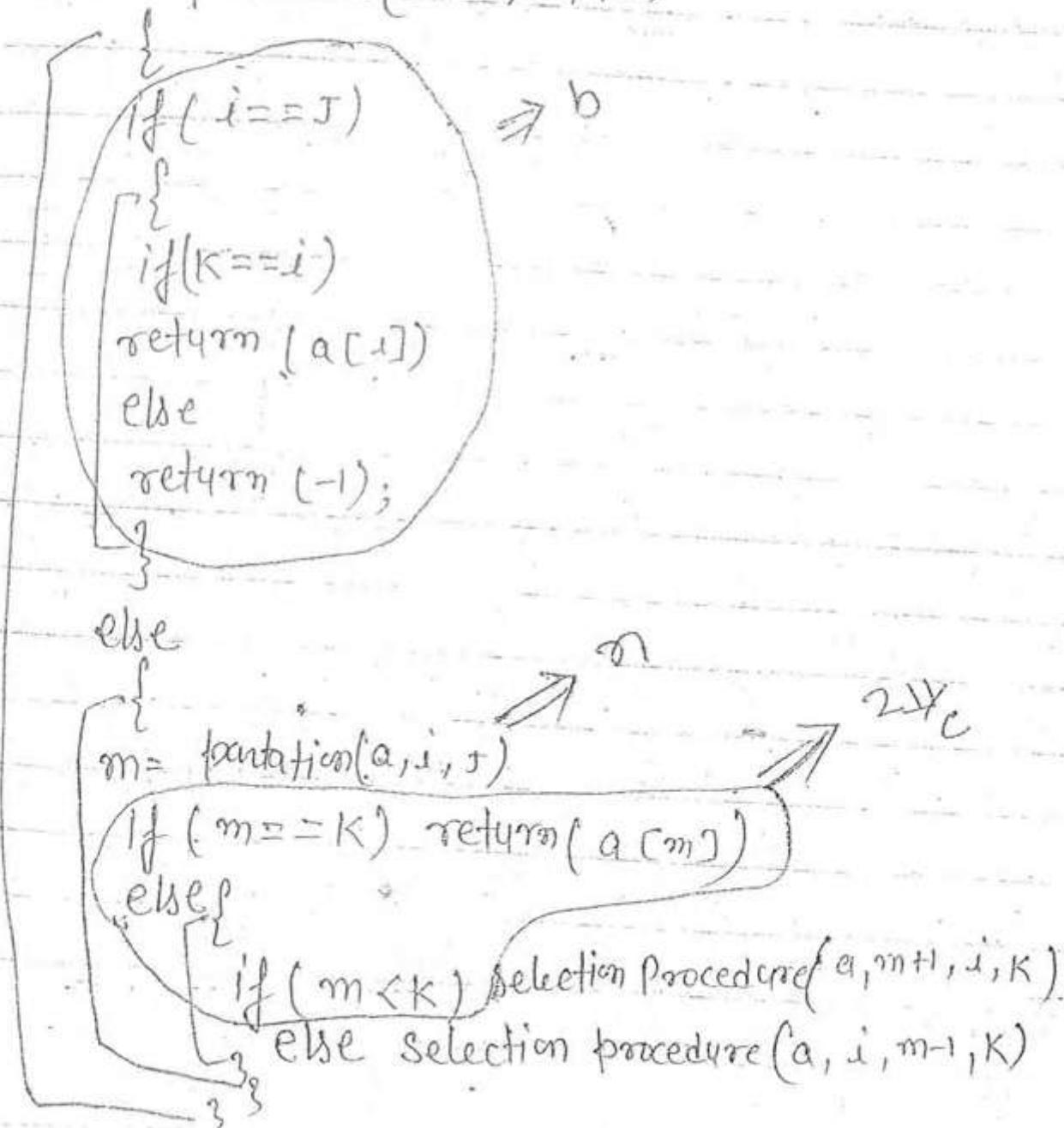
$\Rightarrow (K-1)n$

(2) find smallest element $\Rightarrow n$

$O(Kn)$

Algo

Selection procedure (a, i, j, k)



$$T(n) = \begin{cases} b & \text{if } n=1 \\ n + T(n/2) + 2 \\ \Downarrow \\ T(n/2) + n \Rightarrow O(n) \end{cases}$$

Matrix Multiplication

Without DAE

Matrix Addition -

$A_{n \times n}$

$B_{n \times n}$

$$C_{n \times n} = A_{n \times n} + B_{n \times n}$$

C contains n^2 elements

for every element required one element addition.

$$\text{So } T.C \Rightarrow n^2 * O(1) \\ \Rightarrow O(n^2)$$

```

    for (i = 1 to n)
    for (j = 1 to n)
        C(i, j) = A(i, j) * B(j, i)
    
```

Matrix multiplication

e.g

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2 \times 3}$$

$$B = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}_{3 \times 3}$$

$$C = A * B$$

$$C = \begin{bmatrix} - & - & - \\ - & - & - \\ - & - & - \end{bmatrix}_{2 \times 3}$$

if

$$C_{n \times n} = A_{n \times n} * B_{n \times n}$$

Matrix multiplication

$$\Rightarrow A_{n \times n}$$

$$B_{n \times n}$$

$$C_{n \times n} = A_{n \times n} * B_{n \times n}$$

C - Contains n^2 elements.

for every elements in multiplication

so

$$T.C \Rightarrow n^2 \neq n$$

$$\Rightarrow O(n^3)$$

(1) for ($i=1 \text{ to } n$)

for ($j=1 \text{ to } n$)

$$c[i][j] = 0 \Rightarrow n^2$$

(2) for ($i=1 \text{ to } n$)

for ($j=1 \text{ to } n$)

for ($k=1 \text{ to } n$) $\Rightarrow n^3$

$$c[i][j] = c[i][j] + a[i][k] + b[k][j]$$

$$\underline{\underline{O(n^3)}}$$

Using DAC

- (1) Given two matrix sizes are $\leq 2 \times 2$ then problem is small stop.
- (2) Assume given two matrix are power of 2.
- (3) Assume given matrices are square matrices.

eg

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad \frac{4 \times 4}{2}$$

Matrix A is shown as a 2x2 block matrix. Each block is a 2x2 matrix labeled A₁₁, A₁₂, A₂₁, and A₂₂. The elements of A₁₁ are 1, 2, 5, 6. The elements of A₁₂ are 3, 4, 7, 8. The elements of A₂₁ are 9, 10, 11, 12. The elements of A₂₂ are 5, 16.

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \quad \frac{4 \times 4}{2}$$

Matrix B is shown as a 2x2 block matrix. Each block is a 2x2 matrix labeled B₁₁, B₁₂, B₂₁, and B₂₂. The elements of B₁₁ are a, b, e, f. The elements of B₁₂ are c, d, g, h. The elements of B₂₁ are i, j, m, n. The elements of B₂₂ are k, l, o, p.

$$C = A \otimes B$$

$4 \times 4 \quad 4 \times 4 \quad 4 \times 4$

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

$$C_{11} = A_{11} B_{11} + A_{12} B_{21}$$

$$C_{12} = A_{11} B_{12} + A_{12} B_{22}$$

$$C_{21} = A_{21} B_{11} + A_{22} B_{21}$$

$$C_{22} = A_{21} B_{12} + A_{22} B_{22}$$

∴

$$T(4) = \Theta T(2)$$

$$T(8) = \Theta T(4)$$

$$T(16) = \Theta T(8)$$

$$T(n) = \Theta T(n/2)$$

$$T(n) = \begin{cases} C & \text{if } n \leq 2 \times 2 \\ \Theta T(n/2) + \Theta(n/2)^2 \\ \quad \downarrow \\ \Theta T(n/2) + n^2 \end{cases}$$

$$T(4) = \begin{cases} c & \text{if } n \leq 2 \times 2 \\ 8T(\frac{n}{2}) + 4 \text{ additions of size } \frac{n}{2} \times \frac{n}{2} & \end{cases}$$

$$T(n) = \begin{cases} c & \text{if } n \leq 2 \times 2 \\ 8T(\frac{n}{2}) + 4(\frac{n}{2})^2 & \text{if } n > 2 \end{cases}$$

↓

$$8T(\frac{n}{2}) + n^2$$

↓

$$\underline{\underline{O(n^3)}}$$

Multiply two matrices of size $n \times n$ will take order $O(n^3)$ time in Both cases DAC and without DAC.

So without DAC is better one compare to DAC.

Strassen's matrix multiplication

The decreased number of multiplications from 8 to 7

The increased number of additions from 4 to 16
then the recurrence relⁿ

$$T(n) = \begin{cases} c & \text{if } n \leq 2 \times 2 \\ 7T\left(\frac{n}{2}\right) + cn^2 \\ \downarrow \\ O(n \log_2 7) \\ \downarrow \\ O(n^{2.81}) \end{cases}$$

$\Rightarrow O(n^{2.32})$

Master Theorem

This method will be applicable only those recurrence relⁿ in the following form.

$$\begin{aligned} T(n) &= aT\left(\frac{n}{b}\right) + f(n) \\ &= 2T\left(\frac{n}{2}\right) + n \end{aligned}$$

$$a > 1, b > 1$$

Case 1 :-

$$f(n) = \Theta\left(n^{\log_b a - \epsilon}\right) \text{ where } \epsilon > 0$$

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

Case 2 :-

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

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

Case 3 :-

$$f(n) = \Omega\left(n^{\log_b a + \epsilon}\right) \epsilon > 0$$

and

$$a(f(n/b)) \leq f(n)$$

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

~~Q8~~

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

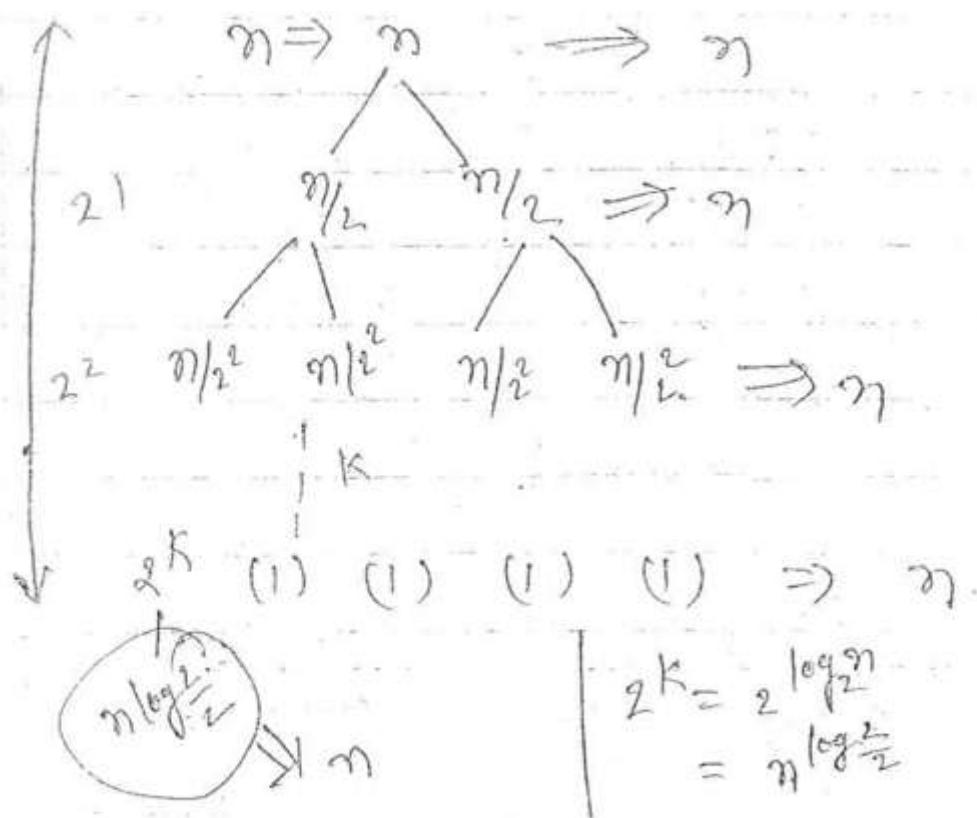
$$a=2, \quad b=2, \quad f(n)=n.$$

$$f(n)=n.$$

$$n^{\log_2 4} = n^{\log_2 2^2} \Rightarrow n$$

Case 2:

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



$$\# T(n) = 2T(n/2) + n^c$$

$$a=2, b=2$$

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

$$n^{\log_2^0} \Rightarrow n^3$$

$$\text{Case 1: } T(n) = \underline{\underline{\Theta(n^3)}}$$

$$\# T(n) = 2T(n/2) + 2.$$

$$a=2, b=2$$

$$f(n) = 1$$

$$n^{\log_b^a} \Rightarrow n$$

$$\text{Case 1: } T(n) = \underline{\underline{\Theta(n)}}$$

$$\# T(n) = 4T(n/2) + n^3$$

$$a=4, b=2$$

$$f(n) = n^5$$

$$n^{\log_2^4} \Rightarrow n^2$$

Case 3:-

$$T(n) = \underline{\underline{\Theta(n^3)}}$$

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

$$a=2, b=2$$

$$n^{\log_b a} = n, \quad f(n) = n \log n.$$

$$\frac{n \log n}{n} = \log n.$$

here $f(n)$ function greater than $n^{\log_b a}$ by $\log n$, which is not polynomial.
So master theorem will not work.

$$\# \quad T(n) = T(n/2) + n$$

$$a=1, b=2$$

$$n^{\log_b a} = n^{\log_2 1} \Rightarrow n^0 \Rightarrow 1$$

$$f(n) \Rightarrow n$$

Case 3 :-

$$\underline{T(n) = \Theta(n)}$$

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

$$a=2, b=1$$

it is not in own
required format.

Conversion

$$T(n) = 2T(\sqrt{n}) + \log n$$

(1) Assume $n = 2^k$

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

(2) Assume $T(2^k) = S(k)$

$$S(k) = 2S(k/2) + k$$

$$a=2, b=2$$

$$f(k) = k$$

$$k^{\log_b a} = k$$

$$k, \log k$$

$$\underline{O(\log_2 n \cdot \log(\log_2 n))}$$

$$\text{# } T(n) = T(\overline{m}) + c$$

$$(i) \quad n^{2K}$$

$$T(2^K) = T(2^{K/2}) + c$$

$$(2) \quad T(2^K) = S(K)$$

$$S(K) = S(K/2) + c$$

$$a=1$$

$$b=2$$

$$f(K) = 1$$

$$K^{\log_2 1} \Rightarrow K^0 \Rightarrow 1$$

$$= \Theta(1 \cdot \log K)$$

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

* Find time complexity for the following C program.

A(n)

{
 }
 {
 if ($n \leq 1$) return 1;
 else
 return (A($\lceil n \rceil$));
 }
}

if ($n \leq 1$) return 1;

else

return (A($\lceil n \rceil$));

$b = \lceil n \rceil \Rightarrow c$

$c, A(b) \Rightarrow T(\lceil n \rceil)$

return (c) $\Rightarrow c$

3.

- a) O(n) b) O($\log n$) , c) O($\log(\log n)$) d) none

~~Q15~~
Let $T(n)$ be the time complexity required to solve A which contain n -etc.

$$T(n) = \begin{cases} b & \text{if } n \leq 1 \\ T(\lceil n \rceil) + c & \text{if } n > 1 \end{cases}$$

$$S(k) = S(k/2) + c$$

$$\Downarrow \\ \log k$$

$$\underline{\underline{O(\log(\log n))}}$$

Find time complexity for following C-Program.

DAA (n)

```
{
    if ( $n \leq 1$ ) return ;
    else
        {
            return (DAA ( $n/2$ ) + DAA ( $n/2$ ) + n) ;
        }
}
```

$\Theta(n)$

$$a = DAA(n/2)$$

$$b = DAP(n/2)$$

$$c = a+b+n$$

return (c)

$$T(n) = \left\{ \begin{array}{l} b \quad \text{if } n \leq 1 \\ T(n/2) + T(n/2) + c \end{array} \right\}$$

$O(n)$

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

$$= 2[2T(n/2) + c] + c$$

$$= 2^2 T(n/2^2) + 2^1 + 2^0 c$$

$$= 2^3 T(n/2^3) + 2^2 c + 2^1 c + 2^0 c$$

↓
14

$$\begin{aligned}
 &= 2^K + \left(\frac{n}{2^K}\right) + \left[2^0 + 2^1 + 2^2 + \dots + 2^{K-1} \right] \\
 &= n + \left[\frac{1(2^K - 1)}{2 - 1} \right] \\
 &= n + n \\
 \Rightarrow & O(n)
 \end{aligned}$$

#

A(n)

```

    {
        if ( $n \leq 1$ ) return;
        else
            return ( $n * A(\sqrt{n})$ )
    }
  
```

{

 $a = \sqrt{n}$ $b = A(a)$ $c = n * b$

return(c);

{

Solv

$$T(n) = \begin{cases} b & \text{if } n \leq 1 \\ T(\sqrt{n}) + c \end{cases}$$

$$\underline{\underline{O(\log(\log n))})}$$

$$1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$$

↓

$$O(n^2)$$

$$1 + 2 + 3 + 4 + \dots + n$$

$\text{Sum} = 0$
 for ($i = 1$ to n) ~~sum~~

$$\text{Sum} = \text{Sum} + i$$

$$\Downarrow O(n)$$

$$n! = O(n^n)$$

$$1 * 2 * 3 * 4 * \dots * n = O(n^n)$$

* fact(n)

```
{
  if ( $n \leq 1$ ) return 1
  else
    return ( $n * \text{fact}(n-1)$ )
}
```

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ T(n-1) + C & \text{otherwise} \end{cases}$$

Cube Root

for ($i = 1$ to 1000)

```

  {
    if ( $i^3 \leq 1000$ )
      elm
      return ( $i-1$ )
  }
  3

```

\Rightarrow

10

$$m^3 < n$$

Conclusion

- 1) Binary Search $\Rightarrow T(n/2) + c \Rightarrow O(\log n)$
- 2) Power of an element $\Rightarrow T(n/2) + c \Rightarrow O(\log n)$
- 3) Find Max, Min $\Rightarrow 2T(n/2) + 2 \Rightarrow O(n)$
- 4) Merge sort $\Rightarrow 2T(n/2) + n \Rightarrow O(n \log n)$
- 5) Quick sort \Rightarrow
 - $2T(n/2) + n \Rightarrow O(n \log n)$
 - $T(n-1) + n \Rightarrow O(n^2)$

6) Selection procedure $\Rightarrow T(n/2) + n \Rightarrow O(n)$

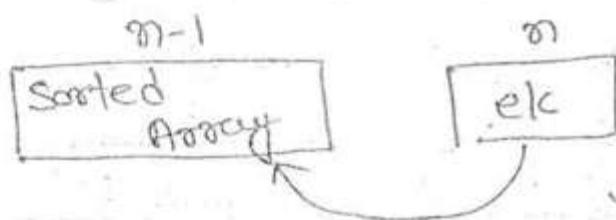
$$O(1)(n/2) + n^2 \Rightarrow O(n^2)$$

7) Matrix multiplication \Rightarrow

$$T(n/2) + n^2 \Rightarrow O(n^2 \cdot 0)$$

(Strassen)

Insertion Sort



eg 10, 50, 60, 70, 80, 30, 40, 90, 100

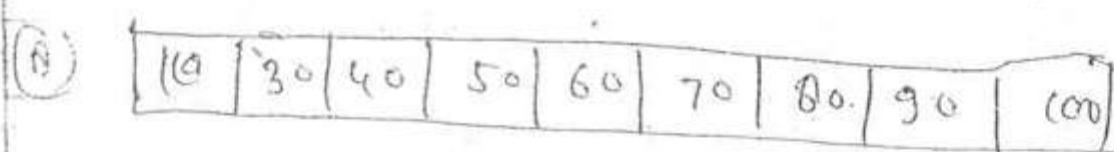
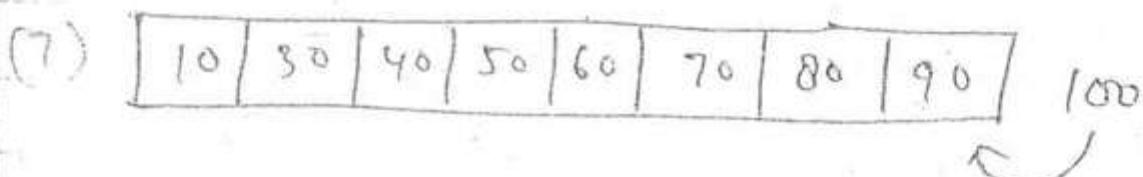
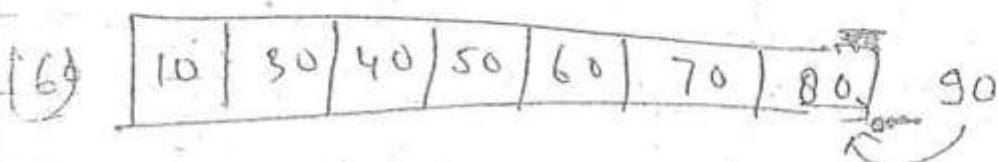
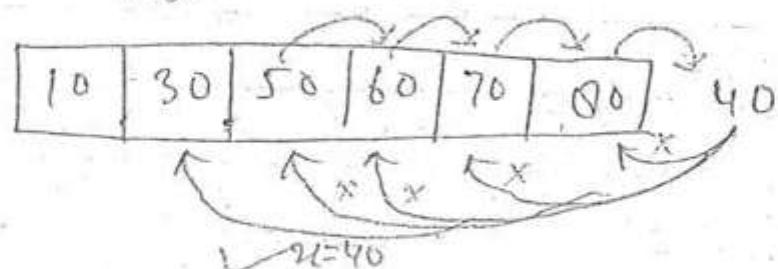
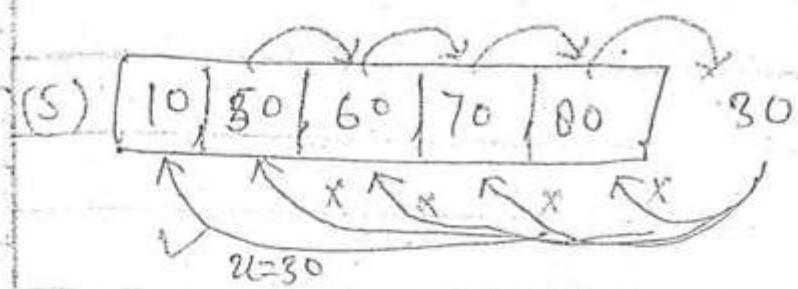


(1) [10] 50

(2) [10 | 50] 60

(3) [10 | 50 | 60] 70

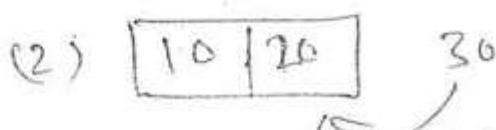
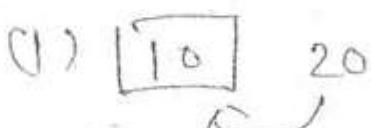
(4) [10 | 50 | 60 | 70] 80

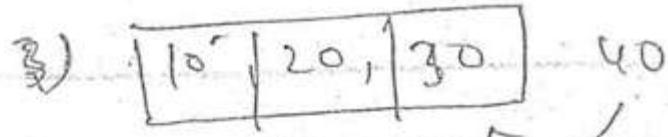


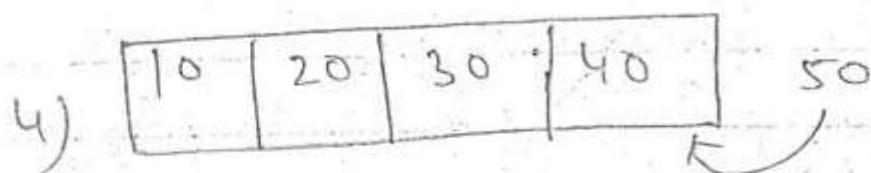
Best case

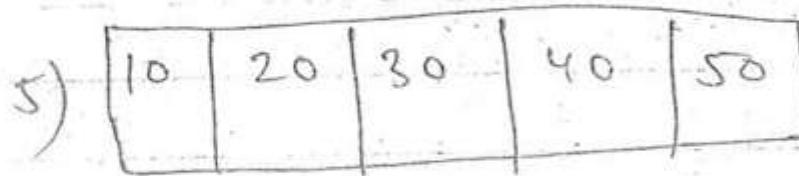
Increasing order

10, 20, 30, 40, 50



3) 

4) 

5) 

Best case of T.C of Insertion sort is

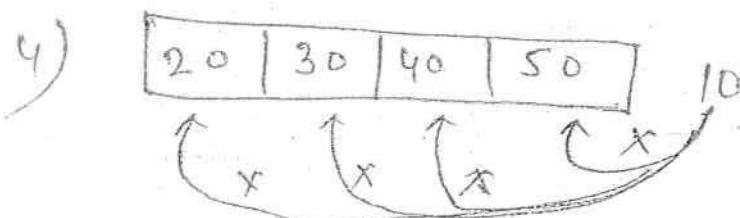
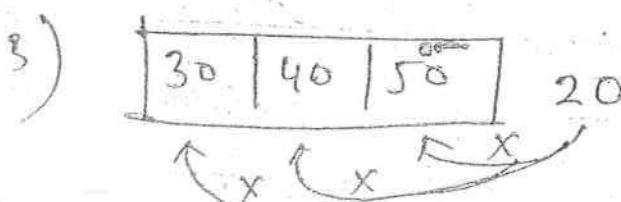
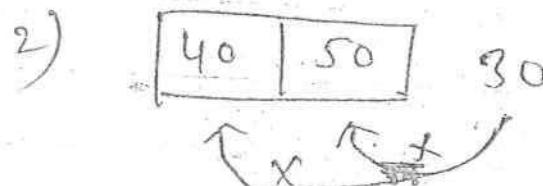
$\underline{\underline{O(n)}}$ in $(n-1)$ time (Θn^2)

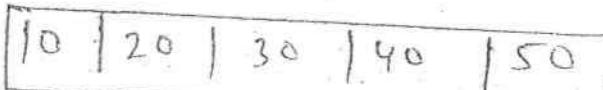
Note For smaller size of input insertion sort is Better.

For greater " " " " Merge
" " " " Better

worst case

50, 40, 30, 20, 10





Comparison	swapt
1 $\Rightarrow n$	1 $\Rightarrow n$
2 $\Rightarrow n$	2 $\Rightarrow n$
3 $\Rightarrow n$	3 $\Rightarrow n$
4 $\Rightarrow n$	4 $\Rightarrow n$
1	1
1	1
$n-1$	$n-1$
n	n

total Comparison $O(n)^2 \Rightarrow n^2 + n^2$

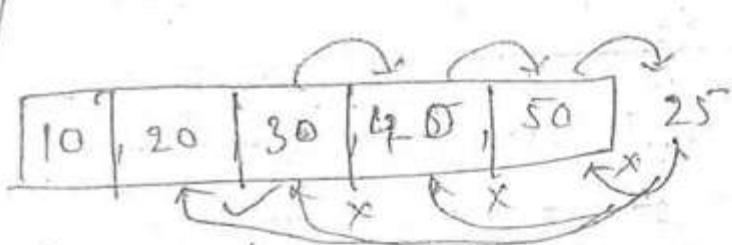
4 swapt $O(n)^2$

$O(n^2)$

Q Why applying Insertion sort to Insert an element we are finding correct place of that element using linear search even though array is sorted. What will be the ~~T.C.~~ of Insertion sort, If you replace linear search at place of Binary

a) Search

- a) remains $O(n^2)$
- b) $O(n \log n)$
- c) $O(n)$
- d) None.



~~Solⁿ~~

Comparison

$$1 \Rightarrow \log n$$

$$2 \Rightarrow \log n$$

$$3 \Rightarrow \log n$$

(

)

)

$$n-1 = \log n$$

swap

$$1 \Rightarrow \log n \quad n$$

$$2 \Rightarrow \log n \quad n$$

$$3 \Rightarrow \log n \quad n$$

(

)

)

$$n-1 = \log n \quad n$$

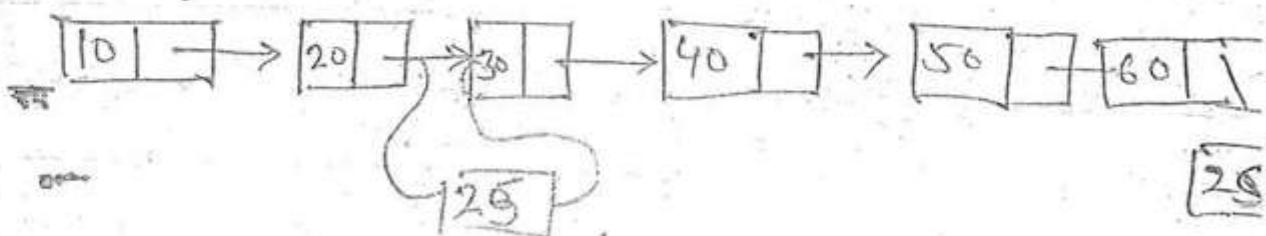
rect

$$\text{total Comparison} = O(n \log n) \Rightarrow n \log n + n^2$$

swapt
hat

$$\text{total Swap} = O(n^2) \Rightarrow O(n^2)$$

#



Comparison

$$1 \Rightarrow n$$

$$2 \Rightarrow n$$

$$3 \Rightarrow n$$

$$n-1 \Rightarrow n$$

$$\frac{n}{2}$$

$$n^2 + 0$$

$$\underline{\underline{O(n^2)}}$$

sweapt

$$0$$

$$0$$

$$0$$

$$0$$

$$0$$

$$\frac{n}{2}$$

$$0$$

Average case also

leads to

$$\underline{\underline{O(n^2)}}$$

Tuesday

~~Topic~~

Heap Sort

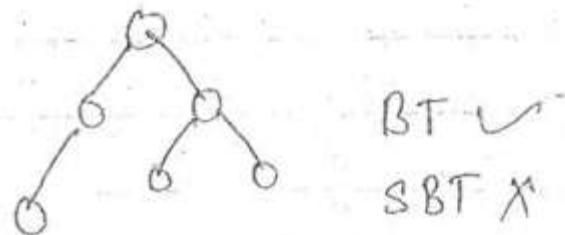
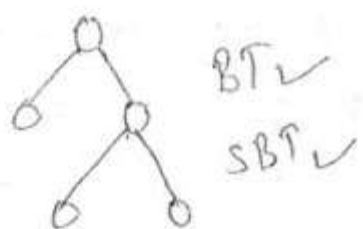
1) Binary Tree.



2) Strict Binary tree (full binary tree) in the given.

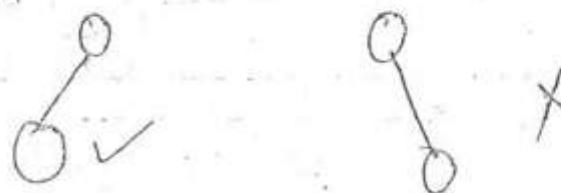
binary tree at every node exactly zero
or two children.

eg

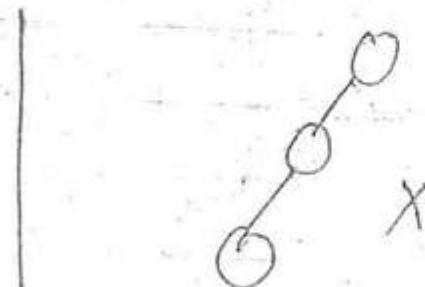
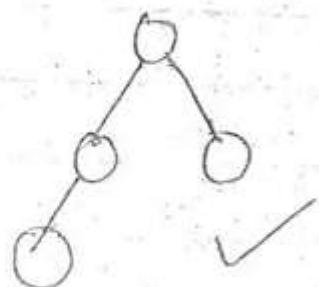


3) Almost Complete Binary tree:-

- i) A binary tree said to be a almost complete binary tree iff
- ii) at every node after completing left only go to right

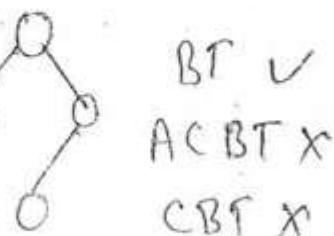
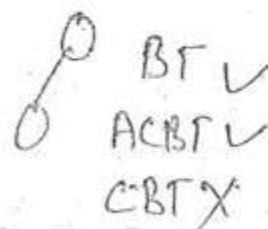
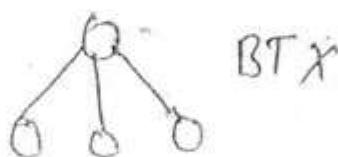
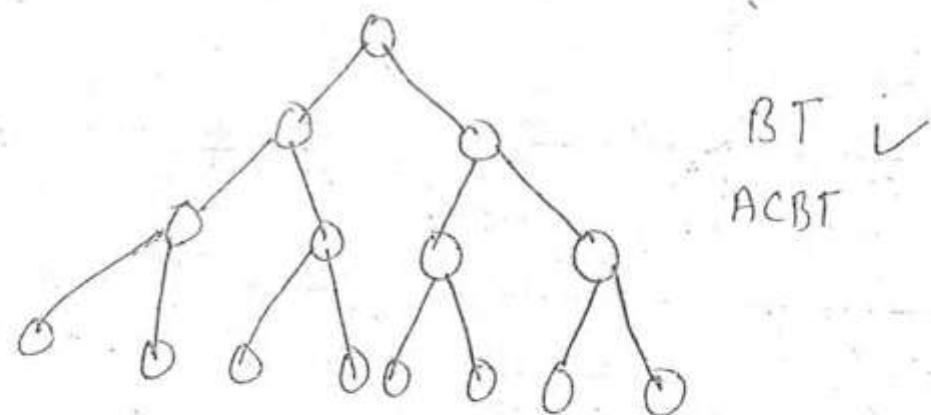


2) at every node after completing present level completely then only go to next level



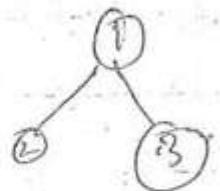
4) Complete binary tree:-

In the almost complete binary tree if last level also filled completely then that particular tree is called complete binary tree.



Q. Construct almost complete binary tree for nodes 1, 2, 3.

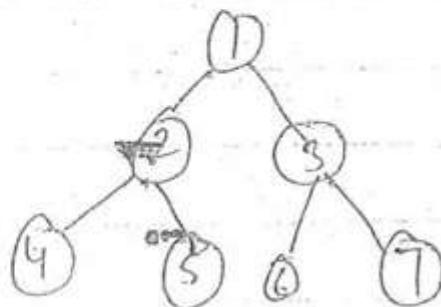
$$n=3 \\ 2^{2-1}$$



$$\log_2 3 \Rightarrow 2$$

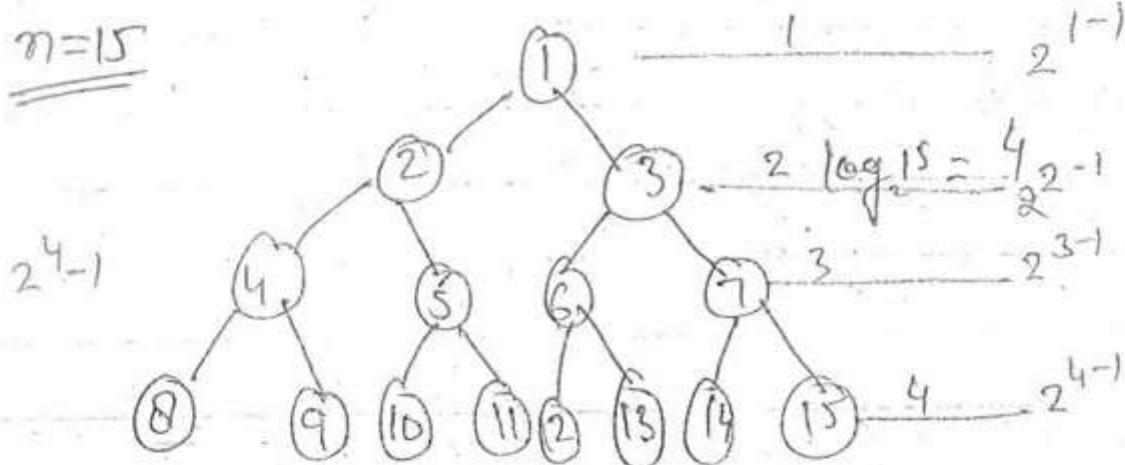
$$\underline{n=7}$$

$$2^{3-1}$$



$$\log_2 7 \Rightarrow 3$$

$$\underline{n=15}$$



$$1 \quad 2^{1-1}$$

$$2 \log_2 15 = 4 \quad 2^{2-1}$$

$$2^{3-1}$$

$$2^{4-1}$$

all are BT, CBT, ACBT

The max^{no} no of node present at level i in almost complete binary tree

$$is = \boxed{2^{i-1}}$$

~~* The max^m no of node present at level i
almost complete binary tree -~~

(total no of nodes) = $2^i - 1$

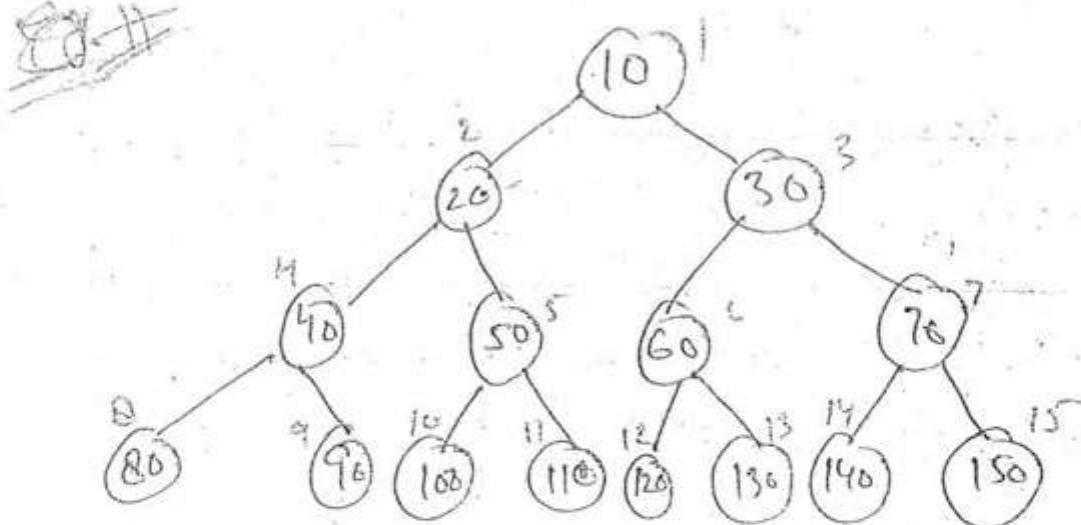
~~* No of levels in almost complete binary tree with n nodes is~~

$$n = 2^i - 1$$

$$2^i = n + 1$$

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

where i is level



10	20	30	40	50	60	70	80	90	100	110	120	130	140	150
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

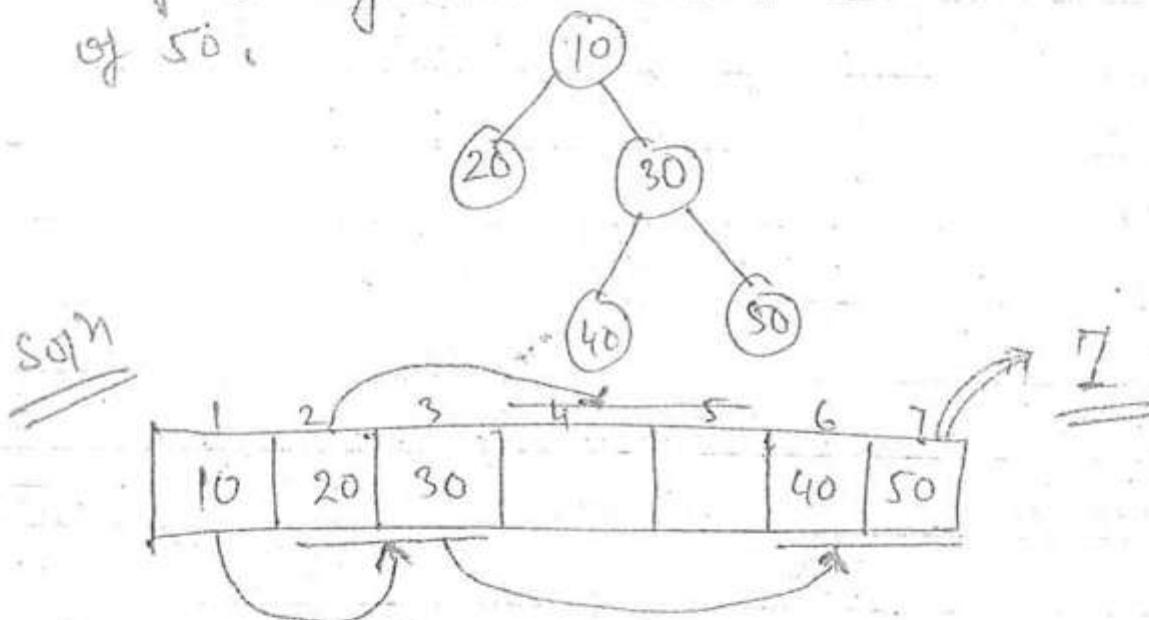
the node present at i th replace

$$\text{left}(i) = 2 \times i$$

$$\text{right}(i) = (2 \times i) + 1$$

$$\text{Parent}(i) = \left\lfloor \frac{i}{2} \right\rfloor$$

Consider the following binary tree in the form of array then what will be the position of 50.



~~Imp~~ Heap Sort

Heap tree →

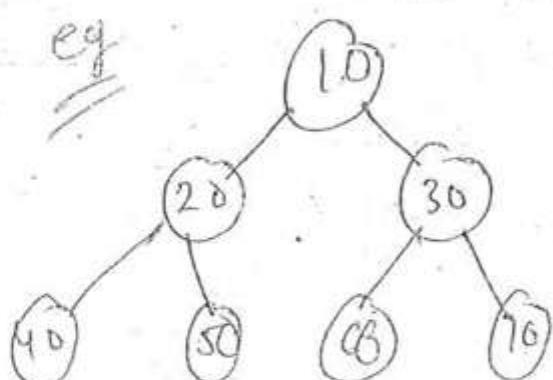
An almost complete binary tree is said to be Heap tree iff.

It should satisfy heap property.
two types of heap property.



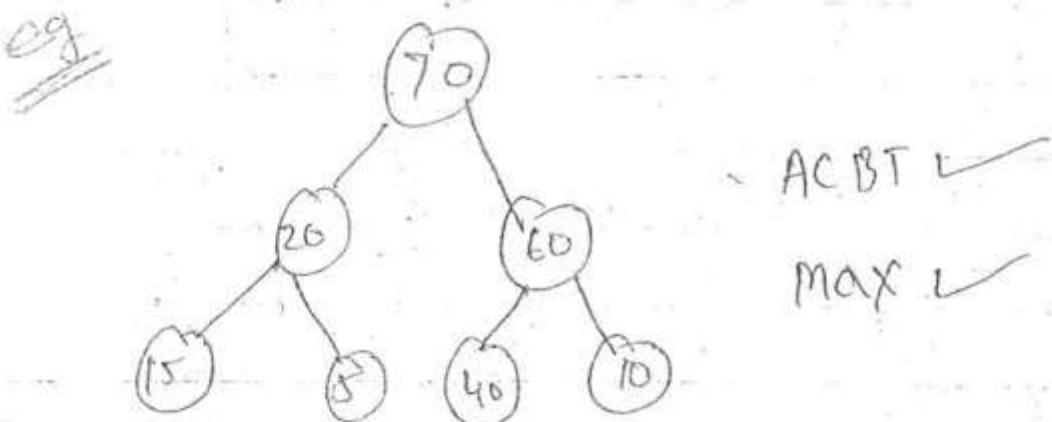
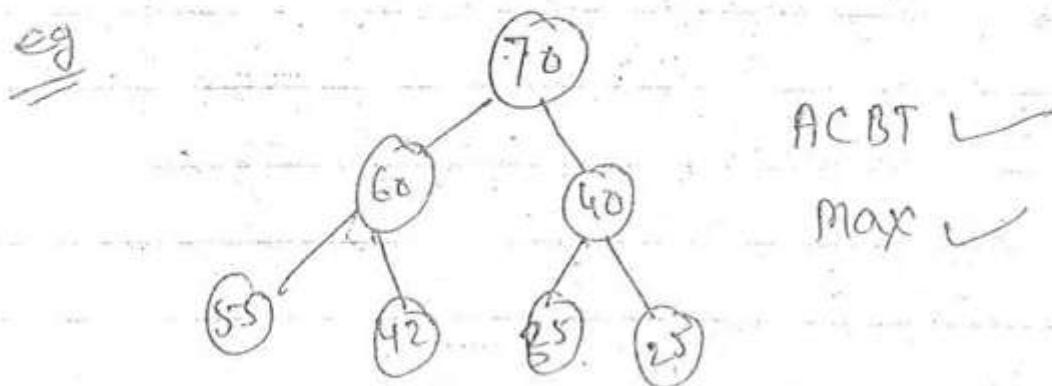
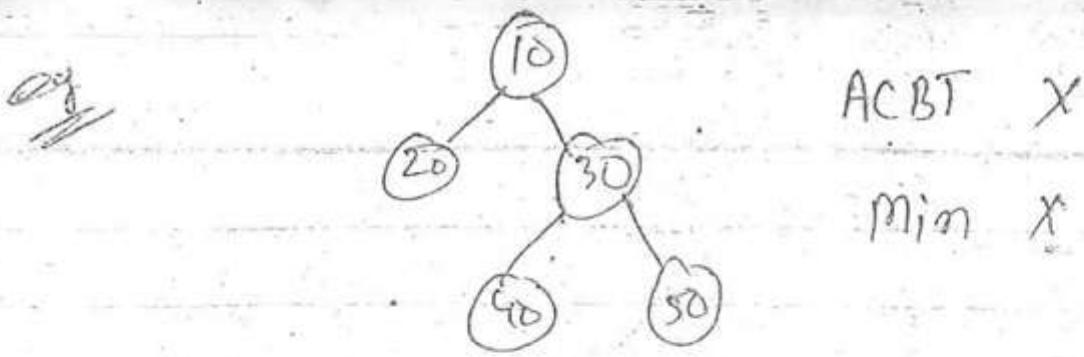
In the almost complete binary tree at every node present is minⁿ comparing their children is called min heap tree

In the almost complete binary tree at every node present is maximum comparing their children is called max heap tree.

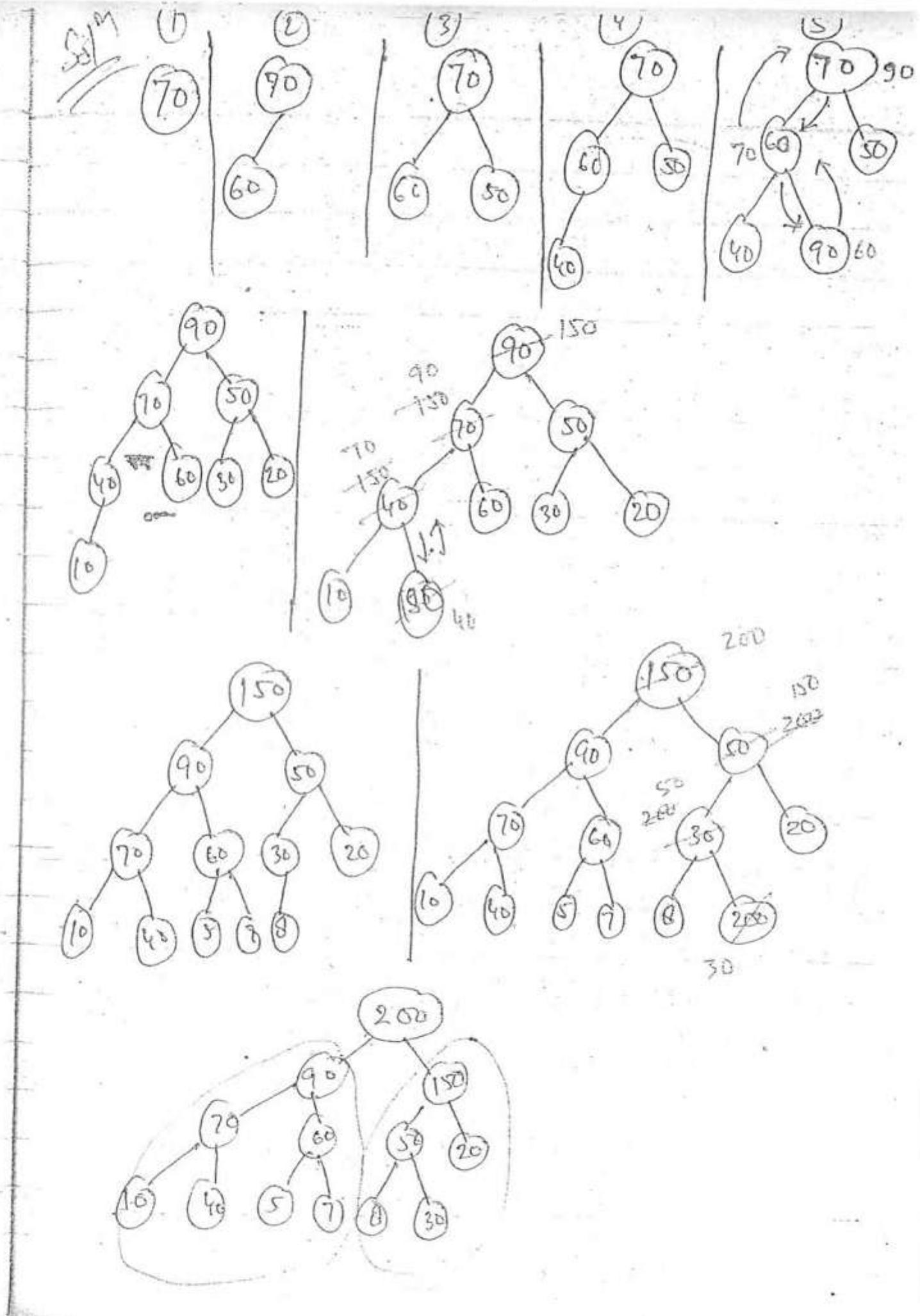


ACBT ✓

Min ✓

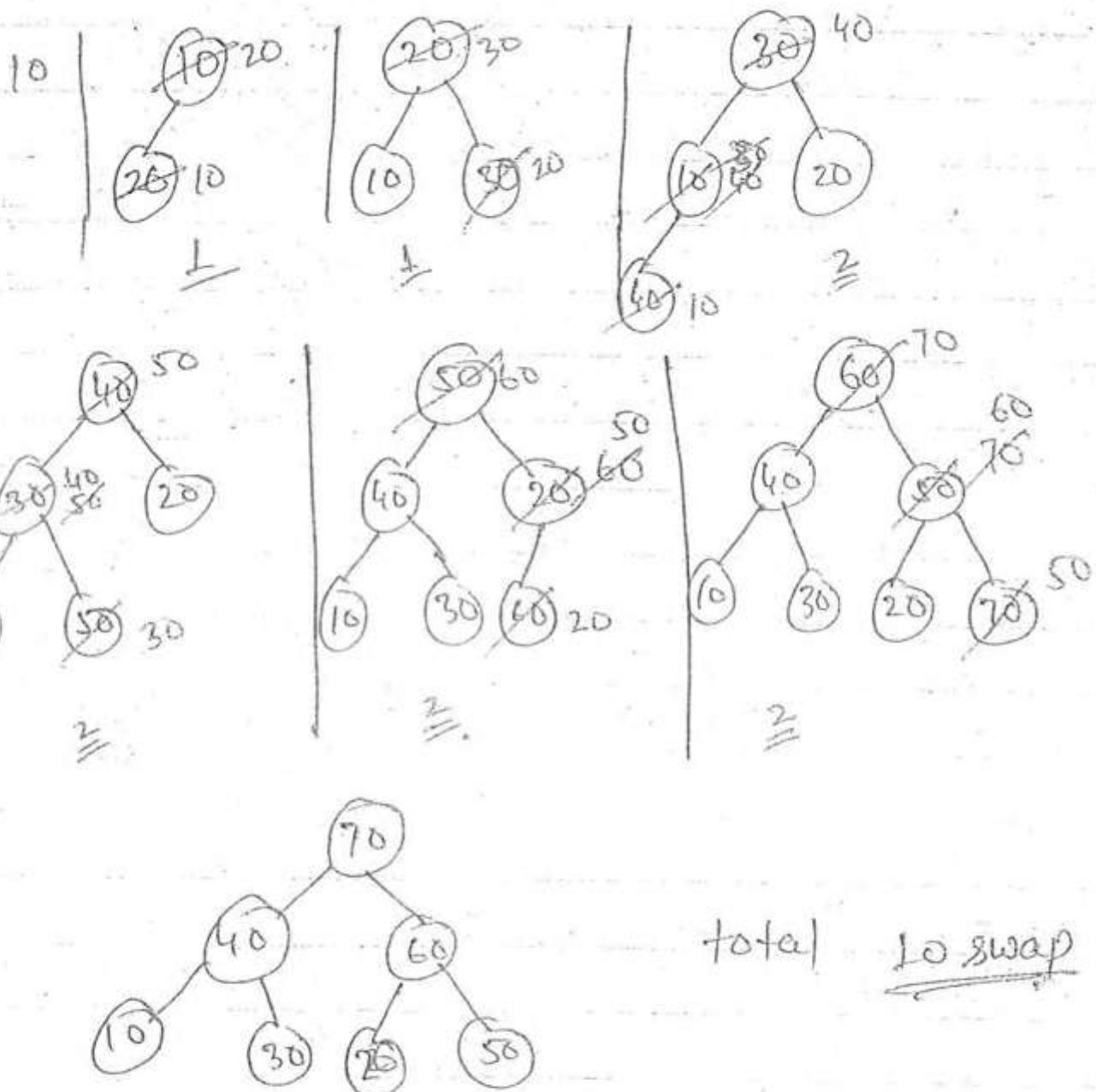


P Consider the max heap tree for the following elements: 70, 60, 50, 40, 90, 30, 20, 10, 150, 5, 7, 8, 200.

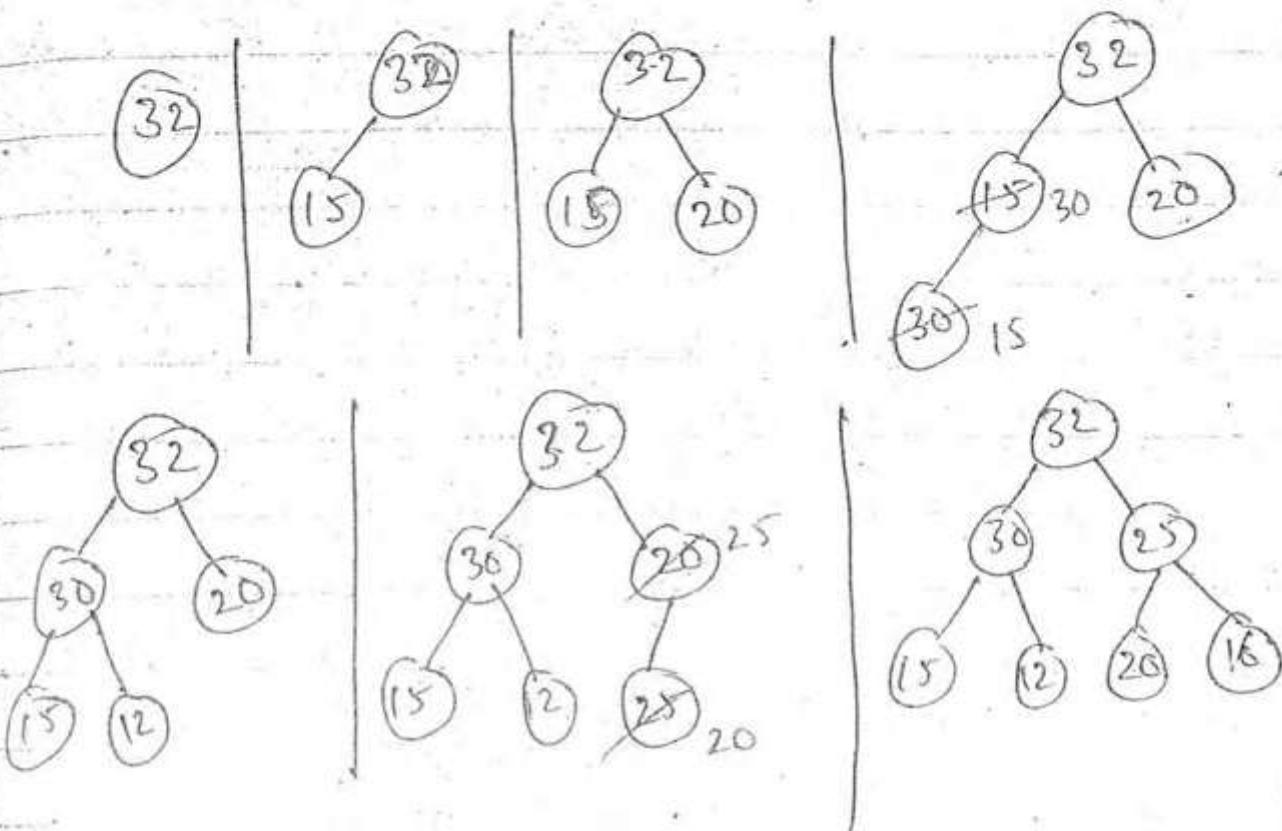


#

construct max heap tree for the following element 10, 20, 30, 40, 50, 60, 70

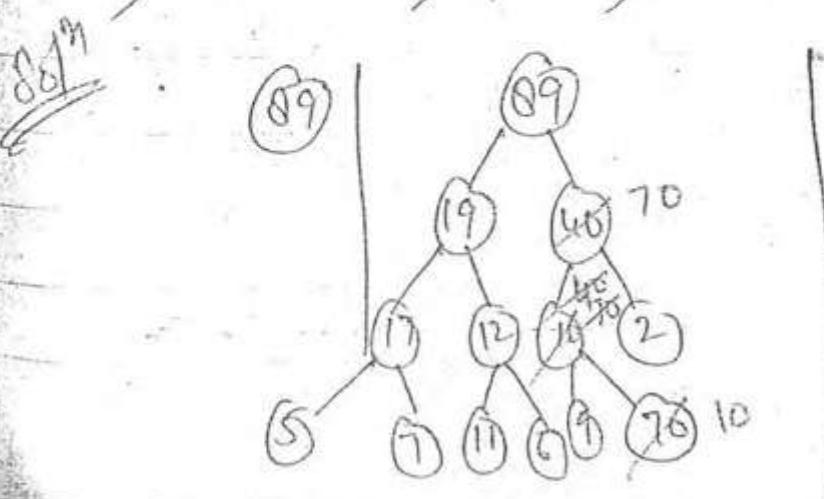


* The elements: 32, 15, 20, 30, 12, 25, 16 are inserted one after another into a max heap. The resultant max heap tree is.

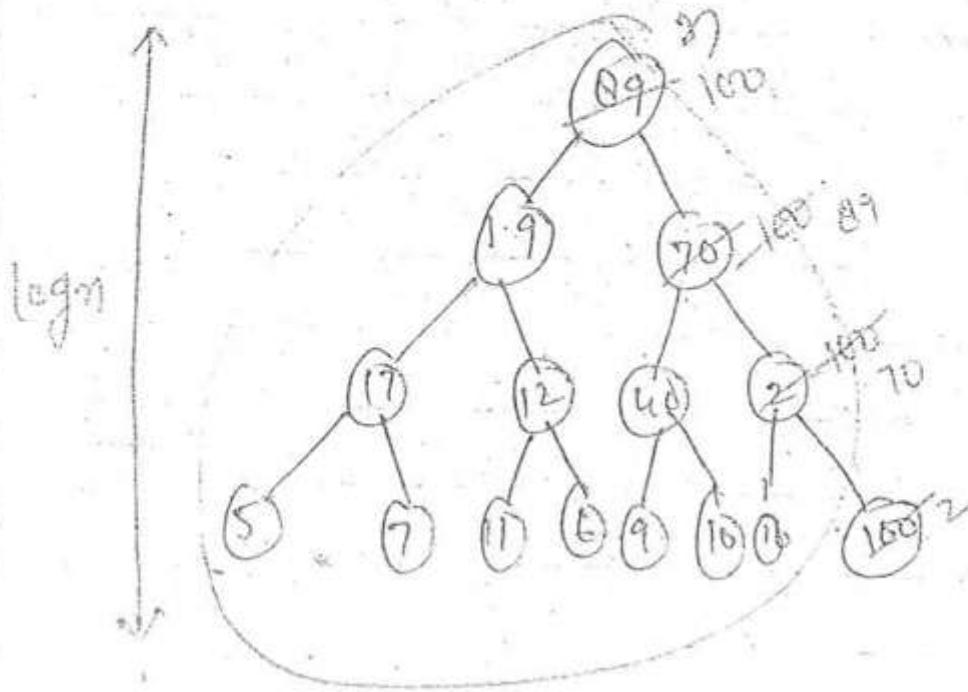


* The minⁿ no of interchanges needed to convert the array: 89, 19, 40, 17, 12, 10, 25, 7, 11, 6, 9, 70 into heap with maxⁿ elem at the root is

- a) 0
- b) 1
- c) 2 ✓
- d) 3



B. How much time it will take to insert an element into maxⁿ or min heap which already contains n elements.



Max heap ~~O(1)~~

Best case $\Rightarrow \mathcal{O}(1)$

Eg. let we have to insert 1

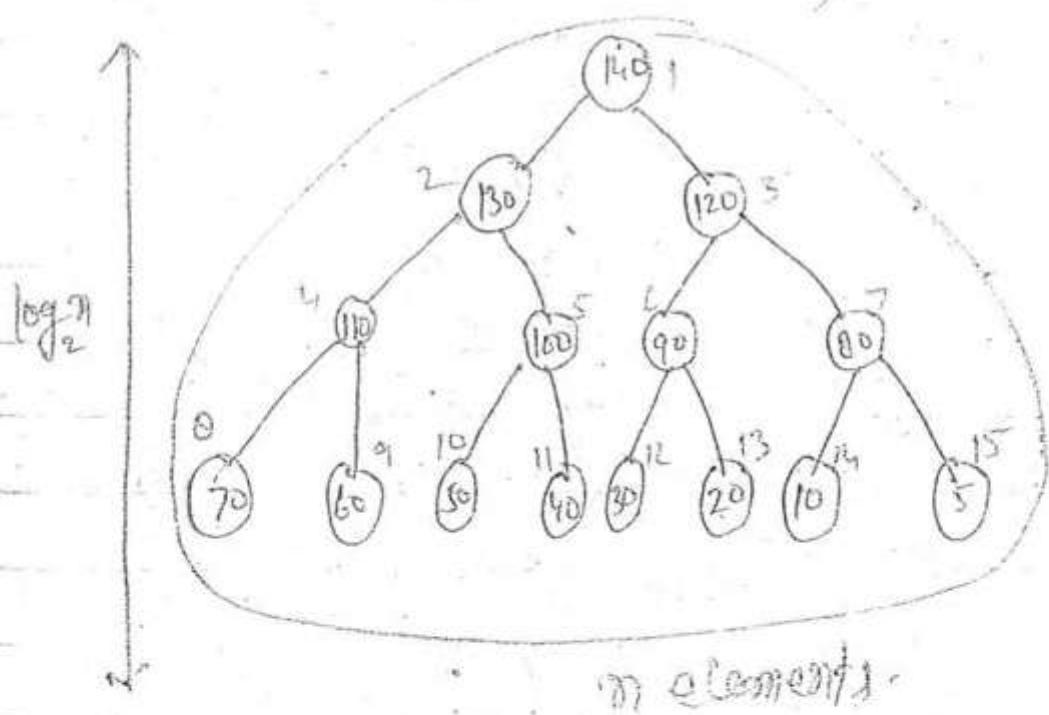
Worst case $\Rightarrow \mathcal{O}(\log n)$

Let we have to insert 100 then all parent will be affected. Hence total no of swapping = no of level in tree.

Note

In order to insert an element into min heap or max heap will take order of $O(\log n)$ times $\Rightarrow O(\log n)$ worst case

Q. How much time it will take to delete an element from min heap or max heap which contain already n elements.



140	130	120	110	100	90	80	70	60	50	40	30	20	10	5
1	2	3	4	5	6	7	8	9	10	11	12	13	14	

Jan

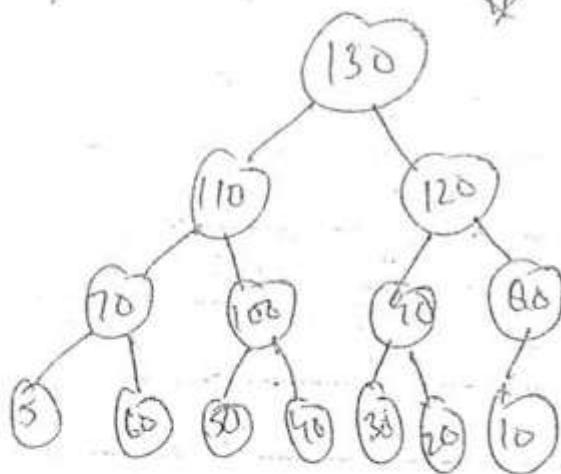
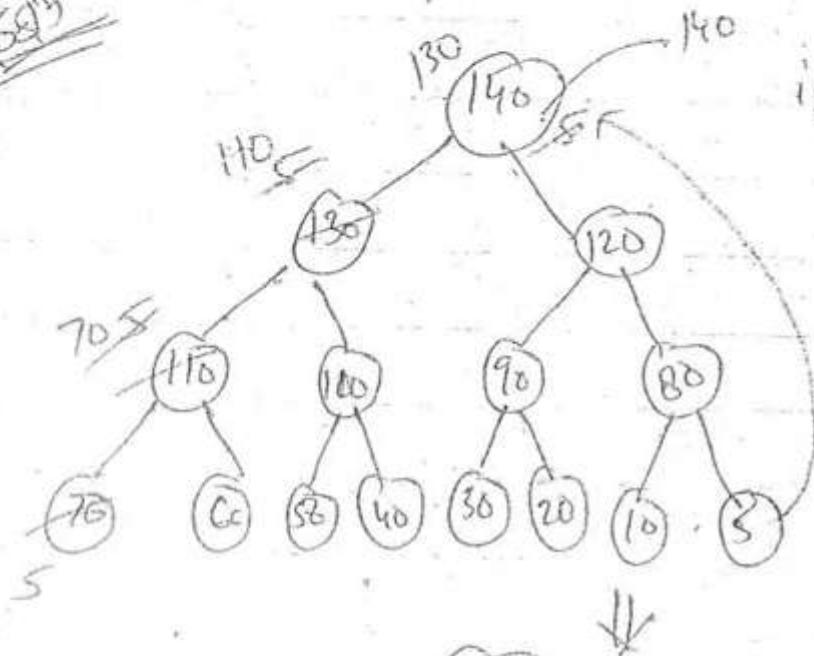
Note \Rightarrow In How to delete an element
 from min heap or max heap which
 Contains already n elements requires
 order of log n times.

~~SDP~~

if 140 Delete it replace
 with 5

SDP

000



140	130	120	110	100	90	80	70	60	50	40	30	20	10	5
-----	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	---

15 2 3 4 5 6 7 8 9 10 11 12 13 14 15
replace

To delete 140 replace it with 5

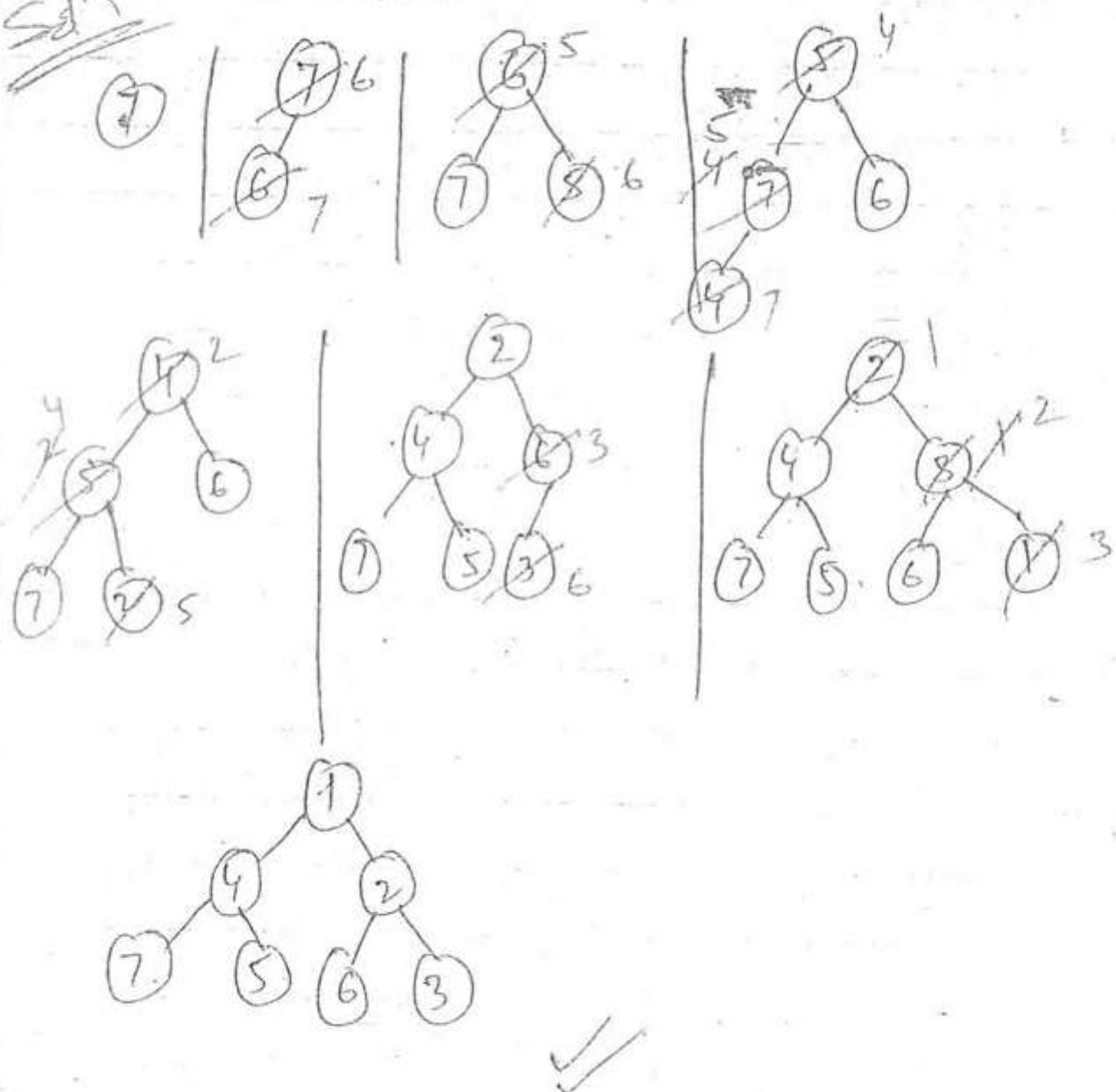
5	130	120	110	100	90	80	70	60
---	-----	-----	-----	-----	----	----	----	----

Q Draw the min heap tree that results from insertion of the following elements one after another into empty min heap.

7, 6, 5, 4, 2, 3, 1.

Show the resulted min heap tree after deleting root from the min heap.

~~S&H~~



Q1 Consider the following max heap implemented using array.

Q2 Which one of the following is max heap

- P1 a) 25, 12, 16, 13, 10, 8, 14
- b) 25, 14, 13, 16, 10, 8, 12
- c) 25, 14, 16, 13, 10, 8, 12
- d) 25, 14, 12, 13, 10, 8, 16

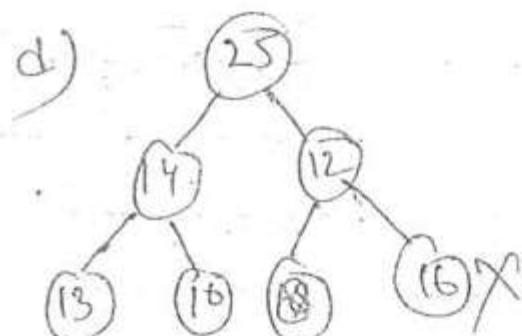
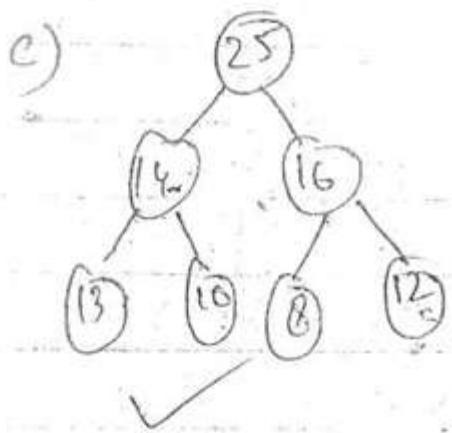
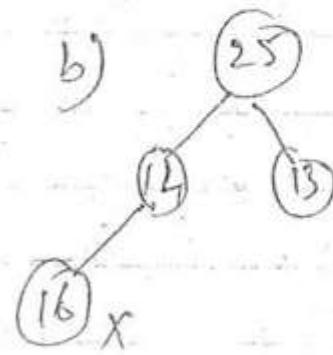
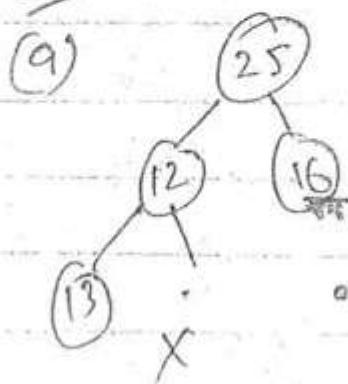
Q2 What are the contents of the array after deleting on the correct answer to the above problem.

- a) 14, 13, 12, 10, 8
- b) 14, 12, 13, 8, 10

c) 14, 13, 8, 12, 10

✓ 14, 13, 12, 8, 10

Solⁿ P₁



Solⁿ P₂

~~Q1~~ A 3-Array max heap is like a binary max heap but insert of 2-children nodes have 3-children.

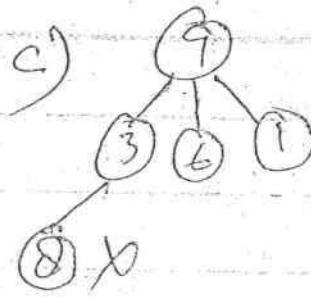
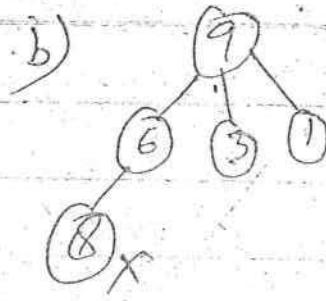
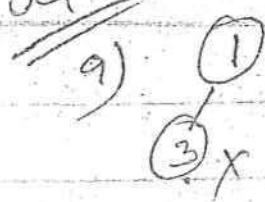
1) which one of the following is a valid sequence of elements in array representation of 3-array max heap.

- a) 1, 3, 5, 6, 8, 9
- b) 9, 6, 3, 1, 8, 5
- c) 9, 3, 6, 8, 5, 1
- d) 9, 5, 6, 8, 3, 1

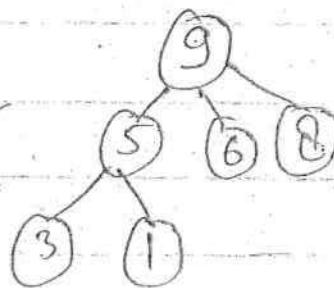
~~Q2~~ Suppose the elements 7, 2, 10 & 4 are inserted in the order into valid 3-array max heap found in the about problem which is true.

- a) 10, 7, 9, 8, 3, 5, 2, 6, 4
- b) 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
- c) 10, 9, 4, 5, 7, 6, 8, 2, 1, 3.
- d) none.

~~SD²~~



d)



2. In a binary search tree, if we want to insert a node with key 10, then which of the following is correct?

a) Insert 10 as a left child of 5.

b) Insert 10 as a right child of 5.

c) Insert 10 as a left child of 8.

d) Insert 10 as a right child of 8.

Heap Sort (Assending order)

(1) Create maxⁿ heap tree by inserting one after another(n) $\Rightarrow O(n \log n)$

(2) Delete one by one element and store in the array from right to left(n)

$$O(n \log n)$$

$$\Rightarrow \underline{\underline{\Theta(\log n)}}$$

Min Heap

(1) finding minⁿ $\Rightarrow O(1)$

(2) deleting minⁿ $\Rightarrow O(\log n)$

eg (3) Finding 100th smallest element

Getting $\Rightarrow (100 \log n) \Rightarrow O(\log n)$

(4) Getting maxⁿ element $\Rightarrow O(n)$

Max heap

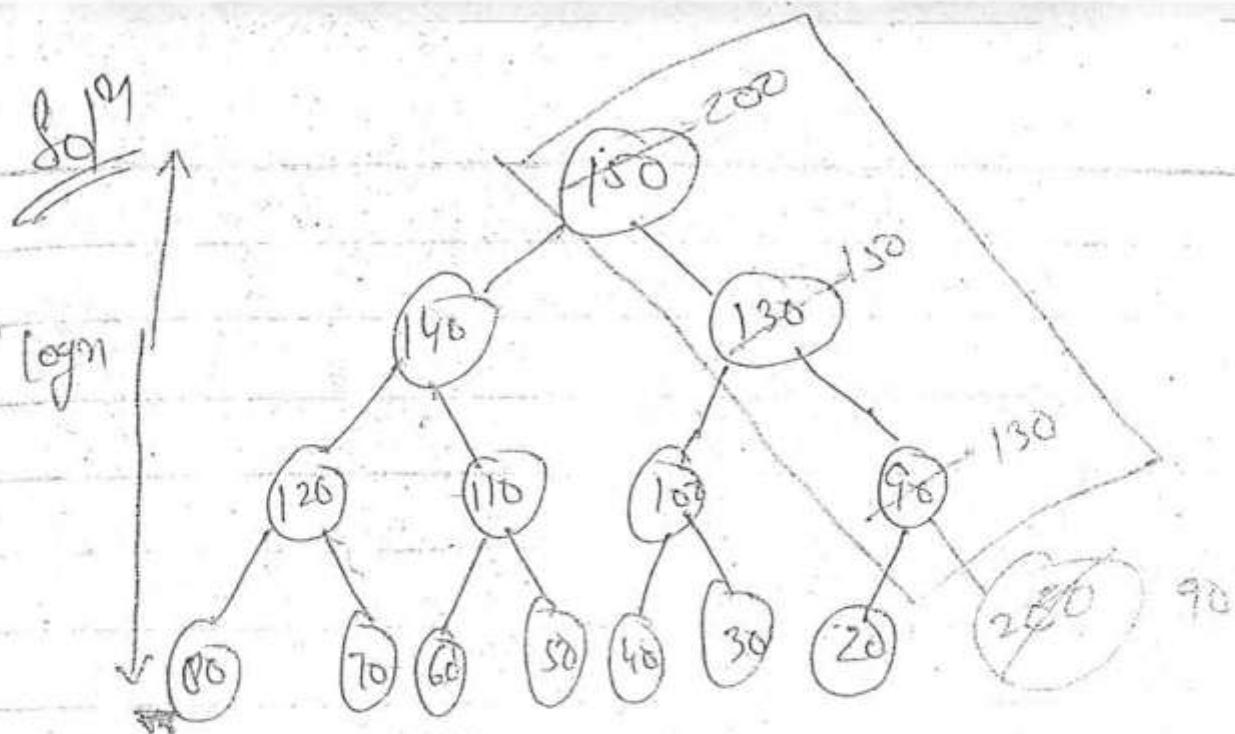
- (1) Finding $\text{Max}^m \Rightarrow O(1)$
- 2) Deleting $\Rightarrow O(\log n)$
- 3) Getting 100^{th} max^m element
 $100 \log n \Rightarrow O(\log n)$
- 4) Getting min^m element $\Rightarrow O(n)$

B Consider the process of inserting n elements into maxheap where max heap is represented using array.

Suppose we perform a binary search on the path from new leaf to root to find the position for the newly inserted element.

The T.C is.

- ~~a) $O(\log n)$~~ b) $O(\log(\log n))$
- c) $O(n)$ d) $O(n \log n)$



$\log(\log n) + \log m$

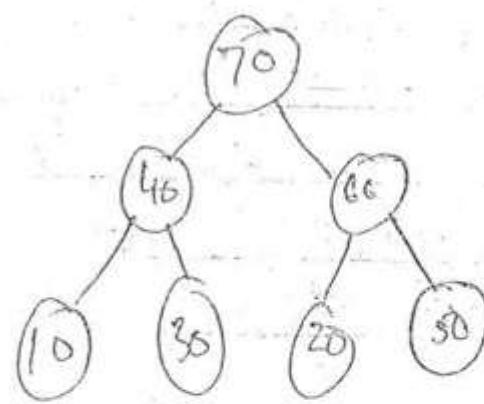
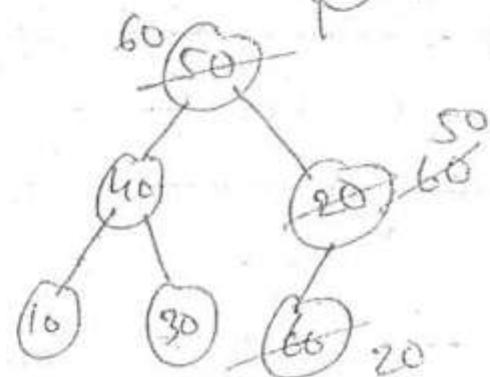
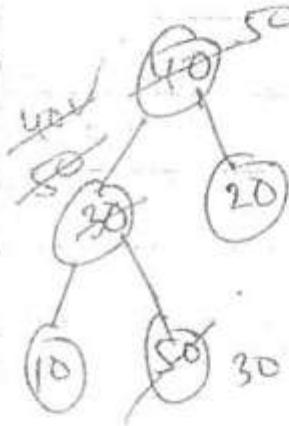
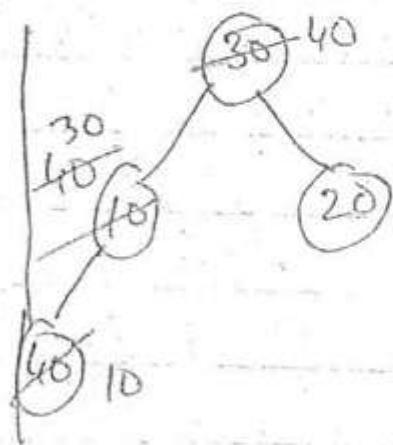
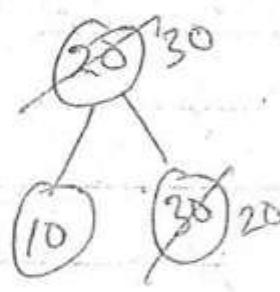
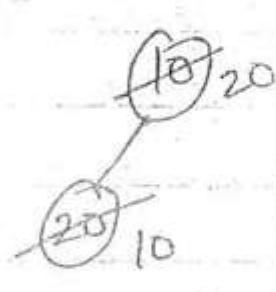
$\Rightarrow O(\log n)$

Binary search is not possible.

Build Heap Method

Note:- Using Build heap method we can create Heap tree which contains n -elements with $O(n)$ time.

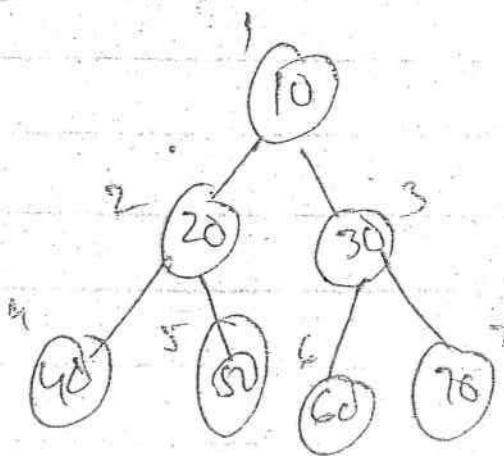
B Create max heap tree for the following elements using build heap method,
 10, 20, 30, 40, 50, 60, 70



$$2+2+2+2+1+1$$

$$= \underline{\underline{10}}$$

Build heap method



Build heap(a, n)

{
for ($i \leftarrow \lfloor \frac{n}{2} \rfloor$ down to 1)
max-heapify(a, i)

Maxheapify(a, i)

{

$l = \text{left}(i)$

$r = \text{right}(i)$

~~max~~

max = i ;

if ($a[l] > a[i]$

 max = l

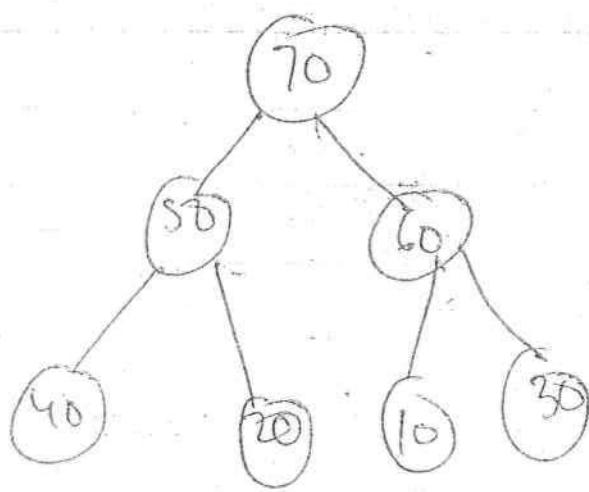
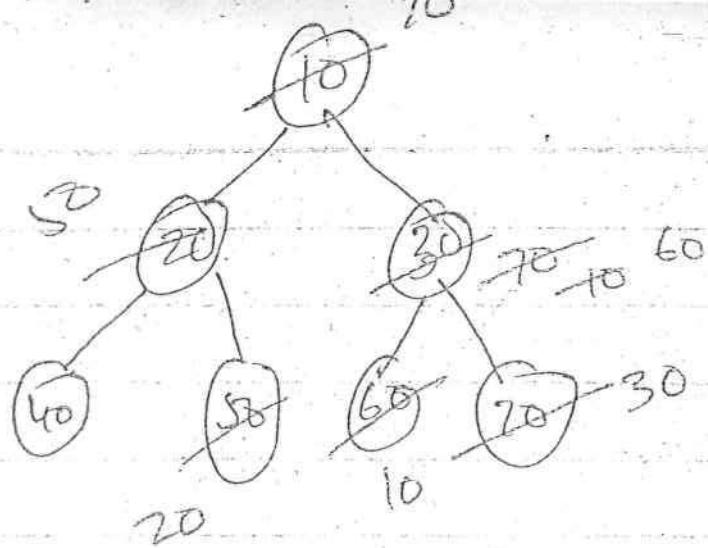
if ($a[r] > a[\text{max}]$)

 max = r ;

swap ($a[i], a[\text{max}]$)

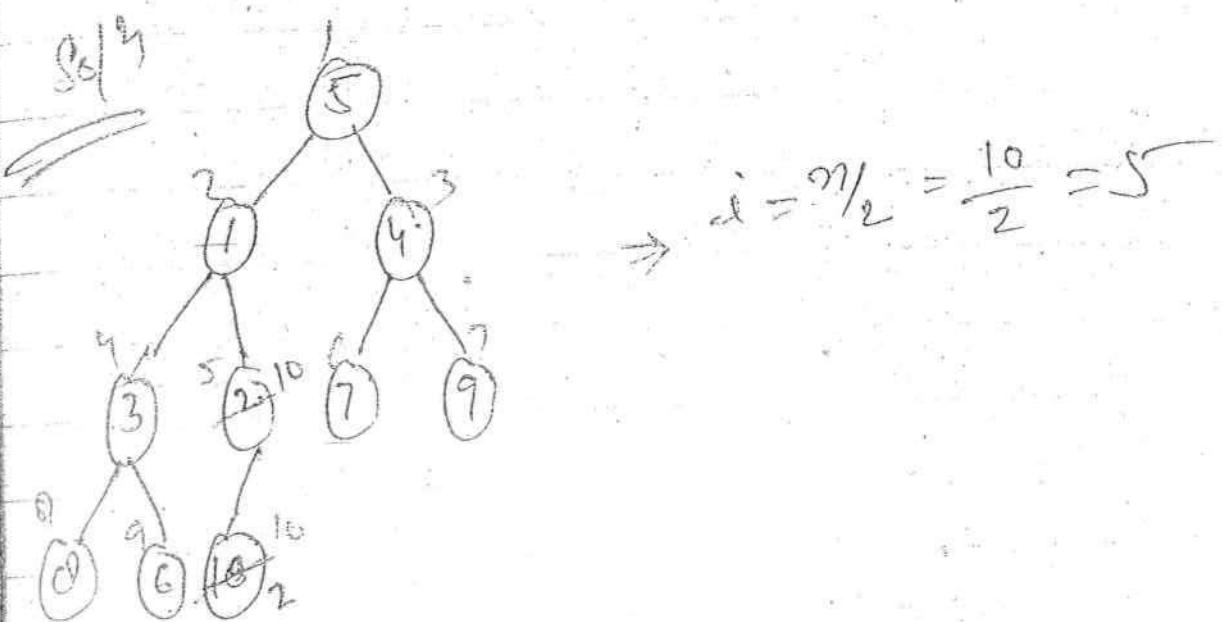
max-heapify(a, max)

3



Construct max heap tree for the following elements using Build heap method.

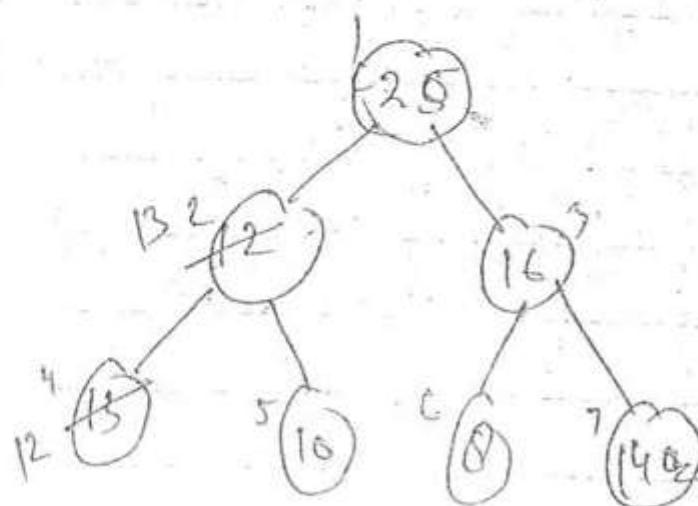
5, 1, 4, 3, 2, 7, 9, 8, 6, 10



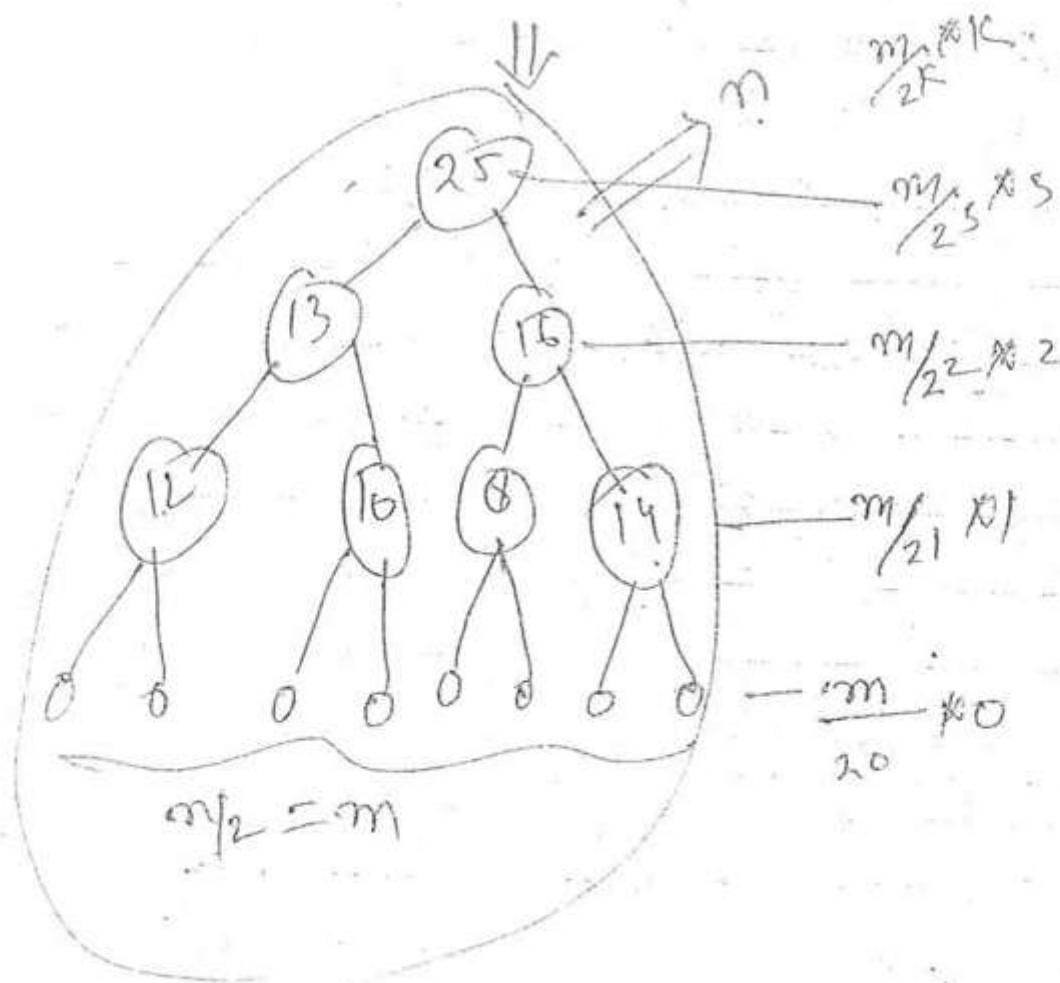
P

Construct a max heap tree for the following using build heap method.

25, 12, 16, 13, 10, 8, 14



$$d = \frac{I}{2} = \frac{3}{2}$$



$$\sum_{n=0}^{\infty} \frac{m}{2^n} \cdot h$$

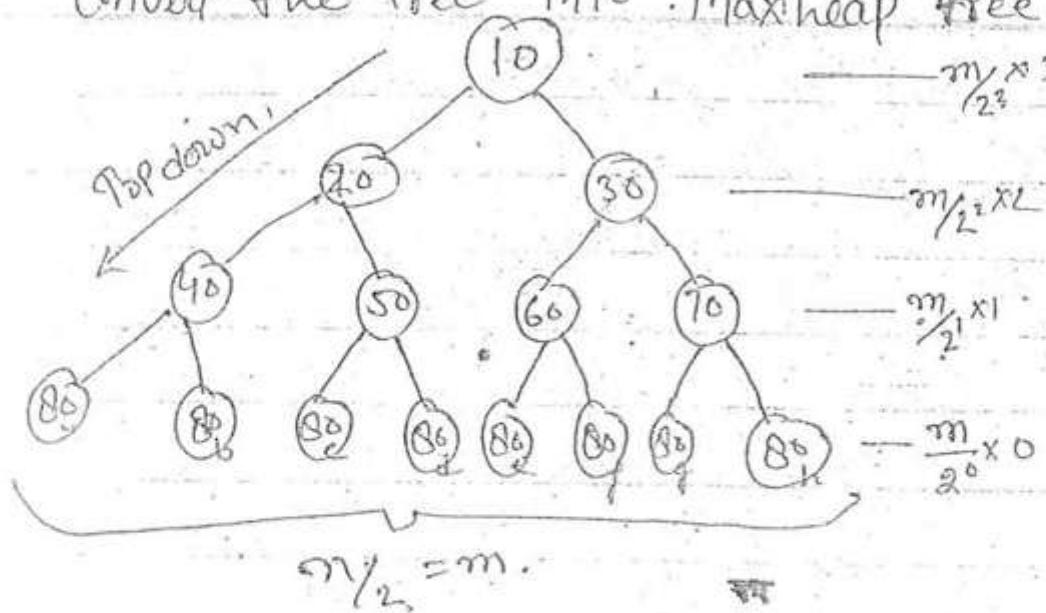
$$= m \left[\frac{0}{2^0} + \frac{1}{2^1} + \frac{2}{2^2} + \frac{3}{2^3} + \frac{4}{2^4} + \dots + \frac{k}{2^k} \right]$$

$= m [\text{constant}]$

$= \frac{m}{2} + \text{constant}$

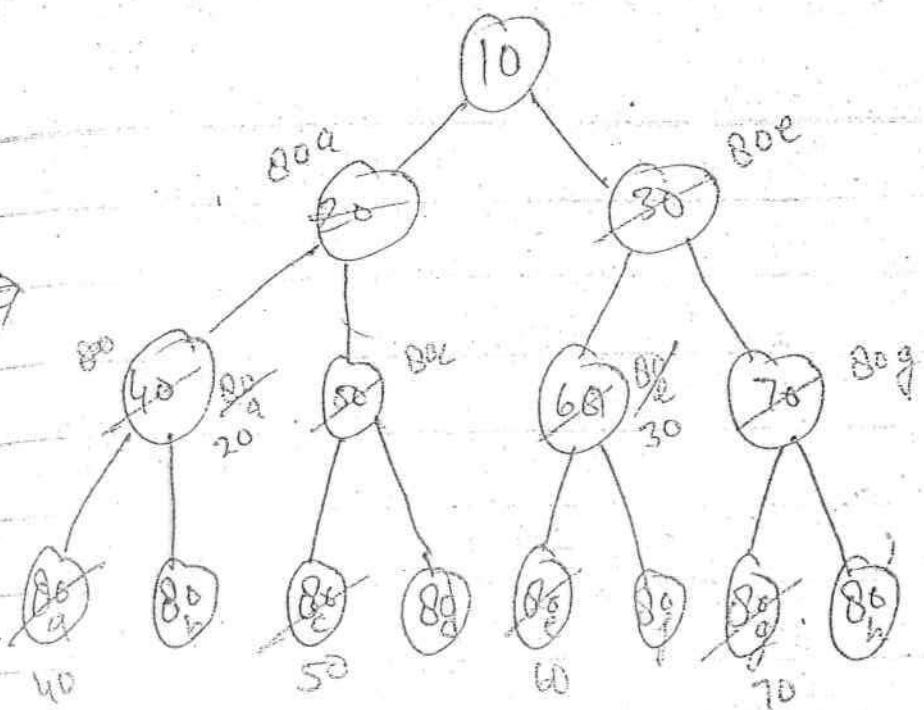
$\underline{\underline{= O(n)}}$

$\frac{P}{=}$ Convert the tree into Maxheap tree -



$$\sum_{h=0}^K \frac{m}{2^h} \cdot h.$$

$$\Rightarrow m \left[\frac{0}{2^0} + \frac{1}{2^1} + \frac{2}{2^2} + \frac{3}{2^3} + \dots \right]$$



B_0^0	B_0^1	B_0^2	B_0^3	B_0^4
---------	---------	---------	---------	---------

Heap sort not ~~stable~~ stable sorting technique.

② In-place

Construction

Selection Sort

A : $(10) \quad 55 \quad 65 \quad 5 \quad 15 \quad 25 \quad 35 \quad 20$
 $i=1$

$$\min = 1, \min = 4;$$

Pass 1 : $(5) \quad (55) \quad 65 \quad (10) \quad 15 \quad 25 \quad 35 \quad 20$
 $i=2$

$$\min = 2$$

$$\min = 4 \text{ or } 1$$

Pass 2 : $(5) \quad (10) \quad ((65) \quad 55 \quad (15)) \quad 25 \quad 35 \quad 20$
 $i=3$

$$\min = 3, 4, 5$$

Pass 3 : $(5) \quad (10) \quad (15) \quad ((55) \quad 65 \quad 25 \quad 35 \quad (20))$
 $i=4$

$$\min = 4, 6, 8$$

Pass 4 : $(5) \quad (10) \quad (15) \quad (20) \quad ((65) \quad (25) \quad 35 \quad 55)$
 $i=5$

$$\min = 5, 6$$

Pass 5 : $(5) \quad (10) \quad (15) \quad (20) \quad (25) \quad ((65) \quad (35) \quad 55)$
 $i=6$

$$\min = 6, 7$$

Pass 6 : $(5) \quad (10) \quad (15) \quad (20) \quad (25) \quad (35) \quad ((65) \quad (55))$
 $i=7$
 $\min = 8$

Part 1: (5) (10) (15) (20) (25) (38) (55) (65)

Selection sort

(1) The min^m no of swap in Selection sort is $(n-1)$

(2) max^m no of swap in Selection sort is $(n-1)$

Insertion sort

(3) Insertion sort min^m no of swap zero.

(4) max^m no of swap $O(n^2)$ for the insertion sort

Quick sort

(5) The min^m no of swap in Quick sort n

(6) " max^m " $- n^2$

Merge sort

(7)

The no of swap in merge sort = 0 swap.

(Best case and worst case)

Bubble Sort

Bubblesort (a)

```

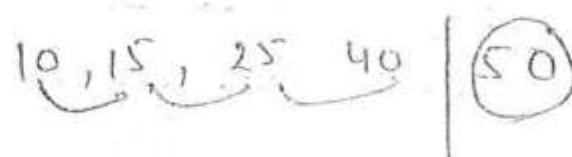
    {
        flag = 0
        for ( i = 1 , i ≤ n-1 ; i++ )
            {
                if ( flag == 0 )
                    for ( j = 1 ; j ≤ n-1 , j++ )
                        {
                            if ( a[j] > a[j+1] )
                                swap ( a[j] , a[j+1] )
                                flag = 1
                            }
                        }
                    }
    }
  
```

modification detection

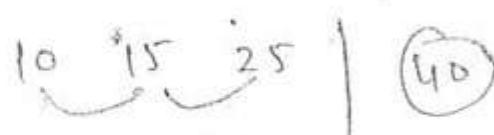
~~eg~~



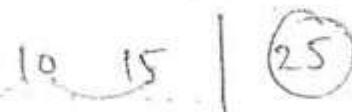
Pass 1



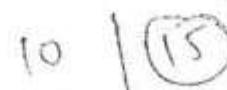
Pass 2

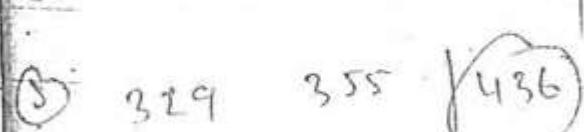
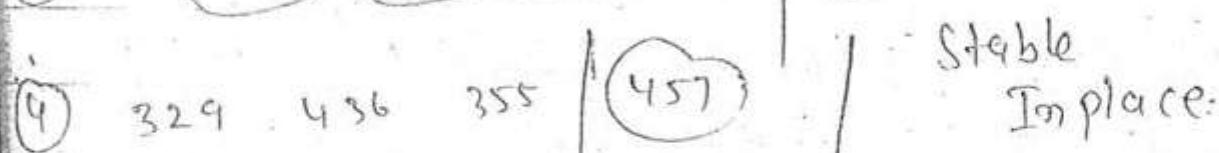
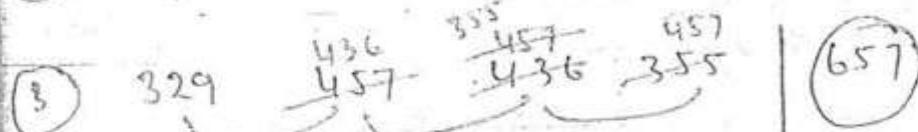
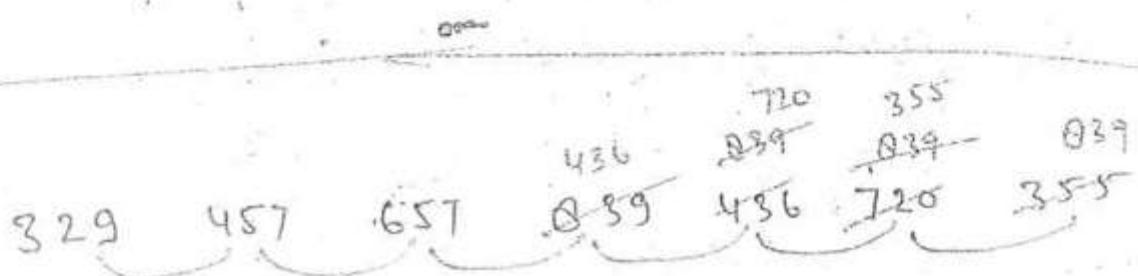
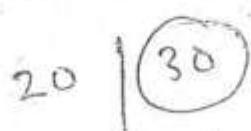
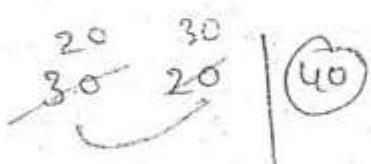
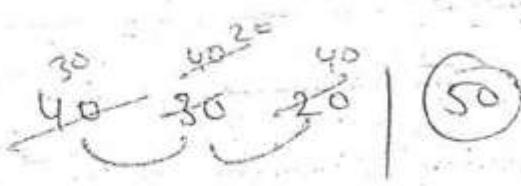
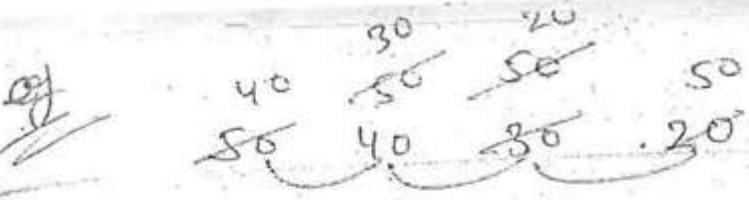


Pass 3



Pass 4





min = 0
max = $O(n^2)$

eg

10, 20, 30, 40 | 50

10 20 30 | 40 50

10 20 | 30 40 50

10 | 20 30 40 50

10 20 30 40 50

if no of flag	with flag
\downarrow	\downarrow
$O(n^2)$	$O(n^2)$
$\sim (n^2)$	$\sim (n)$
\downarrow	
$O(n^2)$	

No:

(5) Optimal Solution.

• (3) Solution space \Rightarrow All possible solⁿ with n input
• Feasible Solution \Rightarrow It is one of the solⁿ in
solⁿ space which will satisfied
our condition.

Optimal solⁿ \Rightarrow It is one of the feasible
solⁿ which optimizes our target:

minimizing cost
maximizing profit

Control Abstraction of Greedy

Greedy (q, n)

Solution = ϕ

for ($i=1$; $i \leq n$; $i++$)

$x = \text{Select}(q, n)$

if (Feasible(x))

add (x, solution)

$f(n)$

L3

return (Solution)

Flow of Control

Select
Feasible
↓ Add

It is the flow of control
But each & every step is iterative
It can choose any student
check any condition --
so it is called as control abstraction
of greedy.

Applications of Greedy

- (1) Greedy Knapsack
- (2) Job sequencing with Deadlines
- (3) Minimum Cost spanning tree
- (4) Huffman Coding
- (5) Optimal merge pattern
- (6) Single source shortest Path
 - (i) Dijkstra's algo
 - (ii) Bellman-Ford algo

Greedy Knapsack

(CPA)

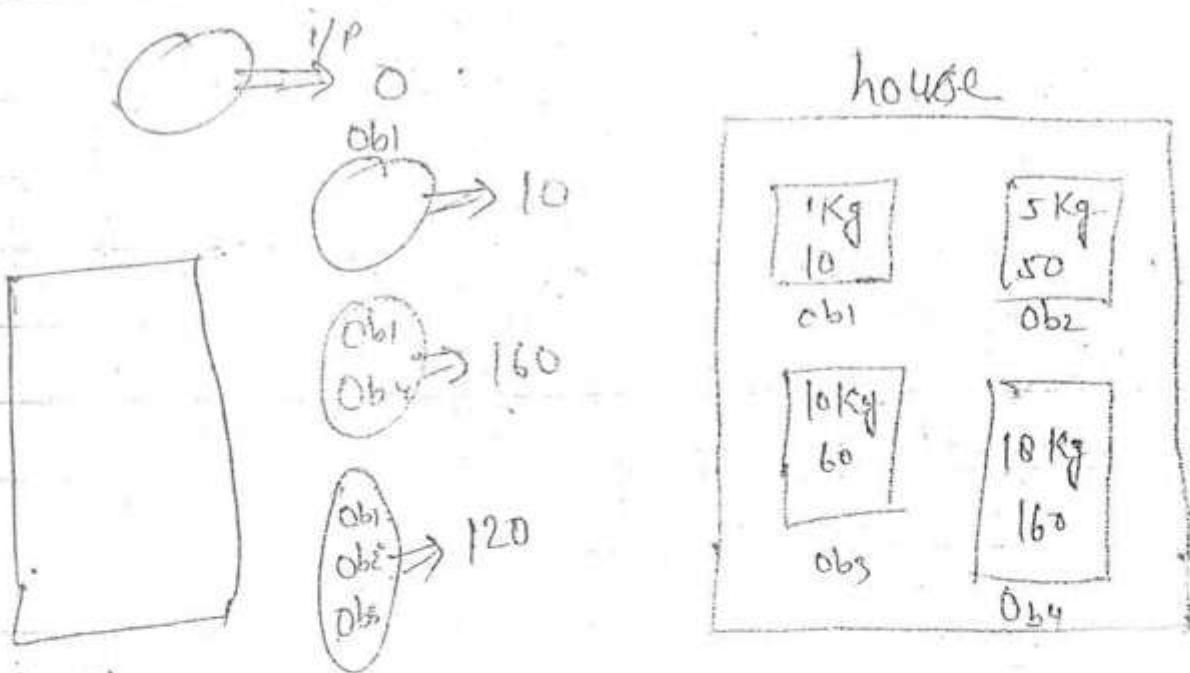
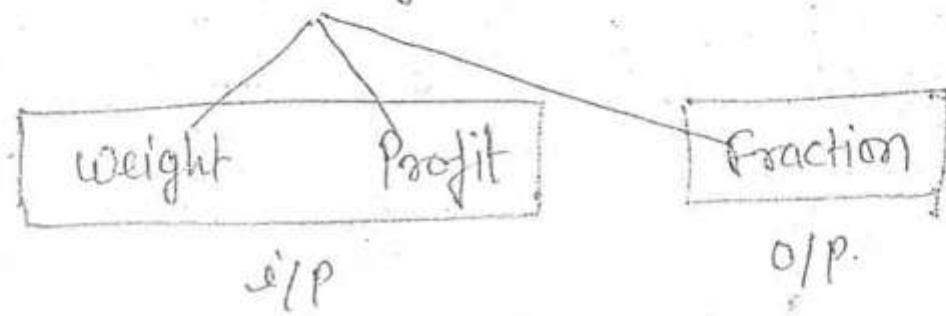
Real Knapsack

(OR)

Fractional Knapsack

$\xrightarrow{x} \xrightarrow{a}$

n-object.



bag

$$C = 20 \text{ Kg}$$

4 - object
(n) $C = 20$

Object Ob₁ Ob₂ Ob₃ Ob₄
Weight w₁ w₂ w₃ w₄
Profit p₁ p₂ p₃ p₄

Problem

$$\sum_{i=1}^n w_i > C$$

Feasible

$$\sum_{i=1}^n w_i * u_i \leq C$$

Optimal

$$\sum_{i=1}^n p_i u_i = \max$$

$$n=3 \quad C=20$$

object	obj ₁	obj ₂	obj ₃
Profit	25	24	15
Weight	10	15	10

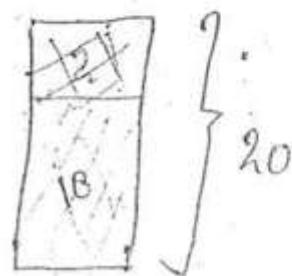
Greedy about profit

$$(1 \quad \cancel{2/5} \quad 0)$$

$$= 1 \times 25 + \frac{2}{15} \times 24 + 0 \times 15$$

$$= 25 + 32$$

$$= 28.2$$



future

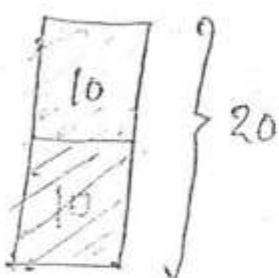
Greedy about weight

$$(0 \quad \cancel{10/15} \quad 1)$$

$$= 0 \times 25 + \frac{10}{15} \times 24 + 1 \times 15$$

$$= 0 + 16 + 15$$

$$= 31$$



present

Greedy about both (P & W)

Obj₁: 18 \Rightarrow 25.

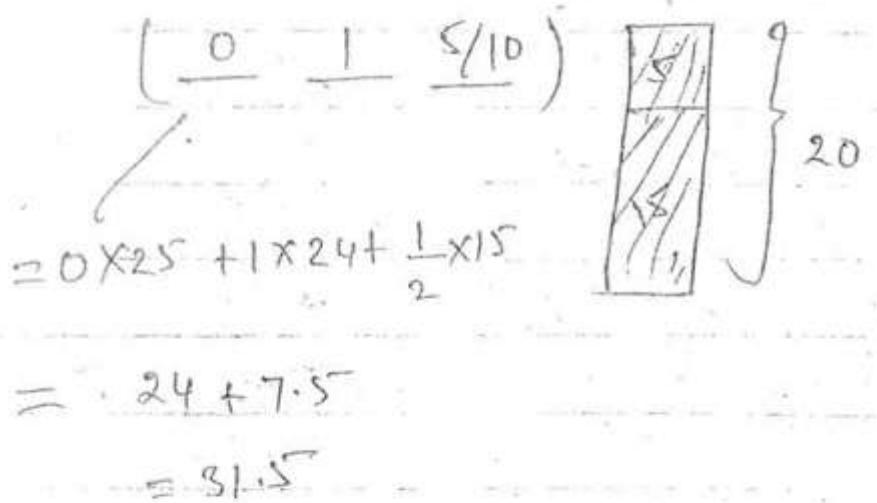
$$1 \Rightarrow \frac{25}{18} = 1.3$$

Obj₂: 15 \Rightarrow 24

$$1 \Rightarrow \frac{24}{15} = 1.6$$

Obj₃: 10 \Rightarrow 15

$$1 \Rightarrow \frac{15}{10} = 1.5$$



Note:- In Greedy Knapsack we will also get optimal soln by considering both profit and weight.

Q

$$n=5 \quad C=12$$

Object Ob₁ Ob₂ Ob₃ Ob₄ Ob₅

Profit 5 2 2 4 5

Weight 5 4 6 2 1

$$\text{Ob}_1: 5 \Rightarrow 5$$

$$1 = 5/5 = 1 \quad (3)$$

$$\text{Ob}_2: 4 \Rightarrow 2$$

$$1 \Rightarrow 2/4 = .5 \quad (2)$$

$$\text{Ob}_3: 6 \Rightarrow 2$$

$$1 \Rightarrow 2/6 = .33 \quad (5)$$

$$\text{Ob}_4: 2 \Rightarrow 4$$

$$1 \Rightarrow 4/2 = 2 \quad (2)$$

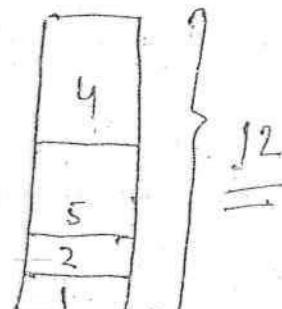
$$\text{Ob}_5: 1 \Rightarrow 5$$

$$1 = 5/1 = 5 \quad (1)$$

(1 1 0 1 1)

$$= 1 \times 5 + 1 \times 2 + 0 \times 2 + 1 \times 5 + 1 \times 4$$

$$= 16$$



$$n=7 \quad C=15.$$

Object $o_{b1} o_{b2} o_{b3} o_{b4} o_{b5} o_{b6} o_{b7}$

Profit 10 5 15 7 6 10 3

Weight 2 3 5 7 1 4 1

then find out Optimal Sol^n

Sol^n

$$1 \rightarrow \frac{10}{2} = 5 \rightarrow \textcircled{2}$$

$$2 \rightarrow \frac{5}{3} = 1.6 \rightarrow \textcircled{5}$$

$$3 \rightarrow \frac{15}{5} = 3 \rightarrow \textcircled{4}$$

$$4 \rightarrow \frac{7}{1} = 1 \rightarrow \textcircled{7}$$

$$5 \rightarrow \frac{7}{1} = 6 \rightarrow \textcircled{1}$$

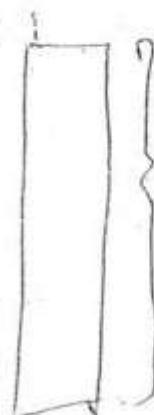
$$6 \rightarrow \frac{10}{4} = 4.5 \rightarrow \textcircled{5}$$

$$7 \rightarrow \frac{3}{1} = 3 \rightarrow \textcircled{3}$$

$$\left(\frac{1}{1} \frac{2}{2} \frac{1}{3} \frac{0}{4} \frac{1}{5} \frac{1}{6} \frac{1}{7} \right)$$

$$= 1 \times 10 + \frac{2}{3} \times 5 + 1 \times 15 + 0 \times 7 + 1 \times 6 + 1 \times 10 \\ + 1 \times 3$$

$$= \underline{\underline{55.33}}$$



~~15/4~~

~~12/3~~

~~10/2~~

~~3/1~~

Algo

(1) $\text{for } (i=1; i \leq n; i++)$ $\rightarrow O(n)$
 $a[i] = p_i/w_i$

(2) Arrange Array in A in decreasing order of
 p_i/w_i ratio. $\rightarrow O(n \log n)$

(3) $\text{for } (i=1, i \leq n, i++)$ $\rightarrow O(n)$
 {
 if ($C \neq 0$)
 {
 $P = P + p_i$
 $C = C - w_i$
 } return (P)
 } take one by one
 object until capacity
 of Knapsack becomes
 zero.
 $\rightarrow O(n \log n)$

(3) $\text{for } (j=1; j \leq n; j++)$
 {
 if ($C > 0$)
 {
 if ($C > w_j$)
 {
 $P = P + p_j, C = C - w_j$
 }
 $P = P + p_j, C = 0$
 }
 }

($\frac{0}{1} - -$)

$\frac{1}{2} \rightarrow$ Solution space for greedy
 $\frac{1}{10}$ Knapsack is infinite

$\frac{1}{10}$

14/7/11

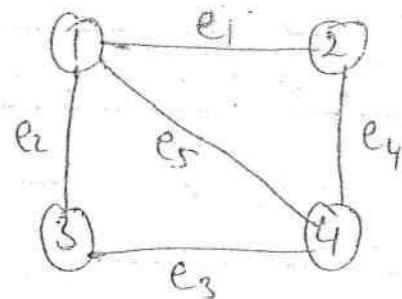
Minimum Cost Spanning Tree

$G(V, E)$

V = Set of vertices.

E = Set of edges.

eg

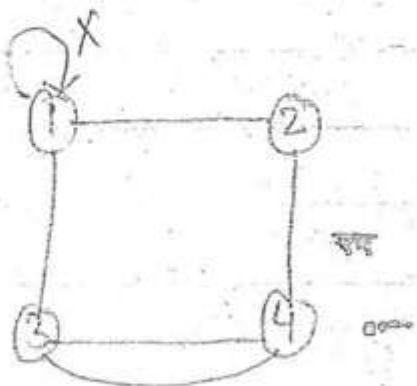


$$V = \{1, 2, 3, 4\}$$

$$E = \{e_1, e_2, e_3, e_4, e_5\}$$

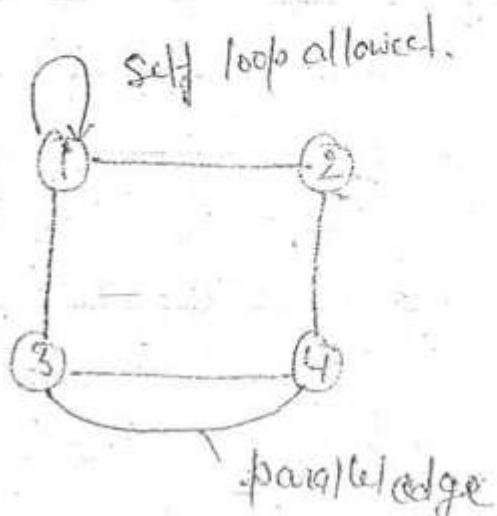
Types of Graph

Simple graph



no-self loop
parallel edge.

Multi Graph.



self loop allowed.

parallel edge

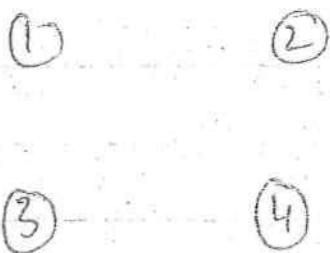
Simple Graph

Types of Graph

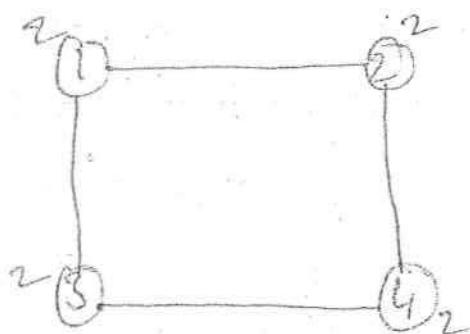
- (i) Null Graph.
- (ii) Regular "
- (iii) Complete "
- (iv) "

edges.

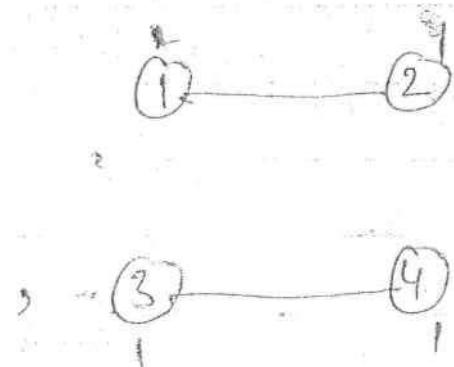
Null Graph



Regular Graph

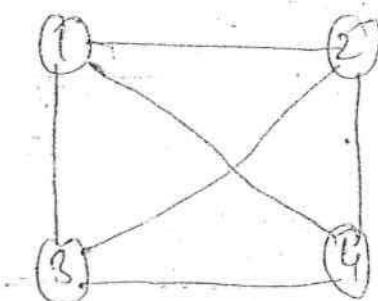


2-regular graph

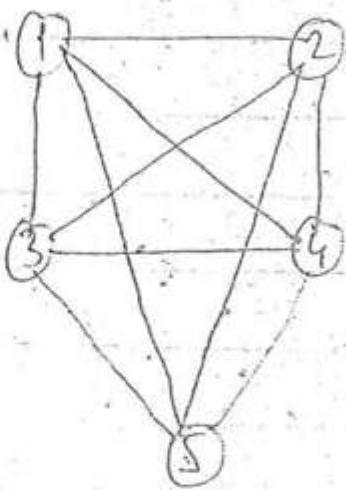


1 regular graph

Complete Graph



K_4 Complete graph



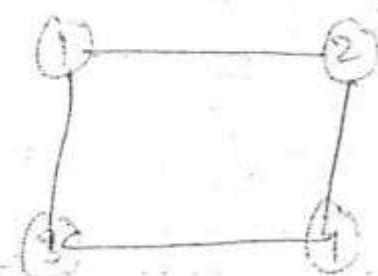
K5 Graph

* Maxⁿ no of edges in n vertex simple graph.
~~Total edge~~ is equal to no of edges in comple graph.

$$E(K_n) = \frac{n(n-1)}{2}$$

* The no of edges in a n vertex K regular

$$= \frac{n \times k}{2}$$

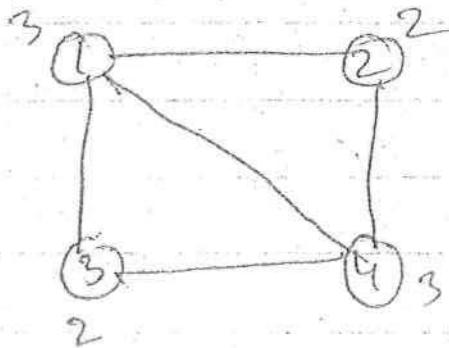


K4 graph

(n-1)

every comple graph is
a regular graph.

Sum of degree theorem (undirected)



$$3+2+2+3 = 2 \times |E|$$

$$\sum_{i=1}^n \text{degree}(v_i) = 2 \times |E|$$

Types of Graphs

undirected graph directed graph



(1,2) ✓

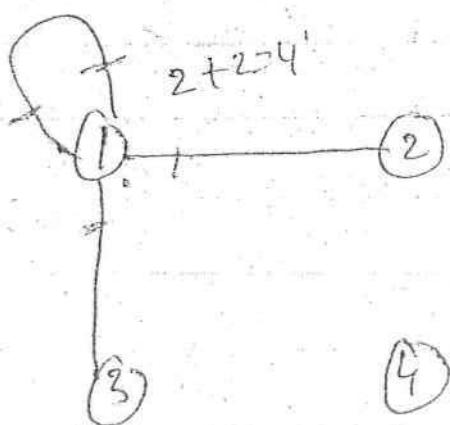
(2,1) ✓



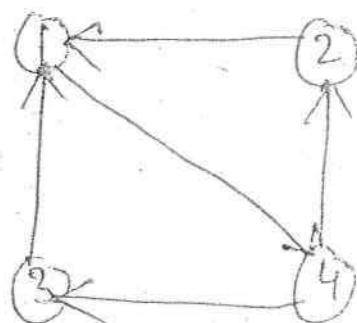
(1,2) ✓

(2,1) ✗

Self loop is counted by degree 2



Directed graph (Sum of degree theorem)



In degree (+)

$$1^+ = 2$$

$$2^+ = 1$$

$$3^+ = 1$$

$$4^+ = 1$$

5

Outdegree (-)

$$1^- = 1$$

$$2^- = 1$$

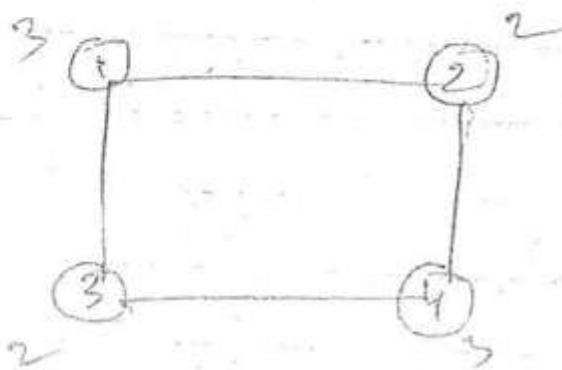
$$3^- = 1$$

$$4^- = 2$$

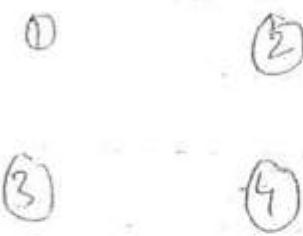
5

$$\sum_{i=1}^n (v_i)^+ = \sum_{i=1}^n (v_i)^- = |E|$$

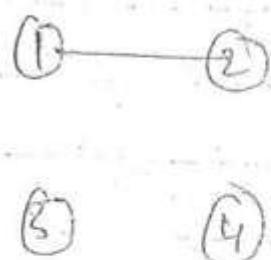
Hand Shaking Lemma



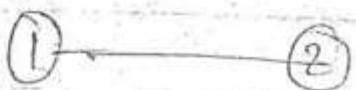
In a given simple graph no of vertices with odd degree is always even.



0 vertices with odd degree



2 vertices with odd degree.



4 - Vertices with odd degree.

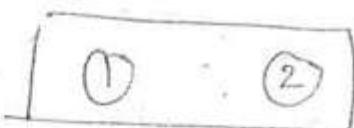
~~Q~~ How many ^{different} simple graphs are possible with n -vertices ?

Soln

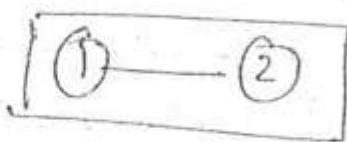
$n=1$



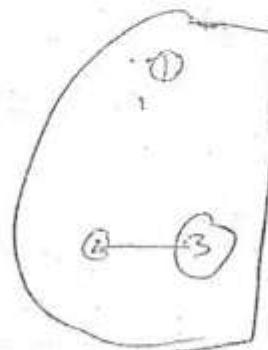
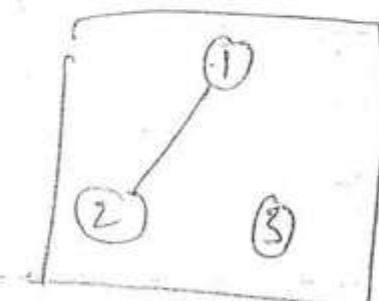
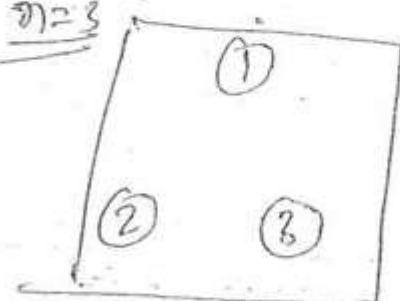
$n=2$

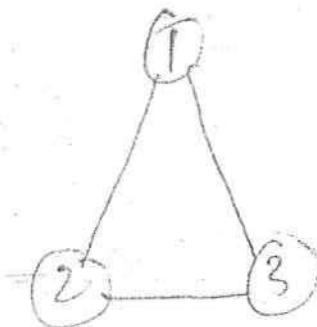
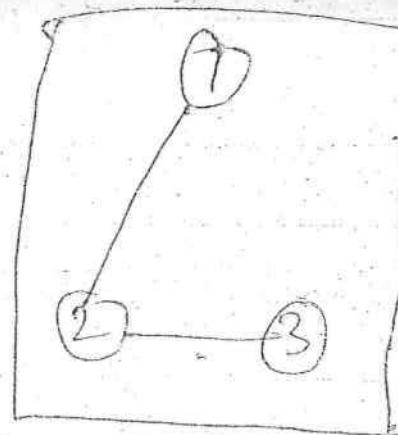
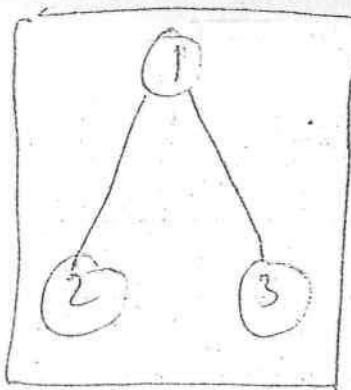
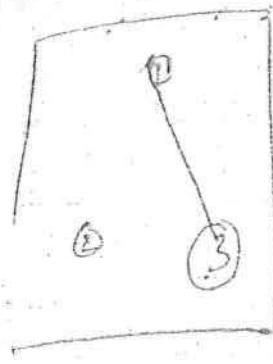


$\Rightarrow 2$



$n=3$





$\Rightarrow \emptyset$

Total simple graph if node is \emptyset

a) n

b) 2^n

c) 2^{n-1}

d) $2^{\frac{n(n-1)}{2}}$

The no of simple graph with n vertices
is equal $\frac{n(n-1)}{2}$ possibly

where $\frac{n(n-1)}{2}$ is max no of edge
written vertices.

DESIGN
ANALYSIS
&
ALGORITHM

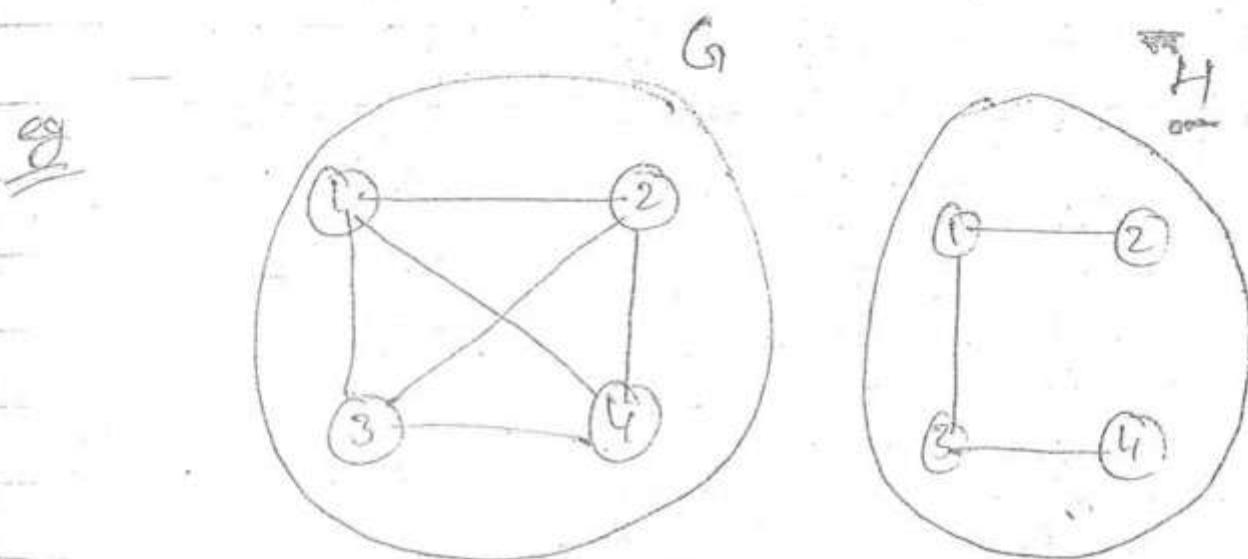
PART-II

(1)

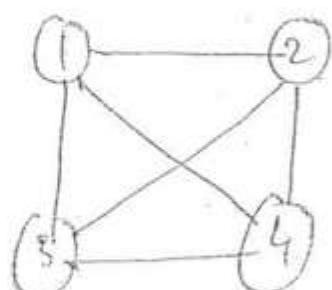
Spanning Tree

A connected subgraph H of a graph G is said to be spanning tree iff:

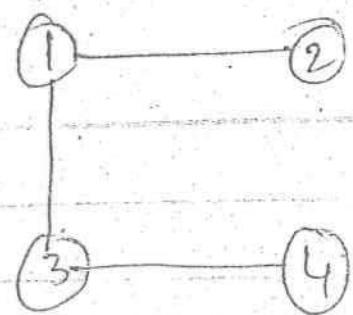
- i) It should contain all the vertices of G .
- ii) " " " $(n-1)$ edges if G contain n vertices.



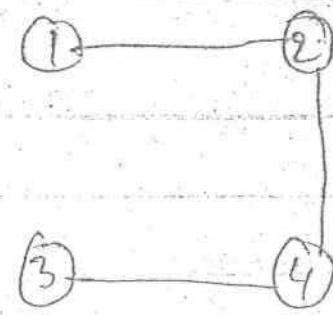
Find all possible different spanning trees for the following graph.



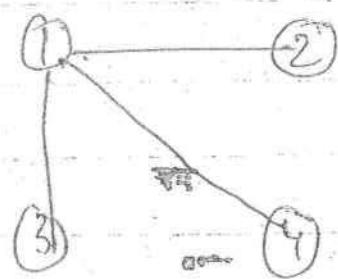
(5)



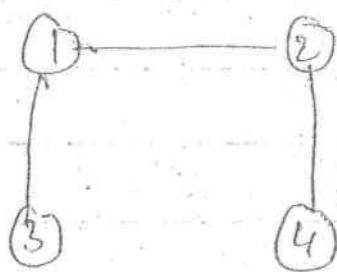
(i).



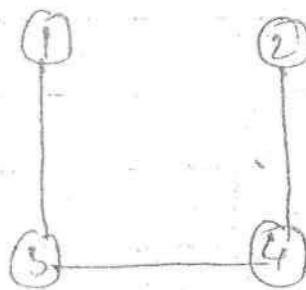
(ii)



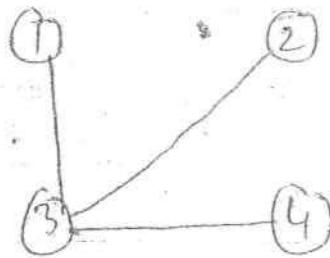
(iii)



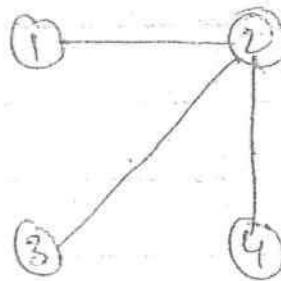
(iv)



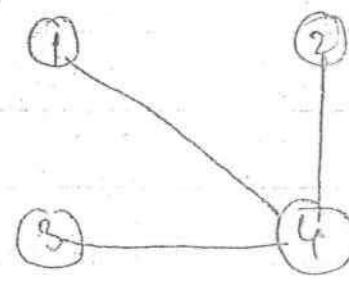
(v)



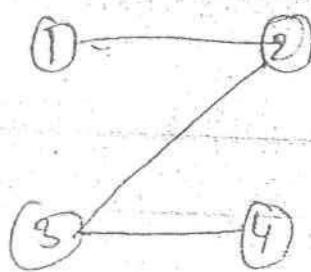
(vi)



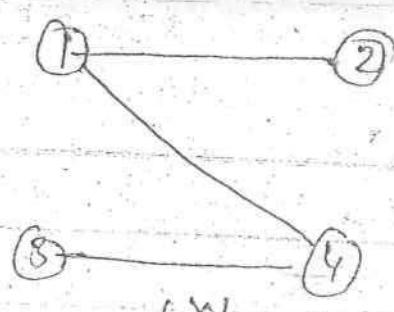
(vii)



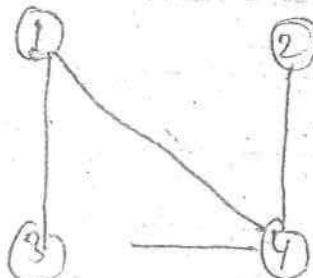
(viii)



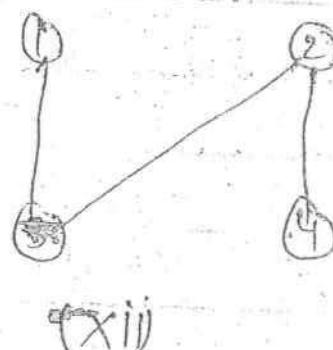
(IX)



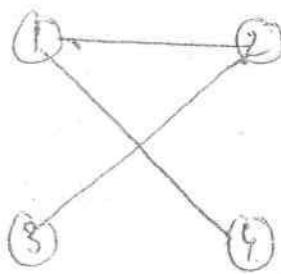
(X)



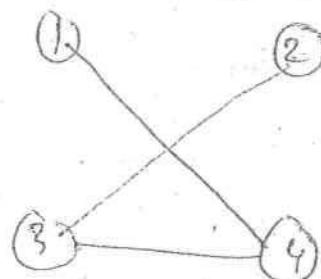
(XI)



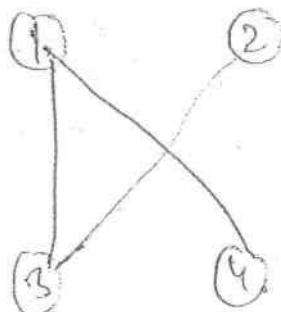
(XII)



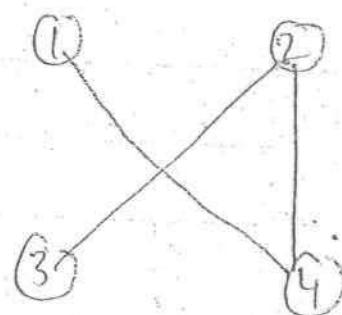
(XIII)



(XIV)



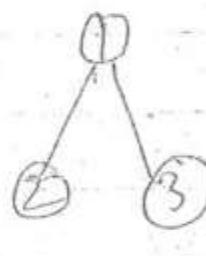
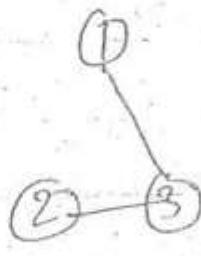
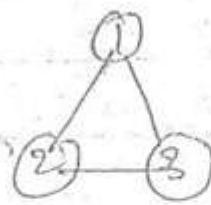
(XV)



(XVI)

"Find all

K_3



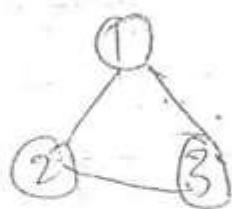
$$ST(K_3) = s \cdot (3^{3-2}) \quad ST(K_3) = 125 (5^{5-2})$$

$$ST(K_4) = 16 \quad (4^{4-2}) \quad ST(K_n) = n^{n-2}$$

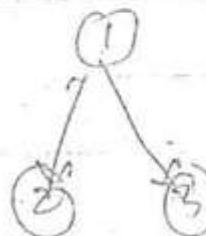
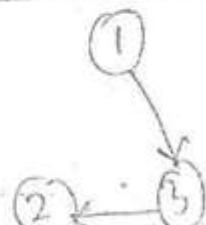
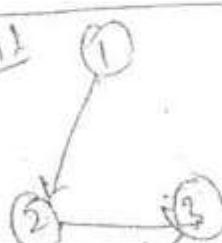
The no. of possible spanning trees in complete graph with n vertices.

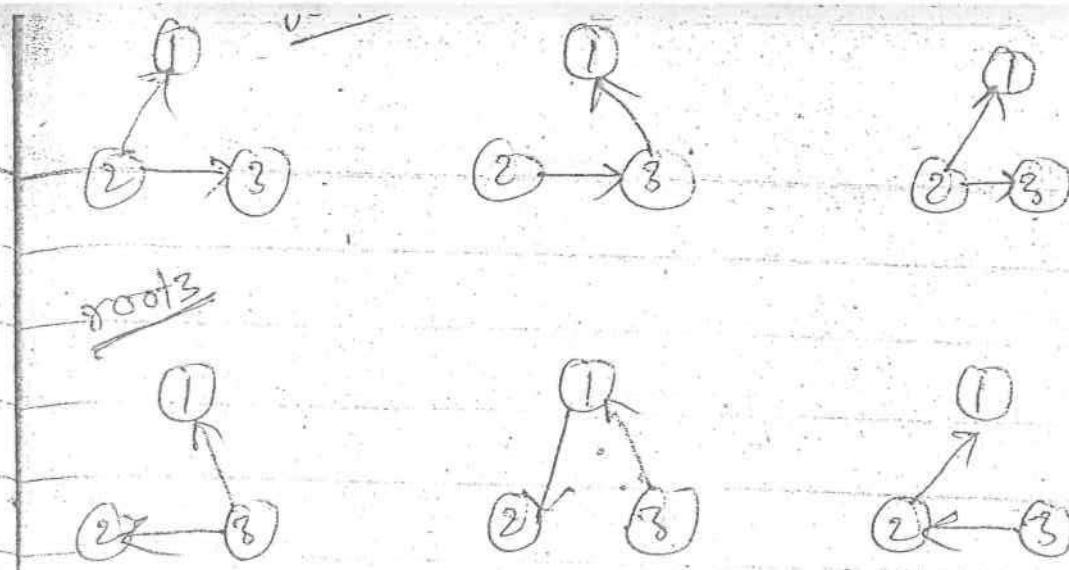
$$= n^{n-2}$$

K_2 =



Root 1



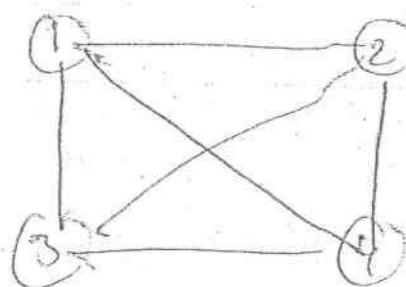


Nodes \rightarrow The no. of root a spanning tree in a graph with n vertices complete graph.

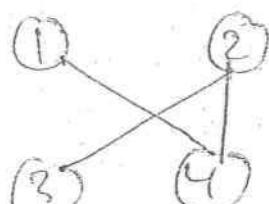
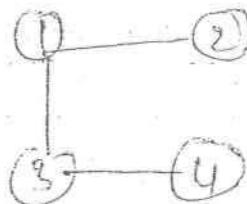
$$= n \times (n^{n-2})$$

#

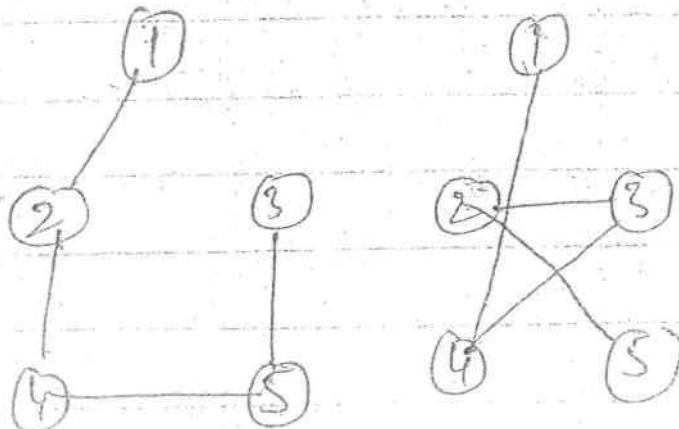
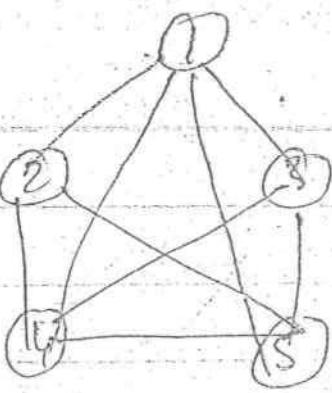
K₄



edge disjoint ST



K_5



$$K_3 = 1 \Rightarrow \left\lfloor \frac{3}{2} \right\rfloor \Rightarrow 1$$

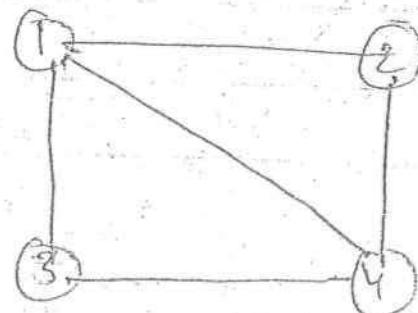
$$K_4 = 2 \Rightarrow \left\lfloor \frac{4}{2} \right\rfloor \Rightarrow 2$$

$$K_5 = 2 \Rightarrow \left\lfloor \frac{5}{2} \right\rfloor \Rightarrow 2$$

$$K_6 = 3 \Rightarrow \left\lfloor \frac{6}{2} \right\rfloor \Rightarrow 3$$

$$\text{EDST}(K_n) = \left\lfloor \frac{n}{2} \right\rfloor$$

Q. How many spanning tree are there for the following graph.



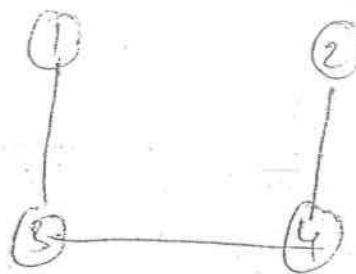
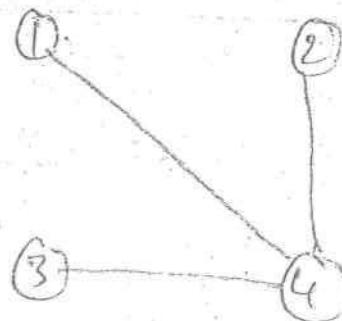
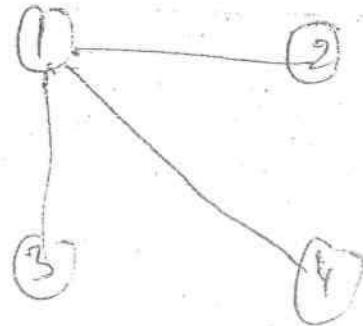
a) 16

b) 10.

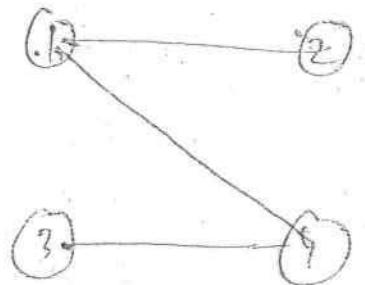
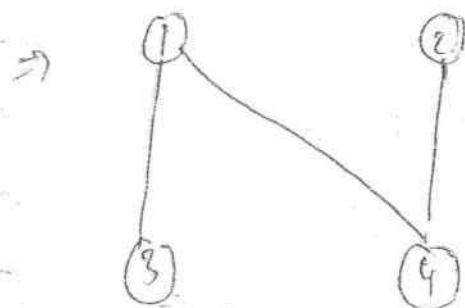
c) 8 ✓

d) 12

Sol.



4



2

Kirchhoff Theorem

D) Find adjacency matrix M for the given graph.

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 4 & 1 & 1 & 1 \end{bmatrix}$$

\Rightarrow Dence graph
(more edge)

Adjacency Matrix $\Rightarrow \underline{\mathcal{O}(V^2)}$

Adjacency Graph Matrix M is good for
Those Graph which has more edges. Such
graph is called Dence Graph (more edge)

Those Graph which has less edges
such graph is called sparse graph (less edge)

③

(i) replace all non diagonal 0's by 1's.

(ii) replace all diagonal 0's by its equal degree.

$$M = \begin{vmatrix} 1 & 1 & 2 & 3 & 4 \\ 3 & -1 & -1 & -1 & \\ \hline 2 & -1 & 2 & 0 & -1 \\ 3 & -1 & 0 & 2 & -1 \\ 4 & -1 & -1 & -1 & 3 \end{vmatrix}$$

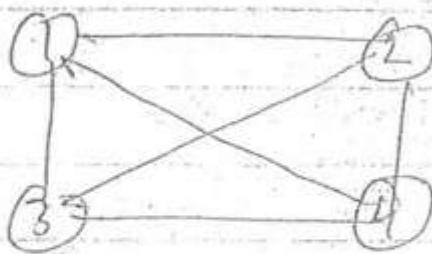
③ Find co-factor of any element which will give no of spanning tree.

$$\begin{vmatrix} 2 & 0 & -1 \\ 0 & 2 & -1 \\ -1 & -1 & 3 \end{vmatrix}$$

$$2(6-1) - 0(\cdot) - 1(0+2)$$

$$= 10 - 2 = 8$$

Find no of spanning tree for the following graph using Kirchhoff Theorem



~~SIⁿ~~

$$\textcircled{1} \quad M_2 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

$$\textcircled{2} \quad M = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

\textcircled{3} find cofact of M_{11}

$$= (-1)^{1+1} \begin{vmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{vmatrix} = \underline{\underline{16}}$$

for a given graph more than 1 ~~path~~
MST is may be possible but min^m cost
be same ..

If the given graph contain Distinct
edge weight then it give max^m ~~one~~
One Max^m set.

Q Let G be an undirected connected graph.
~~connected~~ with distinct edge weight.
Let e_{\max} be the max^m weight and
 e_{\min} be " min^m weight then
which one of the following false.

- (a) Every min^m MST must contain e_{\min} .
- (b) If e_{\max} is in MST, then its removal
must disconnect G .
- (c) No min^m MST contain e_{\max}
- (d) G ~~contain~~ has unique MST

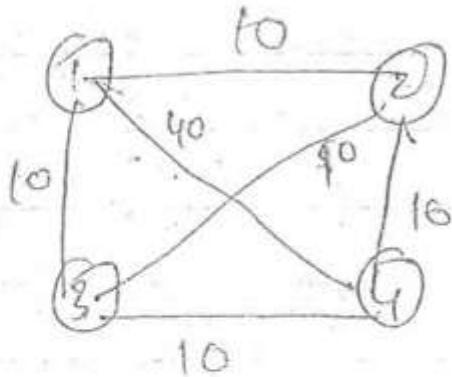
P Let G be an undirected connected graph with n vertices.

Let w be the min^m weight among all edge weight and e be a specific edge with weight w .

then which one of the following is false.

- a) e may be there in MST
- b) If e is not in MST than all edges have same weight w in that cycle.
- c) If every ϕ min^m est should contain an edge with w .
- d) Every MSCT should contain edge e .

Sol^m



$$w = 10$$

Q An undirected graph G_1 has n -nodes. Its adjacent matrix is given by an $n \times n$ square matrix, whose elements are defined as follows.

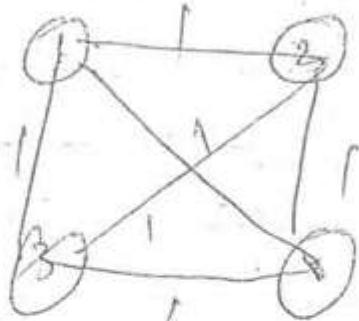
- i) all diagonal elements are 0's. (means)
- ii) all non-diagonal elements are 1's. (complet graph)

True ?

- a) G_1 has no MST
- b) G_1 has unique MST, each of cost $n-1$
- c) G_1 has multiple MST with different cost
- d) G_1 has multiple MST each of cost $n-1$

sol:

K_4



$16 \Rightarrow$ MST

each of cost

= 3

$K_n \Rightarrow n^{n-2}$ MST

each of cost $n-1$

\Rightarrow Let T and T' be 2-spanning tree of connected graph G .

Suppose that an edge e is in T but not in T' and all other remaining edges are same. Then which one of the following is true about T & T' after performing following 2-operations.

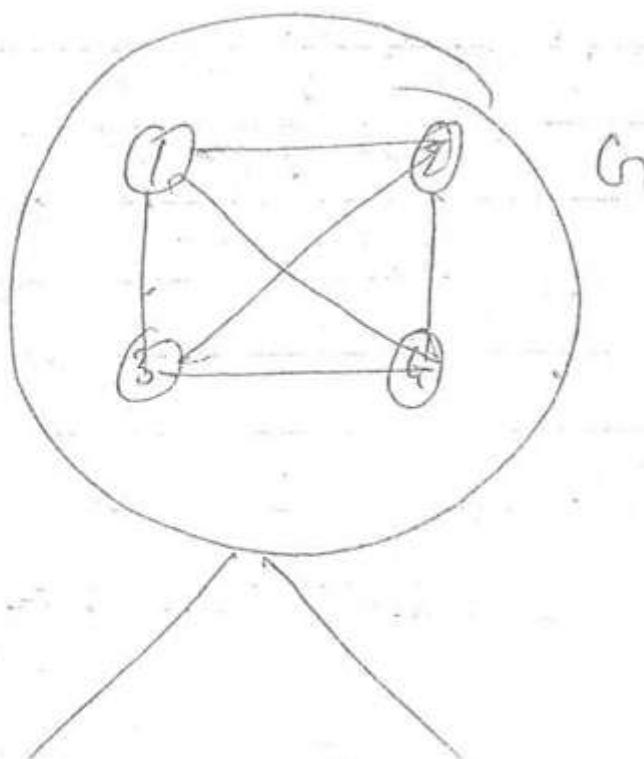
(i) $(T - e) \cup e'$

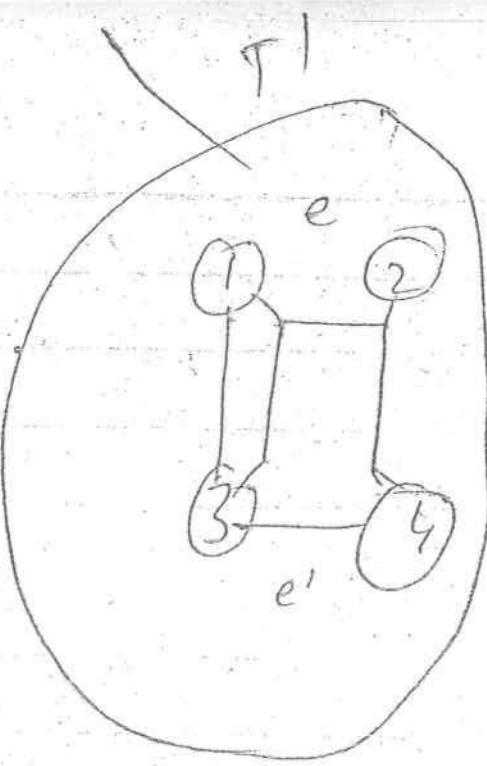
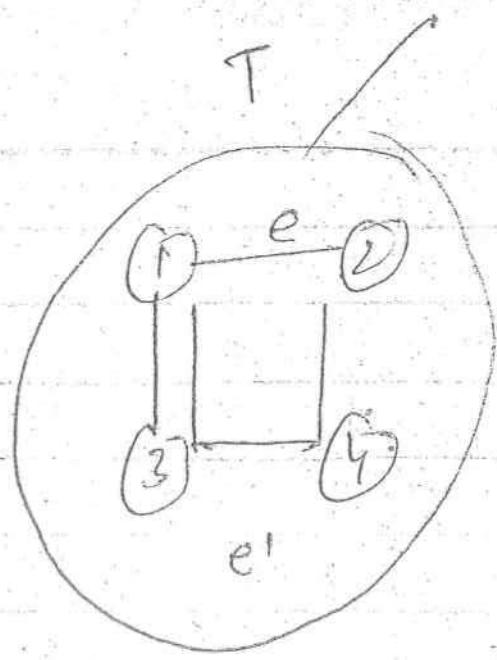
(ii) $(T' - e') \cup e$

~~(iii)~~ both T and T' are still ST

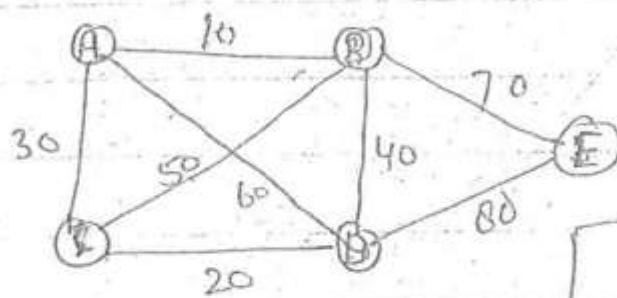
- b) only T is ST
- c) only T' is ST
- d) both are not.

~~soln~~

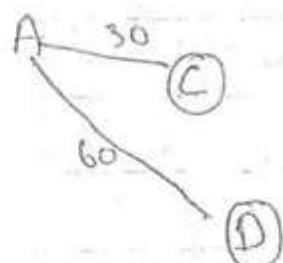
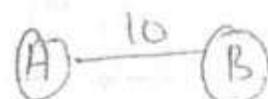
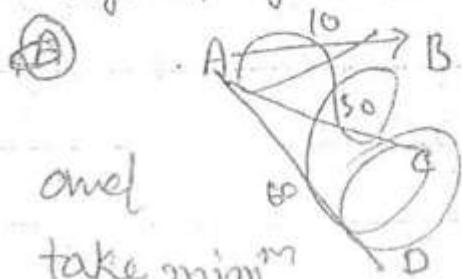




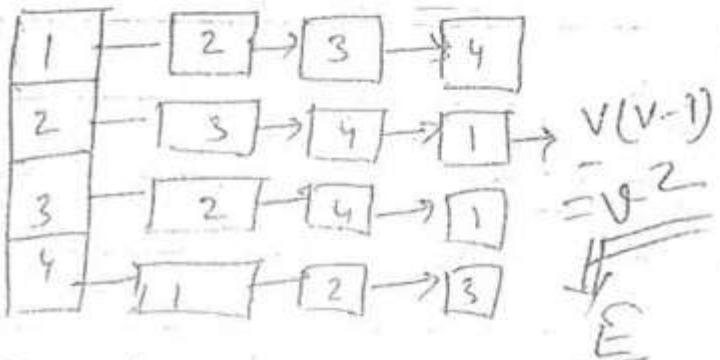
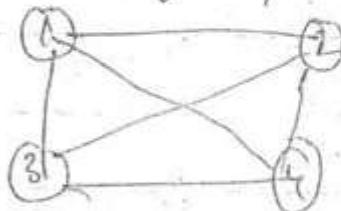
Prim's Algo



(i) Take any vertex & find adjacent of that vertex



Adjacency list representation



① ② no of edges

③ ④



$O(V)$

$O(V+E)$

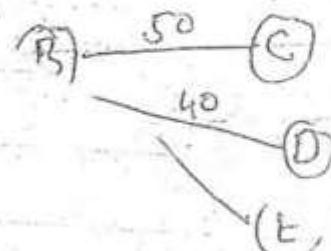
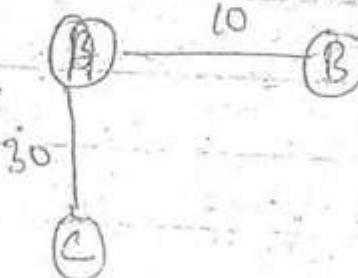
④

No

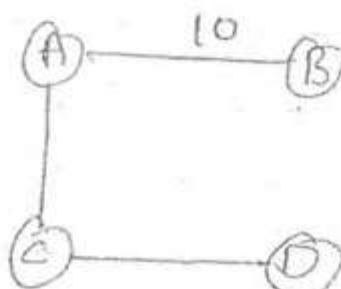
(C)

Pr.

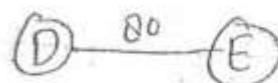
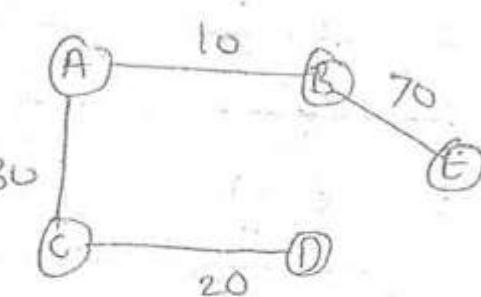
of the new vertex find adjacent and min^m
among vertices & previous



- ③ for the new vertex find adjacent and take min^m
ii) of three adjacent and previous all adjacent.



(4)



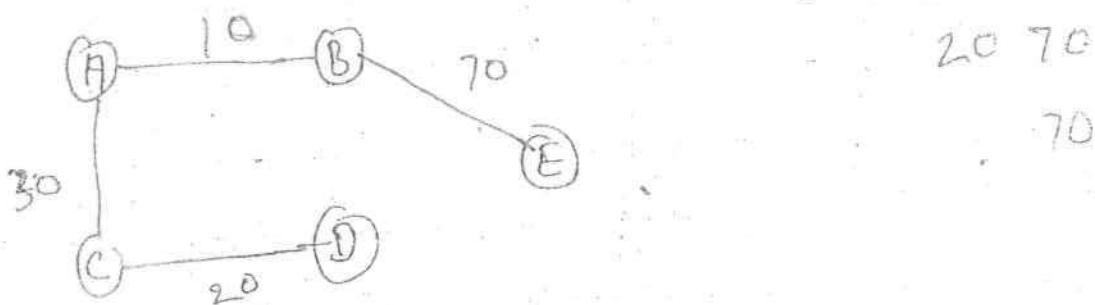
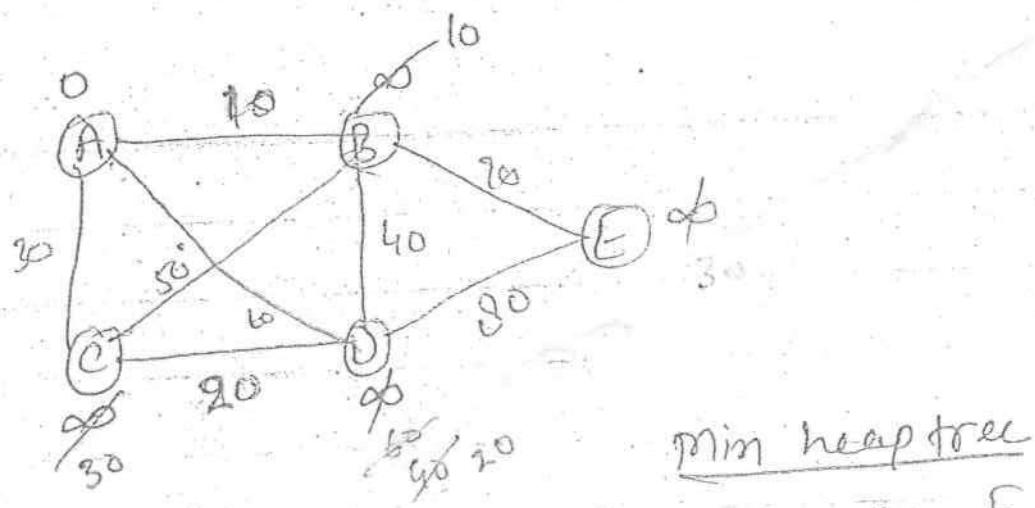
BD X

BC X

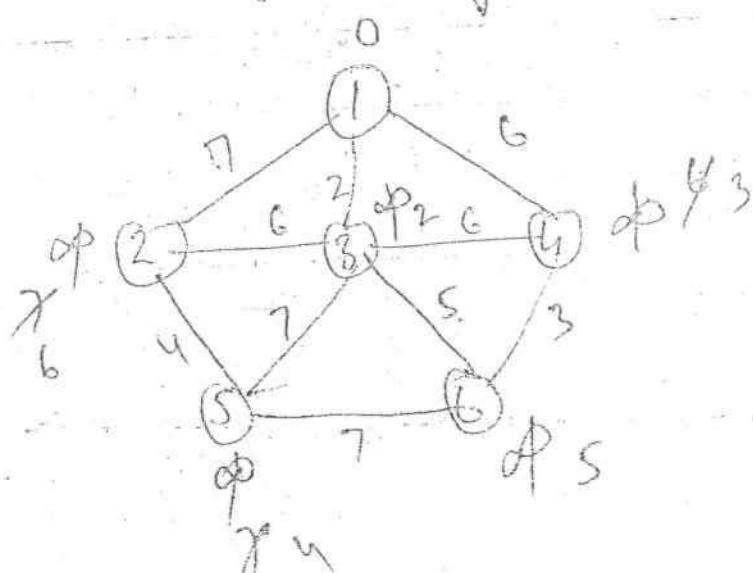
AD X

Note

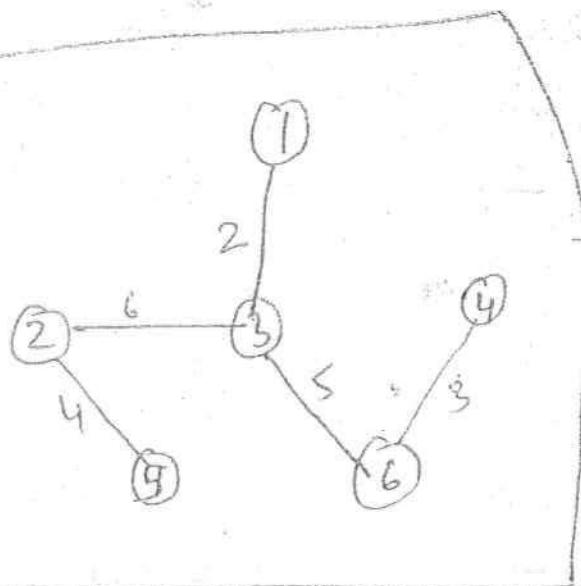
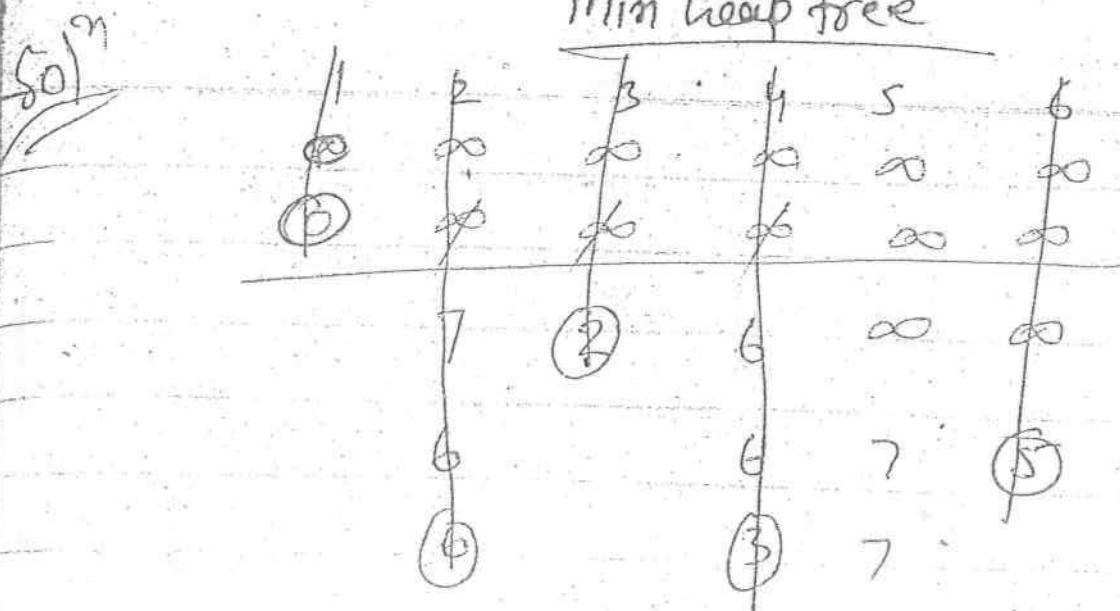
Prim's & Kruskal's algo Both generate some cost
Can generate diff. MST.
But in B/I/O algo Prim's doesn't have any
any forest and Kruskal have forest



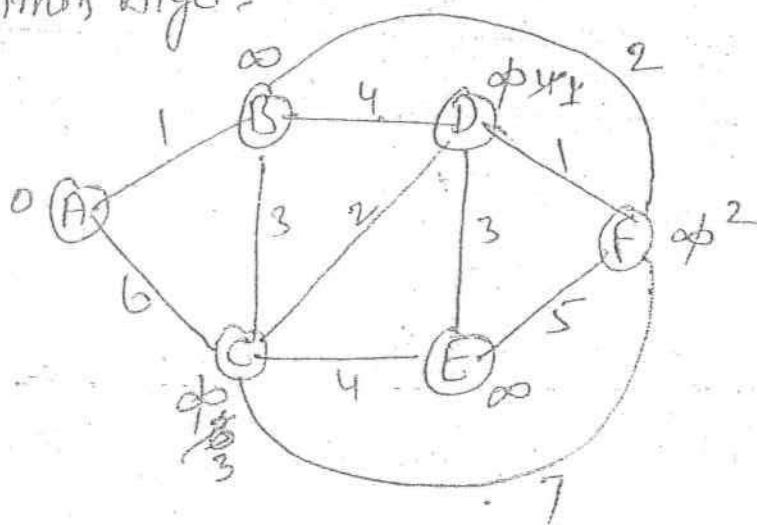
Consider the following graph.



Min heap tree



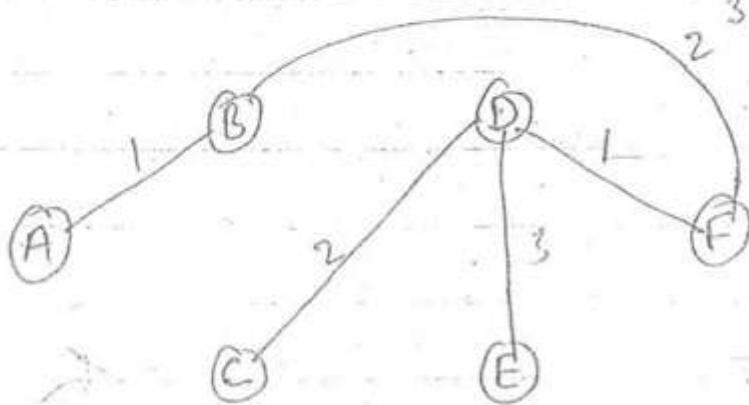
Q Construct min CST for the following Graph using Prim's Algo.



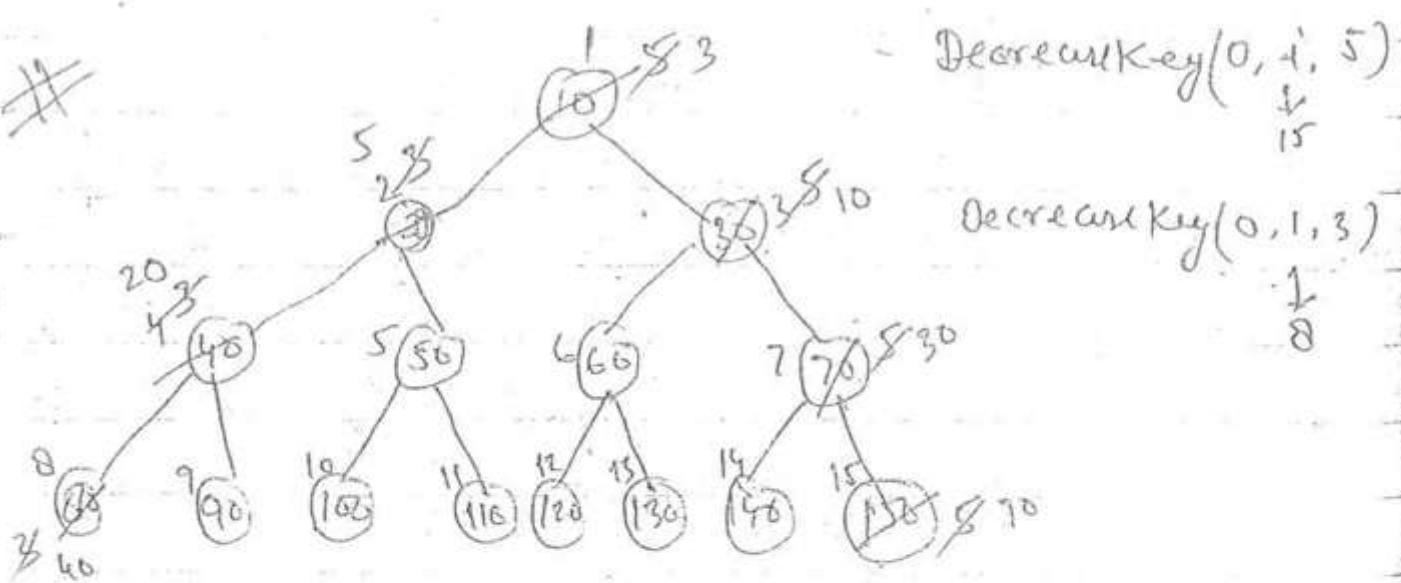
Min heap tree

SIM

A	B	C	D	E	F
∞	∞	∞	∞	∞	∞
0	0	0	0	0	0
1	1	6	0	0	0
2	3	3	4	∞	2
3	3	3	1	5	3
4	2	2	3	3	3
5	1	1	2	2	2
6	0	0	1	1	1
7	0	0	0	0	0



Initialization



It is also operation of heap. To decrease the key in the min heap tree it will take $O(\log n)$ time.

\Rightarrow To increase the key in the max heap tree, it will take $O(\log n)$ time.

Algo

Prims (G, V, E, S)

{ Initialization

① for each vertex $v \in V$

$$v_{\min} = \infty \quad \Rightarrow O(V)$$

② $s_{\min} = 0 \quad \Rightarrow O(1)$

③ Create min heap tree (\emptyset) for all vertices. $\Rightarrow O(V)$

④ while ($\emptyset \neq \emptyset$)

$\Rightarrow O(\log V)$

$u = \text{extract-min}(\emptyset)$

for all v adjacent u

if ($d(u, v) < v_{\min}$)

$$v_{\min} = d(u, v)$$

$\frac{V-1}{2}$

decreases key of
 $O(\log V)$

$$= \cancel{O(V)} + \cancel{O(E)} + \cancel{O(V)} + V(\log V + (V-1)\log V)$$

$$= V \log V + V^2 \log V$$

$$= V \log V + E \log V$$

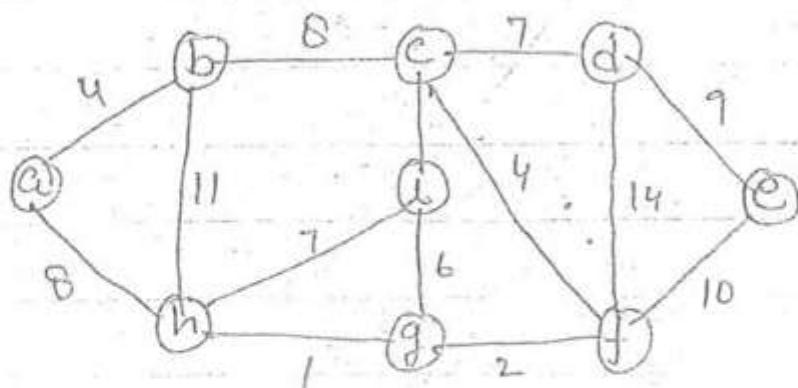
$$= \boxed{O(V+E) \log V}$$

using minheap

$$= \boxed{O(E + V \log V)}$$

using Fibonacci minheap

Consider the following Graph.



Which one of the following sequence of edges of mCST is not true when the algo is started.

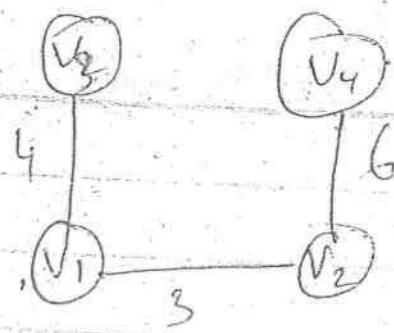
- a) (a,b)(b,c)(c,i)(c,f)(f,g)(g,h)(h,d)(d,e)
- b) (a,b)(a,h)(h,g)(g,f)(f,c)(c,i)(c,d)(d,e)
- c) (a,b)(b,c)(f,c)(c,i)(f,g)(g,h)(h,d)(d,e)
- d) (a,b)(b,c)(h,g)(f,g)(c,f)(c,t)(c,d)(d,e)

disconnected

Graph.
so. Prims not possible

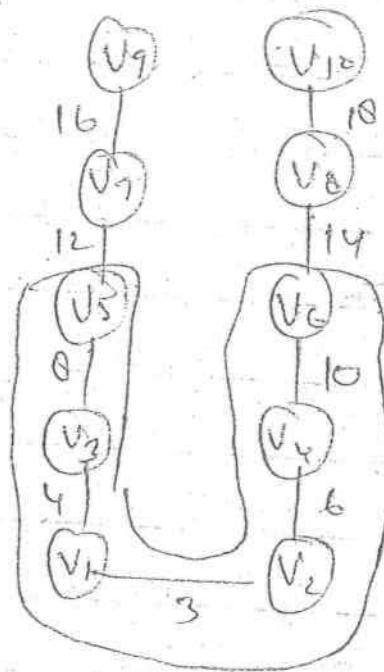
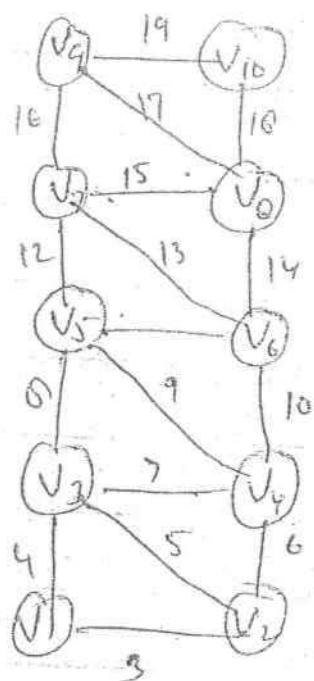
①

Sd²⁴



13

②

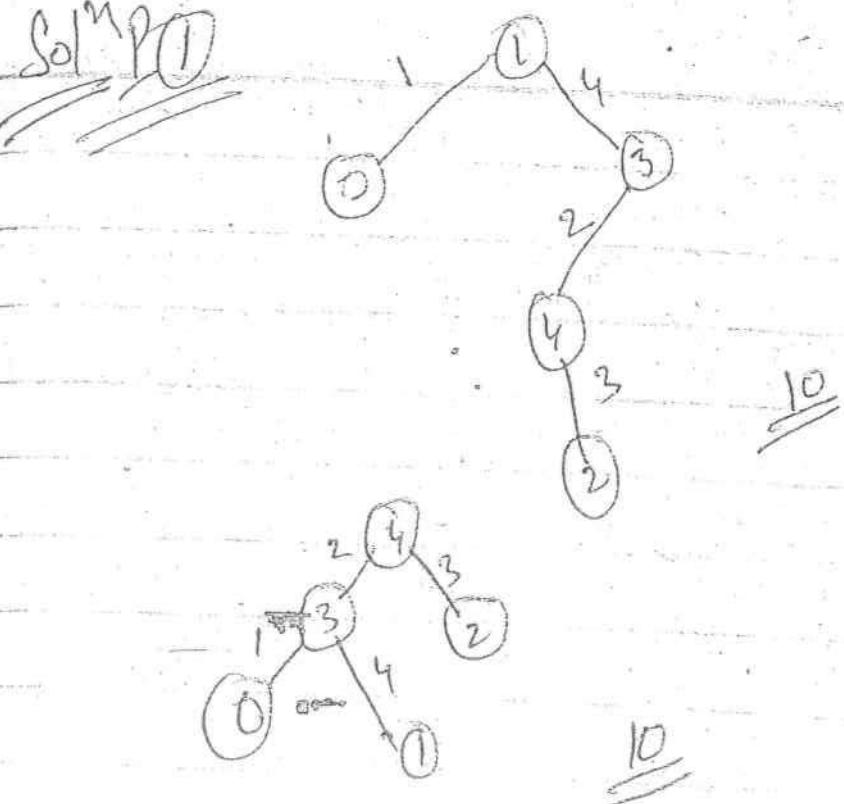


P Consider a ^{complete} undirected graph $G(V, E)$ with vertex set $\{0, 1, 2, 3, 4\}$. Entry w_{ij} in the matrix W below is the weight of the edge (i, j) .

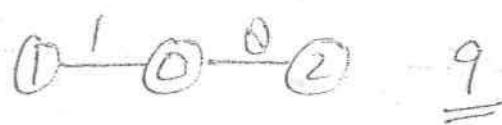
$$W = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 0 & 1 & 4 \\ 1 & 1 & 0 & 12 & 4 & 9 \\ 2 & 0 & 12 & 0 & 7 & 3 \\ 3 & 1 & 4 & 7 & 0 & 2 \\ 4 & 4 & 9 & 3 & 2 & 0 \end{bmatrix}$$

- ① What is the min^m possible weight of MST of G for the above graph, such that vertex 0 is leaf node in T
 a) 7, b) 8 c) 9 d) 10

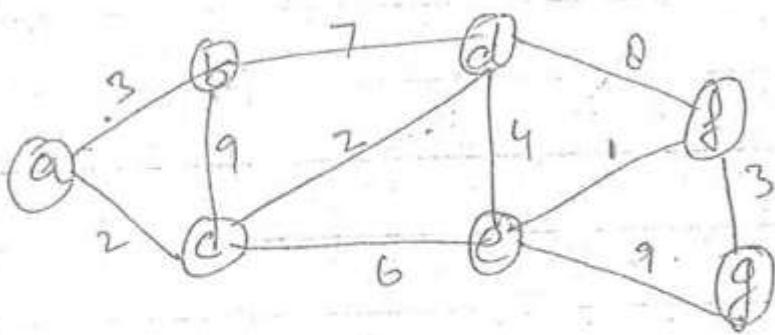
- ② What is the min^m possible weight of a path P from vertex 1 to vertex 2 for the above graph G such that P contains atmost 3 edges
 a) 7 b) 8 c) 9 d) 10



PQ



= Consider the following graph.

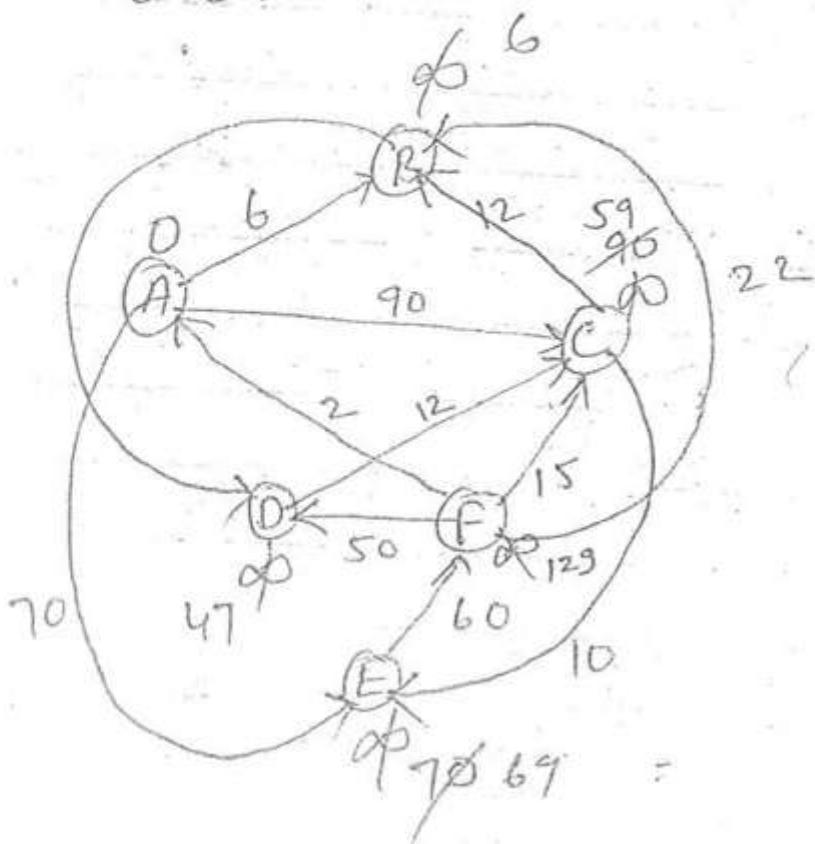


If Prim & Kruskal algo is used to generate MST of above graph, then which one of the following is true.

- a) Prim is 25% decrease comparing with Kruskal.
- b) Kruskal is 10% increase comparing with prim.
- c) Kruskal is 7.3% decrease with Prim algo.
~~10%~~
- d) Prim decreas 10% increas with Kruskal.

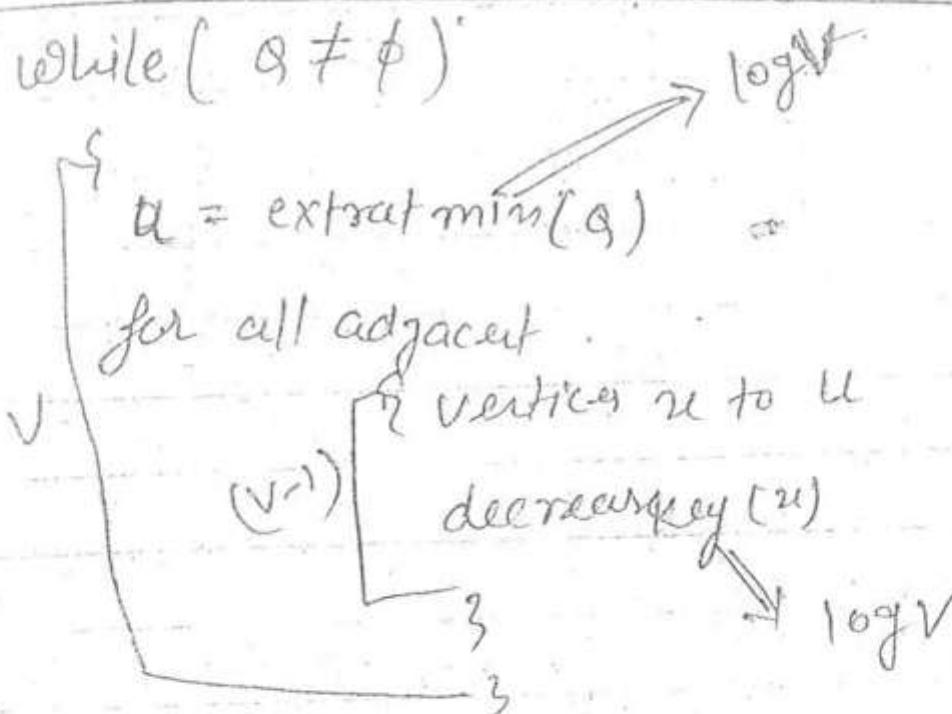
~~Ans~~ Both Prim and Kruskal will give same answer

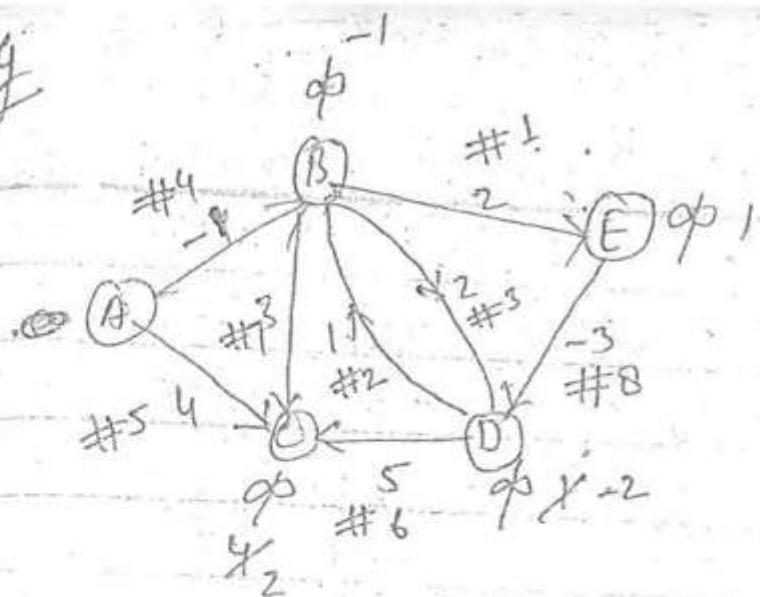
Let G be a directed weighted graph shown below.



- ① Output the sequence of vertices identified by algo when the algo is started from A
- ② Output the sequence of vertices in the shortest path from A to E
- ③ What is the cost of shortest path from A to E

<u>Not in min heap</u>	In min heap					
	A	B	C	D	E	F
	(0)	∞	∞	∞	∞	∞
A		(6)	90	∞	70	∞
AB			90	(47)	70	∞
A, B, B				(59)	70	∞
A, B, D, C					(69)	129
A, B, D, C, E						129
A, B, D, C, E, F						(129)





apply $|V|-1 = 5-1 = 4$ time ($i=4$ time).

Step

	A	B	C	D	E	
0	∞	∞	∞	∞	∞	
0	-1	∞		∞	∞	$i=1$
0	-1	4		∞	∞	
0	-4	2		∞	∞	
0	-1	2	∞	1		$i=2$
0	-1	2	1	1		
0	-1	2	-2	1		
0	-1	2	-1	1		$i=3$
0	-1	2	-1	1		$i=4$

Note -

If given graph contain -ve weight cycle.
 Bellman Ford algo report saying that
 -ve weight cycle ~~not there~~ exist.

Otherwise it will find out shortest path (minimum weight cycle not form source to every vertex)

$O(VE)$

$O(VV^2) = O(V^3)$ = dense

$O(V \cdot V) = O(V^2)$ = sparse

eg

Job Sequencing with Deadline

- ① Single process is available. (at a time for each job)
- ② Interleaving is not allowed.
- ③ Arrived time all of the job are same.
- ④ Every job requires 1-unit of time.

Ques
 $n=4$

Jobs
Profit Deadlines

Jobs	J_1	J_2	J_3	J_4
Profit	P_1	P_2	P_3	P_4
Deadline	d_1	d_2	d_3	d_4

Ques

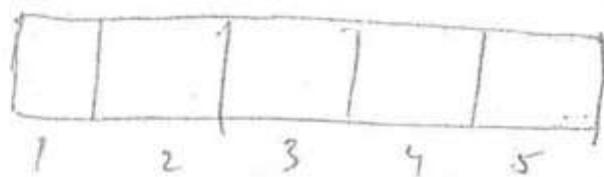
n=6

Jobs	J ₁	J ₂	J ₃	J ₄	J ₅	J ₆
Profit	24	10	22	30	12	10
Deadline	5	3	3	2	4	2

Solⁿ

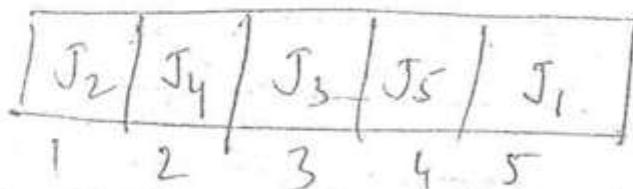
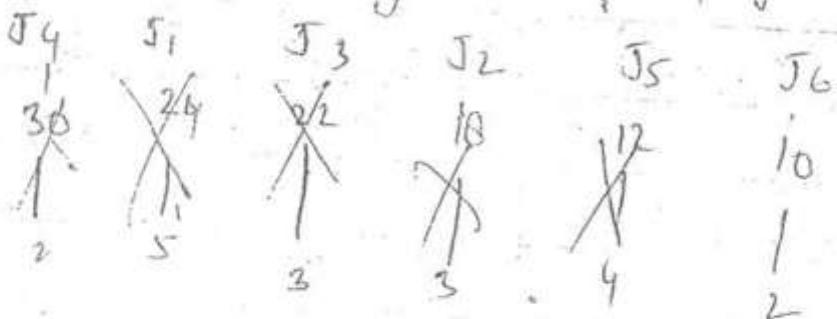
① { }

② { J₁, J₂ } { J₃ } { J₄ } , { J₅ } , { J₆ } .



Solⁿ

↓ Decreasing order of profit



$$= 10 + 30 +$$

$$= 106$$

Job $J_1 J_2 J_3 J_4 J_5$

Profit 10 20 15 5 80

Deadline 3 3 3 4 4.

J₁	J₂	J₃	J₄	J₅
1	2	3	4	

Decreasing order of deadline Profit

J_5	J_2	J_3	J_1	J_4
80	20	15	10	5
1	1			
4	3	3	3	1

J_1	J_3	J_2	J_5
1	2	3	4

$$= 10 + 15 + 20 + 80 = 125$$

Q. $n=9$

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
Profit	15	20	30	10	10	10	23	16	25
Deadline	7	2	5	3	4	5	2	7	3

We are given 9 jobs. Execution of each job requires 1 unit of time.

(1) Which one of the following is true.

a) all jobs are completed.

b) J_1 and J_6 are left.

c) J_1 , & J_8 " left only.

d) $\cancel{J_4}$ & J_6 "

② \max^m Profit

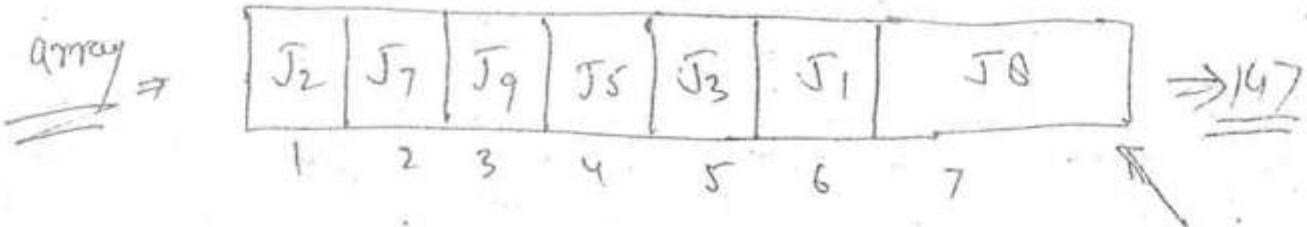
a) 147 b) 165

c) 167 d) 168.

Solⁿ

Decreasing order of profit

J_3	J_9	J_7	J_2	J_4	J_5	J_8	J_1	J_6
X 30	X 25	X 23	X 20	10	X 10	X 6	X 5	10
1	1	1	1	3	4	7	7	5
5	3	2	2	3	4	7	7	5



Algo

① Sort all the jobs in decreasing order of profits. $\Rightarrow O(n \log n)$

② Find max^m deadlines in the given array of n deadline and take the same size array and start from RHS of the array.

For every slot I apply linear search to find a job which contain $\boxed{\text{deadline} \geq d}$

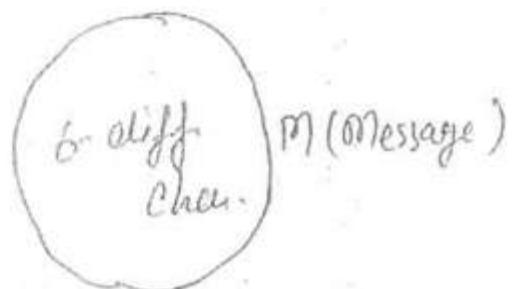
$$O(n^2)$$

$$\underline{O(n^2)}$$

Huffman Coding

- (1) Binary tree
- (2) Encoding Technique
- (3) Data compression mechanism.

eg



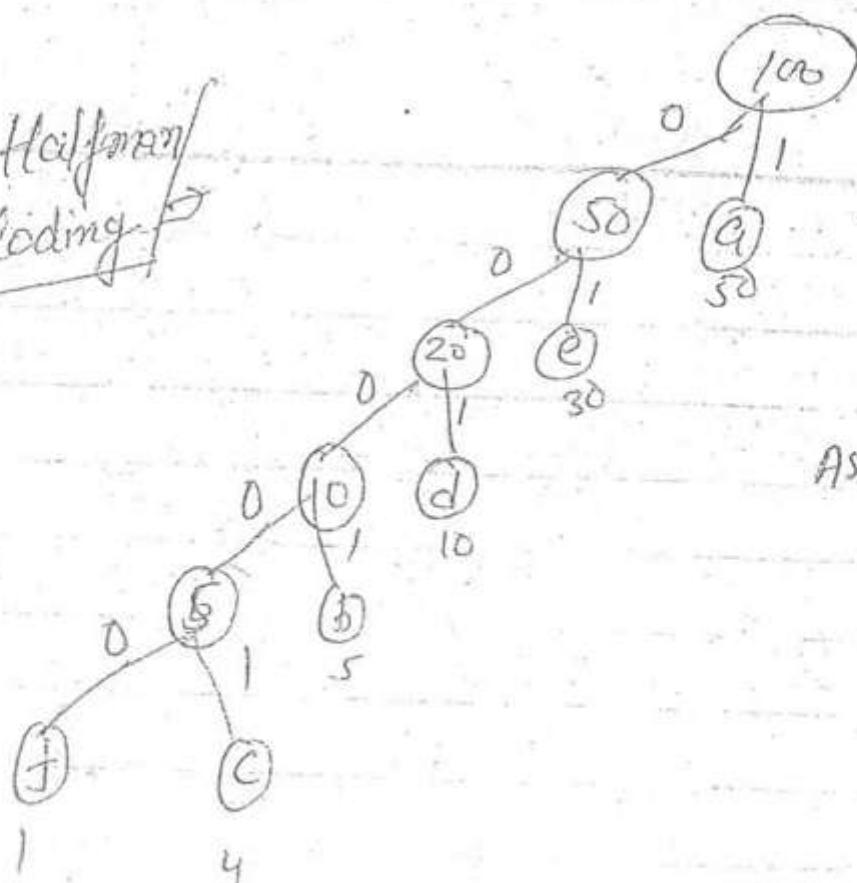
100-character A (ASCII) 3-bit (not ASCII)

a - 50	a - 50x7	a - 000 = 50x3
b - 5	b - 5x7	b - 001 = 5x3
c - 4	c - 4x7	c - 010 = 8x3
d - 10	d - 10x7	d - 011 = 10x3
e - 30	e - 30x7	e - 100 = 30x3
f - 1	f - 1x7	f - 101 = 1x3
	700 bytes	300 bits

M
D/I destination
compress
700 bytes

Compress

? Huffman
Coding



Assume left = 0

Right = 1

$$a = 1$$

$$b = 0001$$

$$c = 00001$$

$$d = 001$$

$$e = 01$$

$$f = 000000$$

after 2ⁿ compression

$$a - 1 * 50 = 50$$

$$b = 5 * 4 = 20$$

$$c = 4 * 5 = 20$$

$$d = 10 * 3 = 30$$

$$e = 30 * 2 = 60$$

$$f = 1 * 5 = 5$$

$$\frac{185}{185}$$

This is Called Huffman Coding

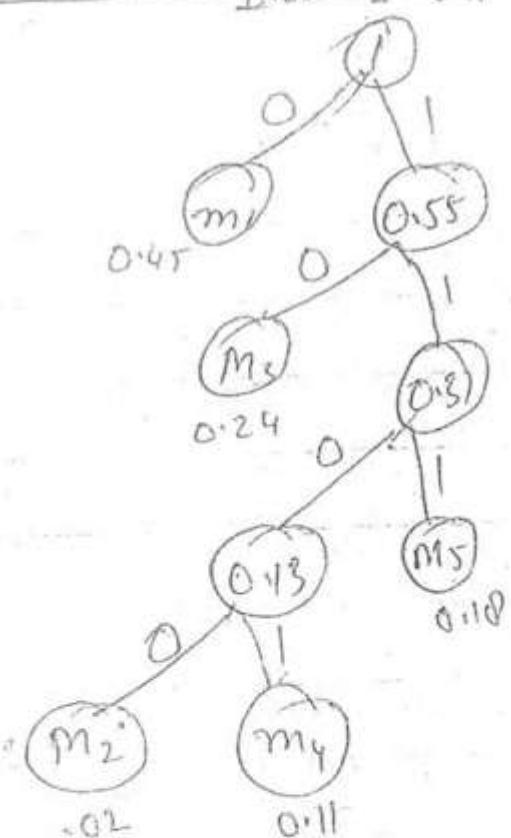
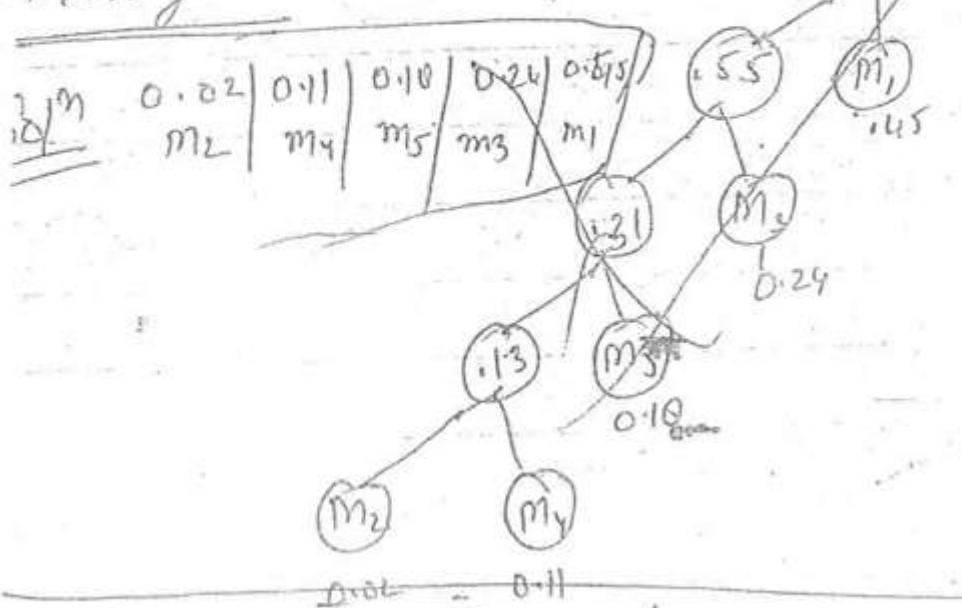
105 bit to perform for
100 char

100 char = 105 bits.

$$1 \quad = \frac{105}{100} = 1.05 \text{ bits/char}$$

Message $M = (m_1, m_2, m_3, m_4, m_5)$

Using Huffman coding find Encoded form of message.



$$\begin{aligned}
 m_1 &= 0 \\
 m_2 &= 1100 \\
 m_3 &= 10 \\
 m_4 &= 1101 \\
 m_5 &= 111
 \end{aligned}$$

Total no of bit to transfer

$$\begin{aligned}
 &= 0.45 \times 1 + 0.02 \times 4 + 0.24 \times 2 \\
 &\quad + 0.11 \times 4 + 0.18 \times 3 \\
 &= 1.99 \text{ bits}
 \end{aligned}$$

Avg no of bit to transfer

$$\frac{1.99}{5} = 1.99 \text{ bit/msg}$$

Q Suppose the letters \rightarrow a, b, c, d and e occur
Probability $\rightarrow \frac{1}{2}, \frac{1}{2^2}, \frac{1}{2^3}, \frac{1}{2^4}$ and $\frac{1}{2^5}$ respectively

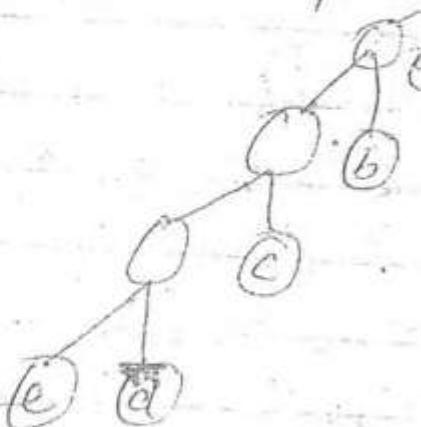
① Which one of the following is Huffman coding for the letters a, b, c, d & e respectively

a) 11, 10, 011, 010, 001

b) 0, 10, 110, 1110, 1111

c) 1, 10, 01, 001, 0001

d) 110, 100, 010, 000, 001



② What is the avg no of bits required per message?

a) 3

b) 3.75

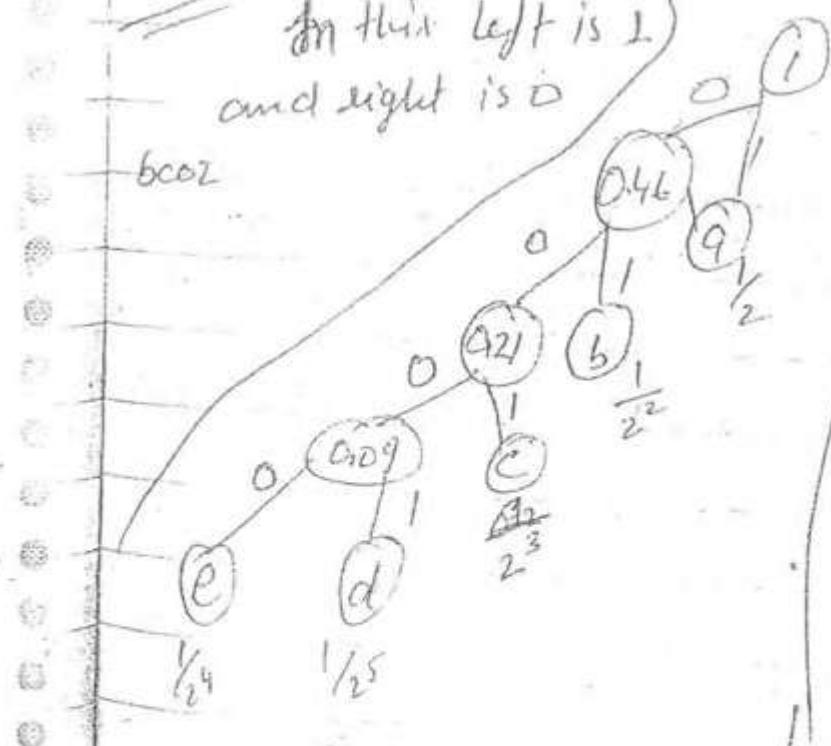
c) 2.10

d) 1.78.

Solⁿ

(in this left is 1)
and right is 0

bcz



$$\begin{cases} a = 0 \\ b = 10 \\ c = 110 \\ d = 1110 \\ e = 1111 \end{cases}$$

Avg no of bits required

$$\sum_{i=1}^n p_i * b_i$$

p_i = Probability of msg m_i

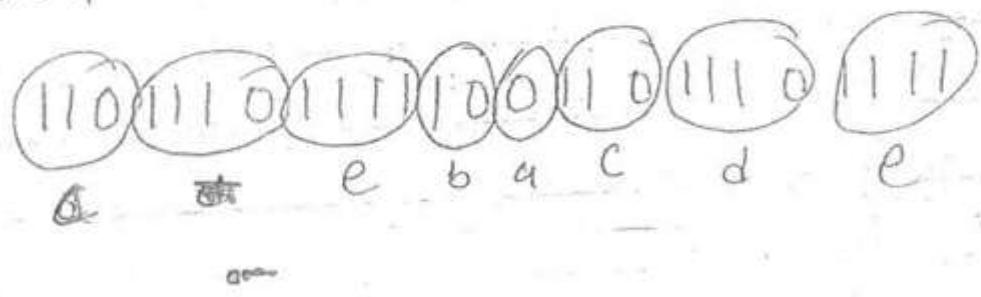
b_i = # of bits to do

represent msg m_i

$$= \frac{1}{2}x_1 + \frac{1}{2^2}x_2 + \frac{1}{2^3}x_3 + \frac{1}{2^4}x_4 + \frac{1}{2^5}x_5$$

$$= 1.78 \text{ bits/msg}$$

3 If the encoded msg is 1101110111100110110111
what will be the equivalent Decoded msg.

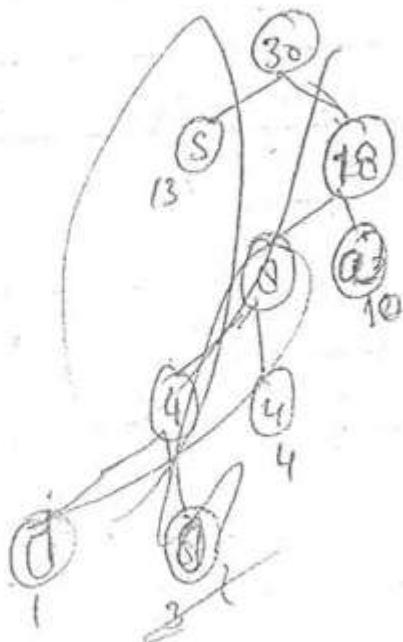


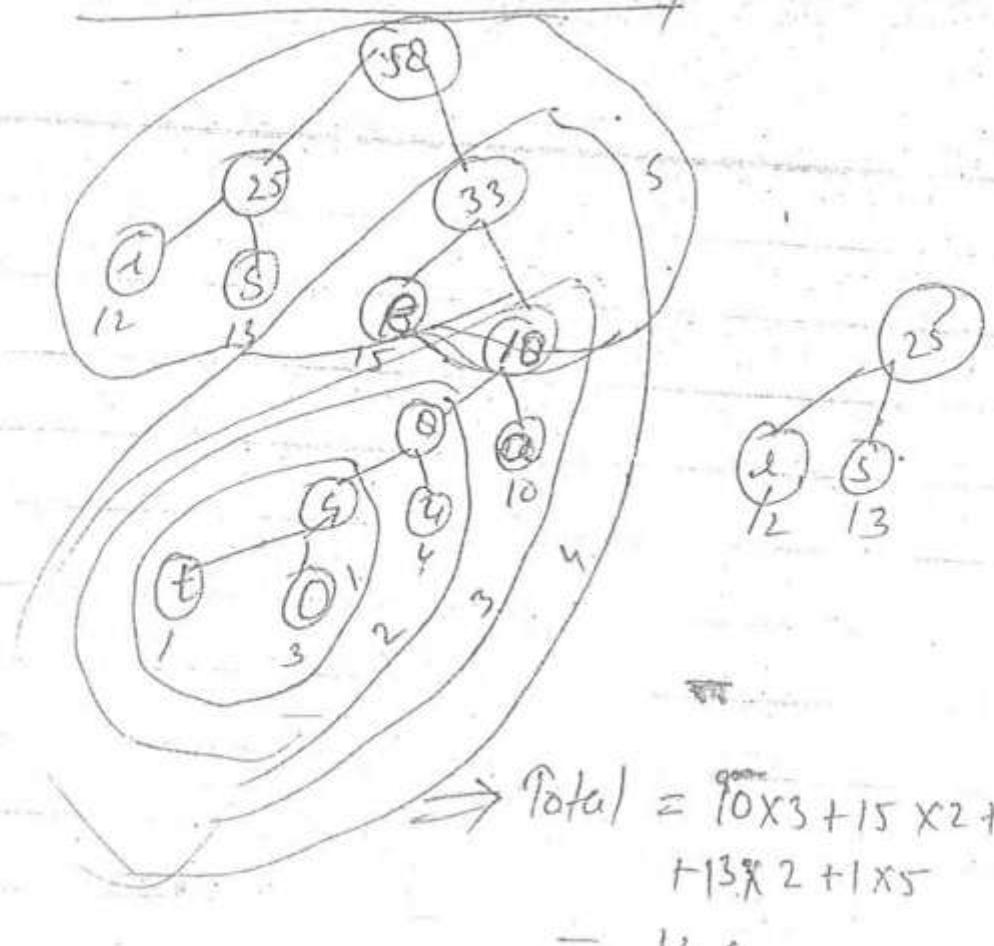
P A file contain char a, e, i, o, u, s and d.
and frequency 10 15 18 3 4 13 1

If we use Huffman coding for data coding
what will be the avg code length?

- a) $\frac{146}{58}$ b) $\frac{144}{58}$ c) $\frac{140}{58}$ d) none.

SOL





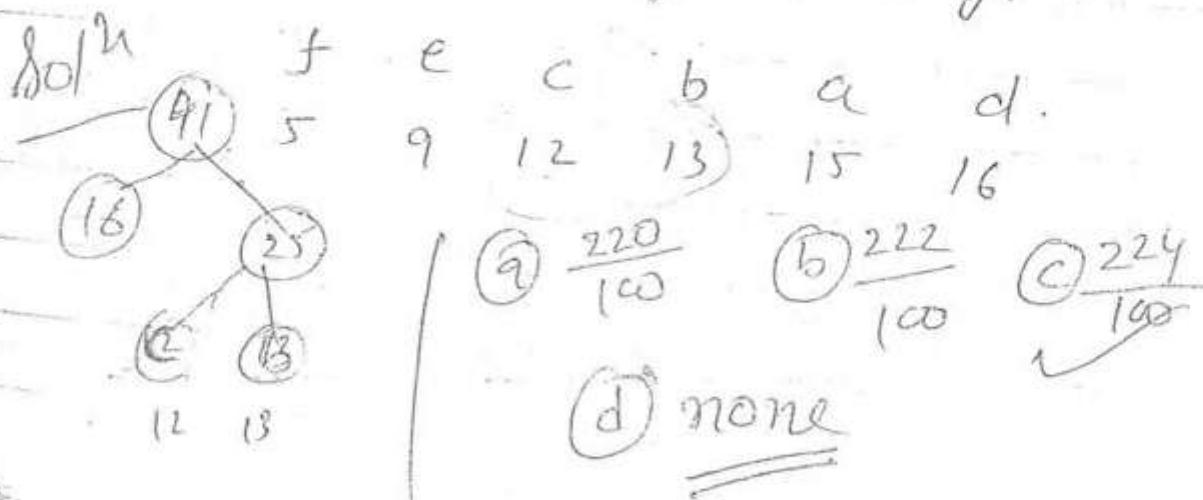
$$\rightarrow \text{Total} = 10 \times 3 + 15 \times 2 + 12 \times 2 + 3 \times 5 \\ + 13 \times 2 + 1 \times 5 \\ = 146$$

$$\text{Avg} = 146 / 58$$

P Consider the frequencies for the chars. a, b, c, d, e and f

a	b	c	d	e	f
45	13	12	16	9	5

If we use Huffman code for Data compression
What will be the avg code length.

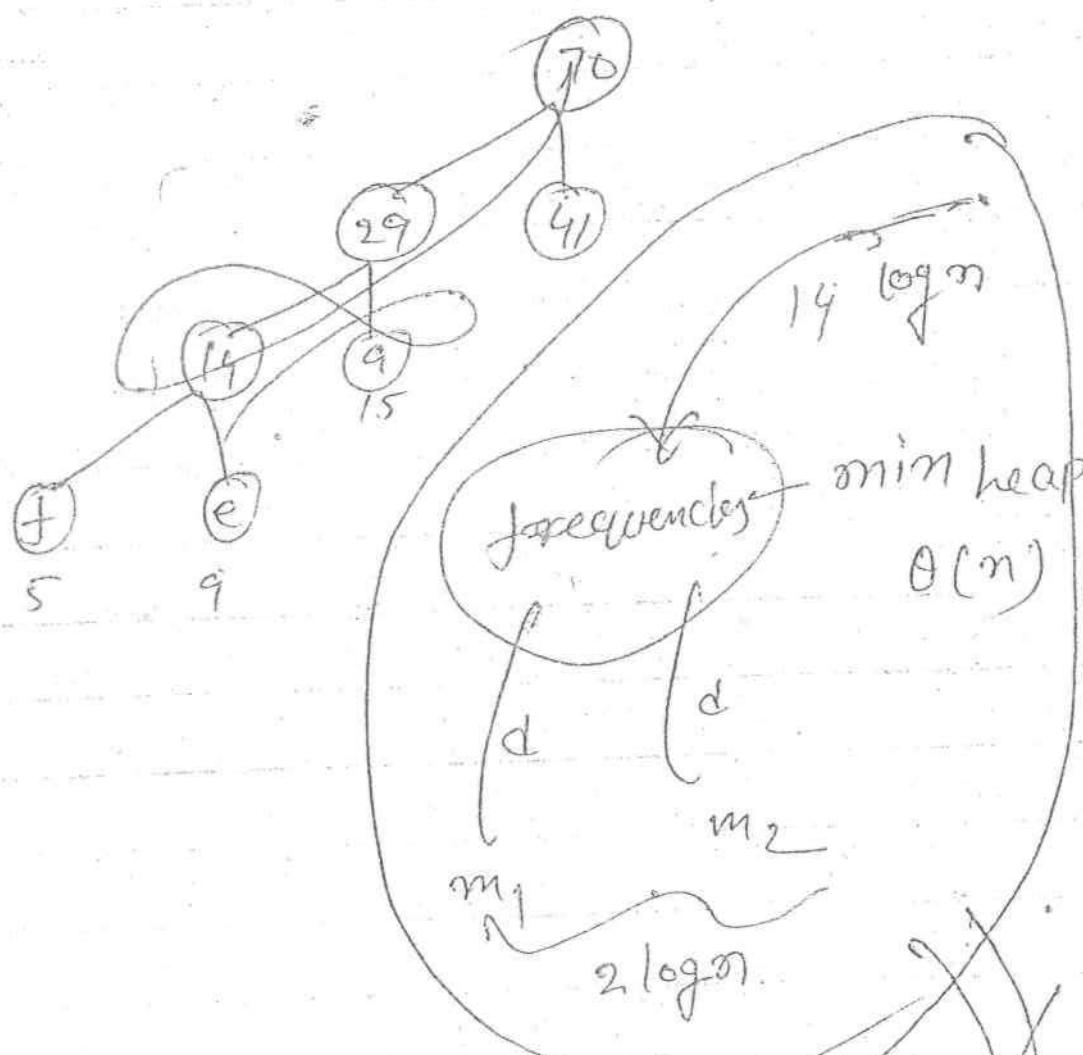


~~Service no 701~~

~~11 Name OLA-LKC~~

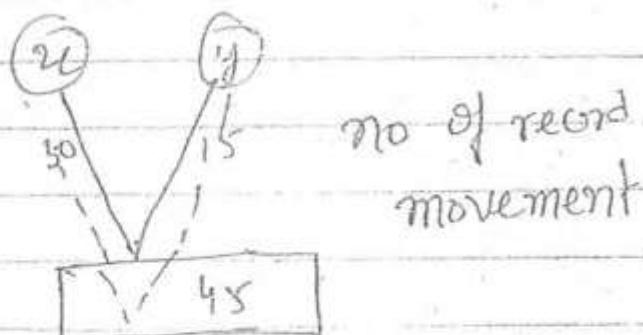
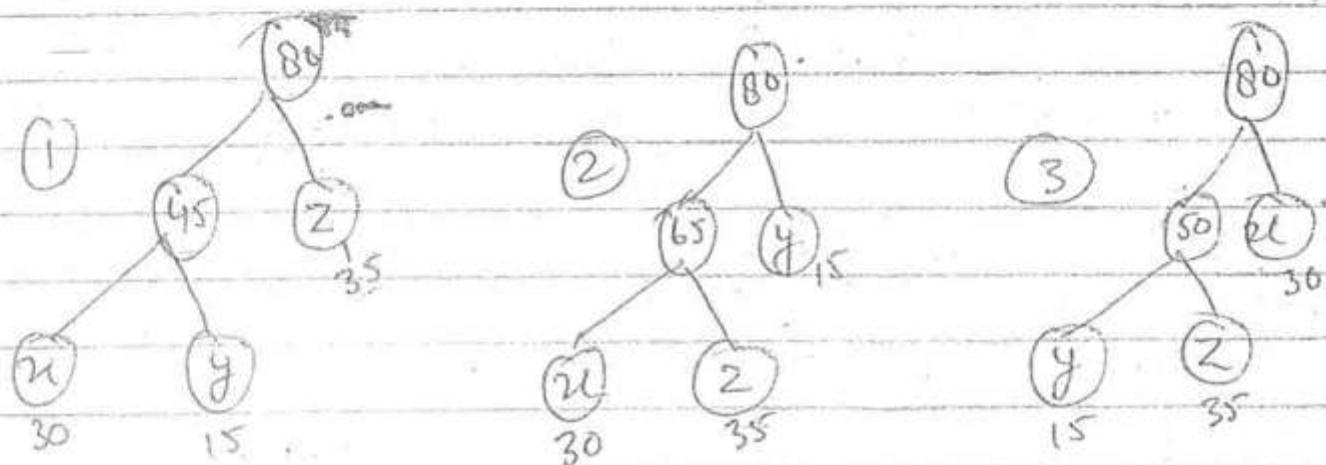
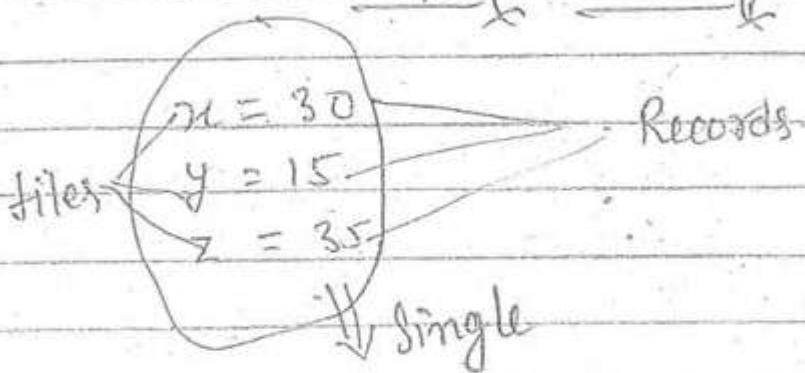
~~Type - Sleeper Coach~~

~~7:30 - 8:15~~



Tuesday
19/07/011

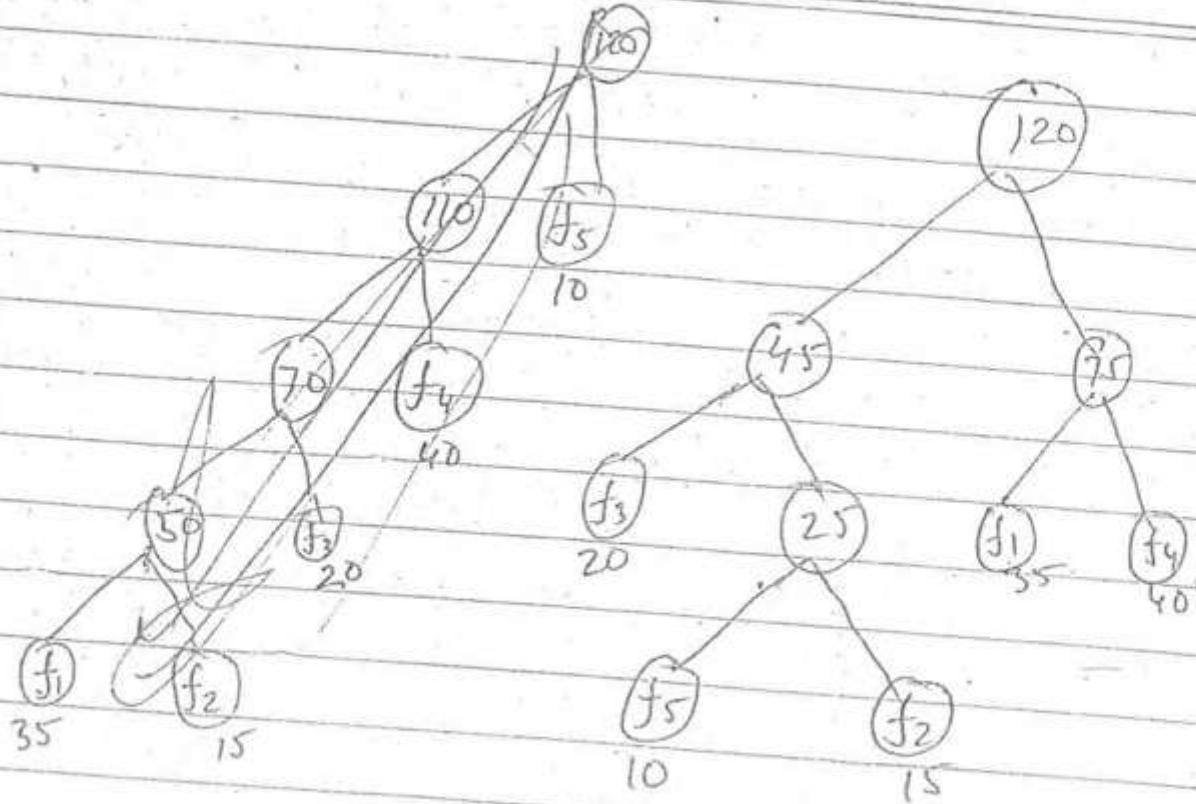
Optimal Merge Pattern



The min^m no of record movement to merge 5-files

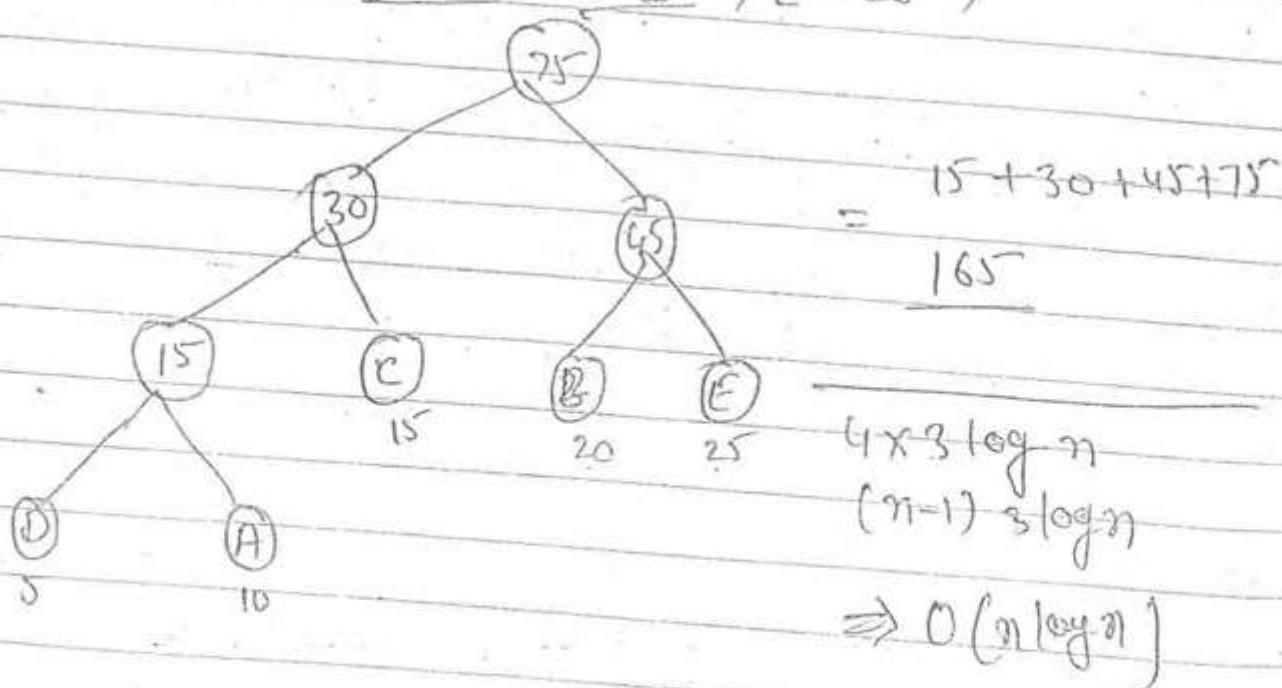
$$f_1 = 35, f_2 = 15, f_3 = 20, f_4 = 40, f_5 = 10$$

So/



Min no of Recvl msg required to merge files.

$$A=10, B=20, C=15, D=45, E=25$$



Conclusion

(1) knapsack $\Rightarrow O(n \log n)$

(2) MST Kruskal $\Rightarrow E \log V$

Prims $\Rightarrow (V+E) \log V$ (B. Min heap)

or

$E + V \log V$ (F. min heap)

(3) Job sequencing
with Deadline $\Rightarrow O(n^2)$

(4) Huffman Coding $\Rightarrow O(n \log n)$

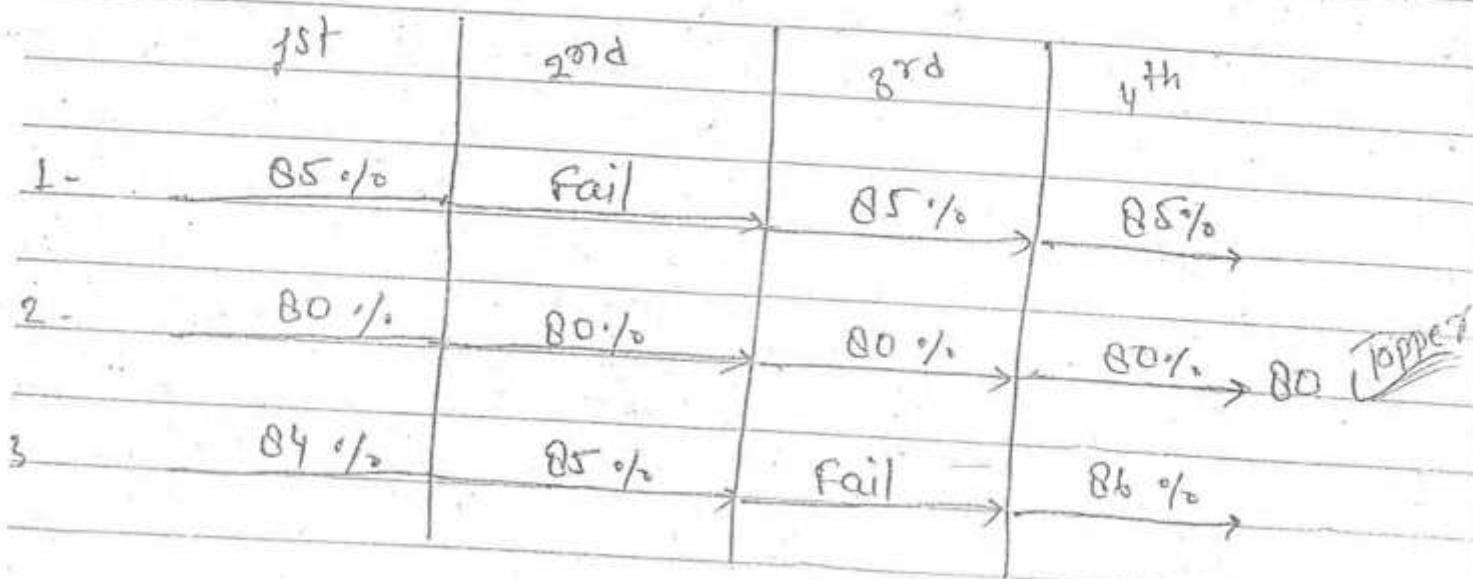
(5) Optimal merge Pattern $\Rightarrow O(n \log n)$

(6) Single source shortest Path

(i) Dijkstra's $\Rightarrow (V+E) \log V$ or
 $E + V \log V$

(ii) Bellman-ford $\Rightarrow O(VE)$

Dynamic Programming (Time is over)



Dynamic programming ^{will} give always = Optimal Solution
Greedy may not give optimal soln.

Application of D.P

① Longest common subsequence.

② 0/1 knapsack.

③ Multistage Graph.

④ Travelling Sales Person.

⑤ Matrix chain multiplication

⑥ Sum of subsets problem

⑦ All pairs of shortest path.

⑧ Optimal cost Binary Search Tree.

(1) Longest Common Subsequence

$\text{--- } x \text{ --- } x \text{ --- } b \text{ --- }$

Subsequence \Rightarrow A subsequence of a given sequence is just the given sequence in which 0 (or) more elements left out.

eg

$$u = (A, B, C, B, D, A, B)$$

1 2 3 4 5 6 7

$$S_1 = (A, A, B) \rightarrow \text{Subsequence}$$

1 6 7

$$S_2 = (A, B, B) \rightarrow \text{Subsequence}$$

1 4 7

$$S_3 = (C, D, B) \rightarrow \text{Subsequence}$$

3 5 7

$$S_4 = (A, A, B, C) \times$$

1 6 7 3

$$S_5 = (A, B, C, B, D) \rightarrow \text{Subsequence}$$

1 2 3 4 5

Common Subsequence Given two sequence x and y
we say that a sequence z is a common
subsequence of x and y if z is a subsequence
of both x and y .

Ex $x = (A, B, B, A, B, D)$

$$y = (A, A, B, B, D, D)$$

$CS_1 = (A, A, D)$ — common subsequence.

$CS_2 = (A, A, B, D)$ — " "

$CS_3 = (A, B, B, D)$ — " "

$CS_4 = () \times$

In the above problem CS_2 and CS_3 are the
longest common subsequence for the given
two sequences x and y .

Ex $x = (A, B, C, B, D, A, B)$

$$y = (B, D, C, A, B, A)$$

$CS_1 = (B, B, A, B)$

longest common subsequence

 $CS_2 = \{A\}$ $CS_3 = \{A, B\}$ $CS_4 = \{B, C, A\}$ $CS_5 = \{\emptyset\}$ $P - w = (A, /B, C, /B, D, A, /B,)$ $y = (P$

$u = (a_1, a_2, a_3, a_4, \dots, a_n)$

$y = (b_1, b_2, b_3, \dots, b_n)$

LCS

 $u = (A, B, C)$

$\overset{3}{\rightarrow} \{ \}$	$\{ A, B \}$	$\{ A, B, C \}$
$\{ A \}$	$\{ A, C \}$	
$\{ B \}$	$\{ B, C \}$	
$\{ C \}$		

① Find all possible Subsets to 1st subsequence
 $x(1 \dots m)$

$$\rightarrow 2^m$$

$$2^n$$

(2) For every subset of x check that subset is there in y (or) not.



$$O(2^m \times n)$$



using Brute force : $O(2^m \times n) \Rightarrow$ Exponential time

\Downarrow
(In technical term means
algo is very slow)

$$x = (A, A, A, A, A)$$

$$y = (\underline{A}, A, A, A, A)$$

longest common subsequence (5,5) = 1 + LCS(4,4)

$$LCS(4,4) = 1 + LCS(3,3)$$

$$LCS(3,3) = 1 + LCS(2,2)$$

$$LCS(2,2) = 1 + LCS(1,1)$$

$$LCS(1,1) = 1 + \underline{LCS(0,0)}_0$$

$$LCS(n,n) = 1 + LCS(n-1, n-1) \text{ if } x[n] = y[n]$$

eg $x = (A, A A B A A B)$
 $y = (\underline{A}, A A A A A A)$

defn $LCS(7,7) = 1 + LCS(6,6)$

↓

$1 + LCS(5,5)$

↓

$1 + LCS(4,4)$

↓

$LCS(3,4)$

↓

$1 + LCS(2,3)$

↓

$1 + LCS(1,2)$

↓

$1 + LCS(0,1)$

eg $x = (B, B, B, B, A, A, A)$

$x = (B, B, B, A, A, A, A)$

$LCS(7,7) = 1 + LCS(6,6)$

↓

$1 + LCS(5,5)$

6 ←

$1 + LCS(4,4)$

$LCS(4,3)$

↓
 $1 + LCS(3,2)$

↓
 $1 + LCS(2,1)$

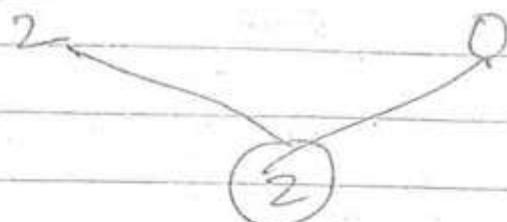
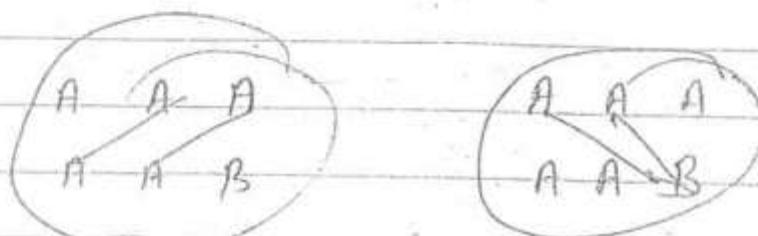
↓
 $1 + LCS(1,0)$

eg $x = \text{A B A B}$
 $y = \text{B A B A}$

if match is not found then
 don't decrement both.

$$\text{LCS}(i, j) = \begin{cases} 0 & \text{if } i=0 \text{ (or) } j=0 \\ 1 + \text{LCS}(i-1, j-1) & \text{if } x[i] == y[j] \\ \max(\text{LCS}(i-1, j), \text{LCS}(i, j-1)) & \text{if } x[i] \neq y[j] \end{cases}$$

eg



Optimal Solution

(i) Optimal Solution of original problem contains optimal soln of subproblem.

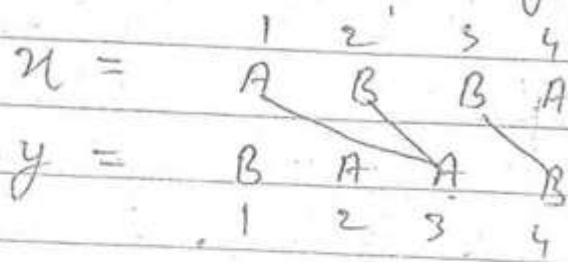
Write a recursive "c" Program to $\text{LCS}(i, j)$,

```
LCS(i, j)
{
    if (i == 0 || j == 0)
        return 0; // exponential O(2^n)

    else
    {
        if (x[i] == y[j])
            return (1 + LCS(i-1, j-1));
        else
        {
            a = LCS(i-1, j); // work
            b = LCS(i, j-1);
            c = max(a, b);
            return c;
        }
    }
}
```

eg

Find LCS for the following Problem.



Solⁿ

$$LCS(4,4) = \max \left\{ \begin{array}{l} LCS(3,3) \\ LCS(4,3) \end{array} \right\}$$

$$LCS(3,4) = 1 + LCS(2,3)$$

$$LCS(2,3) = \max \left\{ \begin{array}{l} LCS(1,3) \\ LCS(2,2) \end{array} \right\}$$

$$LCS(1,3) = 1 + LCS(0,2)$$

$$LCS(0,2) = 0$$

$$LCS(2,2) = \max \left\{ \begin{array}{l} LCS(1,2) \\ LCS(2,1) \end{array} \right\}$$

$$LCS(1,2) = 1 + LCS(0,1)$$

$$LCS(2,1) = 1 + LCS(1,0)$$

(0,1)

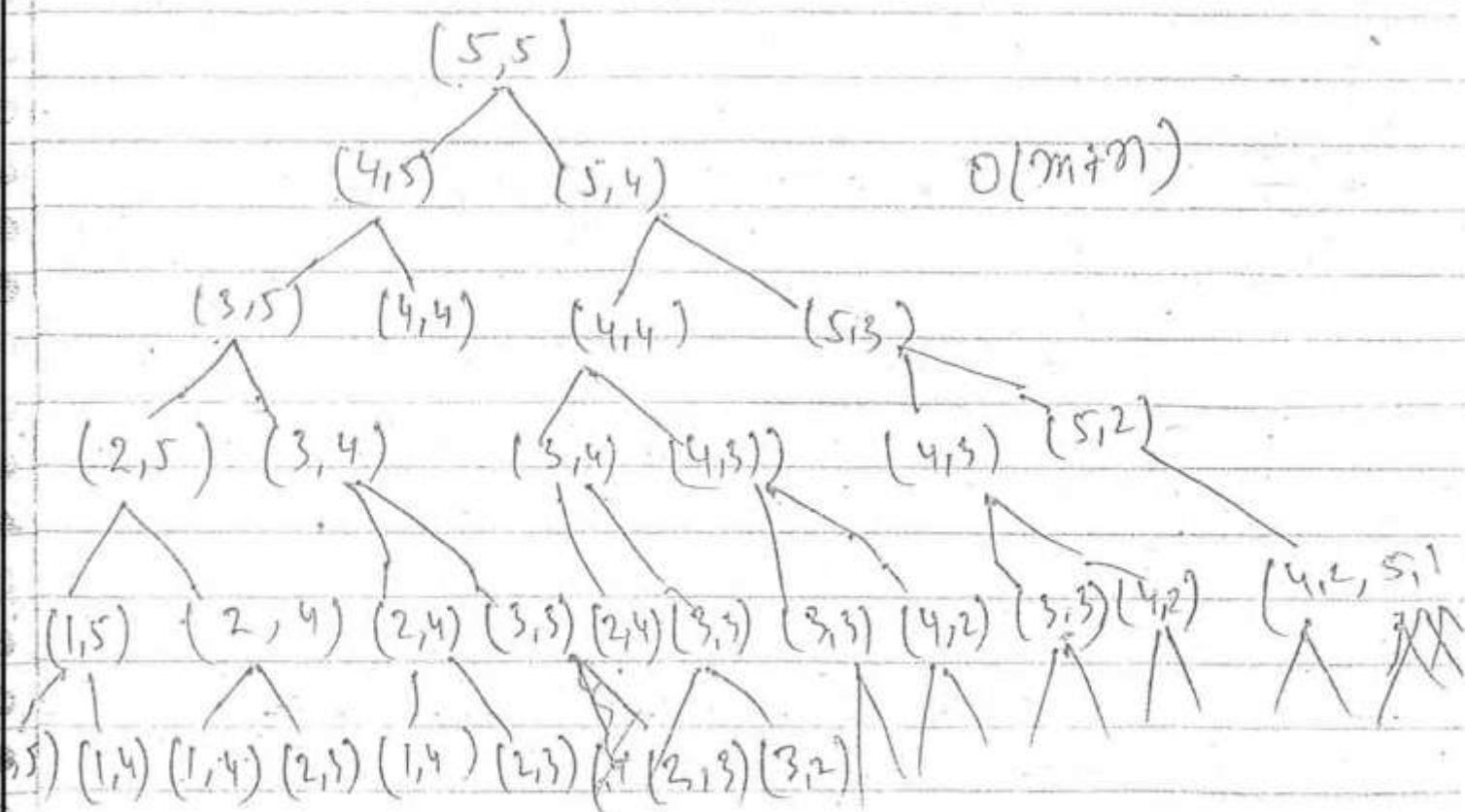
$$LCS(4, 3) = 1 + LCS(3, 2)$$

$$LCS(3, 2) = \max \begin{cases} LCS(2, 2) \\ LCS(3, 1) \end{cases}$$

$$LCS(3, 1) = 1 + \cancel{LCS(2, 0)}$$

e.g. $x = (A, A, A, A, A)$

$$y = (B, B, B, B, B)$$



$$3 \Rightarrow 2^3 - 1$$

$$2^{m+n} = 2^m \cdot 2^n$$

2 Overlapping Subproblem \Rightarrow In most of the recursive program small no. of distinct funⁿ call are present and, more no. of repeated subproblem, so Dynamic programming solves only distinct Subproblem.

Using recursion (Without DP)

$\Rightarrow m+n$ levelled ans.

$\Rightarrow 2^{m+n}$



Using recursion (With DP)

$$m=4, n=5$$

$$\begin{array}{cccc} 15 & 8,5 & 2,5 & 1,5 \\ 14 & 3,4 & 2,4 & 1,4 \end{array} \Rightarrow m \times n$$

$$\begin{array}{cccc} 13 & 3,3 & 2,3 & 1,3 \\ 12 & 3,2 & 2,2 & 1,2 \end{array} \Rightarrow \underline{O(m,n)} O(1)$$

$$\begin{array}{ccc} 11 & 3,1 & 2,1 & 1,1 \end{array}$$



$$O(m,n) \Rightarrow O(n^2)$$

Quadratic

Drawback

Dynamic Programming take more space to store table for every sub problem.

I saw answer
this previously.

Memoization - $\text{LCS}(i, j)$

```

{
    if (i == 0 || (j == 0))
        return 0; // Table[i, j] = 0
    else {
        if (u[i] == v[j])
            if (Table[i-1, j-1] != nil)
                return 1 + Table[i-1, j-1];
            else
                Table[i-1, j-1] = LCS(i-1, j-1);
        else
            return max(1 + Table[i-1, j], 1 + Table[i, j-1]);
    }
}
  
```

$\begin{array}{|c|c|c|c|c|c|} \hline & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & & & & & \\ \hline 2 & & & & & \\ \hline 3 & & & & & \\ \hline 4 & & & & & \\ \hline 5 & & & & & \\ \hline \end{array}$

Content over Table

initially all values of table are nil

else

Contents are in
Table.

{
if (Table[i-1, j] != nil & Table[i, j-1] != nil)

return (max { Table[i-1, j],
Table[i, j-1] })

else

{

if (Table[i, j-1] == nil)

Table[i, j-1] = LSS(i, j-1);

if (Table[i-1, j] == nil)

Table[i-1, j] = LSS(i-1, j)

return (max { Table[i, j-1],
Table[i-1, j] })

}

}

}

eg

$x = A, B, A, B$

$y = B, A, B, A$

	0	1	2	3	4
0	nil	0	nil	nil	nil
1	0	nil	1	nil	nil
2	nil	1	nil	2	nil
3	nil	nil	2	nil	3
4	nil	nil	nil	3	nil

initially all are nil

→

$$LCS(4,4) = \max \left\{ \begin{array}{l} LCS(3,4) \\ LCS(4,3) \end{array} \right\}$$

$$LCS(3,4) = 1 + \underline{LCS(2,3)}$$

$$LCS(2,3) = 1 + \underline{LCS(1,2)}$$

$$LCS(1,2) = 1 + \underline{LCS(0,1)}$$

$$LCS(4,3) = 1 + \underline{LCS(3,2)}$$

$$LCS(3,2) = 1 + \underline{LCS(2,1)}$$

$$LCS(2,1) = 1 + \underline{LCS(1,0)}$$

~~While computing LCS Table we can not say in which order we are computing:~~

eg:-

(2) 0/1 Knapsack Problem

$$n=3$$

Objects. Obj₁ Obj₂ Obj₃

Profit 5 3 4

Weight 3 2 1

$$m=5$$

$$\frac{5}{3} = 1.6$$

$$\frac{3}{2} = 1.5$$

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

Solⁿ (L O I) manually

9

↓

~~eg 2~~

$$m = 3$$

objects obj₁ obj₂ obj₃

Profit 50 29 33

weight 10 6 7

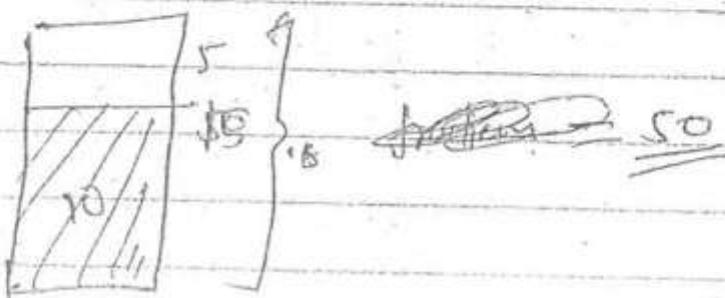
$$m = 15$$

$$\frac{50}{10} = 5$$

10

$$\frac{29}{6} = 4.8$$

$$\frac{33}{7} = 4.7$$



~~manually~~

$$m = 15$$

In DP there is no shortcut becz it is not given
as a optimised soln

P

$$n = 5$$

Object	obj ₁	obj ₂	obj ₃	obj ₄	obj ₅
Weight	3	1	2	4	6
Profit	7	2	1	6	12

$$m = 10$$

86/21

$$\text{OKS}(M, N) = \begin{cases} 0 & \text{if } m=0 \text{ (or) } N=0 \\ \text{OKS}(M, N-1) & \text{if } M < W[N] \\ \max \left\{ \begin{array}{l} \text{OKS}(M, N-1) + P(N) \\ \text{OKS}(M - W[N], N-1) + P(N) \end{array} \right\} & \text{otherwise} \end{cases}$$

M = Capacity of knapsack

N = no. of object

$$M, m$$

$$10 - 5$$

$$4 - 4$$

$$4 - 3$$

$$4 - 2$$

$$3 - 1$$

OKS.

$$12$$

$$7$$

$$2$$

$$\underline{\underline{21}}$$

Recursive Program for 0/1 Knapsack

OLKS(m, n)

{
if ($m == 0$ (or) $n == 0$)

return (0);

else

{
if ($m < w[n]$)

return (OLKS($m, n-1$))

else

return ($\max \left(\begin{array}{l} OLKS(m - w[n], n-1) + p[n] \\ OLKS(m, n-1) + 0 \end{array} \right)$)

$O(n^2)$

(1)

(m, n)

$(m - w_n, n-1)$

$(m, n-1)$

$(m - w_{n-1}, n-2)$ $(m, n-2)$

$(m, 0)$

In worst case

Level = n

Note \Rightarrow In the 0/1 Knapsack Program (recursive)
Step (A) gives worst case behaviour i.e.
every time two possibilities like them in time.
So binary tree 'n' levels.
So time complexity of 0/1 Knapsack = $O(2^n)$

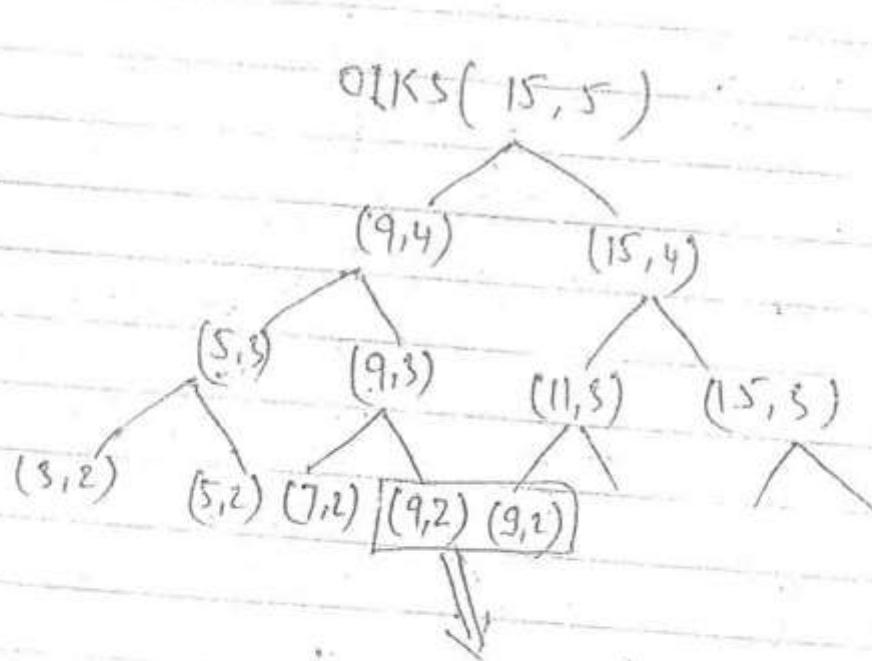
Solⁿ

$$M=15$$

$$n=5$$

Wt

Object	Ob ₁	Ob ₂	Ob ₃	Ob ₄	Ob ₅
Weight	3	1	2	4	6
Profit	7	2	1	6	12



Overlapping Subproblem

(Can be solved only one time in DP)

~~Q~~ How many distinct subproblems O/I KS(M,N) having.

$$M = 15$$

$$N = 5$$

(15, 5) ~~(15, 6)~~

(15, 4) (14, 4) (13, 4) 12, 11, 10 - - - 0

(15, 3) (14, 3) (13, 3)

(15, 2) (14, 2) (13, 2)

(15, 1) (14, 1) (13, 1)

(15, 0) (14, 0) (13, 0)

the no of ^{distinct} subproblem are.

$$O/I KS(M,N) = (M+1) * (N+1)$$

$$= M * N$$



larger

= Exponential

= N.P. Complete.

Dynamic Programming O/I KS

Using Dynamic Programming

Memorization - CLKS (M, N)

{

if (M == 0 (or) N == 0)

{ return (0);

Table [M, N] = 0

else

{

if (M < w[N])

{

if (Table [M, N-1] ≠ Nil)

return (Table [M, N-1])

else

{

Table [M, N-1] = CLKS (M, N-1)

return (Table [M, N-1])

}

else

{

if (Table [M, N-1] ≠ Nil)

Table [M-w[N], N-1] ≠ Nil)

return (max { Table [M, N-1] ,
Table [M-w[N], N-1] + p_N })

else

{

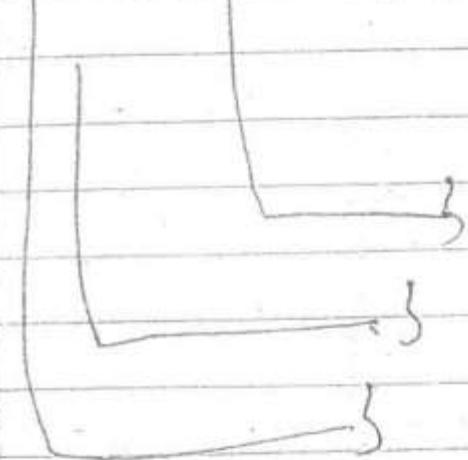
if Table [M, N-1] == nil)

Table [M , N-1] = OIKS { M , N-1 }

if (Table [M - w[N] , N-1] == nil)

Table [M - w[N] , N-1] = OIKS { M - w[N] , N-1 }

return { max { Table [M , N-1]
{ Table [M - w[N] , N-1] + P(N) } } }



Sum of Subsets Problem (NP, Complete)

$$S = \{10, 50, 30, 80, 70, 40, 100\}$$

↓ ↓ ↓ ↓ ↓ ↓ ↓
 1 2 3 4 5 6 7

$$M = 210$$

$$S_1 = \{30, 80, 100\}$$

$$S_2 = \{70, 40, 100\}$$

$$S_3 = \{10, 30, 70, 100\}$$

Recurrence Ref

$$\begin{aligned}
 \text{sos}(N, M) = & \begin{cases}
 0 & \text{if } M=0 \\
 1 & \text{if } N=0 \\
 \text{sos}(N-1, M) & \text{if } w[N] > M \\
 \max \left(\text{sos}(N-1, M-w[N]) \cup w[N] \right) & \text{if } w[N] \leq M
 \end{cases}
 \end{aligned}$$

Max ←
 complete binary tree with
 n-levels $\Rightarrow O(2^n)$

NM

7, 210
6, 110

5, 70
4, 0

9
0

00700400100

$SOS(N-1), m \cdot w(N) \cup w(N)$

- 00700400100

~~-210~~

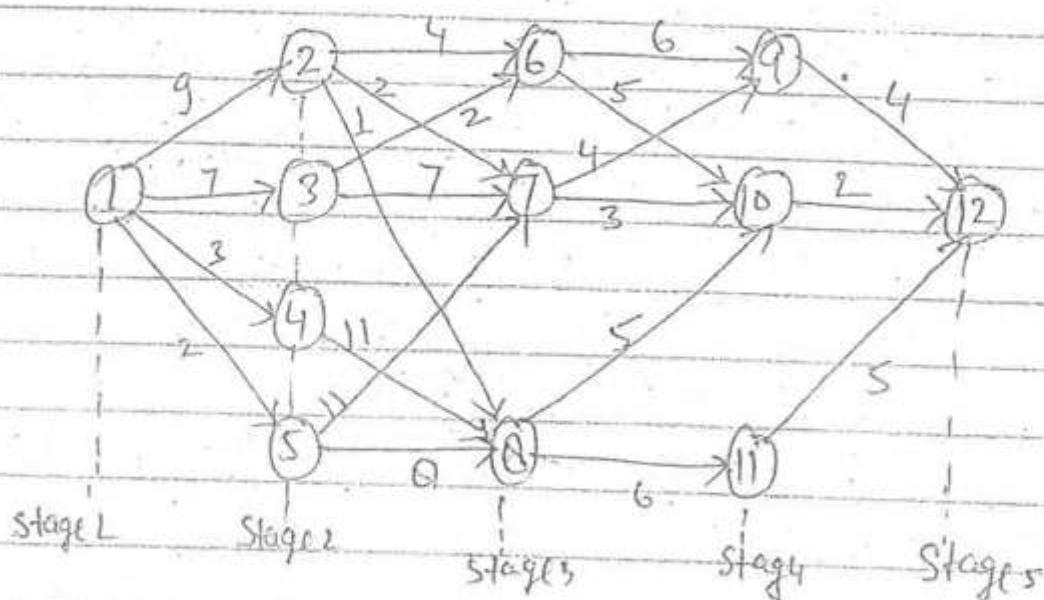
Using Dynamic Programming

$M \times N \Rightarrow O(MN)$



larger
 \hookrightarrow Exponential.

Multi Stage Graph.

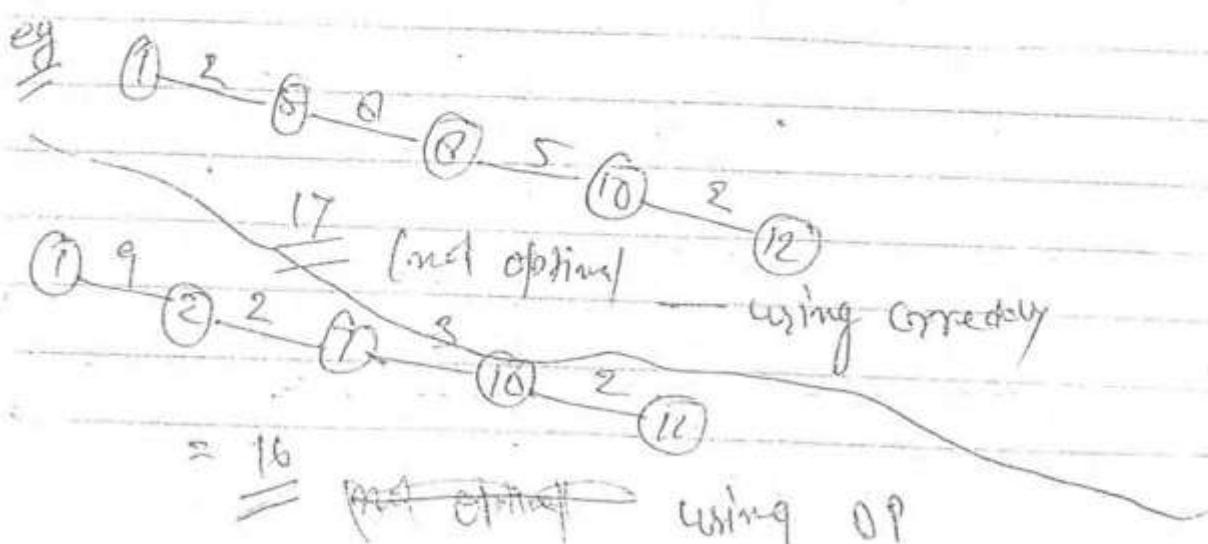


$MSG(1, 1)$

$MSG(2, 5)$

$MSG(3, 7)$

$MSG(s_i, v_j) =$ The min^m cost required from stage s_i to destination v_j .



$O(V+E)$

$$MSG(1,1) = \min_{\substack{16 \\ 1 \\ 2}} \left\{ \begin{array}{l} C(\overline{1,2}) + MSG(\overline{2,2}) \\ C(\overline{1,3}) + MSG(\overline{2,5}) \\ C(\overline{1,4}) + MSG(\overline{2,4}) \\ C(\overline{1,5}) + MSG(\overline{2,5}) \end{array} \right\}$$

$$MSG(2,2) = \min_{\substack{7 \\ 1 \\ 2}} \left\{ \begin{array}{l} C(\overline{2,6}) + MSG(\overline{3,6}) \\ C(\overline{2,7}) + MSG(\overline{3,7}) \\ C(\overline{2,8}) + MSG(\overline{3,8}) \end{array} \right\}$$

$$MSG(3,6) = \min_{\substack{7 \\ 5}} \left\{ \begin{array}{l} C(\overline{8,9}) + MSG(\overline{4,9}) \\ C(\overline{6,10}) + MSG(\overline{4,10}) \end{array} \right\}$$

$$MSG(\overline{4,9}) = \min \left\{ C(\overline{9,12}) \right\}$$

$$MSG(\overline{4,10}) = \min \left\{ C(\overline{10,12}) \right\}$$

$$MSG(\overline{3,7}) = \min_{\substack{5 \\ 3}} \left\{ \begin{array}{l} C(\overline{1,9}) + MSG(\overline{4,9}) \\ C(\overline{7,10}) + MSG(\overline{4,10}) \end{array} \right\}$$

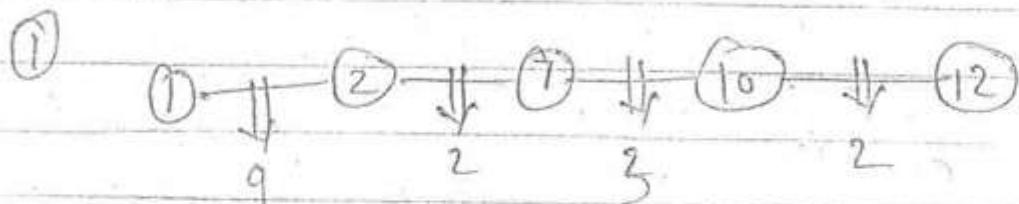
$$MSG(3,8) = \min \left\{ \begin{array}{l} c(\overline{8,10}) + MSG(\overline{4,10}) \\ c(\overline{8,11}) + MSG(\overline{4,11}) \end{array} \right.$$

$$MSG(4,11) = c(\overline{11,12})$$

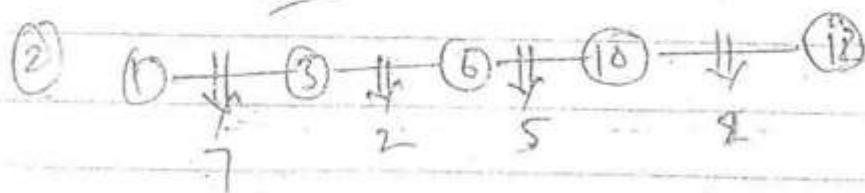
$$MSG(2,3) = \min \left\{ \begin{array}{l} c(\overline{3,6}) + MSG(\overline{3,6}) \\ c(\overline{3,7}) + MSG(\overline{3,7}) \end{array} \right.$$

$$MSG(2,4) = c(\overline{4,8}) + MSG(\overline{3,8})$$

$$MSG(2,5) = \min \left\{ \begin{array}{l} c(\overline{5,7}) + MSG(\overline{3,7}) \\ c(\overline{5,8}) + MSG(\overline{3,8}) \end{array} \right.$$



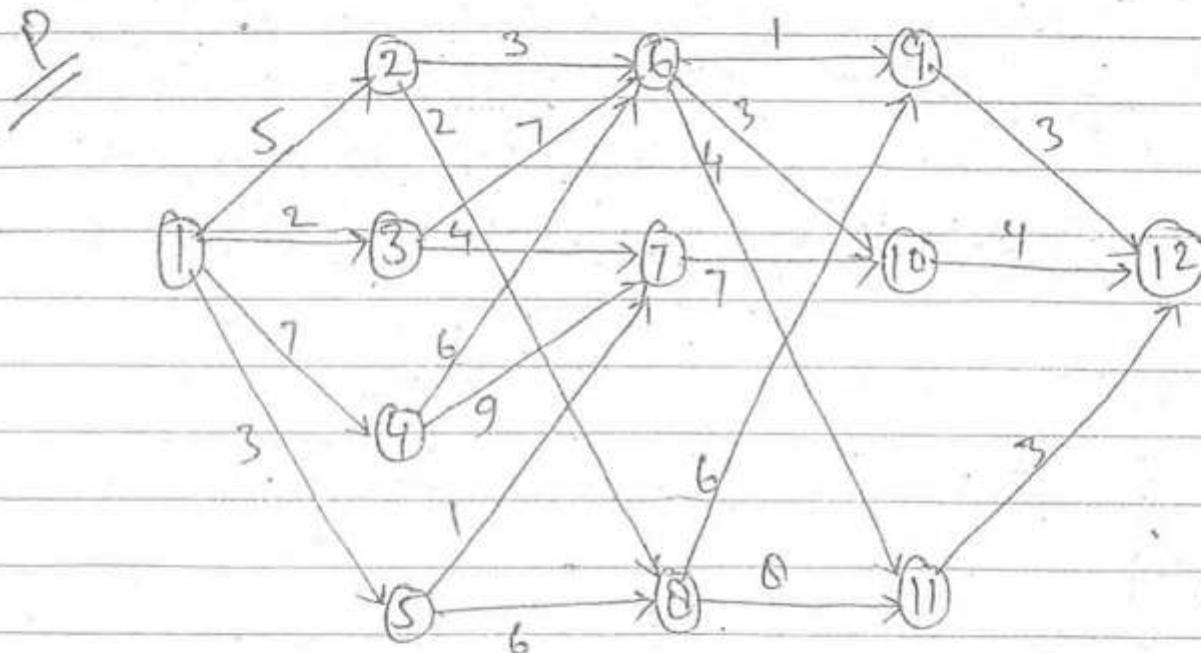
$$= \underline{\underline{16}}$$



$$=\underline{\underline{16}}$$

Note - Every edge is covered only once.
 At every vertex covered only once.

no of fun call = no of vertex



$$MSG(1,1) = \min \left\{ \begin{array}{l} C(\overbrace{1,2}^5) + MSG(\overbrace{2,3}^7) \\ C(\overbrace{1,3}^2) + MSG(\overbrace{2,4}^{11}) \\ C(\overbrace{1,4}^7) + MSG(\overbrace{2,5}^{10}) \\ C(\overbrace{1,5}^3) + MSG(\overbrace{2,6}^{12}) \end{array} \right\}$$

$$MSG(2,2) = \min \left\{ \begin{array}{l} c(\overline{2,6}) + MSG(\overline{\frac{3}{4},6}) \\ c(\overline{2,8}) + MSG(\overline{\frac{3}{9},8}) \end{array} \right\}$$

$$MSG(3,6) = \min \left\{ \begin{array}{l} c(\overline{6,9}) + MSG(\overline{\frac{3}{4},9}) \\ c(\overline{6,10}) + MSG(\overline{4,10}) \\ c(\overline{6,11}) + MSG(\overline{4,11}) \end{array} \right\}$$

$$MSG(3,8) = \min \left\{ \begin{array}{l} c(\overline{8,9}) + MSG(\overline{\frac{3}{4},9}) \\ c(\overline{8,11}) + MSG(\overline{4,11}) \end{array} \right\}$$

$$MSG(2,3) = \min \left\{ \begin{array}{l} c(\overline{3,6}) + MSG(\overline{\frac{4}{3},6}) \\ c(\overline{3,7}) + MSG(\overline{3,7}) \end{array} \right\}$$

$$MSG(3,7) = c(\overline{7,10}) + MSG(\overline{4,10})$$

$$MSG(2,4) = \min \left\{ \begin{array}{l} c(\overline{4,6}) + MSG(\overline{\frac{4}{3},6}) \\ c(\overline{4,7}) + MSG(\overline{\frac{9}{11},7}) \end{array} \right\}$$

$$MSG(2,5) = \min \left\{ \begin{array}{l} C(S, 7) + MSG(3, 7) \\ C(S, 8) + MSG(3, 8) \end{array} \right\}$$

Recurrence Relation

$$MSG(S_i, V_j) = \min \left\{ \begin{array}{l} C(V_j, D) \text{ if } S_i = S_f = 1 \\ C(V_j, k) + MSG(S_{i+1}, k) \\ \forall k \in S_{i+1} \\ (V_j, k) \in E \end{array} \right.$$

D = Destination.

S_f = Final stage.

Euler Graph & Hamiltonian Graph

Path : Sequence of zero (or) more edge

Open Path : $v_s \neq v_E$

Starting vertex is not equal to ending vertex.

Closed Path : $v_s = v_E$

Starting vertex is equal to ending vertex.

Euler Path : In the given graph path every edge of the given graph is covered exactly once.

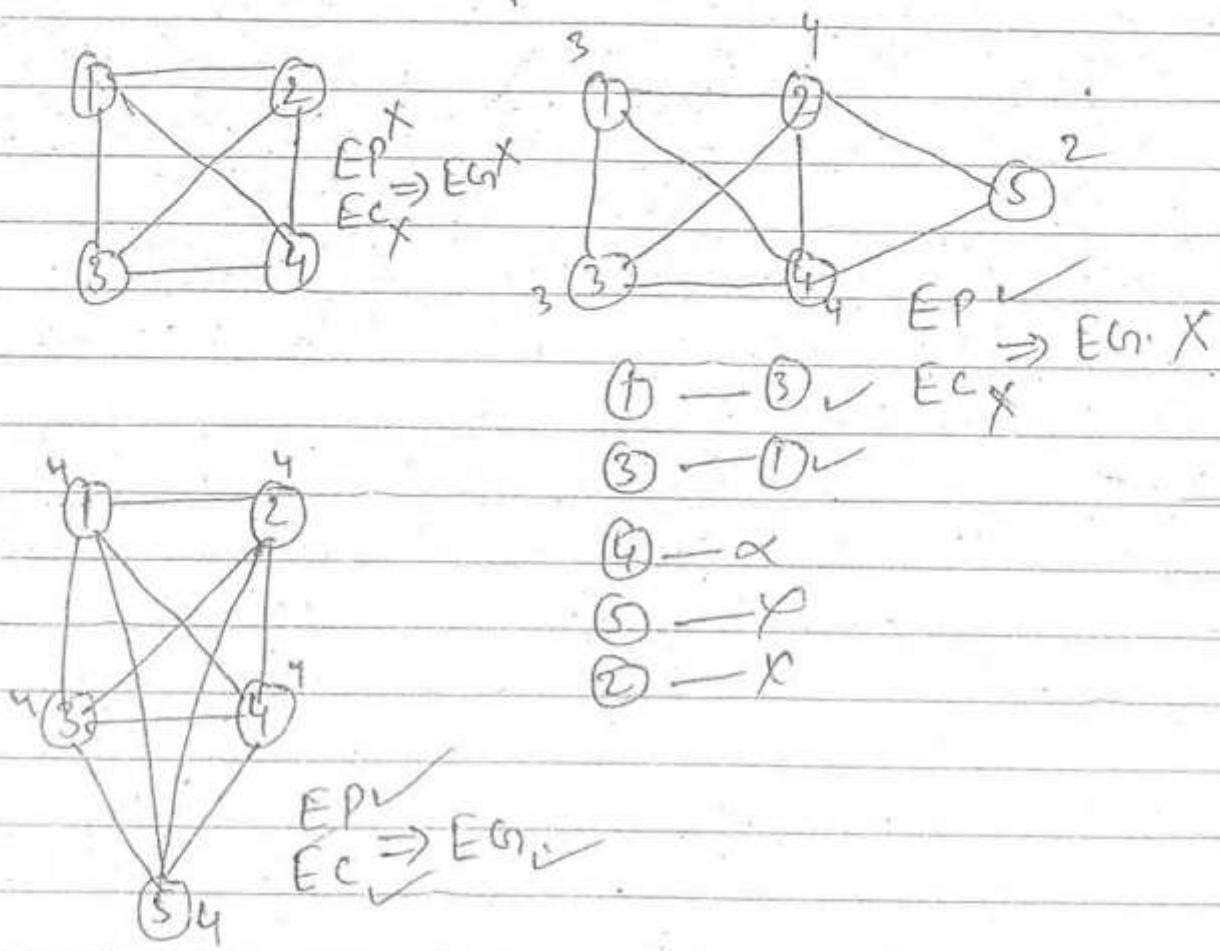
Euler Circuit : If it is an Euler path in which $v_s = v_E$ is called Euler Circuit.

Hamiltonian Path : In the given path every vertex of the given graph is covered exactly once.

Hamiltonian Cycle : It is closed Hamiltonian Path.

Hamiltonian Graph

Consider the following Graph.



In the given graph every vertex degree is even then that graph contain Euler path and Euler Circuit one possible then graphs called Euler graph.

To check a graph is Euler or not $O(|E|)$

Q class problem
bcz
Polynomial
time
complexity

$$= O(V^2)$$

Note In the Given Graph contain exactly two vertices with odd degree. Then too that graph, Euler Path is possible but Euler circuit is not possible.

To that Euler path one of the odd degree vertex is starting other is ending

Note

In the Given Graph contain more than two vertices with odd degree. Even Euler Path is not possible.

Which one of the following is true.

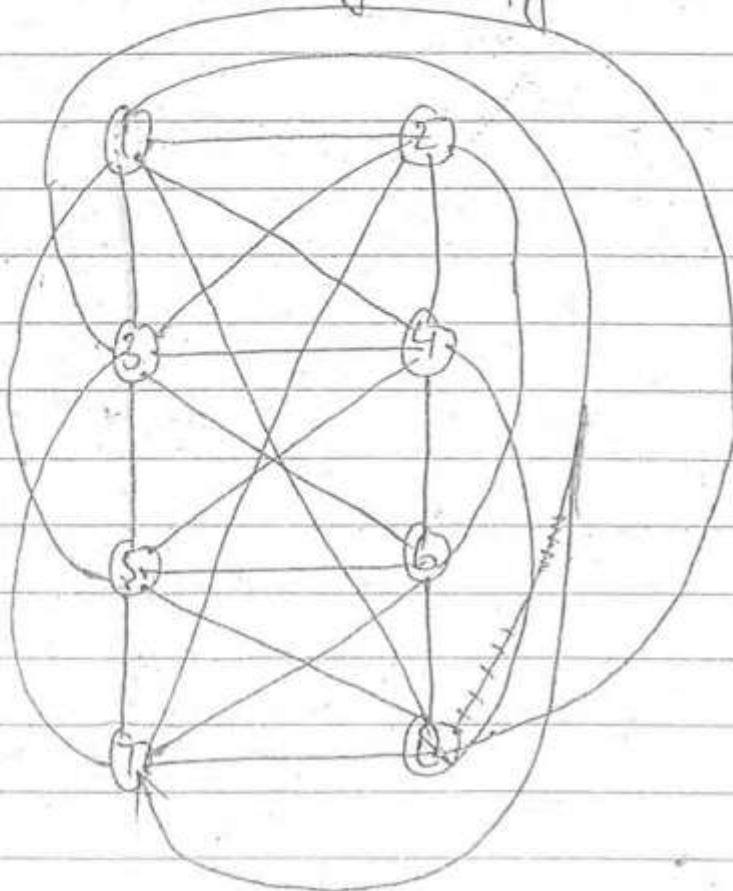
X a) $G \Rightarrow |V|=10, |E|=10$ is E.G.

X b) $G \Rightarrow |V|=50, |E|=100$ is E.G.

X c) $G \Rightarrow |V|=n, |E|=\frac{n(n-1)}{2}$ when n is even is E.G.

✓ d) $G \Rightarrow |V|=n, |E|=\frac{n(n-1)}{2}$ when n is odd
Every vertex degree is even
is E.G.

Q Consider the following graph.

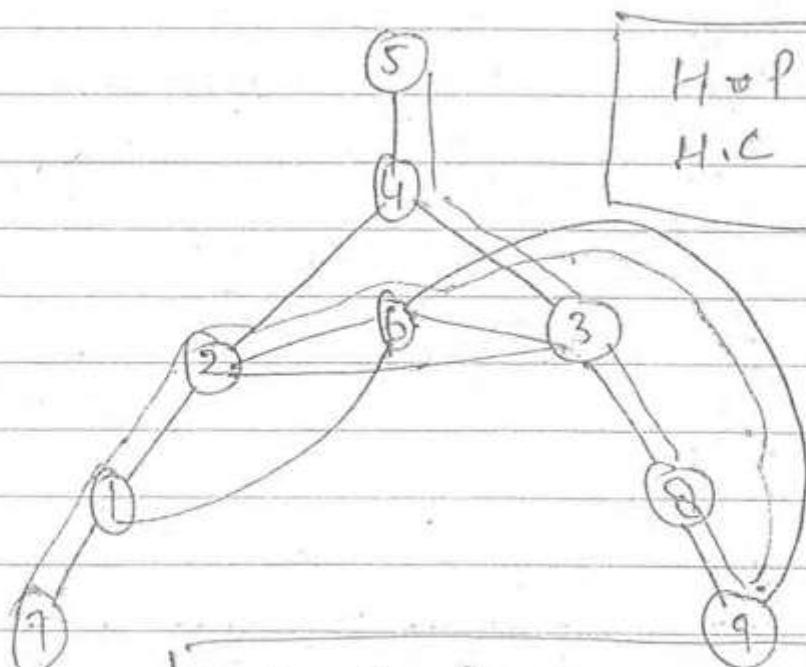


H.P

H.C

1 - 2 - 4 - 6 - 8 - 7 - 5 - 3

Q The following graph is H.G or not



H.P

H.C

3Sat

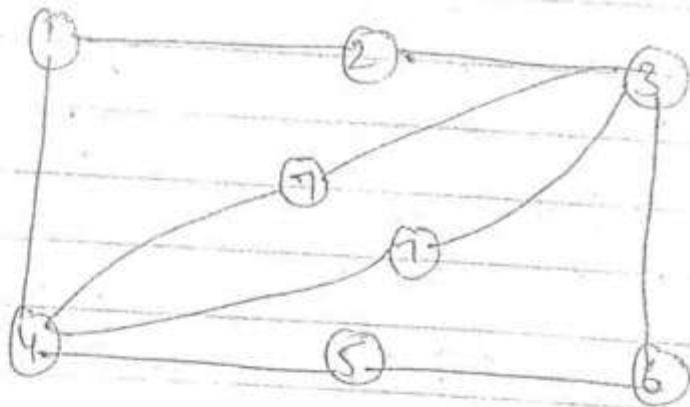
3-SAT
NP
Complete

P-Clean.

5 - 4 - 3 - 0 - 9 - 6 - 2 - 1 - 7

~~Note~~ To check Given graph is H.G or not
there is not polynomial time.

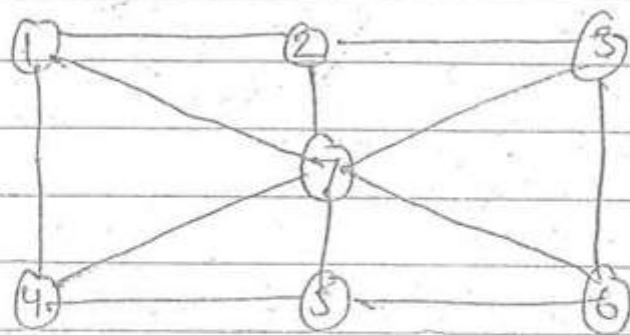
Q Which one of the following is true?



EG ✓

- a) G - both E & H
- b) G - E but not H
- c) G - H but not E
- d) both are not

 which one of the following is true ?

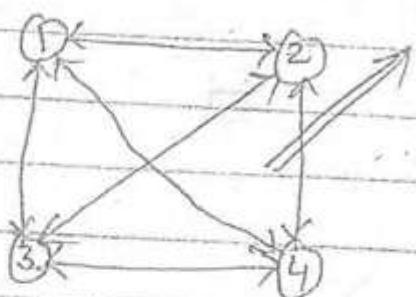


EC X
EP X

~~EG~~ X

- a) G - both E & H
- b) G - E but not H
- c) G - H but not E
- d) both are not

Travelling Sales Person (Hamiltonian tour)



$O(2^n) \Rightarrow$ Exponential (NP-complete)
(Because no overlapping problem)

Travel adjacent vertex exactly once and come back to your home. This condition is allowed in condition if the given graph is Hamiltonian graph.

1 2 3 4

1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

Cost adjacency matrix

of $N = \{1, 2, 3, 4\}$
initially source = 1
 $TSP = (1, (2, 3, 4))$

$TSP(s, V - \{s\})$

= The min cost required to go from s to all other remaining vertices in V exactly once & coming back to source (may be s)

$$\# TSP(1, (2, 3, 4)) = \min \left\{ \begin{array}{l} C(1, 2) + TSP(2, (3, 4)) \\ C(1, 3) + TSP(3, (2, 4)) \\ C(1, 4) + TSP(4, (2, 3)) \end{array} \right\}$$

~~Ans 35~~

$$\cancel{\#} \quad \text{TSP}(2, \{3, 4\}) = \min \left\{ \begin{array}{l} \frac{9}{10} \underbrace{c(2, 3) + \text{TSP}(3, 4)}_{20} \\ \frac{15}{12} \underbrace{c(2, 4) + \text{TSP}(4, 3)}_{15} \end{array} \right\}$$

$$\cancel{\#} \quad \text{TSP}(3, \{4\}) = \underline{c(3, 4)} + \underline{c(4, 1)}$$

$$\cancel{\#} \quad \text{TSP}(4, \{3\}) = \underline{c(4, 3)} + \underline{c(3, 1)}$$

$$\cancel{\#} \quad \text{TSP}(3, \{2, 4\}) = \min \left\{ \begin{array}{l} \frac{13}{12} \underbrace{c(3, 2) + \text{TSP}(2, 4)}_{10} \\ \frac{13}{13} \underbrace{c(3, 4) + \text{TSP}(4, 2)}_{13} \end{array} \right\}$$

$$\cancel{\#} \quad \text{TSP}(2, \{4\}) = \underline{c(2, 4)} + \underline{c(4, 1)}$$

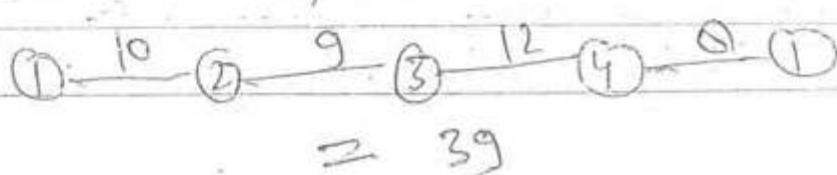
$$\cancel{\#} \quad \text{TSP}(4, \{2\}) = \underline{c(4, 2)} + \underline{c(2, 1)}$$

$$\cancel{\#} \quad \text{TSP}(4, \{2, 3\}) = \min \left\{ \begin{array}{l} \frac{8}{9} \underbrace{c(4, 2) + \text{TSP}(2, 3)}_{15} \\ \frac{9}{9} \underbrace{c(4, 3) + \text{TSP}(3, 2)}_{15} \end{array} \right\}$$

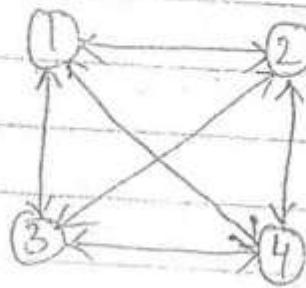
$$\cancel{\#} \quad \text{TSP}(2, \{3\}) = \underline{c(2, 3)} + \underline{c(3, 1)}$$

$$\cancel{\#} \quad \text{TSP}(3, \{2\}) = \underline{c(3, 2)} + \underline{c(2, 1)}$$

Using Greedy



Consider the following Graph.



	1	2	3	4
1	0	5	10	7
2	6	0	11	5
3	9	5	0	6
4	4	11	0	0

Source = 1

$$TSP(1, \{2, 3, 4\}) = \min_{\substack{\text{26}}} \left\{ \begin{array}{l} C(1, 2) + TSP(2, \{3, 4\}) \\ C(1, 3) + TSP(3, \{2, 4\}) \\ C(1, 4) + TSP(4, \{2, 3\}) \end{array} \right\}$$

$$TSP(2, \{3, 4\}) = \min \left\{ \begin{array}{l} C(2, 3) + TSP(3, \{4\}) \\ C(2, 4) + TSP(4, \{3\}) \end{array} \right\}$$

$$TSP(3, \{4\}) = C(3, 4) + C(4, 1)$$

$$TSP(4, \{3\}) = C(4, 3) + C(3, 1)$$

$$\text{TSP}(3, \{2, 4\}) = \min \left\{ \begin{array}{l} c(3, 2) + \text{PSP}(\overline{2 \{4\}}) \\ c(3, 4) + \text{TSP}(\overline{4 \{2\}}) \end{array} \right\}$$

$$\text{TSP}(\overline{2}, \{4\}) = c(\overline{2, 4}) + c(\overline{4, 1})$$

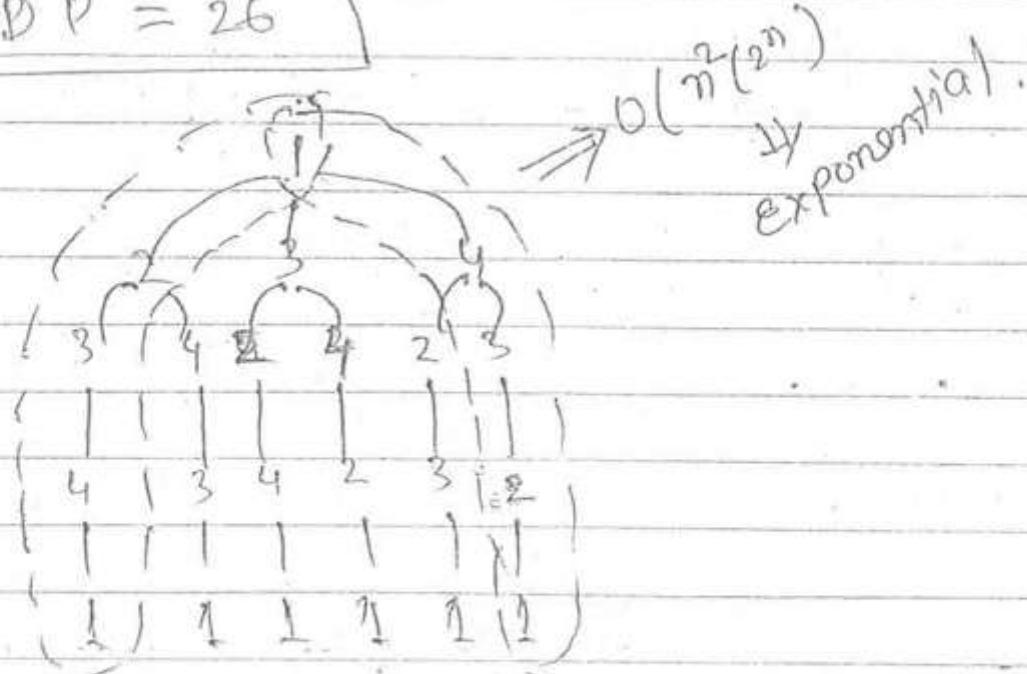
$$\text{TSP}(\overline{4}, \{2\}) = c(\overline{4, 2}) + c(\overline{2, 1})$$

$$\text{TSP}(\overline{4}, \{2, 3\}) = \min \left\{ \begin{array}{l} c(\overline{4, 2}) + \text{PSP}(\overline{2 \{3\}}) \\ c(\overline{4, 3}) + \text{TSP}(\overline{3 \{2\}}) \end{array} \right\}$$

$$\text{TSP}(\overline{2}, \{3\}) = c(\overline{2, 3}) + c(\overline{3, 1})$$

$$\text{TSP}(\overline{3}, \{2\}) = c(\overline{3, 2}) + c(\overline{2, 1})$$

Hence DP = 26



Recurrence Relation

$$TSP(S, V') = \min_{\substack{1, \{2, 3, 4\}}} \left\{ \begin{array}{l} c(S, \text{source}) \text{ if } V' = \emptyset \\ c(S, K) + TSP(K, V' - K) \\ \forall K \in V' \\ (S, K) \in E \end{array} \right\}$$

Day Run

$$TSP(S, V')$$

$$1, \{2, 3, 4\}$$

$$2, \{2, 4\}$$

$$4, \{2\}$$

$$2, \emptyset$$

$$c(S, K)$$

$$1, 3$$

$$3, 4$$

$$1, 2$$

$$TSP(K, V' - K)$$

$$3, \{2, 4\}$$

$$4, \{2\}$$

$$2, \{\emptyset\}$$

③

Matrix chain multiplication (MCM)

$$A_{2 \times 3}, B_{3 \times 2}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$



$$C = A * B$$

$\begin{matrix} 2 \times 2 & 2 \times 3 & 3 \times 2 \end{matrix}$
 $= \boxed{\quad \quad \quad}$
 $\begin{matrix} & & \\ & & \\ & & \end{matrix}$
 2×2

$\text{eg} \quad A_{5 \times 10} \quad B_{10 \times 20}$ $C = A * B$ $C_{5 \times 20} \Rightarrow 100$ $5 \times 20 \times 10$ \Downarrow 1000	$A_{m \times n} \quad B_{n \times p}$ $C = A * B$ $C_{m \times p} \Rightarrow n$ $\Rightarrow mnp$
---	---

~~***~~ Let $A_{m \times n}$, $B_{n \times n}$ Matrix them

$$\text{if } C = A * B$$

then C size is $C_{m \times n}$

to get every element of C , n -multiplied

total multiplication are $= n^2 \times n$

$$\Rightarrow n^3$$

eg

$$A_{5 \times 10}, B_{10 \times 5}; C_{5 \times 20}$$

$$(AB)C \Rightarrow 750$$

they to find
how many min. no of multiplication are
required.

soln

$$(AB)C$$

$$A_{5 \times 10}, B_{10 \times 5}, C_{5 \times 20}$$

$$(AB)$$

$$5 \times 5 \quad C_{5 \times 20}$$

$$(AB)C$$

$$5 \times 20 \times 5$$

$$= 250$$

$$500$$

$$\underline{750}$$

eg

$$A = \begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix}$$

$$B = \begin{Bmatrix} 2 & 3 \\ 4 & 5 \end{Bmatrix}$$

$$C = \begin{Bmatrix} 5 & 4 \\ 5 & 6 \end{Bmatrix}$$

$$(A(BC))$$

$$((AB)C)$$

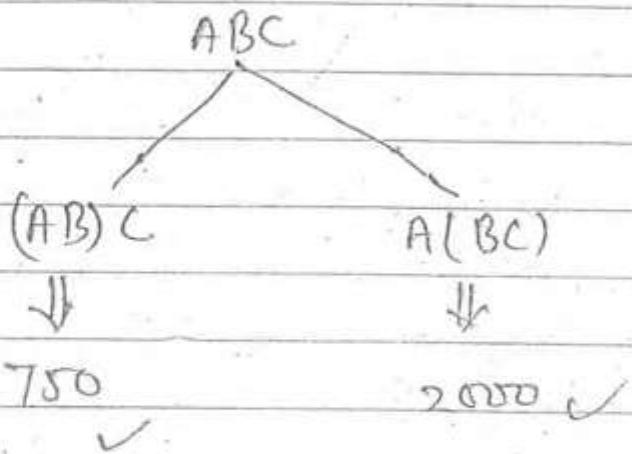
$$A = \begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix}, BC = \begin{Bmatrix} 21 & 26 \\ 37 & 46 \end{Bmatrix}$$

$$AB = \begin{Bmatrix} 10 & 13 \\ 22 & 29 \end{Bmatrix}, C = \begin{Bmatrix} 3 & 4 \\ 5 & 6 \end{Bmatrix}$$

$$A(BC) = \begin{Bmatrix} 95 & 110 \\ 211 & 262 \end{Bmatrix}$$

$$(AB)C = \begin{Bmatrix} 95 & 110 \\ 211 & 262 \end{Bmatrix}$$

Note - Matrix multiplication satisfied associative property

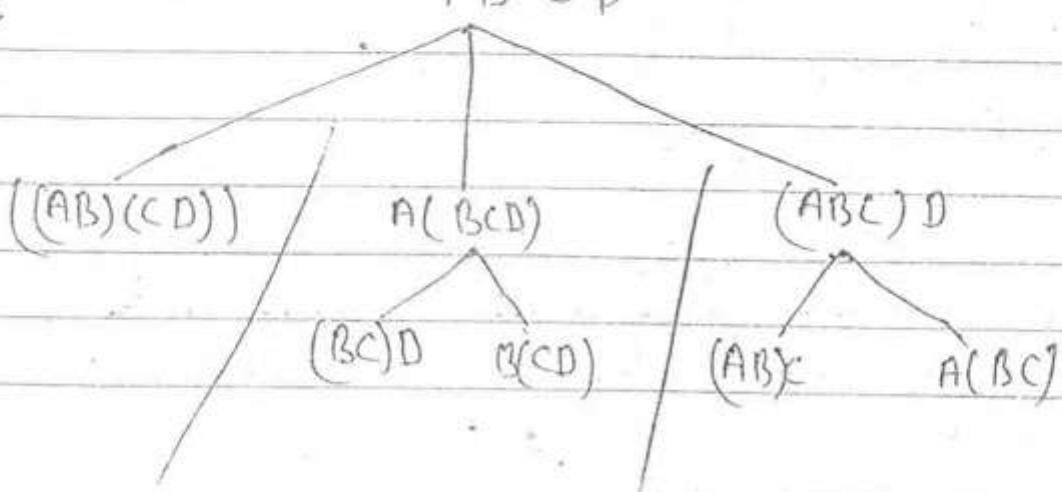


Note → In Matrix chain multiplication we don't want resultant matrix after multiplying n matrix.

In mcm our goal is to multiply chain of n matrices we have to find out min no of multiplication required.

eg. $A_{2 \times 5}, B_{5 \times 10}, C_{10 \times 5}, D_{5 \times 2}$ to find ABCD
how many min^m no of multiplication required?

Sol^m



$$\underline{S_2^2} \quad ((AB)(CD))$$

$$(AB)$$

$$2 \times 10 \times 5 = 100$$

$$(CD)$$

$$10 \times 2 \times 5 = 100 \rightarrow \begin{array}{r} 100 \\ 100 \\ \hline 200 \end{array}$$

$$(ABCD)$$

$$2 \times 2 \times 10 = 40 \quad \underline{\quad}$$

$$100$$

$$100$$

$$40$$

$$240$$

$$\underline{(BC)D}$$

$$\underline{B(CD)}$$

$$BC$$

$$5 \times 5 \times 10 = 250$$

$$CD \quad 10 \times 2 \times 5 = 100$$

$$(BC) \underline{5 \times 5}, D \underline{10 \times 2} = 50$$

$$B \underline{5 \times 10}, (CD) \underline{10 \times 2} = 100$$

$$= 250$$

$$50$$

$$\underline{300}$$

$$= 100$$

$$100$$

$$\underline{200}$$

$$\underline{(AB)C}$$

$$A(BC)$$

$$AB \quad 2 \times 10 \times 5 = 100$$

$$BC \quad 5 \times 5 \times 10 = 250$$

$$(AB) \quad 2 \times 10, C \underline{10 \times 5} = 100$$

$$A \underline{2 \times 5}, (BC) \underline{5 \times 5} = 50$$

$$= 100$$

$$100$$

$$\underline{200}$$

$$= 250$$

$$50$$

$$\underline{300}$$

$$A((BC)D) = A_{2 \times 5} \times (BC)_5 \times D_{5 \times 2} \Rightarrow (ABC)_2 \times 2 \times 5 = 20 \\ = 320$$

$$A(B(CD)) \Rightarrow 220$$

$$(AB)C \times D = 220$$

$$(A(BC))D = 320$$

$$((AB)(CD)) = 240 \quad (((AB)C)D) = 220$$

$$(A((BC)D)) = 320 \quad ((A(BC))D) = 320$$

$$A((B(CD))) = 220$$

* Four matrices M_1, M_2, M_3, M_4 of dimensions $P \times q$, $q \times r$, $r \times s$, $s \times t$ respectively can be multiplied.

In several ways with diff no of property if

$$P=10, q=100, r=20, s=5, t=80, \text{ Hence } n$$

min^m no of multiplications $M_1 \times M_4$ is required

- a) 2,40,000
- b) 44,000
- c) 19,600
- d) 2,50,000

SFM

$M_1 M_2 M_3 M_4$

$((M_1 M_2) (M_3 M_4))$

$(M_1 M_2 M_3) M_4$

$M_1 (M_2 M_3 M_4)$

$(M_1 M_2) M_3 \quad M_1 (M_2 M_3)$

$M_2 (M_3 M_4) = (M_2 M_3) M_4$

$$(M_1 M_2) \quad (M_3 M_4)$$

$10 \times 100 \quad 100 \times 20 \quad 20 \times 5 \quad 5 \times 80$

$$M_1 M_2 = 10 \times 20 \times 100 = 20,000$$

$$M_3 M_4 = 20 \times 80 \times 5 = 8000$$

$$(M_1 M_2) (M_3 M_4) = 10 \times 80 \times 20 = 16000$$

$10 \times 20 \quad 20 \times 80$

$$= 20000$$

$$8000$$

$$16000$$

$$\underline{44000}$$

$$m_2(m_3 m_4)$$

$$m_3 m_4 = 8000 \checkmark$$

$$m_2(m_3 m_4)$$

$$100 \times 20 \quad 26 \times 80$$

$$= 100 \times 80 \times 20$$

$$= 160000 \checkmark$$

$$m_1(m_2(m_3 m_4))$$

$$10 \times 100 \quad 100 \times 80$$

$$= 10 \times 80 \times 100$$

$$= 80000 \checkmark$$

$$= 2,40,000$$

$$(m_2 m_3) m_4$$

$$m_2 m_3 \\ 100 \times 20 \quad 20 \times 5$$

$$= 100 \times 5 \times 20 = 10000 \checkmark$$

$$(m_2 m_3) m_4$$

$$100 \times 5 \quad 5 \times 80$$

$$= 100 \times 80 \times 5$$

$$= 40000 \checkmark$$

$$m_1((m_2 m_3) m_4)$$

$$10 \times 100 \quad 100 \times 80$$

$$= 10 \times 80 \times 100$$

$$= 80000 \checkmark$$

$$= 1,30,000$$

$$(M_1 M_2) M_3$$

$$M_1 M_2 = 20,000 \quad \checkmark$$

$$(M_1 M_2) M_3$$

$$10 \times 20 \quad 20 \times 5$$

$$= 10 \times 5 \times 20$$

$$= 1000 \quad \checkmark$$

$$((M_1 M_2) M_3) M_4$$

$$10 \times 5 \quad 5 \times 00$$

$$= 10 \times 00 \times 5$$

$$= 4000 \quad \checkmark$$

$$= 25000$$

$$M_1 (M_2 M_3)$$

$$M_2 M_3 = 10,000$$

$$M_1 (M_2 M_3)$$

$$10 \times 100 \quad 100 \times 5$$

$$= 10 \times 5 \times 100$$

$$5000 \quad \checkmark$$

$$(M_1 M_2 M_3) M_4$$

$$10 \times 5 \quad 5 \times 00$$

$$= 10 \times 00 \times 5$$

$$= 4000 \quad \checkmark$$

$$= 19000 \quad \checkmark$$

#

$$\overbrace{A B C D}^{A B C D} A B C$$

$$A (B C D)$$

$$(A B) (C D)$$

$$(A B C) D$$

$$A (B C)$$

$$(A B) C$$

Recurrence Reln for MCM

$$MCM(i, j) = \min_{\substack{i \leq k \leq j \\ \text{min}}} \left\{ \begin{array}{l} 0 \quad \text{if } i=j \\ MCM(i, k) + MCM(k+1, j) \\ + NMRTMTM \end{array} \right\}$$

no of multiplication required
to multiply the total multiplication

$$k=1 \quad A(BCD)$$

$$k=2 \quad (AB)(CD)$$

$$k=3 \quad (ABC)D$$

Ex $A_{2 \times 5}, B_{5 \times 2}, C_{2 \times 5}, D_{5 \times 2}$ find mcm of 1 to 4.
matrix is equal to

$$\begin{aligned} & \cancel{80^m} \\ MCM(1, 4) &= \min \left\{ \begin{array}{l} \cancel{MCM(1, 1)} + MCM(2, 4) + \cancel{20} \\ \cancel{20} + MCM(3, 4) + \cancel{20} \\ MCM(1, 5) + MCM(4, 4) + \cancel{20} \end{array} \right\} \\ & \checkmark O(n) \end{aligned}$$

$$MCM(2,4) = \min \left\{ \begin{array}{l} MCM(2,2) + MCM(3,4) + 20 \\ MCM(2,3) + MCM(4,4) + 50 \end{array} \right. \left. \begin{array}{l} 0 \\ 50 \end{array} \right\}$$

~~$O(n)$~~

$$MCM(3,4) = MCM(3,3) + MCM(4,4) + 20$$

0 0

(CD) $C_{2 \times 5} \quad D_{5 \times 2}$ $2 \times 2 \times 5 = 20$	$B(CD)$ $5 \times 2 \quad 2 \times 2$ $5 \times 2 \times 2 = 20$
---	--

$$MCM(2,3) = MCM(2,2) + MCM(3,3) + 50$$

0 0

(BC) $B_{5 \times 2} \quad C_{2 \times 5}$ $5 \times 5 \times 2 = 50$	$(BC)D$ $5 \times 5 \quad 5 \times 2$ $5 \times 2 \times 5 = 50$
---	--

$$A(BCD)$$

$2 \times 5 \quad 5 \times 2$

$$ABCD$$

$2 \times 2 \times 5 \Rightarrow 20$

$$\overbrace{mcm(1,2)}^{20} = \underbrace{mcm(1,1)}_0 + \underbrace{mcm(2,2)}_0 = 20$$

$$(AB)_{2 \times 2 \times 5} = 20$$

$$(A \bar{B}) \frac{(CD)}{2 \times 2} = 0$$

$$mcm(1,5) = \min \left\{ \begin{array}{l} mcm(1,1) + mcm(\cancel{2,3}) + \cancel{50} \\ \cancel{40} \\ mcm(1,2) + mcm(3,5) + \cancel{20} \end{array} \right\} = 20$$

$$A(BC)$$

$$2 \times 5 \quad 5 \times 2$$

$$10 \times 5 = 50$$

$$(AB)(C)$$

$$2 \times 2 \quad 2 \times 5$$

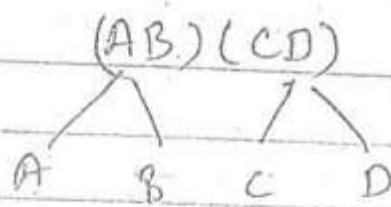
$$10 \times 2 = 20$$

$$(AB)(B)$$

$$2 \times 5 \quad 5 \times 2$$

$$= 2 \times 2 \times 5$$

$$= 20 \cancel{\pi}$$



How many different funⁿ calls.

	1	2	3	4
1	X			
2		X		
3	X			
4		X	X	X

$$\frac{n \times n}{2} = \frac{n^2}{2} \Rightarrow n^2 \text{ fun } O(n)$$

We have to find min $O(n^3)$

In Every funⁿ call we have to find minⁿ

$$\Rightarrow O(n^3)$$

	1	2	3	4
1	0	20	40	40
2	0	0	50	40
3	0	0	0	20
4	0	0	0	0

All pair shortest path



Single source shortest path

① Dijkstra's $\Rightarrow O((E + V) \log V)$

② Bellman-Ford $\Rightarrow O(VE)$



Idea

for ($i=1, i \leq n, i++$)

{
Dijkstraalgo(v_i)
}

① Dijkstra algo $\times |V|$



$V \times ((E + V) \log V)$



$O((V^2 E + V^3) \log V)$

② Bellman-Ford $\times |V|$

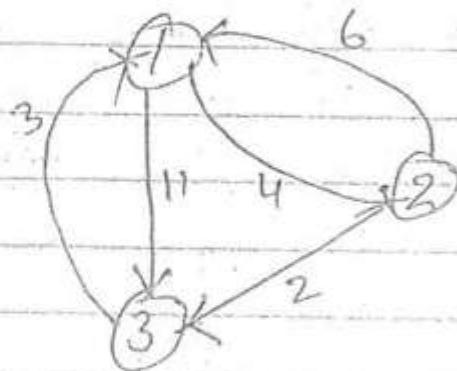


$|V| \times (VE)$



$O(V^2 E)$

All pair shortest Path (SP)



$A^K(i, j) \equiv$ The minⁿ cost required to go from i to j. By taking intermediate vertex numbers.

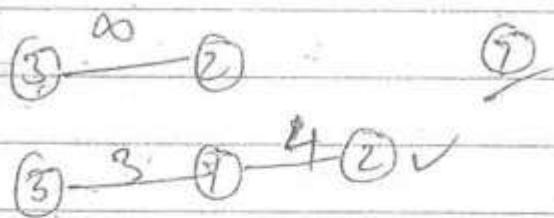
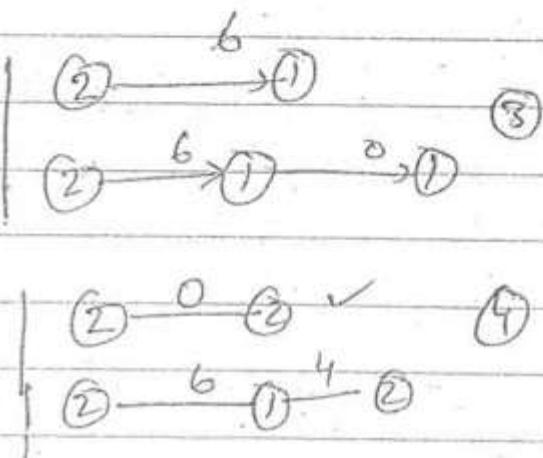
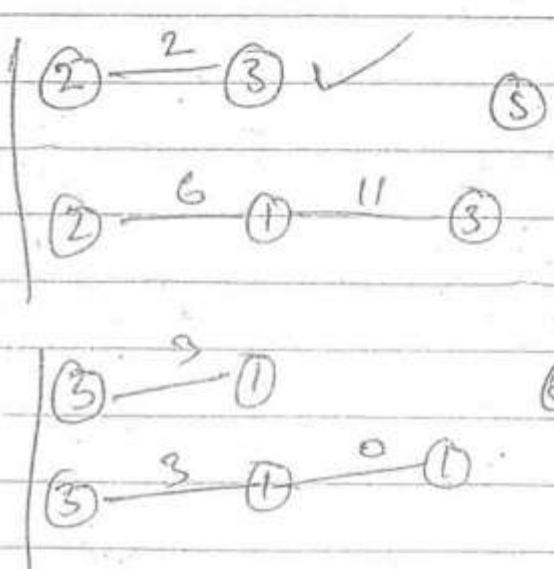
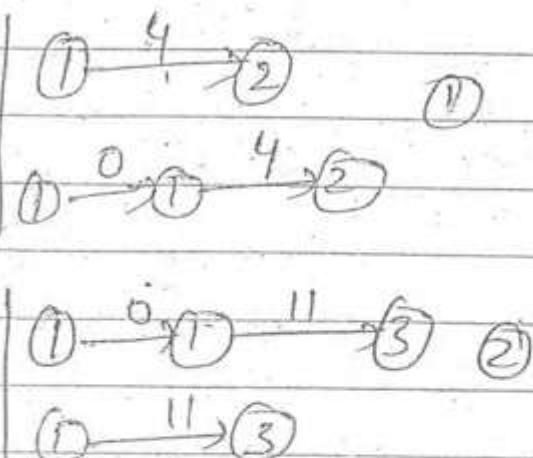
$A^K(i, j) \rightarrow$ The minⁿ cost required to go from i to j. By taking intermediate vertex number not greater than K.

~~without SP~~

A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

(1) \rightarrow (j)

A	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0



A^2	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

A^3	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

| ① — 4 — ② — 6 — ① |

| ① — ② — ② |

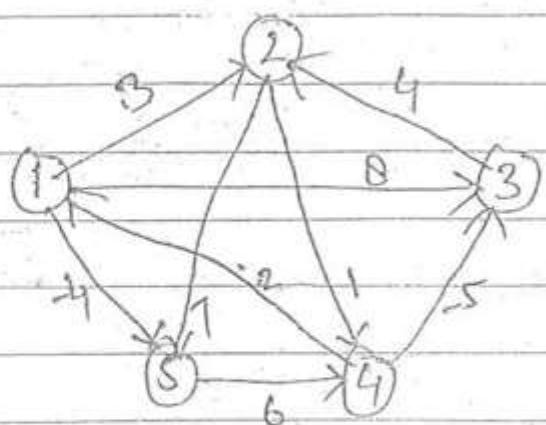
| ① — 11 — ③ |

Floyd-Warshall's All-pairs shortest path algo

$$A^K(i, j) = \min \left\{ \begin{array}{l} \text{Previous value} \\ \frac{A^{K-1}(i, j), A^{K-1}(i, k) + A^{K-1}(k, j)}{\Phi} \\ \text{via } K \end{array} \right\}$$

$$A^3(1, 3) = \min \{ A^2(1, 3), A^2(1, 2) + A^2(2, 3) \}$$

P



Adj

A ⁰	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

A ¹	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

A²

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	4	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

A³

	1	2	3	4	5
1	0	3	4	4	-4
2	∞	0	∞	1	7
3	∞	4	4	5	11
4	2	-1	-5	0	2
5	∞	∞	∞	6	0

A⁴

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	0	5	1	6	0

A^5	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	0	5	1	6	0

for ($i=1$, $i \leq |V|$, $i++$)

for ($j=1$, $j \leq |V|$, $j++$)

for ($s=1$, $s \leq |V|$, $s++$)

$$A^K(i, j) = \min \left\{ A^{K-1}(i, j), A^{K-1}(i, s) + A^{K-1}(s, j) \right\}$$

$\therefore O(n^3)$

~~22-7"~~

Graph Traversal

- ① BFT (Breadth first traversal)
- ② DFT (Depth first traversal)

Graph traversal is not unique & Pre-traversal is unique.

BFT (V)

{

visited (v) = 1;

add (v, q);

while ($q \neq \emptyset$)

{

v = dekte (q)

for all w adjacent to v;

{

if (w is not visited)

{

visited (w) = 1;

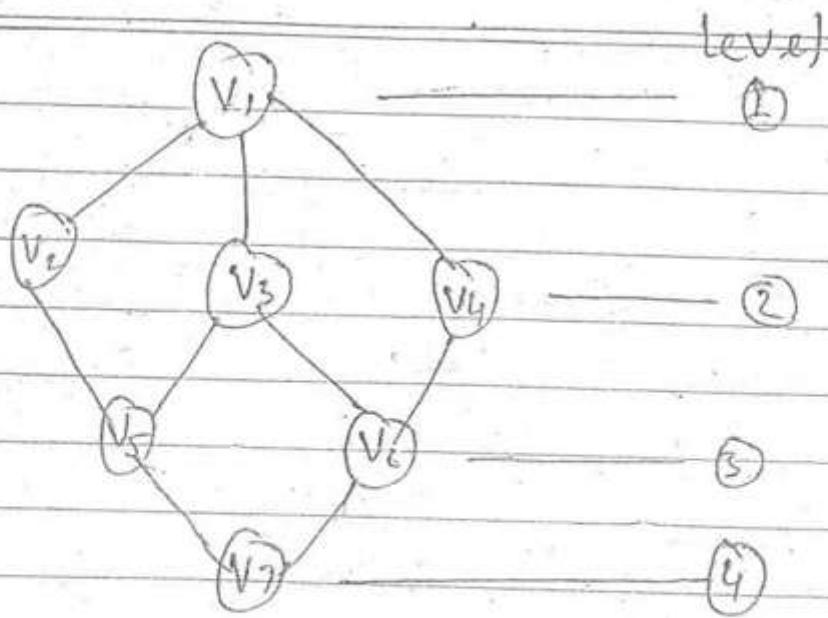
add (w, q);

}

}

}

}



$w(v_2, v_3, v_4)$ adjacent to $u(v_1)$

$w(v_1, v_5)$ adjacent to $u(v_2)$

- (1) The data structure we are using BFT is Queue.
- (2) BFT nothing but visited level by level becoz of random the BFT is called as level order traversal.

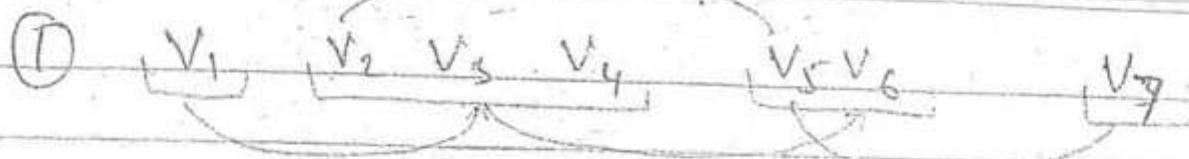
$v_1, v_2, v_3, v_4, v_5, v_6, v_7$
 1 2 3 4

- (3) the time complexity of BFT :

$$O(V + V^2)$$

$$O(V + E)$$

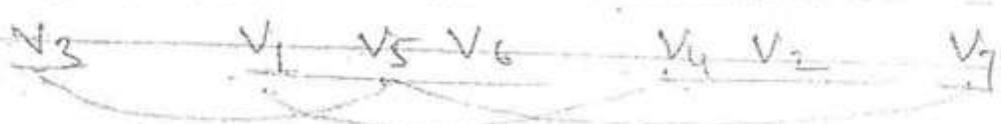
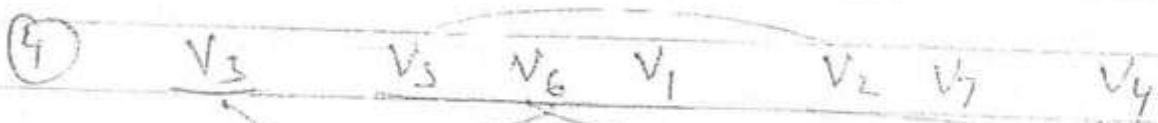
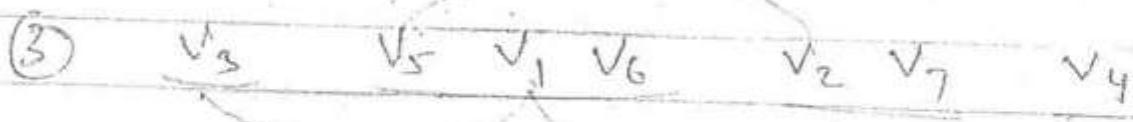
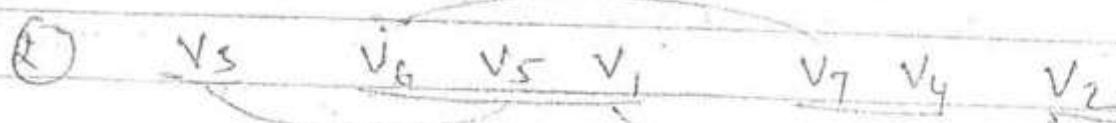
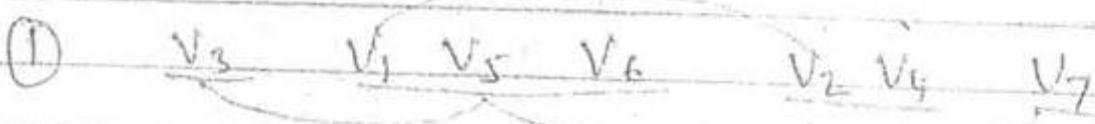
— ↓
 $\underline{O(V+E)}$



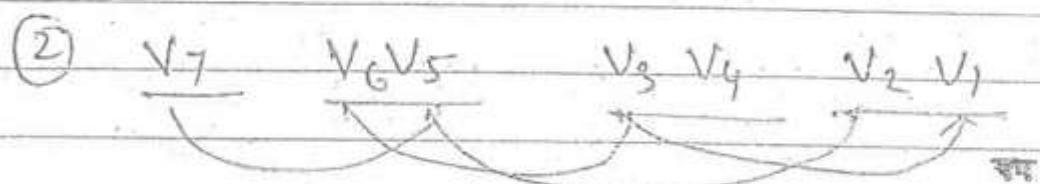
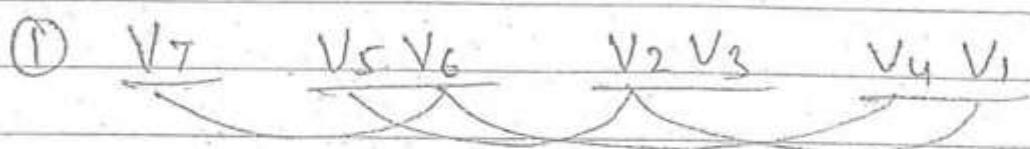
②

③

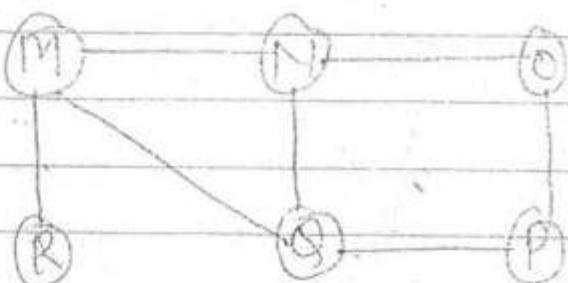
find 4 different answers if you start from
 v_3 in BFT.



find 2-different answers if start from
 v_7 in BFT.



2 Consider the following graph.

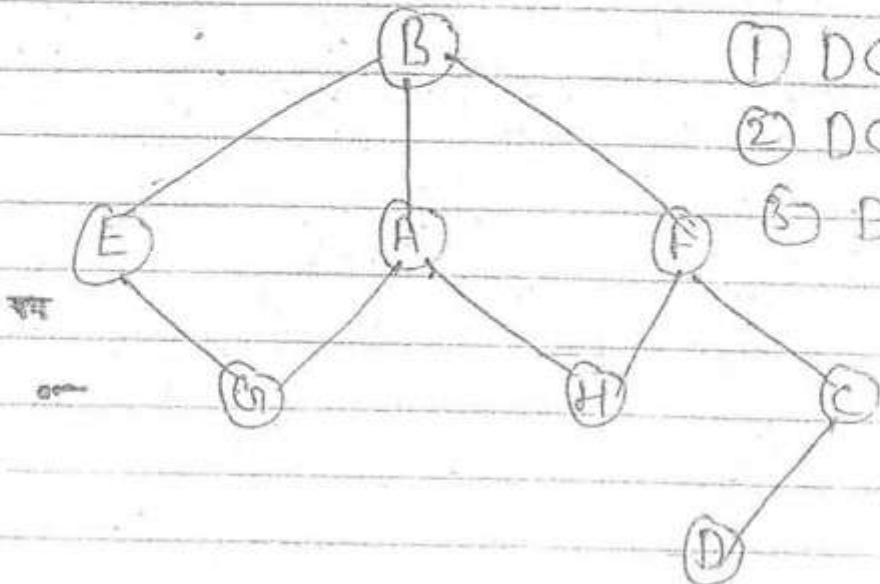


BFT applying on the above graph which one of the following is one possible order of visiting nodes in BFT.

- a) MNOPQR
- b) NQMOPR
- c) QMN PRO
- d) QMN POR

Q Consider the following Graph.

Find 3-different BFT starting from D.



① DCFBHEAG

② DCFHBAEG

③ DCFBHAEGL

④ DCFAEBHGE

⑤ DACEFBHAGEG

⑥ DCFAEBHAGEG

Q Consider undirected Graph G.

Let a BFT of G be done starting from a node r .

Let $d(r, u)$ and $d(r, v)$ be the length of the shortest paths from r to u and r to v respectively in G .

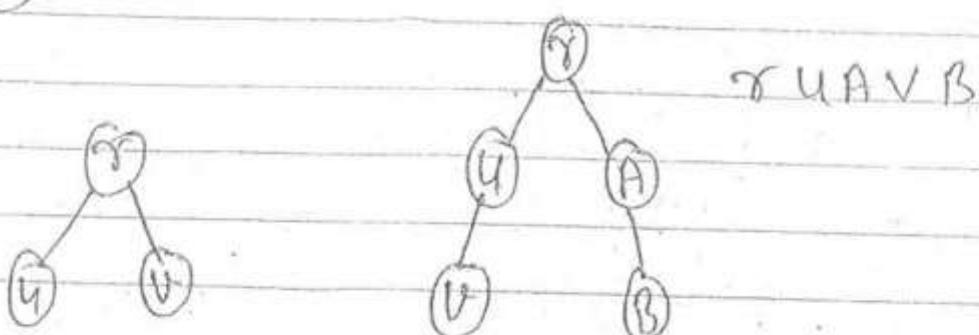
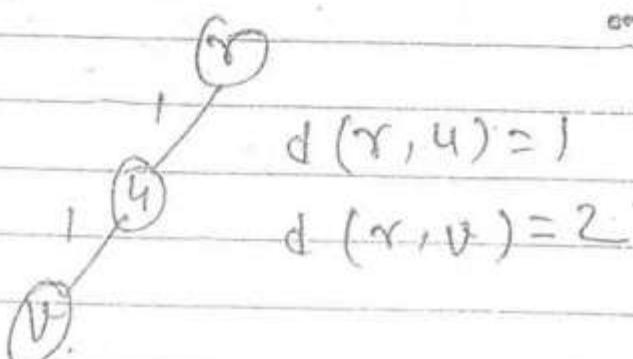
if u visited is visited before v during BFT,
which one of the following statements are
correct.

a) $d(\gamma, u) < d(\gamma, v)$

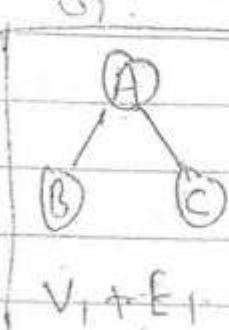
b) $d(\gamma, u) > d(\gamma, v)$

c) $d(\gamma, u) \leq d(\gamma, v)$.

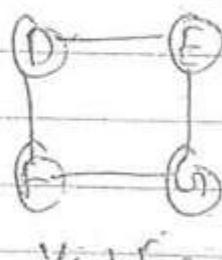
d) $d(\gamma, u) \geq d(\gamma, v)$



$d(\gamma, u) \leq d(\gamma, v)$



$V_1 + E_1$

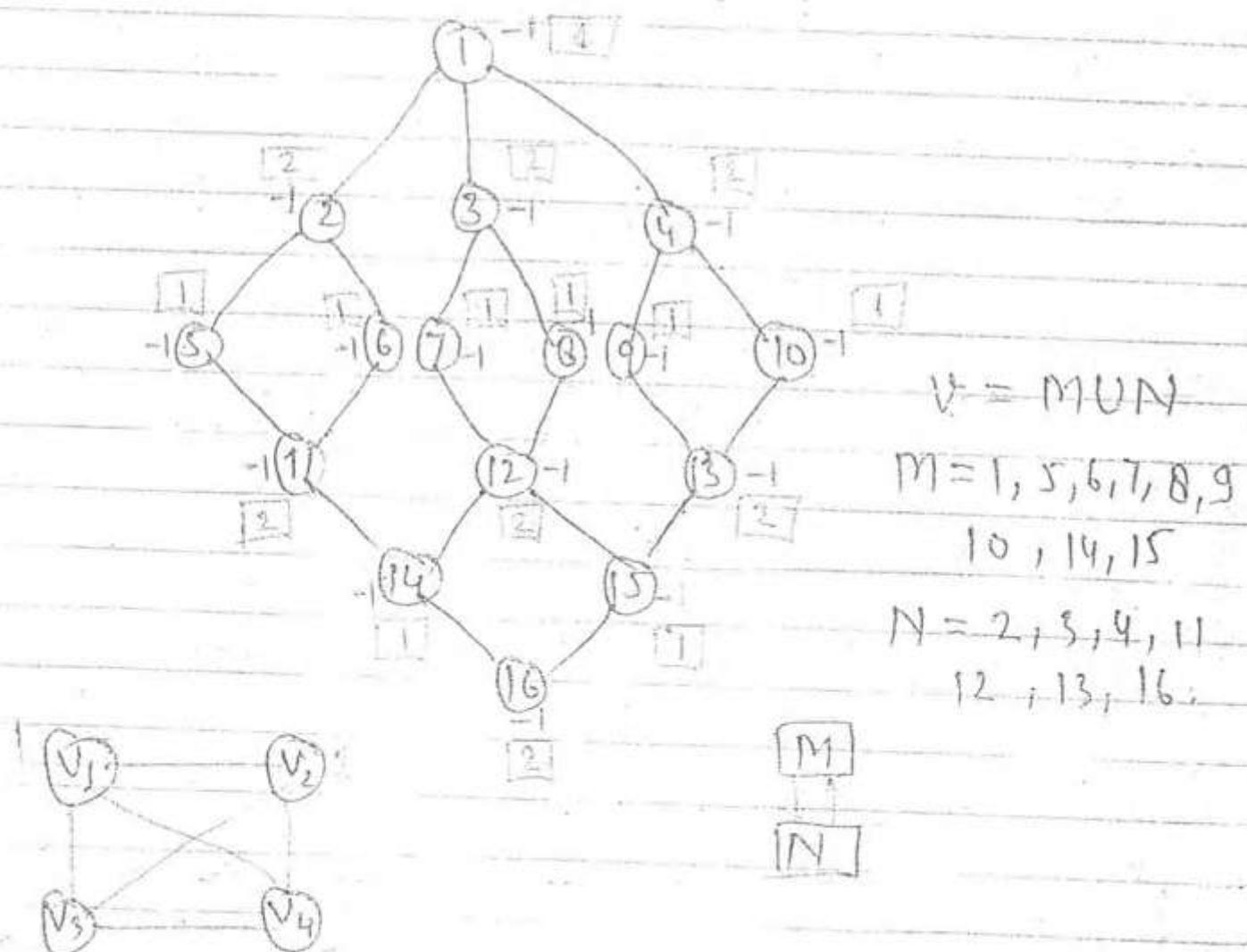


$G(A, B, C, D, E, F, G, H, I)$

$\Rightarrow O(V+E)$

Applications of BFT

- ① Using BFT algo we can check given graph is completed or not.
- ② Using BFT algo we can find out no of connected component in the given graph.
- ③ Using BFT algo we can find out given derived graph contain cycle or not.
- ④ Using BFT algo we can find out given graph is bipartite graph or not.



Fold boundary Method

Key = 123456789

M = 1000

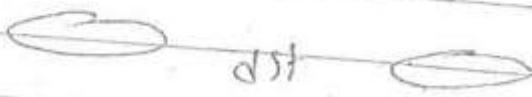
123

789

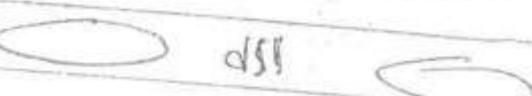
654321

9n	12345678
	9

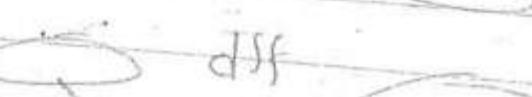
K₁



K₂



K₃



C₆₇₈₉

Mid Square

Key

9452

M = 1000

↓↓

(9452)²

↓↓

8931034

403	9452 8931034
-----	-----------------

Digit Extraction Method (Truncation)

$$m = 1000$$

e.g. $\underline{3} \underline{7} \underline{9} \underline{4} \underline{5} \underline{2} = 394$

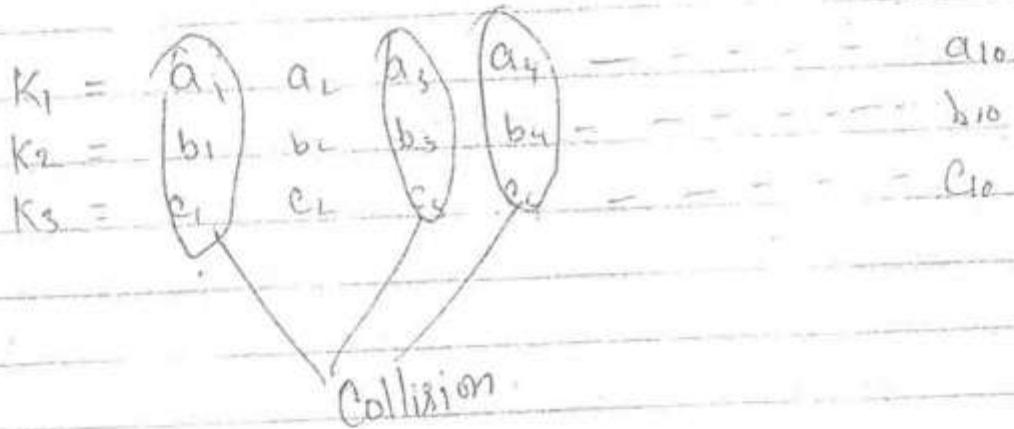
1 3 4

1 2 1 2 6 7 = 112

5 7 0 0 4 5 = 388

112	1 2 1 2 6 7
394	3 7 9 4 5 2

- Using digit extraction method selected digit are extracted from the key and use as the hash address.



Folding Method

① fold shift Method

Key = (1 2 3) (4 5 6) (7 8 9)

$$m = 1000$$

123
456
789
(136) 0

$$736$$

$$\overline{8}$$

$$144$$

$$144$$

$$123456789$$

bcz result always contain 1st P-number of bit (P-LSB bit)

If the given no of m is prime no than no of collision will be s.

Given no is prime number and not base to power of 2 or 10 is always preferable.

eg

$$\textcircled{1} \quad M = 2^p \times$$

result always contain
P-bit

\textcircled{2} \quad M is Prime.

It is advisable.

\textcircled{3} \quad M is prime

not close to power of 2
is good choice.

Types of Hash function

- ① Division modulo Method.
- ② Digit extraction Method. (Trunction)
- ③ Folding Method.
- ④ Mid Square

① Division Modulo Method \Rightarrow

Hash Table Size = M

Key

Key Modulo M.

e.g.

Hash Table Size = 10

Key = 625

$$\textcircled{S} = 625 \bmod 10$$

e.g

$$m = 2^4$$

Key = 15

$$15 \bmod 16 = 15$$

② In division modulo method m should not be powers of 2 means powers of 2 cannot apply

1) Direct address is not possible practically.

2) In order to eliminate all the problems present in DAT we are going to Concept of Hashing..

Hash address

Keys

$25 \bmod 10$

0	
1	61
2	
3	33
4	
5	25
6	
7	57
8	40
9	

Keys

H
A
S
H
I
N
G
F
U
I
N
O
C
W

Hash Table

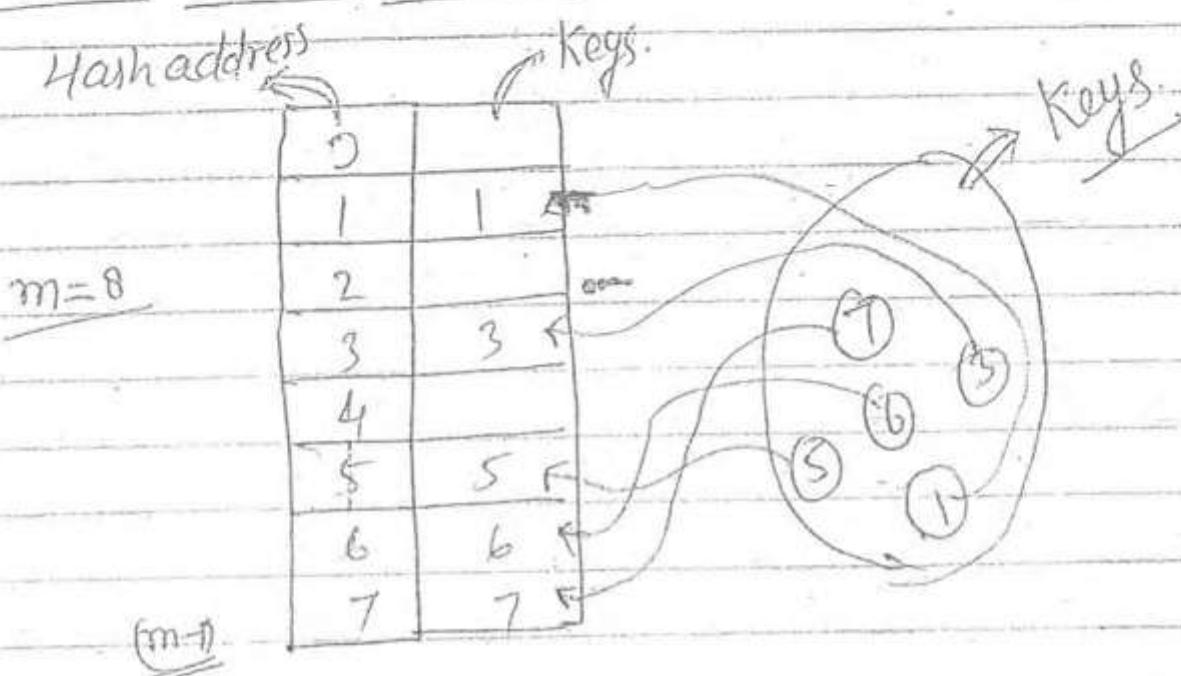
Producing the same Hash address that is already produced is called Collision.

The purpose of hash function is reducing the given key into ~~the~~ one of the Hash function.

Hashing

- ① It is one of the searching Technique.
- ② Average $\Rightarrow O(1)$

Direct address Table



Hash Table

- ① In direct address / Hashing Table Key is the address without any manipulation
- ② Even Though no of Keys are very less but one of the Keys may contains 64 bit.
- ③ The range of Keys determines size of the Hash Table.

Consider the following C Program

Border(root)

{

 if (root != null)

{

 Border(root->left);

 Pointt (root->data);

 AOrder (root->right);

}

}

AOrder(root)

{

 if (root != null)

{

 Pointt (root->data),

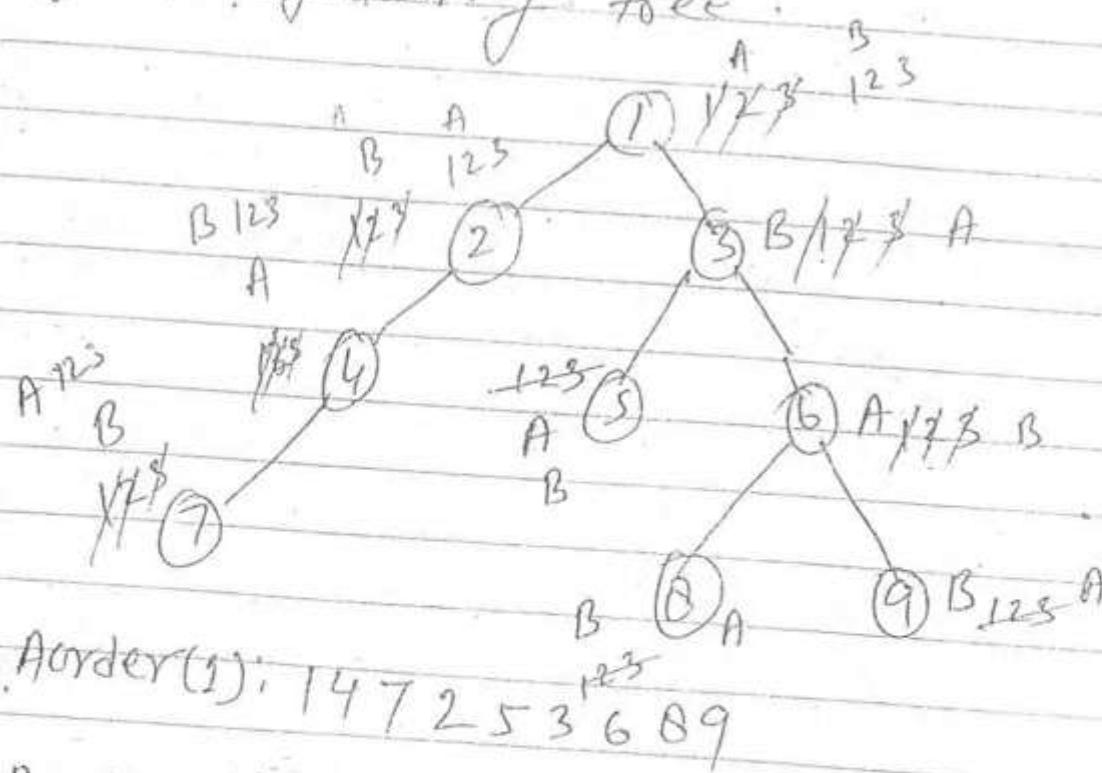
 Border (root->left);

 Border (root->right);

}

}

What will be the output of the above program
for the following tree.

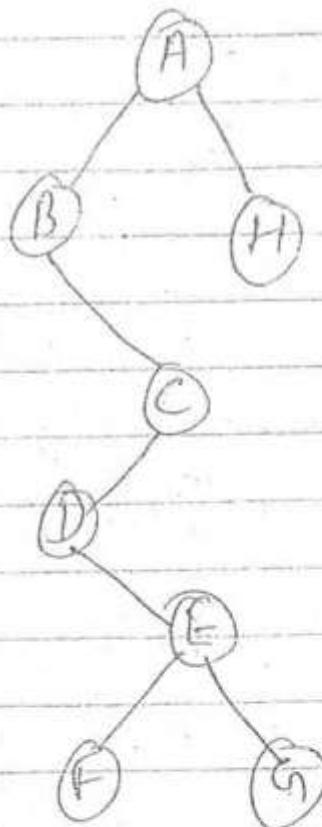


Q Construct the following C program. What will be the O/P if we apply the given program on binary search tree.

```
{  
    if (root != null)  
    {  
        printf("root-data")  
        To (root → left)  
        printf("root → data")  
        To (root → right)  
        printf("root → data")  
    }  
}
```

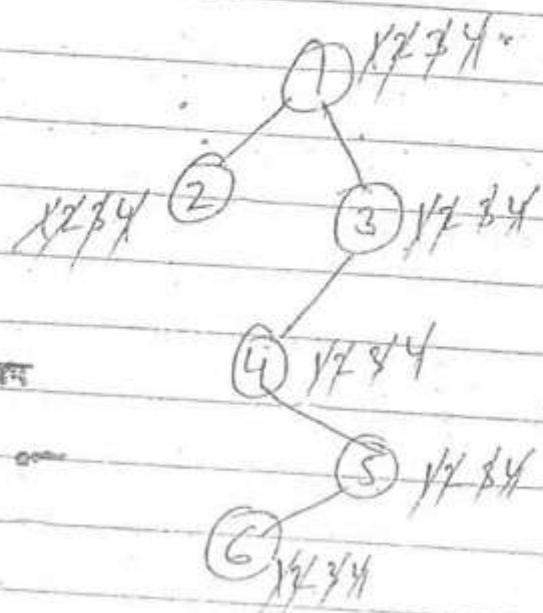
Sol

A B B C D D E F F F
E G G G E D C C B A
H H H A



A B B

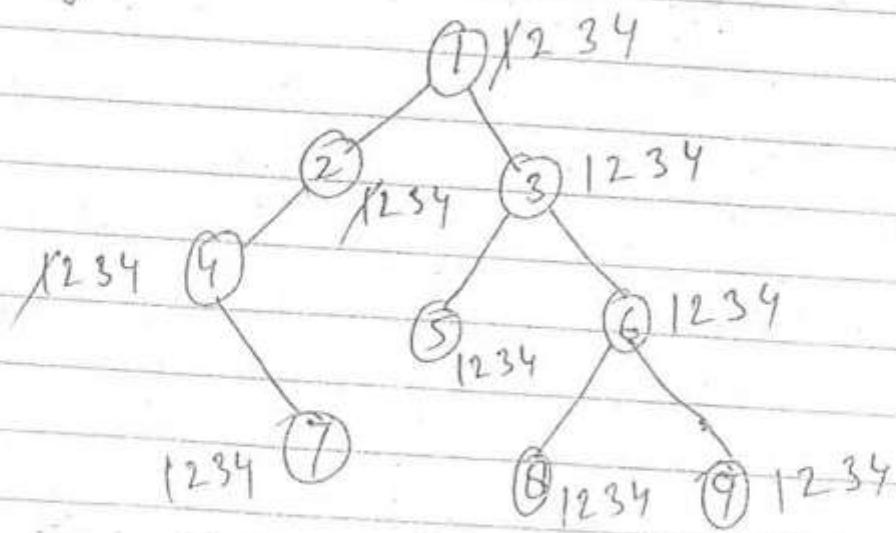
③ Printf(root.data)
 ④ Do [root → right]
 {
 }
 3



Soln

~~1 2 3 4 5 6~~
 1 2 2 1 3 4 4 5 6 6 5 3

What will be the o/p for the above code of the following tree.

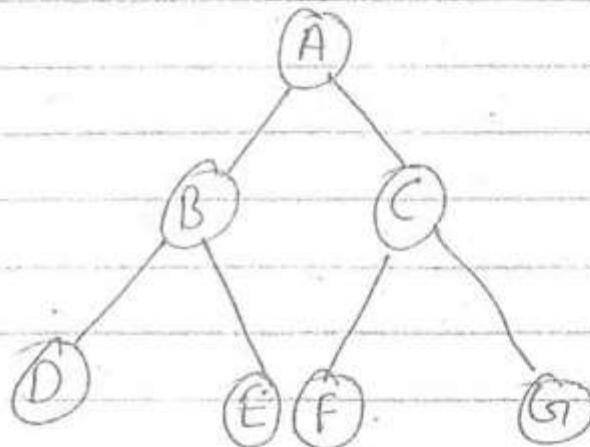


m

1 2 4 4 7 7 2 1 3 5 5 3 6 0 8 6 9 9

~~#~~
ABC (Root)

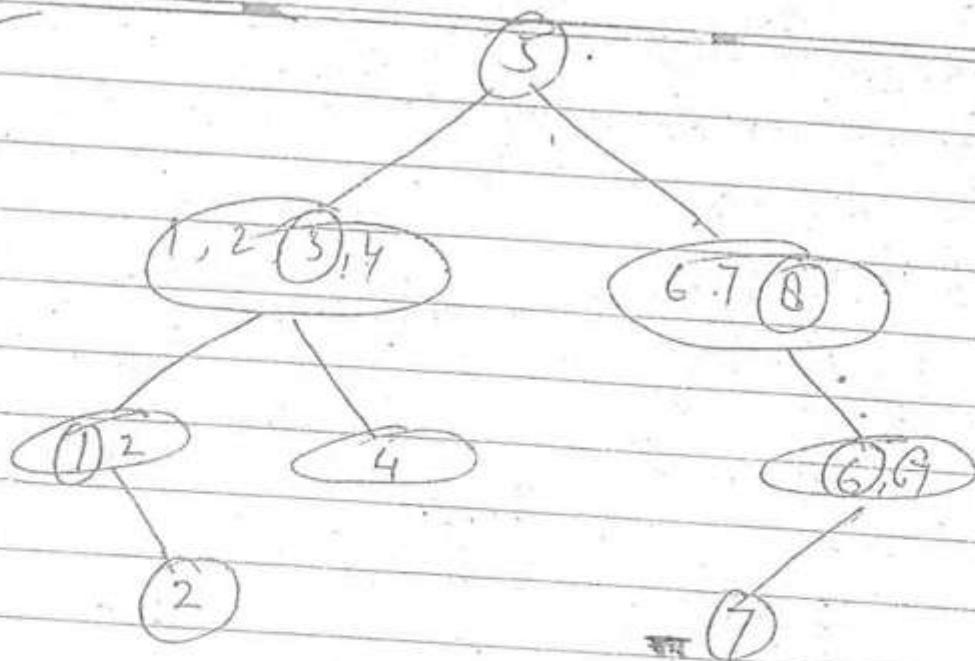
```
if (root != null)
{
    ① ABC ( root → right )
    ② Printt ( root → data )
    ③ ABC ( root → left )
}
```



C, C F, A, E B, D

Q What will be the output for the following program .

```
Do (root)
{
    if (root != null)
        ① Printt (root - data)
        ② Do (root → left)
```



Post order $2, 1, 4, 3, 7, 8, 6, 5$

Note - In order traversal of binary search tree will give always ascending order of element.

For every element of PreOrder or Postorder, apply Linear search to find LST and RST in InOrder.

$O(n)$

\downarrow
 $O(n^2)$

PreOrder-Postorder

eg PreOrder - A B D E C F G

Postorder - D E B F G C A

↓

(A) Unique binary tree
not possible.

Note - Using Preorder and Inorder we can construct unique binary tree.

Using Postorder - Inorder \Rightarrow Unique binary tree.

Using PreOrder - Postorder \Rightarrow Not Unique.

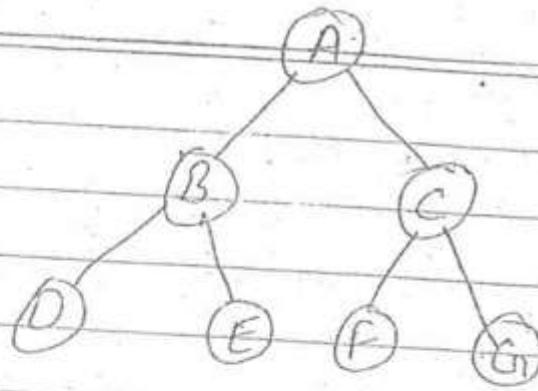
↳

A binary search tree contains following Preorder.

PreOrder \Rightarrow 5, 3, 1, 2, 4, 6, 8, 7

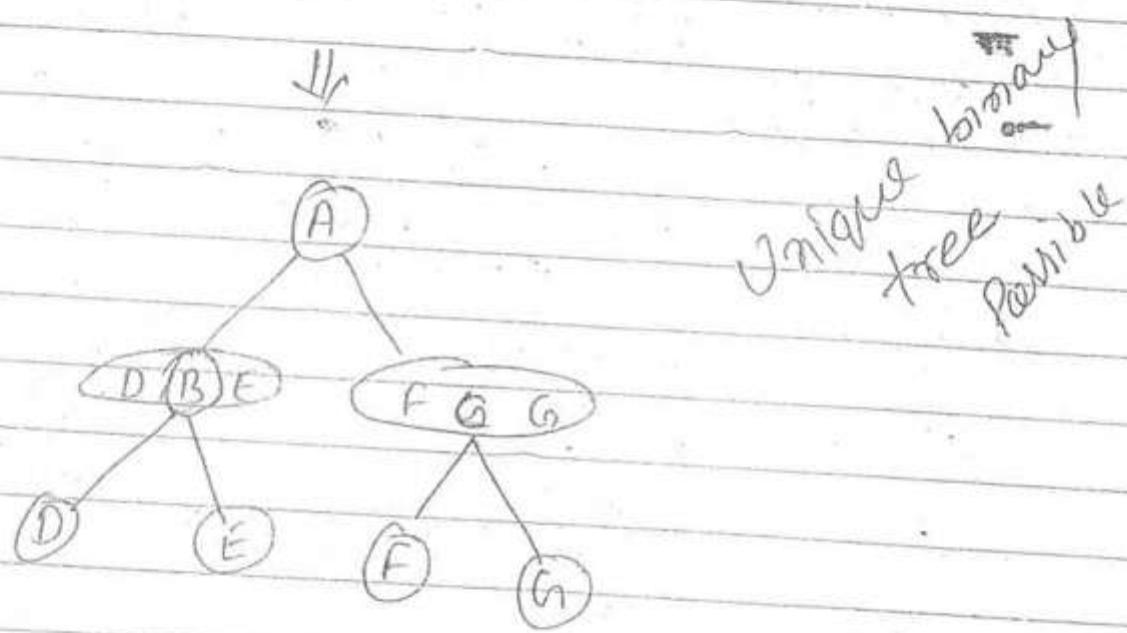
PostOrder \Rightarrow

- a) 8, 7, 6, 5, 4, 3, 2, 1
- b) 1, 2, 3, 4, 8, 7, 6, 5
- c) 2, 1, 4, 3, 6, 7, 8, 5
- d) 2, 1, 4, 3, 7, 8, 6, 5



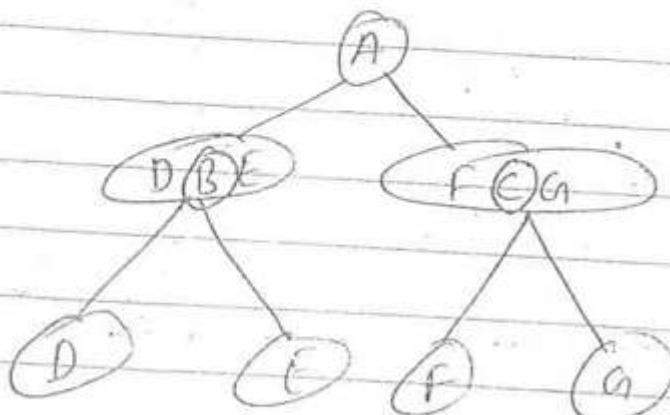
PreOrder - A B D E C F G

InOrder - D, B, E A F C G.

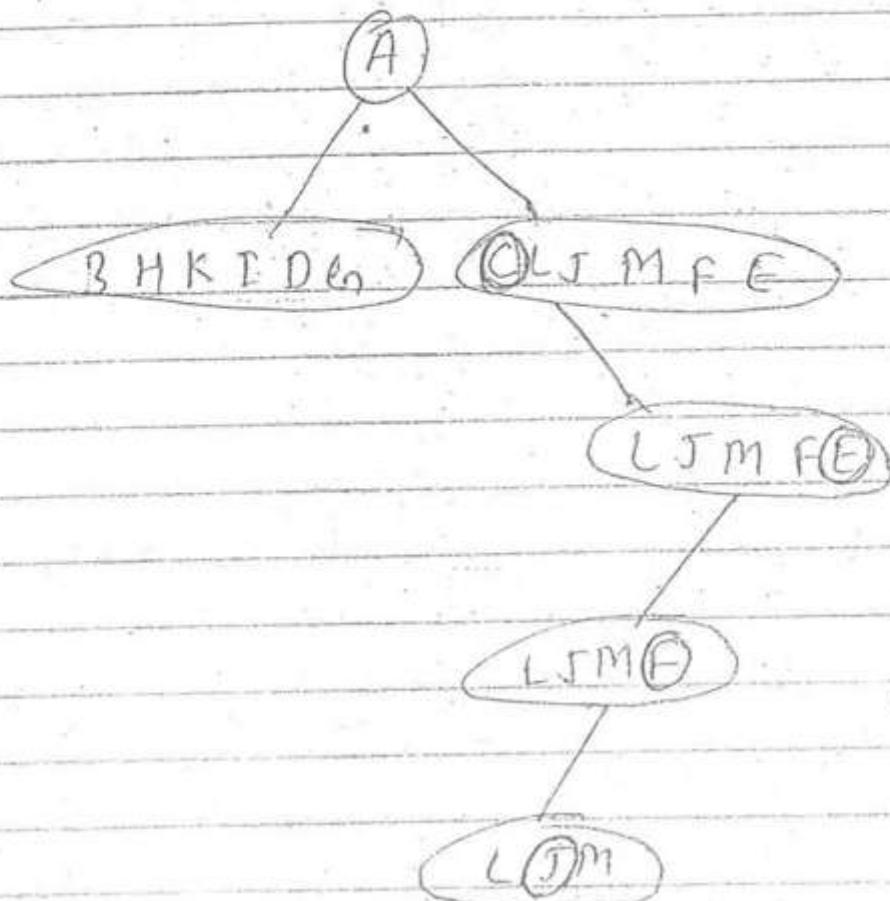


PostOrder - D E B F G C A

InOrder - D B E A F C G.

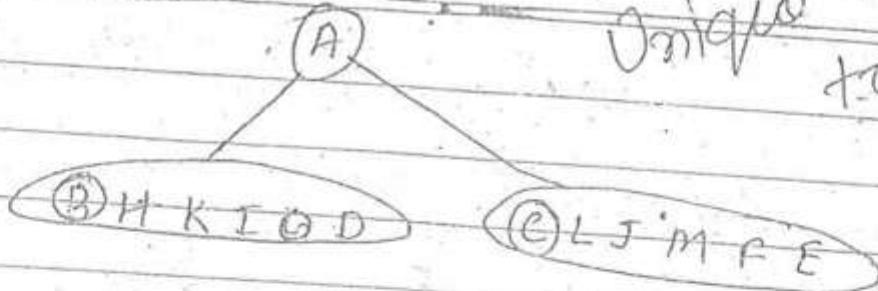


PostOrder - InOrder

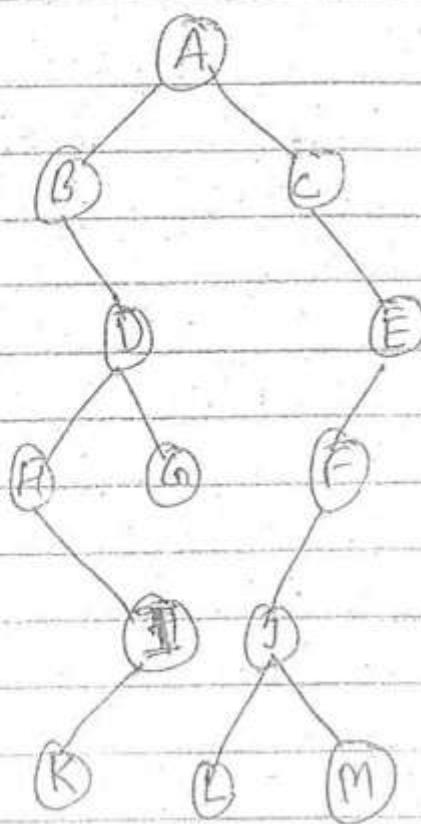


Unique binary
tree possible

PreOrder - Inorder



Only binary tree possible



PreOrder - (R, LST, RST) \Rightarrow A, B, D, H, I, K, G, C, F, J, L, M

PostOrder - (LST, RST, Post) \Rightarrow K, I, H, G, D, B, L, M, J, F, E, C, A

InOrder - B, H, K, I, D, G, A, C, L, J, M, F, E

Postorder

Postorder (Root)

```

{
    if (root != null) {
        Postorder (Root → left)
        Postorder (Root → right)
        Printt (Root → data)
    }
}

```

~~- O(n)~~

InOrder

InOrder (root)

```

{
    if (root != null) {
        InOrder (Root → left),
        Printt (Root → data),
        InOrder (Root → right)
    }
}

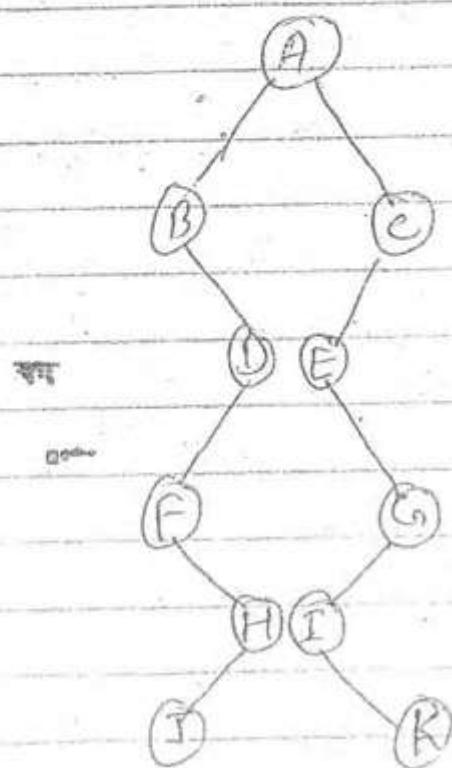
```

~~- O(n)~~

29/01

Monday

- Q. Find Preorder, Inorder and Postorder for the following binary tree.



Preorder - A, B, D, F, H, J, C, E, G, I, K.

Postorder - J, H, F, D, B, K, I, G, E, C, A.

Inorder - B, F, J, H, D, A, E, G, I, K, L, C.

PreOrder

PreOrder(Root) $\Theta(n)$

```

if (root != null)
{
    {
        cout << root->data
        Preorder (root->left)
        Preorder (root->right)
    }
}
  
```

① int u
② u=10 int u=10

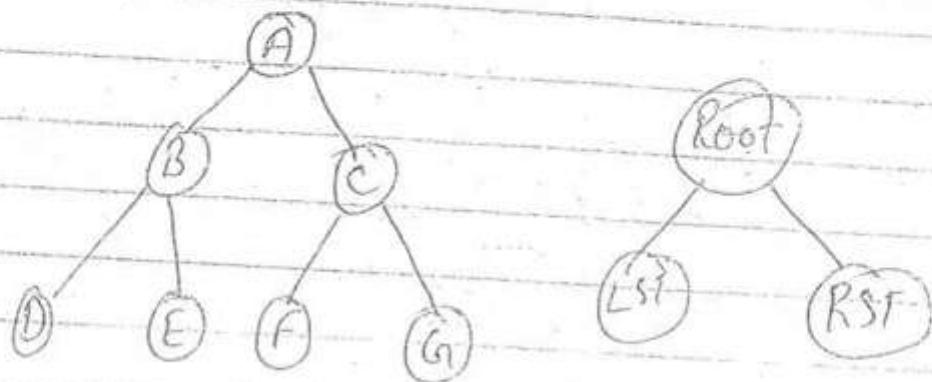


u	10
&u	1000
reference operator	
x(&u)	10
dereference operator	

Tree Traversal (Unique)

- ① InOrder - (LST , Root , Right Sub Tree (RST))
- ② PreOrder (Root , LST , RST)
- ③ PostOrder (LST , RST , Root)

Eg

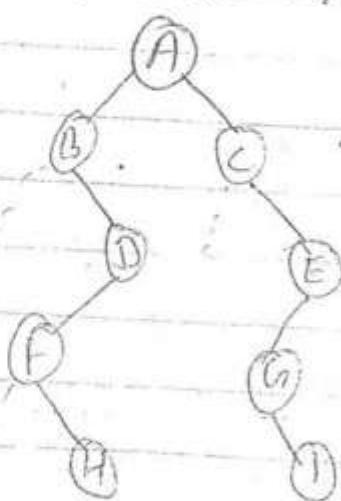


Preorder - A, B, D, E, C, F, G

Postorder - D, E, B, F, G, C, A

InOrder - D, B, E, A, F, C, G

Eg Find Preorder, Postorder, and Inorder in the following tree:



Preorder - A, B, D, E, H, C, E
G, I

Postorder - H, F, D, B, A, E, C, A

InOrder - B, D, H, F, A, G, C, E
I, G, I, E

every strongly connected graph is
also weakly connected.
but every weakly connected graph
not be strongly connected.

सम

गोप

graph b/w

Directed

undirected.

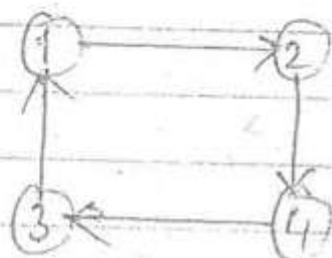
Connected
disconnected.

Strongly

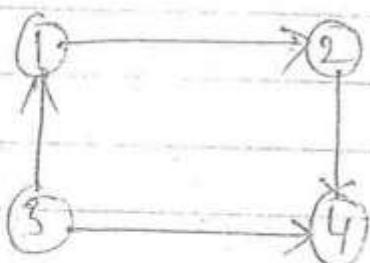
Weakly

Connected

Connected



In the given directed graph b/w
every two vertices there is a path.
strongly connected.



not strongly connected

In the given directed graph after removing
edge direction if then graph is called
weakly connected

Biconnected Component

A graph is said to be Biconnected component if it is contains zero articulation point.

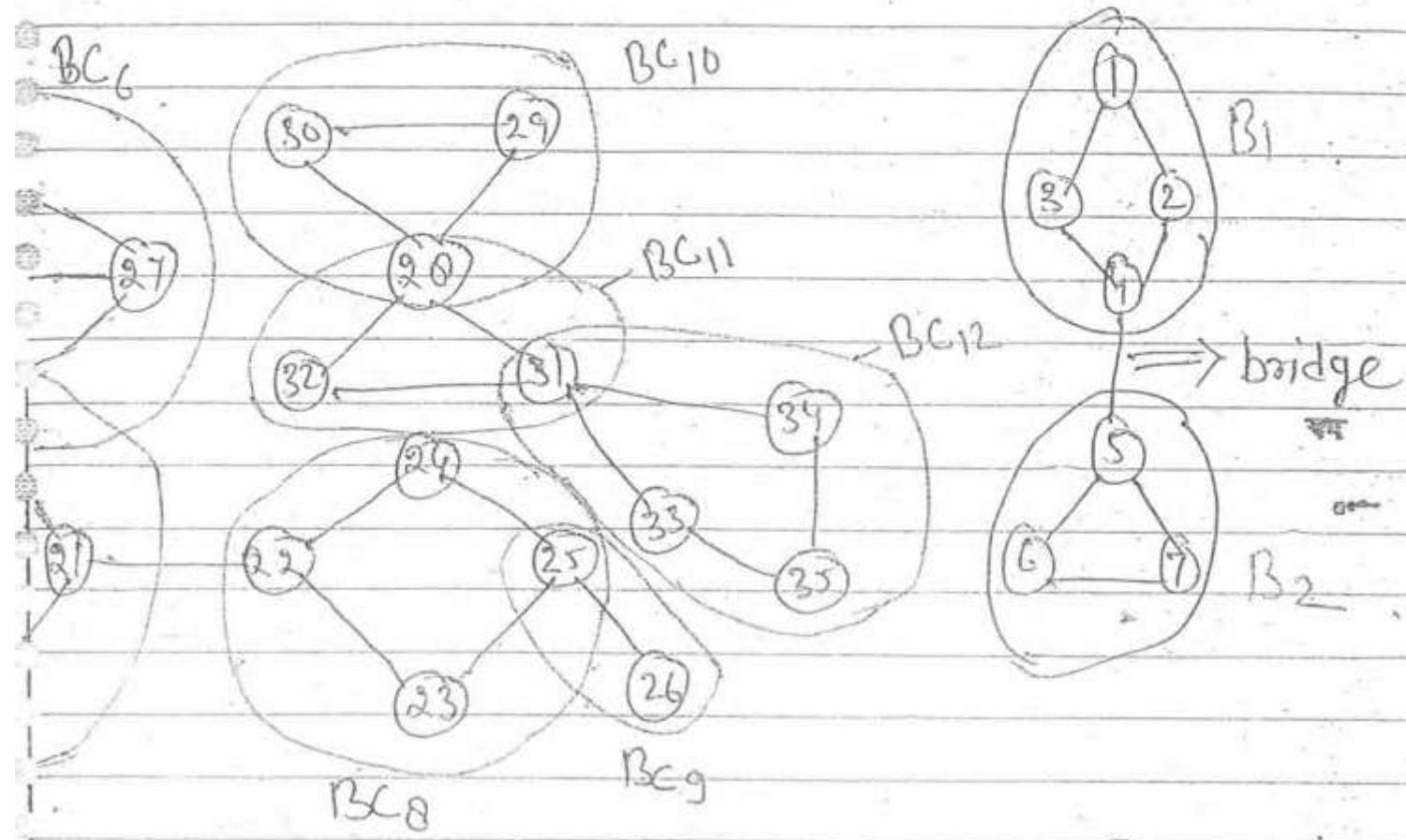
Note -

A max^m connected Subgraph which contains zero articulation point is called biconnected component.

Using DFT we can find out no. of biconnected component in given graph.

Using DFT we can find out given graph cycle or not.

Using DFT we can find out given graph is strongly connected or not.

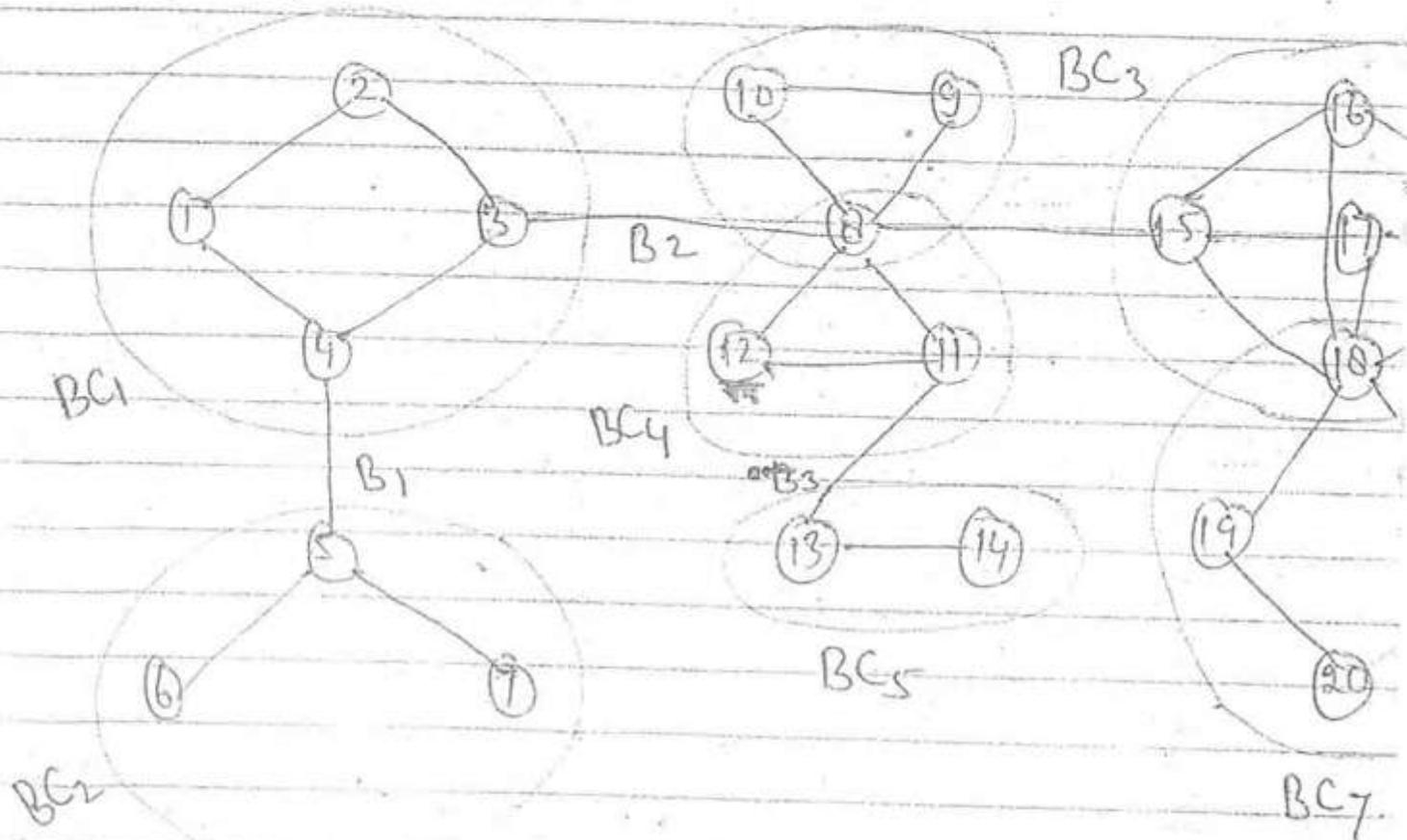


Using DFT algo we can find out bridge also.

Bridge \Rightarrow Any connected graph by deleting an edge if the graph is disconnected then the edge is called bridge.

Using DFT algo we can find out bridge also.

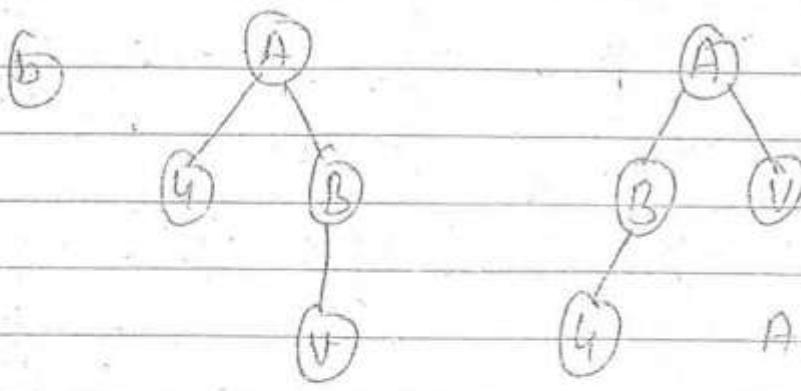
Find Out the no of articulation point of following graph.



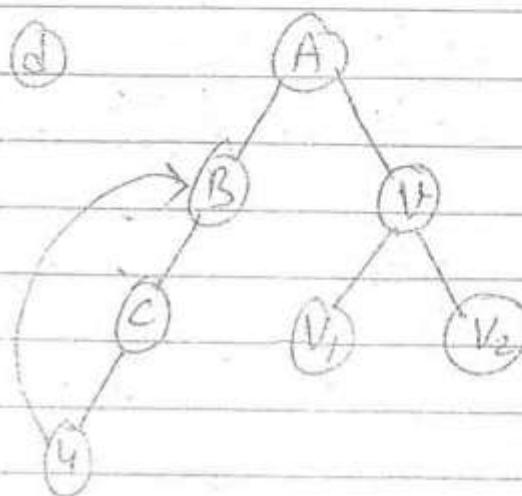
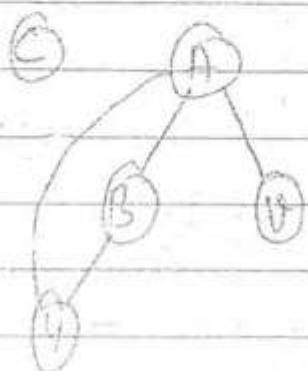
articulation point $\Rightarrow 14$
bridge $\Rightarrow 8$.

Note ① In a particular computer network the number of routers, which are behaving like a articulation point.

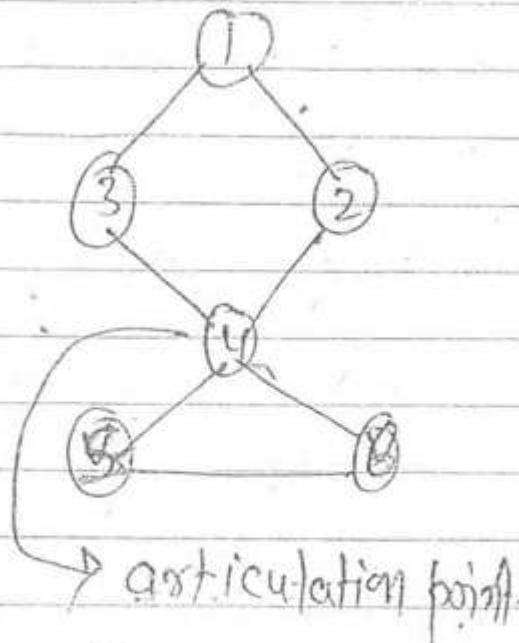
② Using DFT we can find all possible articulation point in given graph.



A B C D E



Articulation Point



In a connected graph a vertex is set to be articulation point iff if final graph is disconnected after removing that vertex together with its incident edge.

→ articulation point

Soln

(i) A, B, E, G, F, D, C, H

A, B, F, D, C, H, E, G

(ii) A, B, E, G, F, C, H, D

(iii) A, D, F, C, H, B, E, G.

Let G be undirected graph.

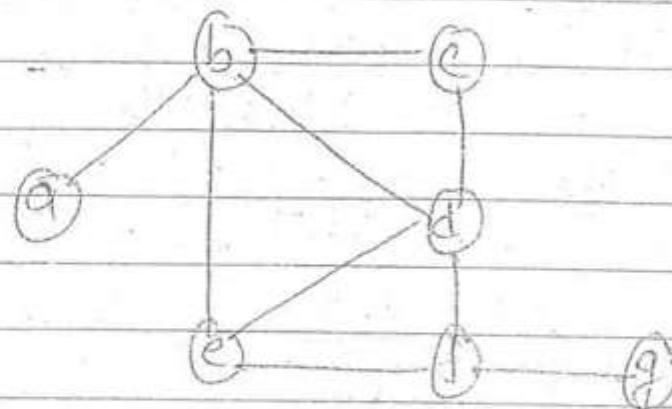
Consider a DFT of g and let

T be the resulting DFT search tree. Let u be a vertex in G and v be the first new vertex visited after visiting u in the traversal.

Which one of the following is true?

- a) $\{u, v\}$ must be an edge in G .
- b) If $\{u, v\}$ is not an edge in G , then $\{u, v\}$ must have same parent.
- c) If (u, v) is not an edge in g then v is leaf node in G .
- d) If $\{u, v\}$ is not an edge in G then u is leaf in T .

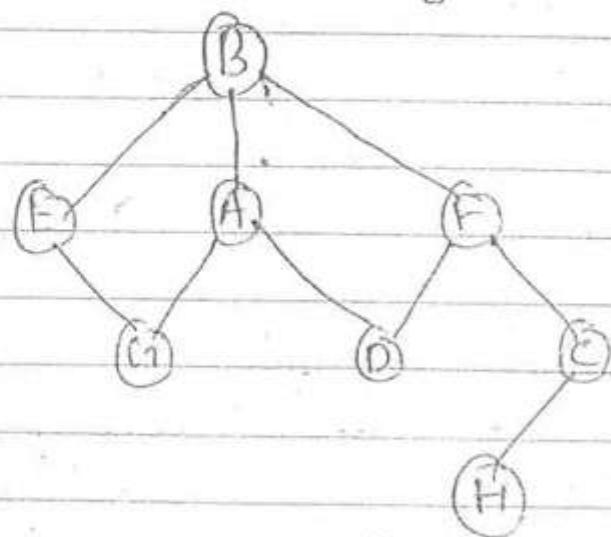
Consider the following graph.



a possible DFT for above graph is.

- a) b, a, e, c, d, f, g.
- b) d, b, a, e, f, c, g.
- c) b, c, d, e, f, a, g.
- d) ~~d, e, b, a, c, f, g.~~

Consider the following graph.



find 3-different
DFT starting
from A.

$V_1, V_5, V_7, V_0, V_6, V_2, V_3, V_4$

find DFT by considering always ascending order starting from V_5

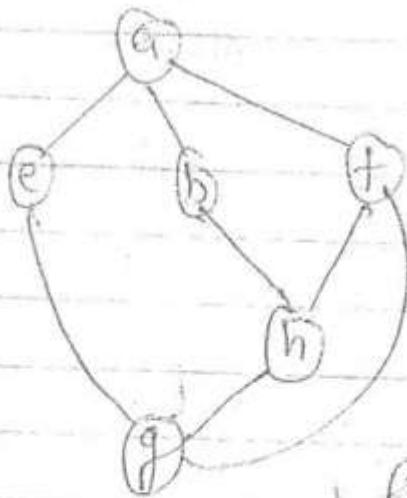
$V_3, V_1, V_2, V_6, V_0, V_7, V_4, V_5$

find 3 DFT starting from V_0 .

$V_0, V_6, V_2, V_1, V_3, V_4, V_7, V_5$

$V_0, V_7, V_5, V_4, V_1, V_4, V_3, V_6, V_2$

$V_0, V_6, V_3, V_1, V_5, V_7, V_4, V_2$



Among the following sequence which are DFT of the above Graph.

(I) a, b, e, g, h, f

(II) a, b, f, e, h, g

(III) a, b, f, h, g, e

(IV) a, f, g, h, b, e

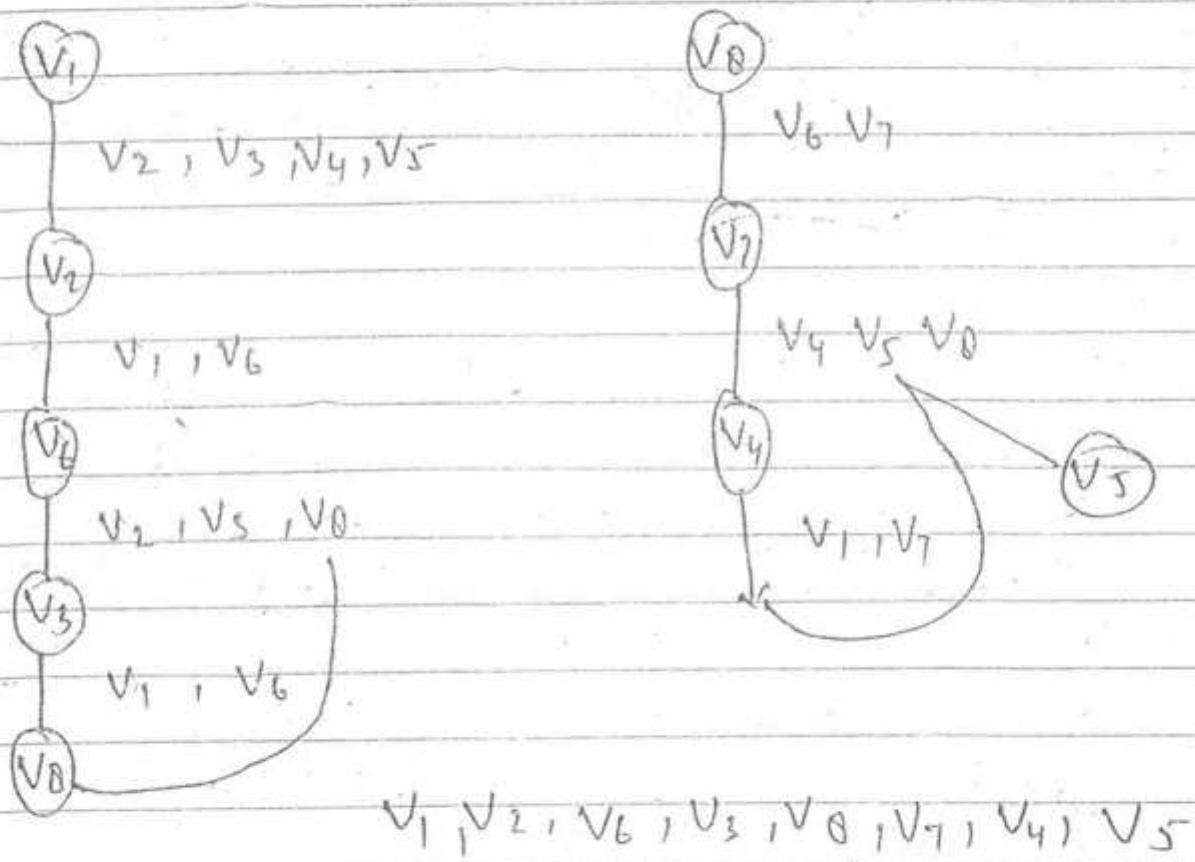
vertex \Rightarrow DFT

for every vertex called DFT

$$\Rightarrow O(V^2 + V)$$

$$O(V+E)$$

- (i) The ds we are using the DFT is stack
- (ii) The time complexity of DFT $O(V+E)$



DFT (v_2)

- ①
- ② $w = v_1, v_6$
- ③

DFT (v_5)

- ①
- ② $w = v_3, v_8$
- ③

DFP (v_3)

- ①
- ② $w = v_1, v_6$

DFT (v_0)

- ①
- ② $w = v_6, v_7$
- ③
- ④ DFT (v_7)

- ①
- ② $w = v_4, v_5, v_0$
- ③

DFT (v_4)

- ①
- ② $w = v_1, v_7$

DFT (v_5)

- ①
- ② $w = v_1, v_7$

DFT

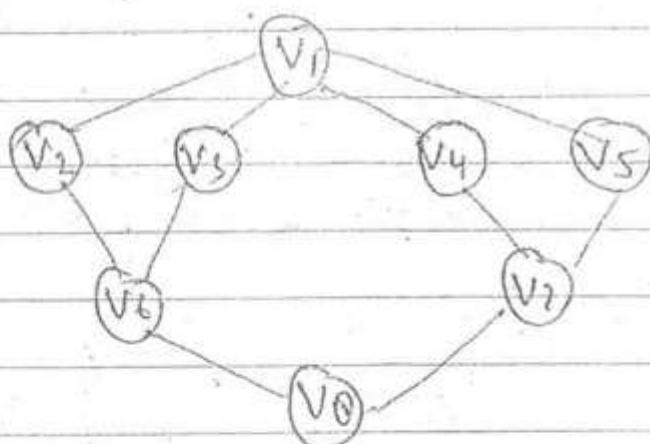
DFT(v)

① visited(v) = 1

② for all w adjacent to v

 ③ if (w is not visited)

 ④ DFT(w);



V₁ V₂ V₆ V₅ V₈ V₇ V₉ V₅

DFT(v_1)

①

② $w = v_2 v_3 v_4 v_5$

③

④ DFT(v_2)

Collision Resolution Technique

Chaining

open addressing (Rehashing)

Linear
Probing

Quadratic
Probing

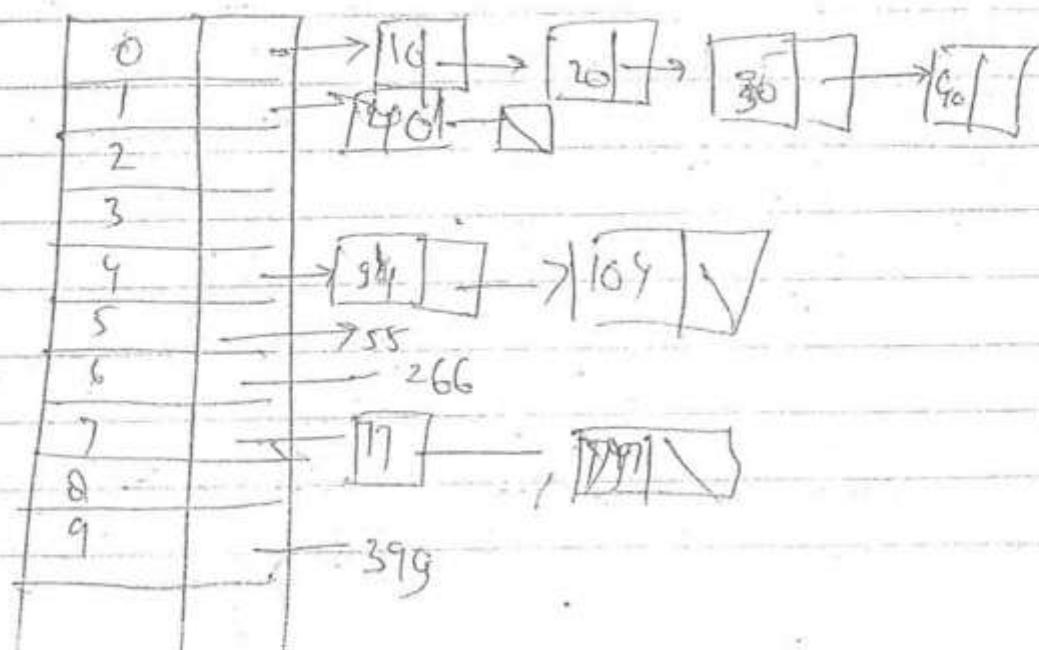
Double
Hashing

Collision - Resolution by Chaining

Key: 10 77 94 899 461 565 104 266 747

Hash function, Key mod 10

Collision Resolution Technique = Chaining



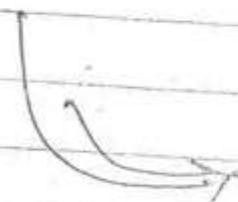
Drawback

- D We are taking extra space extra to store
 - ④ When collision occurs, instead using a variable space. Even table have empty space.
- D In the worst case is be having linear searching.
- ③ Unlimited collisions will be handled by this method.
- D Does not required pre-knowledge of how many are coming into hash.
- 2 If the element are stored in hash table using Chaining, we can delete those element easily whenever not required.

Open Addressing

- No storage require for linked list.

If I hash to particular slot in hash table, it already filled. I will hash again with different hash function.



	0	
	1	
$h(369, 0)$	2	365
	3	
$h(369, 1)$	4	560
$h(369, 2)$	5	
	6	
	7	457
	8	
$h(369, m-1)$	9	

Linear Packing

Keys: 25, 30, 77, 55, 62, 78, 99, 52, 101, 41.

Hash funⁿ = Key modulo 10

Collision Resolution technique = Linear Packing

$$h'(K) = K \bmod m$$

$$h(K, i) = (h'(K) + i) \bmod m$$

$$\forall i \in \{0, 1, 2, 3, \dots, m-1\}$$

0	30
1	10)
2	62
3	52
4	41
5	25
6	55
7	77
8	70
9	99

$$\checkmark h'(25) = 25 \bmod 10 \\ = 5$$

$$h(25,0) \doteq (h'(K) + 0) \bmod 10 \\ = 5$$

$$\checkmark h'(30) = 30 \bmod 10 \\ = 0$$

$$h(30,0) = (0+0) \bmod 10 \\ = 0$$

$$\checkmark h'(77) = 77 \bmod 10 \\ = 7$$

$$h(77,0) = (7+0) \bmod 10 \\ = 7$$

$$\checkmark h'(55) = 5$$

$$h(55,0) = (5+0) \bmod 10 \\ = 5$$

$$h(55,1) = (5+1) \bmod 10 \\ = 6$$

$$\checkmark h'(62) = 2$$

$$h(62,0) \doteq 2$$

$$\checkmark h'(52) = 2$$

$$h(52,0) = (2+0) \bmod 10 \\ = 2$$

$$h(52,1) = (2+1) \bmod 10 \\ = 3$$

$$h'(41) = 1$$

$$h(41,0) = (1+0) \bmod 10 \\ = 1$$

$$h(41,1) = (1+1) \bmod 10 \\ = 2$$

$$h(41,2) = (1+2) = 3$$

$$h(41,3) = (1+3) \bmod 10 \\ = 4$$

Q. Let $h(K) = K \bmod 7$

Keys : 29, 36, 16, 30

Collision Resolution Technique :- Linear probing.

While calculating the no of collision inserting the given key :

0	
1	29
2	36
3	16
4	30
5	
6	

$$h'(K) = K \bmod 7$$

$$h'(29) = 29 \bmod 7 = 1$$

$$h(29, 0) = (1 + 0) \bmod 7 \\ = 1$$

Collision = 4

Q. Keys : 12, 10, 13, 2, 3, 23, 5 and 15 are

inserted into an initially empty, hash-table length 10, using linear probing.

$$\text{hash fun} = K \bmod 10$$

0	
1	10
2	12
3	13
4	2
5	3
6	23
7	5
8	15
9	

= Hash Table size $m = 11$
 hash funⁿ is given as follows.

$h(\text{Key})$

{

int u ;

$$u = (\text{Key} + 5) * (\text{Key} + 5);$$

$$u = u / 16;$$

$$u = u + \text{Key};$$

$$u = u \% 11;$$

return u ;

}

Keys = 48, 23, 1, 0, 15, 31, 4, 7, 11, 3

Show the resultant hash table.

0	48	\checkmark Key = 48 $u = 48 \times 48;$
1	0	$u = (48 \times 48) / 16 = 48 \times 3$
2	31	$u = (48 \times 3) + 48; = 187$
3	1	$u = 0$
4		return (0)
5	7	\checkmark Key = 23 $u = 23 \times 23$
6	23	$u = 49$
7	15	$u = 49 + 23 = 72$
8	11	$u = 6$
9	4	
10	3	\checkmark Key = 1 $u = 2 + 1 = 3$ $u = 36$ $u = 36 / 16 \equiv 2$
		$u = 3 \% 11 = 3$ return (3)

A Hash table of length 10 was open addressing with hash fun $h(K) = K \bmod 10$
C.R.F = linear probing

After inserting 6 values into an empty hash table,
the table is shown below.

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

Q) When one of the following choices given a possible order in which the key values could have been inserted in the table.

- a) 46, 42, 34, 52, 23, 33
- b) 34, 42, 23, 52, 33, 46
- c) 46, 34, 42, 23, 52, 33
- d) 42, 46, 33, 23, 34, 52

i) How many different insertion sequence of the key values using the same hash function and linear probing result in the hash table shown above.

- a) 10 b) 20 c) ~~30~~ d) 40

~~Self~~ $\left[\begin{array}{l} 46, 34, 42, 25 \\ \hline 52, 33 \end{array} \right]$

$$4! = 24$$

$\left[\begin{array}{l} 34 \\ 42 \\ 25 \\ \hline 52 \end{array} \right] \quad \left[\begin{array}{l} 46 \\ 33 \end{array} \right]$

$$\frac{4!}{6} = 6$$

$$\text{total} = 24 + 6 = 30$$

~~Problems~~

- ① In linear probing we have probing one slots after another..
- ② If an empty slots then long runs of occupied slots increases average searching time. This problem is known as primary clustering.
- * In the worst case linear probing will take $O(n)$ time. to search for particular element.
- * In linear probing deleting is difficult i.e. delete one element it creates problem to others.

Quadratic Probing

$$h'(k) = k \bmod m$$

$$h(k, i) = (h'(k) + c_1 \cdot i + c_2 \cdot i^2) \bmod m$$

$\forall i \in \{0, 1, 2, 3, \dots, m-1\}$

0	10
1	35
2	20
3	
4	
5	15
6	30
7	25
8	30
9	49

$$h \text{ fun} = \text{Key mod } 10$$

$$\text{Keys} = 10, 15, 38, 49, 20, 30, 25$$

$$C \cdot R \cdot T = QP$$

$$c_1 = 1, c_2 = 1$$

$$0 = 10 \bmod 10$$

$$h(10, 0) = 0$$

$$\checkmark 5 = 15 \bmod 10$$

h

$$\checkmark h'(30) = 30 \bmod 10 \\ = 0$$

$$h(30, 0) = (h'(30) + c_1 \cdot 0 + c_2 \cdot 0) \bmod 10 \\ = (0 + 0 + 0) \bmod 10 \\ = 0$$

$$\checkmark \frac{h'(20)}{0} = 20 \bmod 10$$

$$h(20, 0) = (h'(20) + 0 + 0^2) \bmod 10 \\ = 0 + 0 + 0 \\ = 0$$

$$h(20, 1) = (h'(20) + 1 + 1 + 1 \cdot 1^2) \bmod 10 \\ = 0 + 1 + 1 \\ = 2$$

$$h'(30)$$

$$0 \equiv 30 \pmod{10}$$

$$h(30,0) = (h'(30) + 0 + 0^2) \pmod{10}$$
$$= 0$$

$$h(30,1) = (h'(30) + 1 + 1^2) \pmod{10}$$
$$= 0 + 1 + 1$$

$$h(30,2) = (h'(30) + 1 \cdot 2 + 1 \cdot 2^2) \pmod{10}$$
$$= (0 + 2 + 4) \pmod{10}$$

$$= \underline{\underline{6}}$$

$$\checkmark h'(40) = 40 \pmod{10}$$

$$= 0$$

$$h(40,0) = (0 + 0 + 0^2) \pmod{10}$$
$$= 0$$

$$h(40,1) = (0 + 1 + 1^2) \pmod{10}$$
$$= 2$$

$$h(40,2) = (0 + 3 + 2^2) \pmod{10}$$
$$= 6$$

$$h(40, 3) = (0 + 3 + 3^2) \bmod 10$$
$$= \underline{\underline{3}} \quad \underline{\underline{2}}$$

$$h(40, 4) = (0 + 9 + 4^2) \bmod 10$$
$$= \underline{\underline{0}}$$

$$h(40, 5) = (0 + 9 + 5^2) \bmod 10$$
$$= \underline{\underline{0}}$$

$$h(40, 6) = (0 + 9 + 6^2) \bmod 10$$
$$= \underline{\underline{2}}$$

$$h(40, 9) = (0 + 9 + 9^2) \bmod 10$$
$$= \underline{\underline{0}}$$

Note

- ① Comparing with linear probing quadratic probing is better.
- ② In the worst case quadratic probing will be $O(n)$. searching time.
- ③ Quadratic probing is depending upon c_1 & c_2 and hash table size.

Key 35

Key 45

$$h(35) = 35 \bmod 10 \\ = 5$$

$$h(35,0) = 5 + 0 + 0^2 \\ = 5$$

$$h(35,1) = 5 + 1 + 1^2 \\ = 7$$

$$h(35,2) = 5 + 2 + 2^2 \\ = 1$$

$$h(45) = 45 \bmod 10 \\ = 5$$

$$h(45,0) = 5 + 0 + 0^2 \\ = 5$$

$$h(45,1) = 5 + 1 + 1^2 \\ = 7$$

$$h(45,2) = 5 + 2 + 2^2 \\ = 1$$

$$h(45,3) = 5 + 3 + 3^2 \\ = 7$$

If two keys having same starting hash address, they both will follow some path. This problem is known as Secondary clustering bcoz of secondary clustering average seeking little bit increases.

Keys

= 10, 22, 31, 4, 15, 18, 17, 88, 59

$$M = 11$$

$$c_1 = 1, c_2 = 3.$$

$$h'(k) = k \quad ; \quad h(k, i) = (h(k) + c_1 i + c_2 i^2) \bmod M$$

$$C.R.T = Q.P$$

0	22
1	59
2	
3	88
4	4
5	17
6	11
7	10
8	15
9	31
10	10

$$\checkmark h(10, 0) = (10 + 0 + 0) \bmod 11 \\ = 10$$

$$\checkmark h(22, 0) = (22 + 0 + 0) \bmod 11 \\ = 0$$

$$\checkmark h(31, 0) = (31 + 0 + 0) \bmod 11 \\ = 9$$

$$\checkmark h(15, 0) = (15 + 0 + 0) \bmod 11 \\ = 4$$

$$h(15, 1) = (15 + 1 + 3 \cdot 1^2) \bmod 11 \\ = 9$$

$$\checkmark h(10, 0) = (15 + 0 + 0) \bmod 11 \\ = 7$$

$$\checkmark h(17,0) = (17+0+0) \bmod 11$$
$$= 6$$

$$\checkmark h(88+0) = (88+0+0) \bmod 11$$
$$= 0$$

$$h(88,1) = (88+1+3) \bmod 11$$
$$= 4$$

$$h(88,2) = (88+2+3 \cdot 2^2)$$
$$= (102) \bmod 11 = 5$$

$$\checkmark h(59,0) = (59+0+0) \bmod 11$$
$$= 4$$

$$h(59,1) = (59+1+3) \bmod 11$$
$$= 63 \bmod 11 = 8$$

$$h(59,2) = (59+2+3 \cdot 2^2) \bmod 11$$
$$= 7$$

$$h(59,3) = (59+3+3 \cdot 3^2) \bmod 11$$
$$= 1$$
$$\checkmark$$

L.P > Q.P

more searching

less searching

Double Hashing (Brook Hashing) Pedicnic

L.P



Primary Clustering



Quadratic Probing



S.C



Double Hashing

$$h(LK, i) = (h_1(K) + i \cdot h_2(K)) \bmod m$$

At $i \in \{0, 1, 2, 3, \dots, m-1\}$

$$h_1(K) = K \bmod m$$

$$h_2(K) = 1 + (K \bmod (m-1))$$

eg

$$m = 10$$

Keys = 10, 25, 14, 16, 20, 30, 35, 45, 65.

0	10
1	65
2	35
3	20
4	14
5	25
6	16
7	45
8	30
9	

$$h_1(10, 0) = 10 \bmod 10$$

$$= 0$$

$$\checkmark h_1(10, 0) = (0 + 0 \cdot h_2(k)) \bmod 10$$

$$= 0$$

$$h_1(25) = 25 \bmod 10$$

$$= 5$$

$$h_1(25, 0) = 5 + 0 \cdot h_2(k) \bmod 10$$

$$= 5$$

$$h_1(14) = 14 \bmod 10$$

$$= 4$$

$$h_1(14, 0) = (4 + 0 \cdot h_2(k)) \bmod 10$$

$$= 4$$

$$h_1(16) = 16 \bmod 10$$

$$= 6$$

$$h_1(16, 0) = (16 + 0 \cdot h_2(k)) \bmod 10$$

$$= 6$$

$$h_1(20) = 20 \bmod 10$$

$$= 0$$

$$h_1(20,0) = (0 + 0 \cdot h_2(5)) \bmod 10$$

$$= 0$$

$$h_2(20) = 1 + (20 \bmod 9)$$

$$= 1 + 2$$

$$= 3$$

$$h(20,1) = (0 + 1 \cdot 3) \bmod 10$$

$$= 3$$

$$h_1(30) = (0 + 0 \cdot h_2(30)) \bmod 10$$

$$= 0$$

$$h_2(30) = 1 + 3 \bmod 9$$

$$= 1 + 3 = 4$$

$$h(30,1) = (0 + 1 \cdot 4) \bmod 10$$

$$= 4$$

$$h(30,2) = (0 + 2 \cdot 4) \bmod 10$$

$$= 0$$

$h_1(35) = 35 \bmod 10$
= 5

$$h_2(35)_2 = 1 + 35 \bmod 9$$
$$= 1 + 8 = 9.$$

$$h(35,0) = (5 + 0 \cdot h_2(5)) \bmod 10$$
$$= 5$$

$$h(35,1) = (5 + 1 \cdot 9) \bmod 10$$
$$= 4$$

$$h(35,2) = (5 + 2 \cdot 9) \bmod 10$$
$$= 3$$

$$h(35,3) = (5 + 3 \cdot 9) \bmod 10$$
$$= 2$$

$$h_1(45) = 45 \bmod 10$$
$$= 5$$

$$h_2(45) = 1 + 45 \bmod 9$$
$$= 1$$

$$h(45,0) = (5 + 0 \cdot 1) \bmod 10$$
$$= 5$$

$$h(45,1) = (5 + 1 \cdot 1)$$
$$= 6$$

$$h(45,2) = (5 + 2 \cdot 1)$$
$$= 7$$

- * In linear probing whenever collision is occurred we will always add i by $+1$. It does not matter what key it is bcoz of that reason average searching time increases.
- * In Quadratic probing whenever 2 keys having same starting hash address we will always add some quadratic eqn to both the keys which will leads to secondary cluster.
- * In double hashing Bcoz whenever 2 key having same starting hash address we will add one random no for the first key and another random no for 2nd keys which is generated by the second hash function.
- * In double hashing S.C is completely eliminating bcoz we are adding 2 different random no for both the keys.

27/07/011

eg

$$M=13, h_1(K) = K \bmod M, h_2(K) = 1 + K \bmod 11$$

29, 69, 72, 98, 10, 50

$$h(K, i) = (h_1(K) + i h_2(K))$$

0	
1	
2	
3	29
4	69
5	98
6	
7	72
8	
9	
10	10
11	50
12	

$$h_1(K) = 29 \bmod 13$$

$$h_1(29) = 3$$

$$h(29, 0) = (3 + 0 \cdot h_2(K)) \\ = 3$$

$$h(69, 0) = (4 + 0 \cdot h_2(K)) \\ = 4$$

$$h(72, 0) = (7 + 0 \cdot h_2(K)) \\ = 7$$

$$h_1(98) = 98 \bmod 13 \\ = 7$$

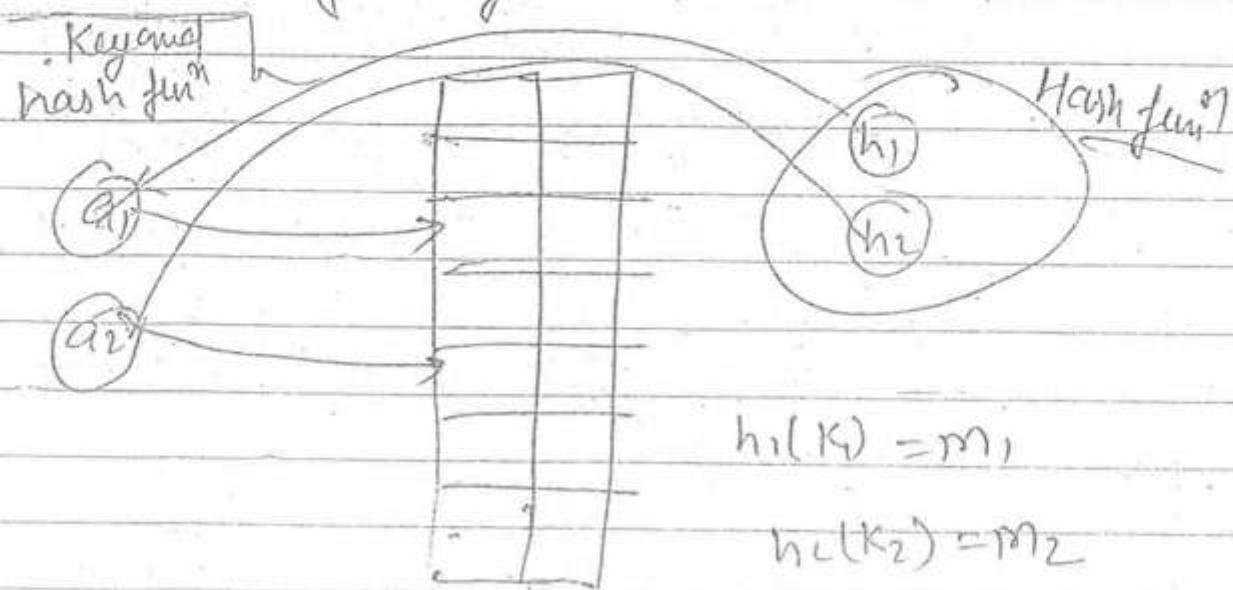
$$h_2(98) = 1 + 98 \bmod 11 \\ = 1 + 10 \\ = 11$$

$$h(98, 0) = (7 + 0 \cdot h_2(K)) \quad \text{mod} \\ = 7$$

$$h(98, 1) = 7 + 1 \cdot 11 \\ = 10 \bmod 13 \\ = 5$$

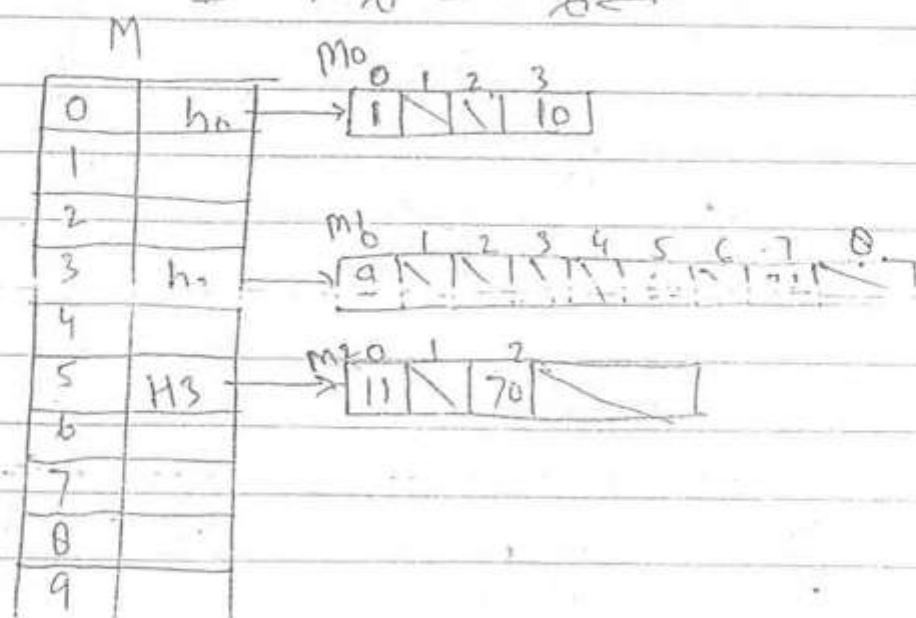
Badness of Hashing

For any choice of hash funⁿ there exist bad set of keys that all hash to same slot.



The solⁿ to this problem picks hash funⁿ randomly, which is known as Universal Hashing.

Perfect Hashing



$$h(1) = 0 \quad h_0(1) = 0$$

$$h(10) = 0 \quad h_0(10) = 3$$

$$h(9) = 5 \quad h_2(9) = 0$$

$$h(60) = 3 \quad h_2(60) = 5$$

$$h(72) = 3 \quad h_2(72) = 7$$

$$h(11) = 5 \quad h_3(11) = 0$$

$$h(70) = 5 \quad h_3(70) = 2$$

Note—

~~Data~~ instead of making linked list of key having to slot
if we can use small secondary hash function
with small table size.

* Total no of collision ~~in~~ in second level of hashing
is $\max^n(y_2)$ no collision.

* The worst case time complexity to search an
element is $O(1)$.

P, NP, NPC, NPH

Problems

Solvable

Unsolvable

Halting Problem.

Solvable

x.L
(accepted)
(rejected)

x.e.L
(accepted)
(rejected)
infinite

Decision
Problems

(Y or N)

optimization
Problem

Travelling
sales person
Problem.

P NP

P-Class Problem : - A problem A is in P-class
iff there exist a polynomial time
algo to solve the problem (less time complexity)

Set of P-Class : - A problem is in P class there
exist deterministic time polynomial
algo solve it .

- (1) Linear Search $\Rightarrow \Theta(n)$
- (2) Bubble Sort $\Rightarrow \Theta(n^2)$
- (3) Quick Sort $\Rightarrow \Theta(n \log n)$
- (4) Euler Graph $\Rightarrow \Theta(V^2)$
- (5) Prim's algo $\Rightarrow (E+V) \log V$
- (6) Merge Sort $\Rightarrow \Theta(n \log n)$
- (7) LCS $\Rightarrow \Theta(n^2)$
- (8) All pair shortest Path $\Rightarrow \Theta(n^3)$
- (9) Matrix chain multiplication $\Rightarrow \Theta(n^3)$
- (10) Binary Search \Rightarrow

NP- Class Problem

→ A Problem P is in NP iff there exist non deterministic time polynomial time algo to solve that problem.

Set of NP class :-

ND Sorting (a, n)

```

    {
        for( i = 1 to n ) → with guess.
        {
            find ith smallest element
            {
                } →  $\Theta(n)$ 
        }
    }
  
```

without guess to solve a problem, the problem is called P-class and without guess is called NP class.

Set of NP-Class

$x \rightarrow x$

(1) Travelling Sales Person.

(2) Hamiltonian

(3) 0/1 Knapsack

(4) Sum of Subsets

(5) Linear Search

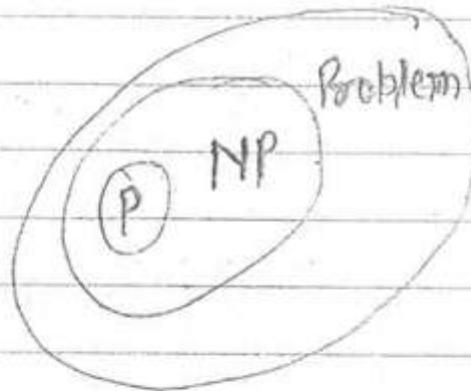
(6) Quick Sort

(7) Merge Sort

(8) Euler Graph.

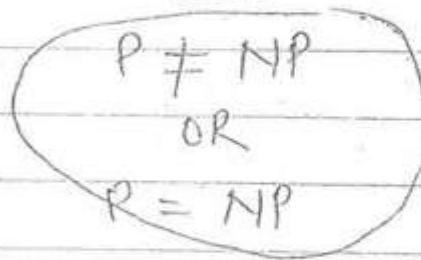
every P class problem is NP class problem.

$$P \subseteq NP$$

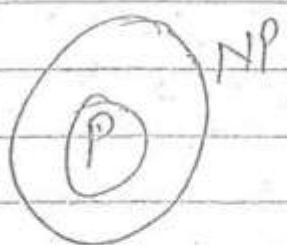


$$P \subseteq NP$$

OR



P & NP



$\gamma - (1\text{-million dollar})$



PCNP

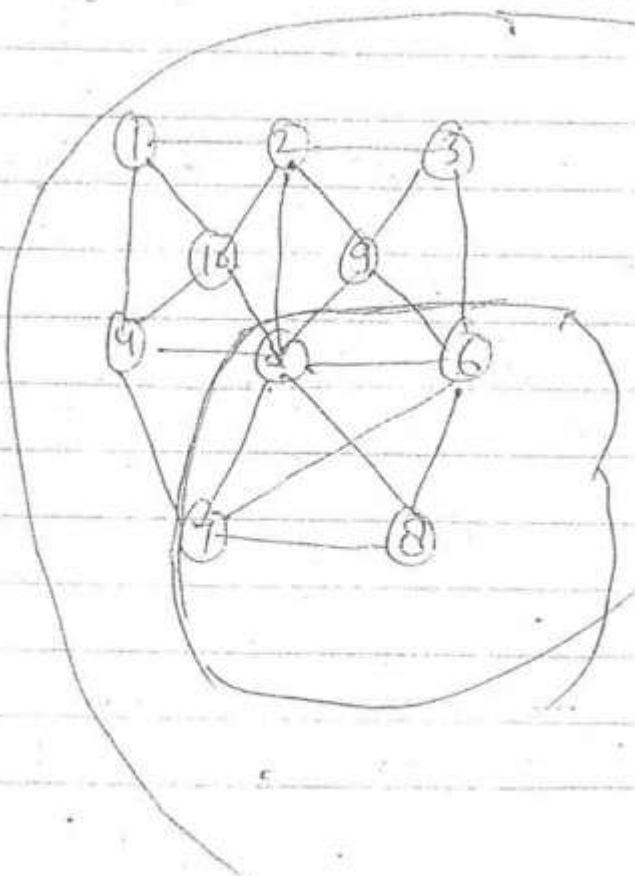
NP

I/P : Graph $G(V, E)$; K

Q : Does G - contain K -Clique

Prover

Verifier



P-Class Problem \Rightarrow Solving entire problem with polynomial time. Is called P-Class Problem.

NP-Class Problem \Rightarrow

Verifying the given correct input in polynomial time is called NP-Class Problem.



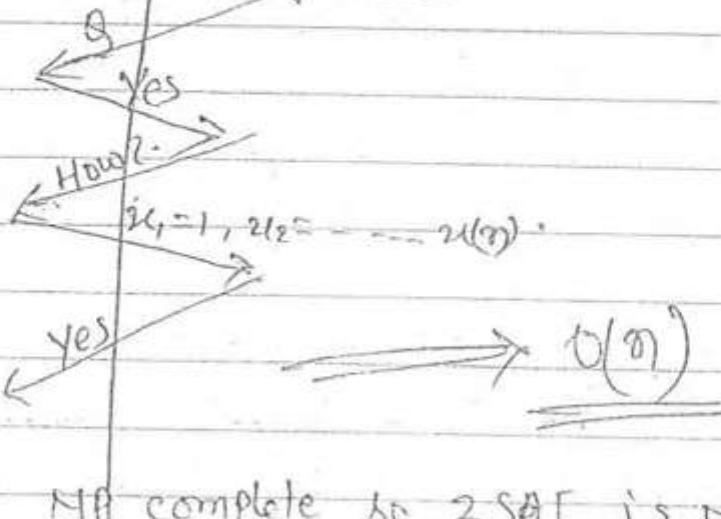
i/p : A boolean expression (CNF) and each term of CNF contain 3-variables.

$$\text{eg. } (u_1 \vee u_2 \vee u_3) \wedge (u_1 \vee u_2 \vee u_3) \wedge (u_1' \vee u_2' \vee u_3') \\ \wedge (u_1 \vee u_{100} \vee u_{50}) \wedge (u_{99}' \vee u_1' \vee u_{100}')$$

Q : Does the given boolean expression contain truth values for all variables u_1, \dots, u_n such that given expression becomes true ?.

~~Idⁿ~~

Prover / Verifier



3-SAT is NP complete so 2-SAT is NP complete

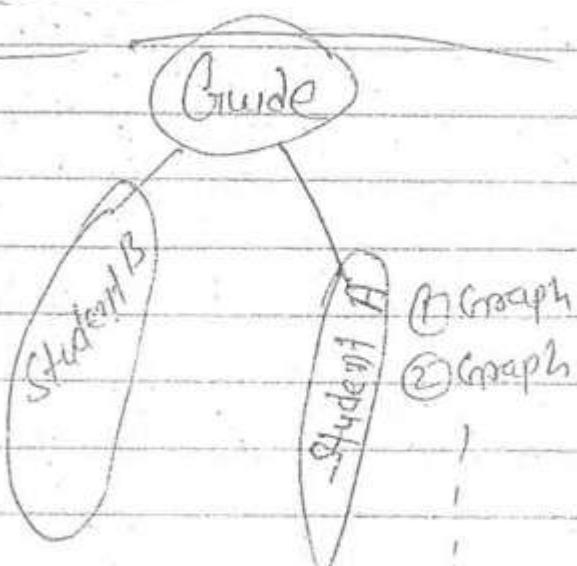
#8

Polynomial time reduction

$$A : bu + c$$

$$B : au^2 + bu + c$$

$$\downarrow \text{[Polynomial]} \quad \text{effort}$$
$$N : bu^2 + bu + c$$



$$\frac{1}{16} \cdot A - NP \subset$$

Let A - known

B - unknown.

If A is polynomially reduced to B

OR

$$A \leq_p B$$

then B is also known problem

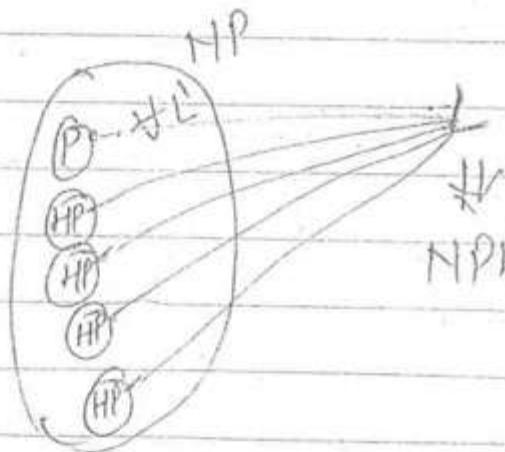
NP-Hard

Let L be the given problem which we want to prove NP-hard.

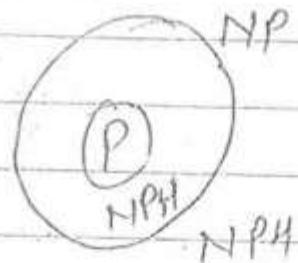
If L' is polynomially reduced to L for all L' belongs to NP then L is NPH

$$L' \leq_p L, \forall L' \in \text{NP} \text{ then}$$

L is NPH



(My Problem L is at least
Hardest as NP-Problem)

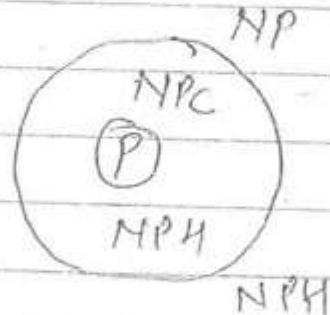


NP-complete

A problem L is NP-complete iff.

① $L \in \text{NPH}$

② $L \in \text{NP}$



Note

① If L is NP-complete and
 L is polynomially reduced to A ie $L \leq_p A$

then A is NPH \xrightarrow{NPC}

② If L is P-class Problem and.

A is polynomially reduced to L ie $A \leq_p L$

then A is also p-class Problem $\xleftarrow{ }$

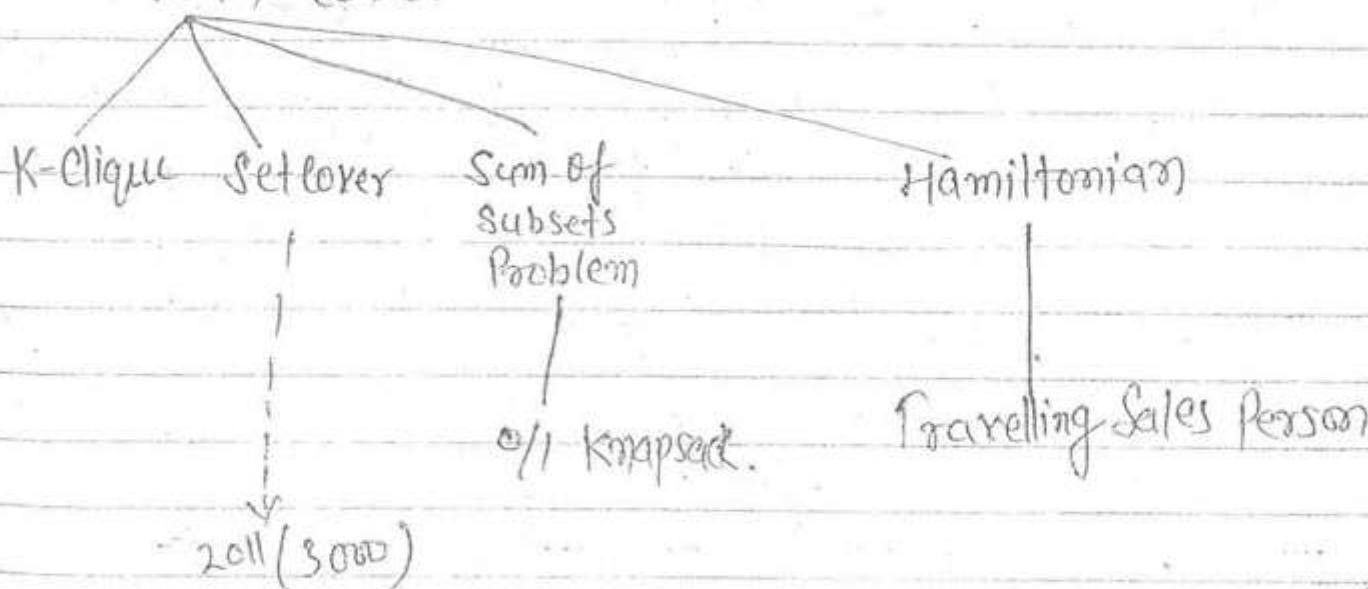
thus if $L \in NP$

$A \leq_p L$ then A is also NP-class. $\xleftarrow{ }$

(i) 3-SAT \Rightarrow NP complete.



Vertex cover



- \Leftrightarrow If any NP Complete Problem is polynomial time solvable then $P = NP$
- \Rightarrow Any Problem in NP-Complete is not solvable in polynomial time then $P \neq NP$

~~Q~~ A, B, and C are three decision problems and A is NPC then which one of the following is true.

- a) $A \in NP \setminus NP\text{-Complete}$
- b) $\forall L \in NP, L \leq_p A$
- c) If $C \in NP$ and $A \leq_p^{NP\text{-Complete}} C$ then C is NPC (Note 1)
- d) If $B \in NP$ and $B \leq_p A$ then B is NPC (Note 2)

~~Notes~~ (1)

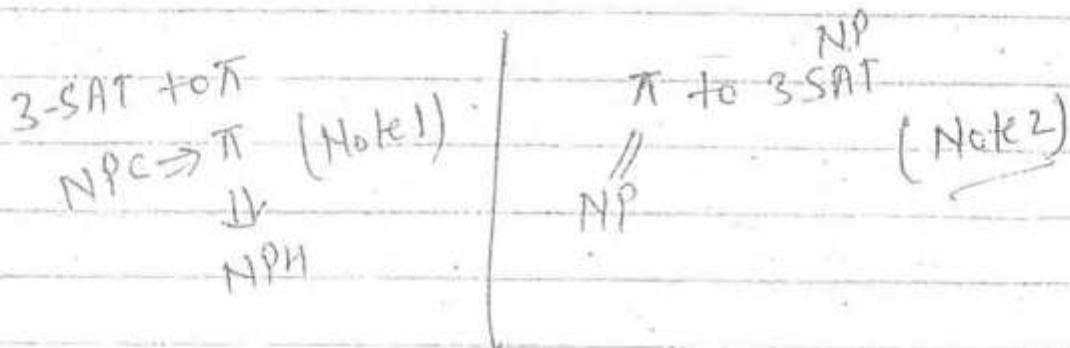
Ram and Shyam want to show that a problem π is NP-complete.

Ram shows a polynomial time reduction from π to 3-SAT.

Shyam shows $\pi \leq_p 3\text{-SAT}$.

Then which one of the following is true.

- a) π is NPC
c) π is NP
- b) π is NPH
d) π is P.

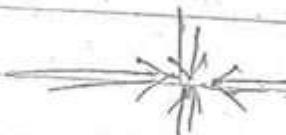


Let S be a NPC and Q and R be 2 other problems not known to be in NP.

If $S \leq_p^{\text{NPC}} R$ & $Q \leq_p^{\text{NP}} S$ then which of the following is true?

- a) R is NPC b) R is NPH
c) Q is NPC d) Q is NPH.

OBSI (Optimal Cost Binary Search Tree)



Any Comparison Based sorting will take $\Omega(n \log n)$ time complexity in Worst Case

W.C.

B.C.

Avg.

Q.Sort $\Rightarrow n^2$

$n \log n$

$n \log n$

M.S. $\Rightarrow n \log n$

$n \log n$

$n \log n$

H.S. $\Rightarrow n \log n$

$n \log n$

$n \log n$

Selection $\Rightarrow n^2$

n^2

n^2

Inversion $\Rightarrow n^2$

n

n^2

Bubble Sort $\Rightarrow n^2$

n^2

n^2

Sorting

Comparison Based
Quick Sort, Merge Sort

Non-Comparison Based

Counting Sort

Radix Sort

Bucket Sort

Heap Sort

Selection Sort

Insertion Sort

Bubble Sort

Counting Sort (Assume all the elements are in the range (0-K))

CountingSort (a, n, k)

{
① for ($i=0$; $i \leq K$; $i++$) $\rightarrow O(K)$.
 $c[i] = 0$;

② for ($i=1$; $i \leq n$; $i++$) $\rightarrow O(n)$
 $c[a[i]] = c[a[i]] + 1$;

③ for ($i=1$; $i \leq K$; $i++$) $\rightarrow O(K)$
 $c[i] = c[i] + c[i-1]$;

④ for ($j=n$; $j \geq 1$; $j--$) $\rightarrow O(n)$

{
 $b[c[a[j]]] = a[j]$;

$c[a[j]] = c[a[j]] - 1$;

{ } $\rightarrow O(n+K)$

$(n+K)$

$O(n)$

A :

2	5	3	0	2	3	0	3
1	2	3	4	5	6	7	8

① C: [0 0 0 0 0]

0	1	2	3	4	5
---	---	---	---	---	---

② C: [2 0 2 2 1 3]

1	0	1	2	0	1
0	1	2	3	4	5

C: [2 0 2 3 0 1]

0	1	2	3	4	5
---	---	---	---	---	---

// how many times
it's present in the
given array.

③ C: [2 1 2 4 1 7 8 7 0 1]

0	1	2	3	4	5
---	---	---	---	---	---

2	2	4	7	7	0
0	1	2	3	4	5

// how many elements
are present such
that all the elements
are $\leq i$

C: [2 2 4 3 7 0 1] \Rightarrow C: [1 2 2 4 7 7]

0	1	2	3	4	5
0	1	2	3	4	5

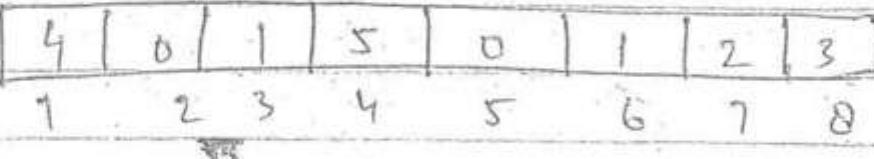
B5 [0 0 2 2 3 3 3 5]

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

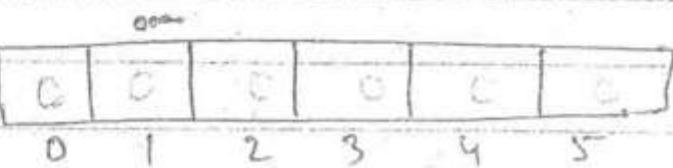
Note -

- (1) Counting search is not in place sorting techni'
- (2) " " " stable sorting .

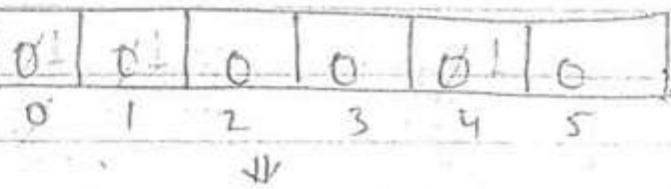
(0-5)

A: 
1 2 3 4 5 6 7 8

4	0	1	5	0	1	2	3
1	2	3	4	5	6	7	8

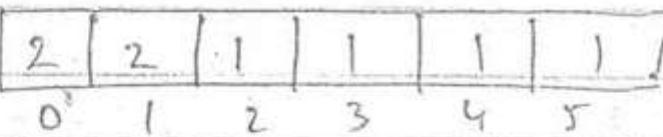
(1) c: 
0 1 2 3 4 5

0	0	0	0	0	0
0	1	2	3	4	5

(2) c: 
0 1 2 3 4 5

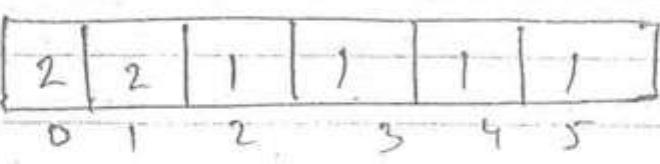
↓

0	0	0	0	0	0
0	1	2	3	4	5

c: 
0 1 2 3 4 5

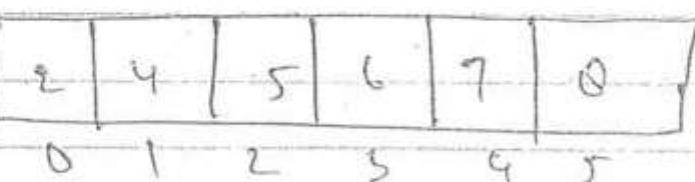
↓

2	2	1	1	1	1
0	1	2	3	4	5

(3) c: 
0 1 2 3 4 5

↓

2	2	1	1	1	1
0	1	2	3	4	5


0 1 2 3 4 5

2	4	5	6	7	0
0	1	2	3	4	5

④	c:	0 2 4 5 6 7
		0 1 2 3 4 5

c:	0 2 4 5 6 7
	0 1 2 3 4 5

B:	0 0 0 1 1 1 2 3 4 5 6 7 8
	0 1 2 3 4 5 6 7 8

Radix Sort

i/p : 375 569 451 786 420 825 143 137 007 911
 1 2 3 4 5 6 7 8 9 10

Pass 1 : 420 451 911 143 375 825 786 137 007 569
 1st LSB (10¹)

Pass 2 : 007 911 420 825 137 143 451 569 375 786
 2nd LSB (10²)

Pass 3 : 007 137 143 375 420 451 569 786 825 911
 3rd LSB
 $\Rightarrow O(dn)$ $d_n = \text{no of digit}$ (10³)
 $\Rightarrow O(n)$

$10n \times 3$

\downarrow

$30n$

\downarrow

$O(n)$

eg

I/P : 329 457 657 839 436 720 355
1 2 3 4 5 6 7

Pass 1 : 720 355 436 457 657 329 839
1ⁿLSB

Pass 2 : 720 329 436 839 355 457 657
2ⁿLSB

Pass 3 : 329 355 436 457 657 720 839
3ⁿLSB

Radix sort not in place
 but stable

$O(n^3)$

Bucket Sort

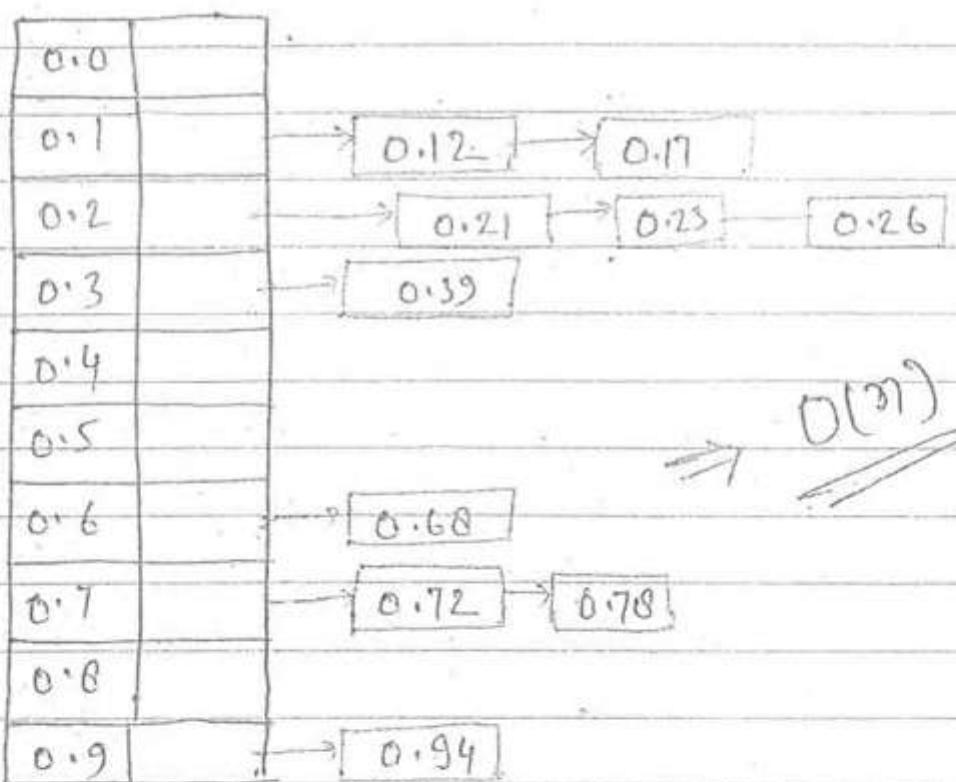
(Assume uniform distribution)

Assume that all the elements are in the range $[0, 1]$; [including 0 Excluding 1]

i/p $\frac{70}{100}, \frac{17}{100}, \frac{39}{100}, \frac{26}{100}, \frac{72}{100}, \frac{94}{100}, \frac{21}{100}, \frac{12}{100}, \frac{23}{100}, \frac{68}{100}$

$70, 17, 39, 26, 72, 94, 21, 12, 23, 68$

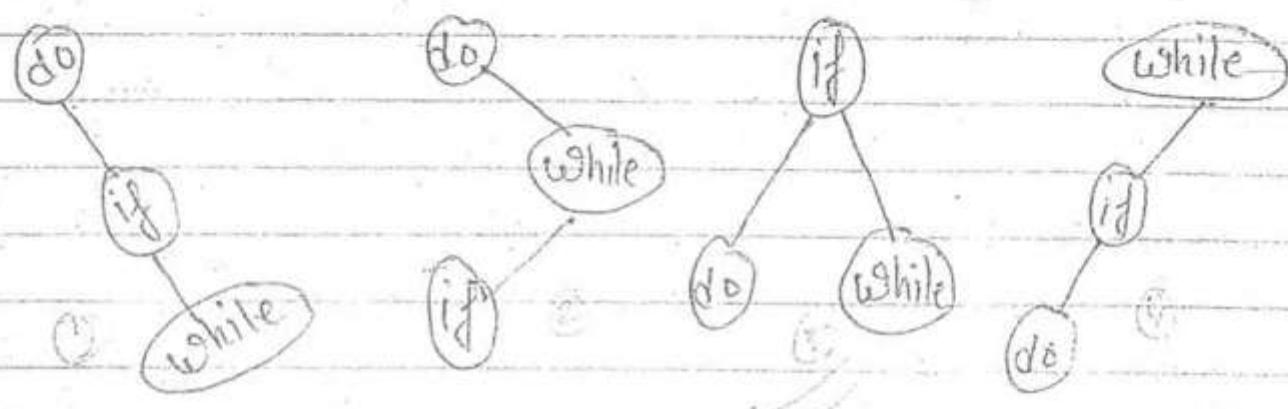
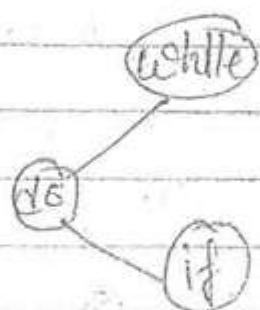
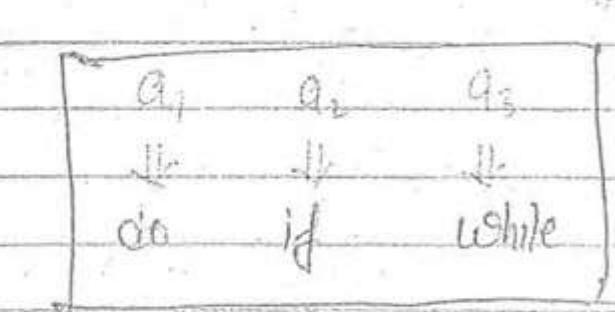
$M = 10$



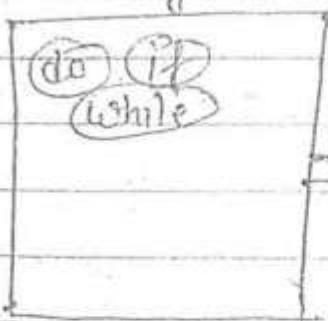
$12 | 17 | 21 | 23 | 26 | 39 | 68 | 72 | 70 | 94$

{ Bucket sort also not in place }

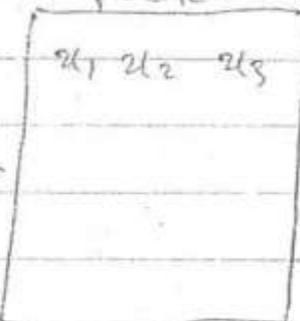
Optimal Cost Binary Search tree (OBST)



English



French



100

①

$$do = 20$$

$$20 \times 1 = 20$$

$$if = 50$$

$$50 \times 2 = 100$$

$$white = 30$$

$$30 \times 3 = 90$$

$$\underline{210}$$

$$(2) \quad 20 \times 1 = 20$$

$$50 \times 3 = 150$$

$$30 \times 2 = 60$$

$$\underline{230}$$

$$(3) \quad 20 \times 2 = 40$$

$$50 \times 1 = 50$$

$$30 \times 2 = 60$$

$$\underline{\underline{150}}$$

$$(4) \quad 20 \times 3 = 60$$

$$50 \times 2 = 100$$

$$30 \times 1 = 30$$

$$\underline{190}$$

$$(5) \quad 20 \times 3 = 40$$

$$50 \times 3 = 150$$

$$30 \times 1 = 30$$

$$\underline{\underline{220}}$$

For the given 3 ~~prob~~ keywords do, if, while with the frequency 20, 50, 30 respectively. 5 difference BST. In those 5 BST optimal one is (3)

Cost of a BST purely Based on:

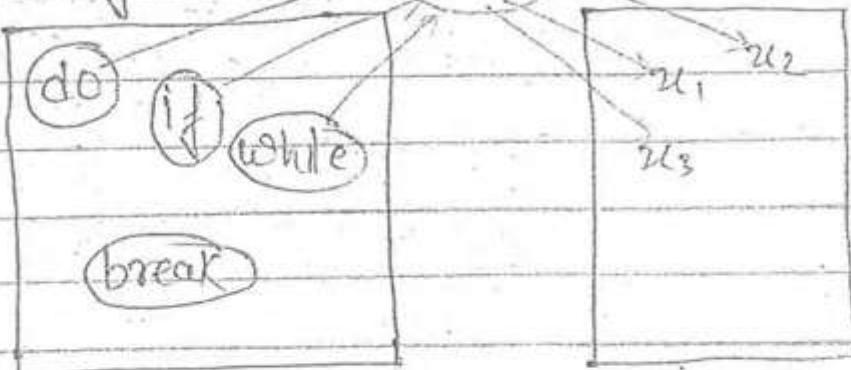
(i) no of leafs.

(ii) highest frequency keywords should be there near to root.

English

BST

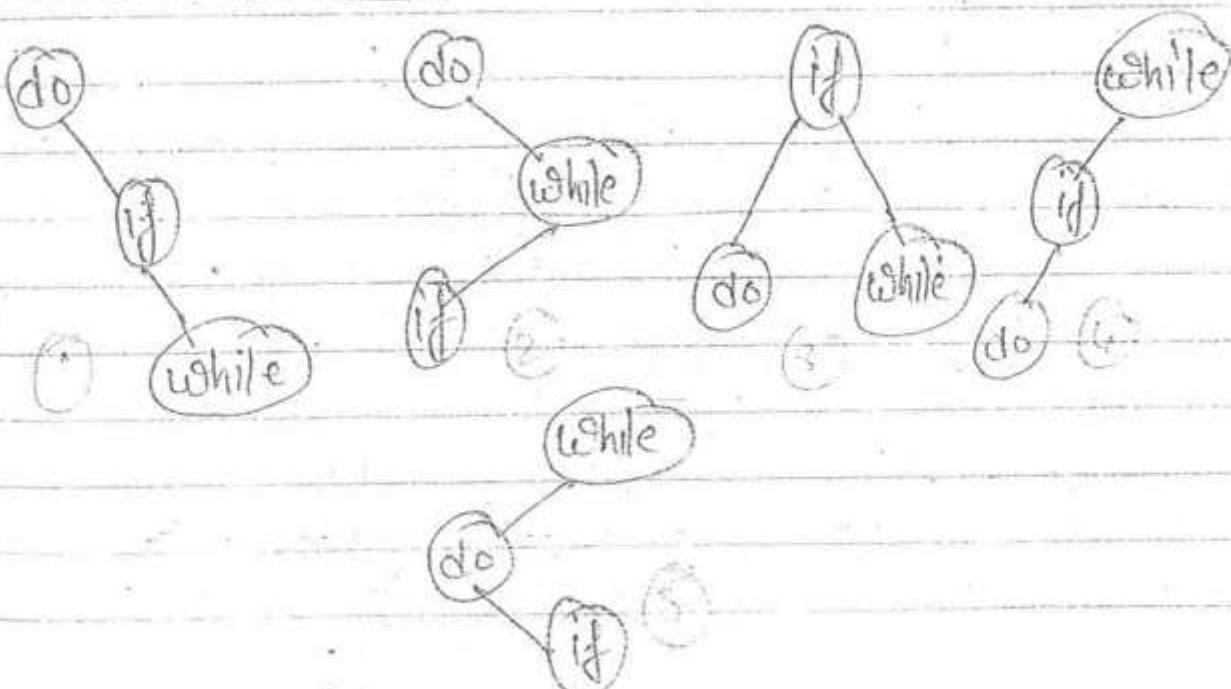
French.



Cost of any BST going will be going to cost required are success full mode \leftarrow cost required for all unsuccessful mode.

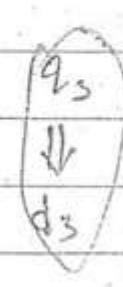
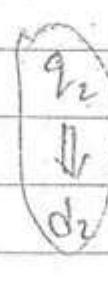
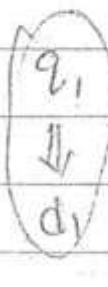
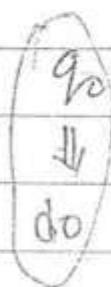
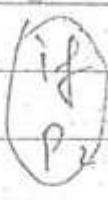
$$\text{Cost(BST)} = \text{Cost}(\text{successfull mode}) + \text{Cost}(\text{unsuccessfull mode})$$

do , if , while



3

$$n=3$$



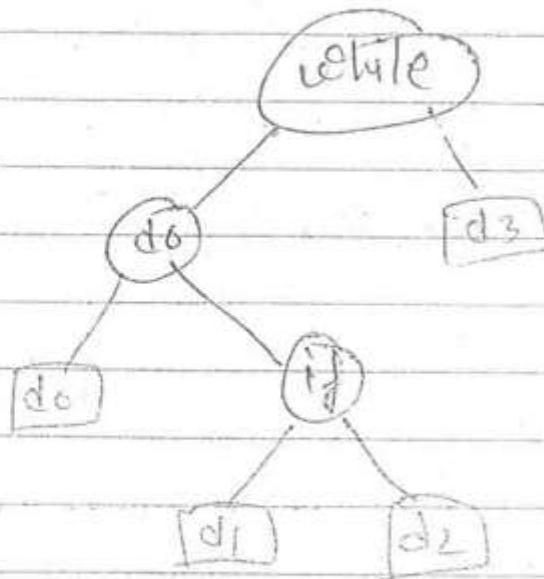
ans

$$\text{Cost (BSR)} = \text{Cost}(\text{successful node}) + \text{Cost}(\text{unsuccessful node})$$

$$= \sum_{i=1}^n p_i l_i + \sum_{i=0}^m q_i l_i$$

$p_i \Rightarrow$ Probability of node

$l_i \Rightarrow$ level of nodes



$$\text{Cost (SSN)} = 20 \times 2 + 50 \times 3 + 30 \times 1$$

+

$$\text{Cost (USS)} = q_0 \times 3 + q_1 \times 4 + q_2 \times 4 + q_3 \times 2$$

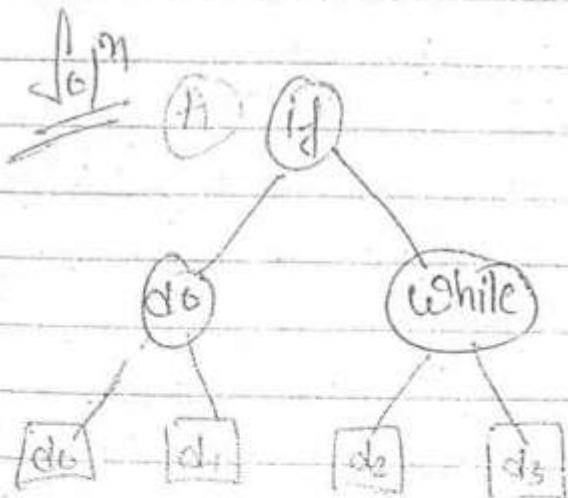
Σ

$$n = 3$$

$$a_1, a_2, a_3 = (\text{do}, \text{if}, \text{while})$$

$$(p_1 - p_3) \quad 0.5 \quad 0.1 \quad 0.05$$

$$(q_0 - q_5) \quad 0.15 \quad 0.1 \quad 0.05 \quad 0.05$$

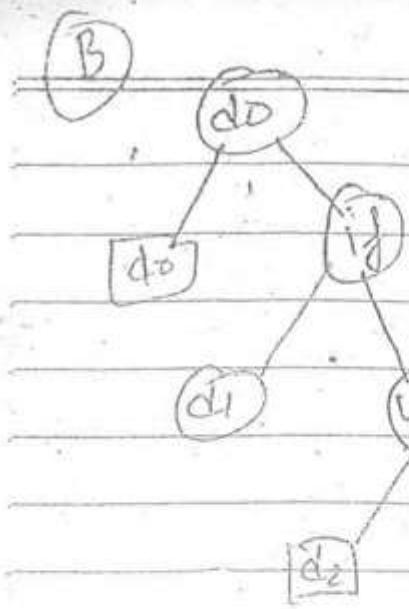


$$\begin{aligned} \text{Cost (SSN)} &= 0.1 \times 1 + 0.5 \times 2 + 0.05 \times \\ &= 1 + 1.0 + 0.1 = 1.20 \end{aligned}$$

$$\begin{aligned} \text{Cost (USS)} &= 0.15 \times 3 + 0.1 \times 3 + 0.05 \times 3 \\ &\quad + 0.05 \times 3 \\ &= 0.45 + 0.3 + 0.15 + 0.15 \\ &= 1.05 \end{aligned}$$

$$\text{Cost (BSI)} = 1.20 + 1.05$$

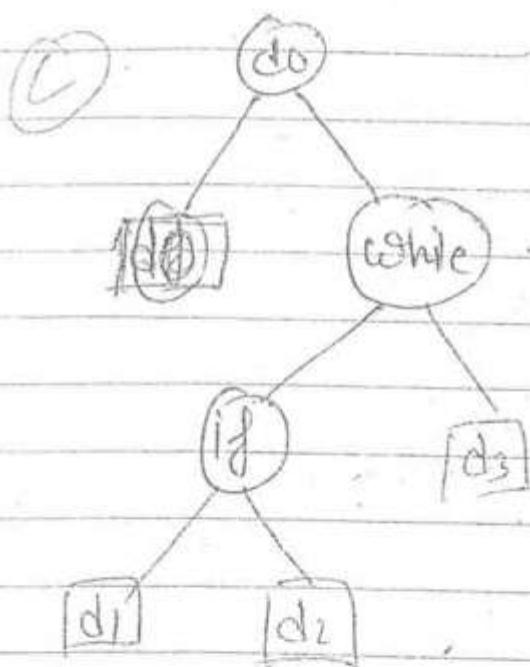
$$= \boxed{2.25}$$



$$\begin{aligned}
 \text{Cost(SSN)} &= 0.5 \times 1 + 0.1 \times 2 + 0.05 \times 3 \\
 &= 1.5 + 1.2 + 0.15 \\
 &= 1.85
 \end{aligned}$$

$$\begin{aligned}
 \text{Cost(USN)} &= 0.15 \times 2 + 0.1 \times 3 + 0.05 \times 4 \\
 &\quad + 0.05 \times 4 \\
 &= 1.30 + 1.3 + 1.20 + 1.20 \\
 &= 1.00
 \end{aligned}$$

~~Cost(BST) = 1.85~~

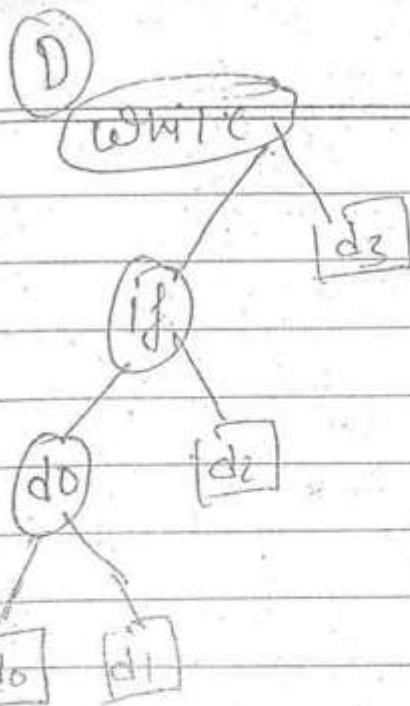


$$\begin{aligned}
 \text{Cost(SSN)} &= 0.5 \times 1 + 0.05 \times 2 + 1 \times 3 \\
 &= 1.5 + 1.0 + 1.3 \\
 &= 1.9
 \end{aligned}$$

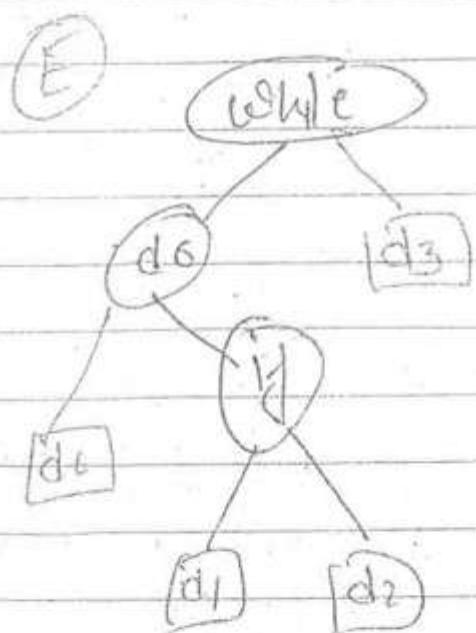
$$\begin{aligned}
 \text{Cost(USN)} &= 0.15 \times 2 + 0.1 \times 4 + 0.05 \times 4 \\
 &\quad + 0.05 \times 3 \\
 &= 1.30 + 1.4 + 1.20 + 1.15 \\
 &= 1.05
 \end{aligned}$$

~~Cost(BST) = 1.9 + 1.05~~

~~= 1.95~~



$$\text{Cost(BST)} = 3.00$$



$$\text{Cost(BST)} = 2.50$$

In the above 5 BST ~~there be~~ tree B is optimal cost binary tree becoz higher probability mode it self ^ and next higher probability go to next on root.

Sometimes even though root is not higher probability node we may get optimal cost BST.

So finally we have to cover all possible combination.

X + -

Y |

$$W(i, j) = \sum_{K=i}^j p_k + \sum_{K=i+1}^j q_k$$

do if while
 $p_1 - p_3 = 0.15 \quad 0.1 \quad 0.05$
 $q_1 = q_3 = 0.15 \quad 0.1 \quad 0.05 \quad 0.05$
 $q_2 \quad q_1 \quad q_2 \quad q_3$

$e[i, j]$ = The min cost required for the optimal cost BST which contains i to j elements.

$$e[a_1, a_2, a_3] = \min \begin{cases} e[1, 2] + e[2, 3] + p_1 + W[1, 2] + W[2, 3] \\ e[1, 1] + e[3, 3] + p_2 + W[1, 1] + W[3, 3] \\ e[1, 2] + e[4, 5] + p_3 + W[1, 2] + W[4, 5] \end{cases}$$

Dots

$$e[i, j] = \min \left\{ \begin{array}{l} \left. \begin{array}{l} e[i, i-1] + e[i+1, j] + p_i + w(i, i-1) \\ + w(i+1, j) \end{array} \right| i \text{ as the root} \\ \left. \begin{array}{l} e[i, j-1] + e[j+1, j] + p_j + w(i, j-1) \\ + w(j+1, j) \end{array} \right| j \text{ as the root} \end{array} \right\}$$

OR
↓

$$e[i, j] = \min \left\{ \begin{array}{l} \left. \begin{array}{l} e[i, r-1] + e[r+1, j] + p_r + w(i, r-1) \\ + w(r+1, j) \end{array} \right| 1 \leq r \leq j \end{array} \right\}$$

$m=1$

a_1
↓

p_1

$q_0 \ q_1$

(q)

[do] [di]

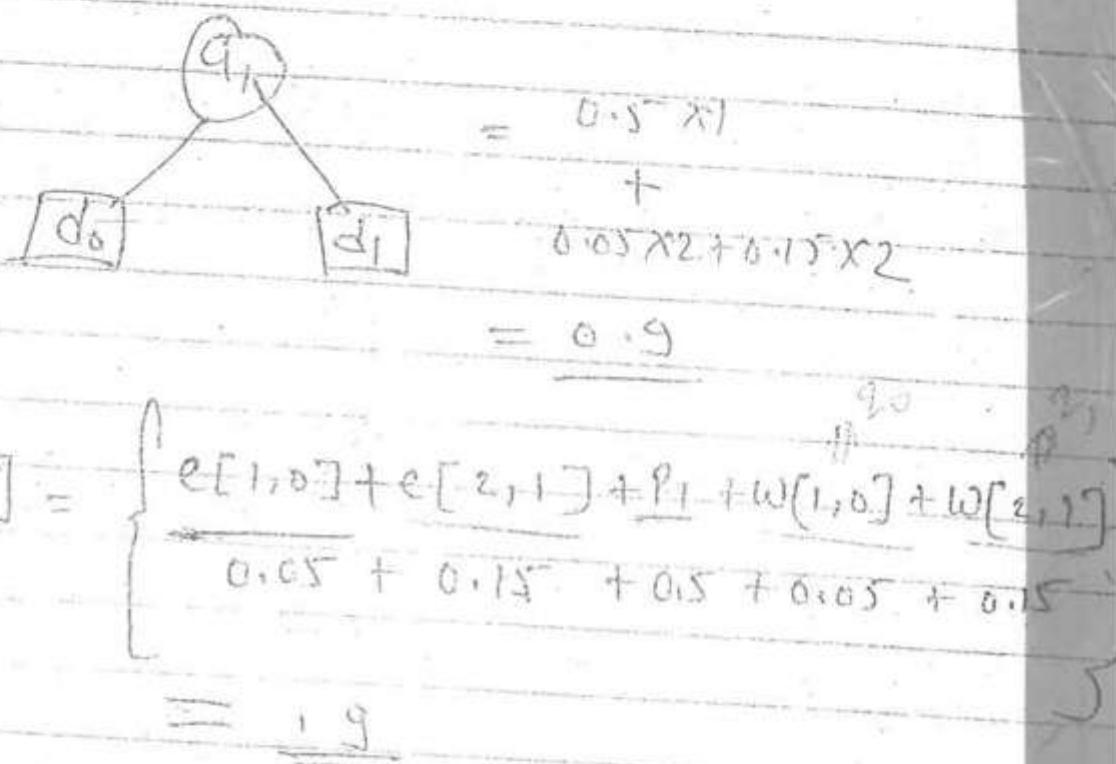
28.3 : A.A.D

~~Q~~
 $n=1$

$a_1 = d_0$

$p_1 = 0.5$

$q_0, q_1 = 0.05, 0.15$



$$e[1,1] = \begin{cases} e[1,0] + e[2,1] + p_1 + w[1,0] + w[2,1] \\ 0.05 + 0.15 + 0.5 + 0.05 + 0.15 \end{cases}$$

$$= 1.9$$

D.A.A. · 235