

Computer Notes

DESIGN
ANALYSIS
&
ALGORITHM

PART-I

(1)

Asymptotic Notation1. O - Notation.2. Ω - Notation.3. Θ - NotationQ) Let $f(n)$ & $g(n)$ be two type functions.Q) 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- constants $c > 0$ & $n_0 > 0$

such that

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

$$0 < f(n) \leq g(n)$$

Q.

$$f(n) = n$$

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

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

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

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

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

~~Af~~

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

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

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

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

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

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

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

Ω -Notation (lower bound)

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

there exist a constant $c > 0$ & $n_0 > 0$

such that

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

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

Θ -Notation ($L.B = U.B$)

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

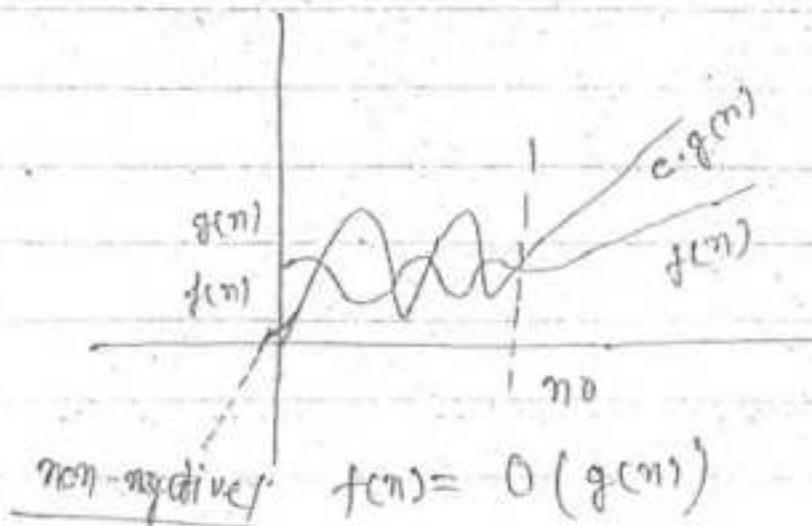
(1) $f(n) = O(g(n))$

&

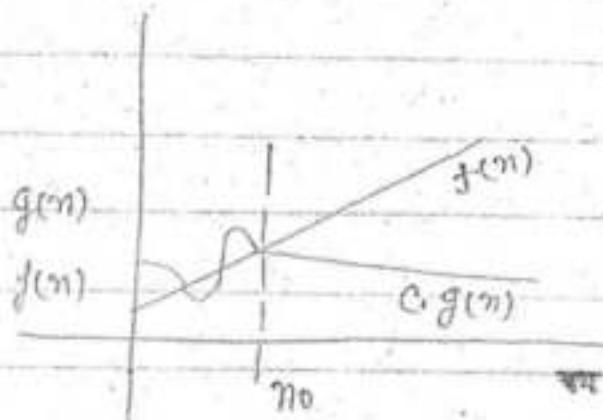
(2) $f(n) = \Omega(g(n))$

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

Big - O- Notation

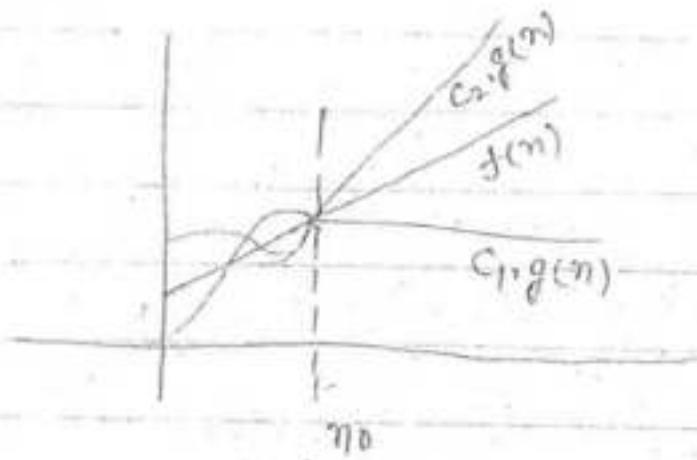


Omega - Notation



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

Theta - Notation



$n \rightarrow$

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

$$\# A = n^2$$

$$B = n^3$$

$$A = \Theta(B)$$

$$\# A = n^2$$

$$B = n^2$$

$$A = \Theta(B)$$

$$\# A = n^2$$

$$B = n$$

$$A = \Omega(B)$$

g

P

↓

A

↓

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

Worst case $n^2 (\Theta(n^2))$

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

Types of Time Complexity

- i) Constant Time complexity $\Rightarrow O(1)$
- ii) Logarithmic " $\Rightarrow O(\log n) \Rightarrow \overline{\log n}$
- iii) Linear " $\Rightarrow O(n)$
- iv) Quadratic " $\Rightarrow O(n^2)$
- v) Cubic " $\Rightarrow O(n^3)$
- vi) Polynomial " $\Rightarrow O(n^k), k \geq 3$
- vii) 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

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

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

↓

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

↓

↓

#

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

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

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

$$\frac{n^2 \log n}{n} > \frac{n (\log n)^{10}}{(\log n)^9}$$

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

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

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

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

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

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

Which one of the following is true or false?

a) $100 n \log n = O\left(\frac{n \log n}{100}\right)$ // 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) $0 < n < 4$ then

$$n^n \text{ is } O(n^4)$$

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

L (a) $100 \cdot n \log n = O \frac{n \log n}{100}$

$$n \log n \leq c \cdot \underline{n \log n}$$

$$\Downarrow \frac{10000}{1000000}$$

$$1000000$$

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 2^n 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

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

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

$$3) a^{\log_b n} = n^{\log_b a} \quad \checkmark$$

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

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

$$6) \log_b a = \sqrt[\log_a b]{a}$$

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

Σn^3

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 $\sqrt{n^2}$ ✗

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_5 .

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

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

(1) $f_3(n) = n \log_2(n)^5$

(3) $f_4(n) = n^{\log_2 7}$

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) $f_2 f_3 f_4 f_1$ (②)

Sol^m

$2^m \quad n^{\log n}$

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

$n^m \quad 2^m$

$n^m \quad 2^m$

$n^{3/2} \quad n \log n$

$n^m \quad n \log n$

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

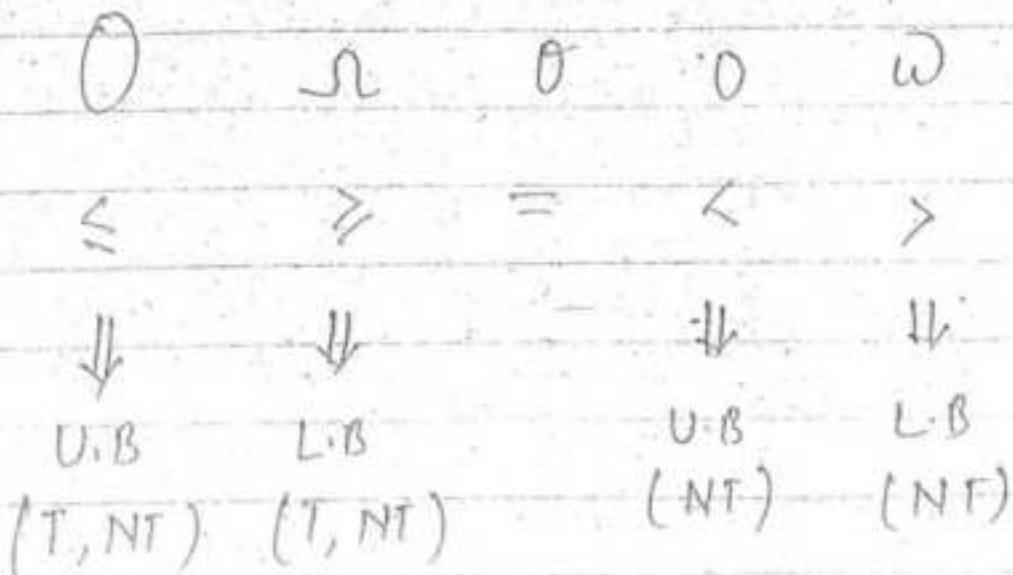
$n^2 < n^5$

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

O

$n^2 > n$ — NTUB
 $n^2 = n^2$ — T.L.B.

$$\boxed{n^2 > n} \rightarrow NT LB$$



$f(n) = n!$ [not comparable]
 $g(n) = n^{\sin \theta}$

$f(n) = \sqrt[n]{n}$ } not comparable,
 $g(n) = n^{\sin \theta}$

$$\sqrt[n]{n} > n^{\sin \theta} \quad (\sin \theta = 0)$$

$$\sqrt[n]{n} < n^{\sin \theta} \quad (\sin \theta = 1)$$

$O(1)$

$\log(\log n)$

$\log n$

$\log n$

$\overline{f(n)}$

n

n^2

n^3

n^K

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

Properties of Asymptotic Notation.

i) Reflexive \Rightarrow

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

if

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

\Downarrow

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

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

$$B = \{a, b, c\}$$

2. Symmetric

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

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

then

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

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

$$\boxed{M = M^T}$$

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

then

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

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

then

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

transitive

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

$$\text{If } f(n) = O(g(n))$$

$$2 < 4$$

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

then

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

) ~~If~~ Similarly π, θ also satisfied transitive.

Properties

L

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

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

$$\text{If } f(n) = O(g(n))$$

then

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

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

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

1

$$\text{If } f(n) = O(g(n))$$

$$g(n) \leq h(n)$$

then

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

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

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

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

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

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

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

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

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

↓

$$g(n) = h(n), \quad f(n) < g(n)$$

$$f(n) < g(n) = h(n)$$

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

$$\frac{S(n)}{2} = T_1(n) = \Theta(n^2)$$

$$T_2(n) = \Theta(n^2)$$

$$T(n) = \sum_{i=1}^n i$$

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

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

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

$$= \frac{n(n+1)(2n+1)}{6} = \Theta(n^3)$$

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

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

$$= \frac{n \times n \times n \times \dots}{n}$$

$$= O(n^n)$$

(IV)

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

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

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

$$= O(u^n)$$

(V)

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

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

$$= 2$$

$$= O(1)$$

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

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

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

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

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

Q Consider the following two functions. The fun
are give like a

$$g_1(n) = \begin{cases} n^3 & 0 \leq n \leq 10,000 \\ n^2 & n > 10,000 \end{cases}$$

$$g_2(n) = \begin{cases} n & 0 \leq n \leq 100 \\ n^3 & n > 100 \end{cases}$$

for $g_1(n)$ and $g_2(n)$ what is the relation g .

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

~~$f(n)$~~

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



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

In big-O notation only n starting is available

$$f_1(n) = n^3 \quad 0 \leq n \leq 10,000$$

$$o(n) = n^2 \quad n \geq 10,000$$

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

$$n^3 \quad n > 100$$

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

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

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

17/06/11

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

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

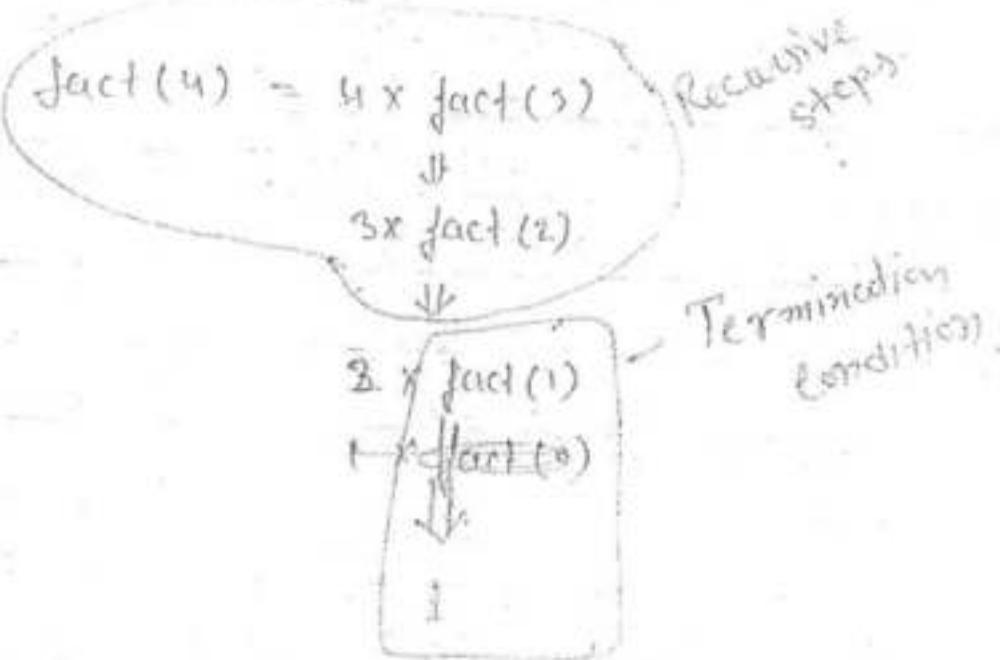
worst $n \quad n^2 [\Omega(n^2)]$

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

Recursion

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

e.g.



In this funⁿ to funⁿ 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.

(ex)

eg

$$\text{Mul}(2, 3) = 2 + \text{Mul}(2, 2)$$



$$2 + \text{Mul}(2, 1)$$



$$2 + \boxed{\text{Mul}(2, 0)}$$



0

Recursive steps.

Termination condition

eg

$$\text{Pow}(2, 3) = 2 * \text{Pow}(2, 2)$$



$$2 * \text{Pow}(2, 1)$$



$$2 * \boxed{\text{Pow}(2, 0)}$$



1

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)

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

e.g. ✓ $\text{GCD}(5, 50) = 5$

✓ $\text{GCD}(5, 7) = 1$

✓ $\text{GCD}(0, 5) = 5$

✓ $\text{GCD}(5, 0) = 5$

✓ $\text{GCD}(0, 0) = 1$

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

$$\begin{array}{r} \downarrow \\ 5 \cancel{|} \begin{array}{l} 50 \\ 50 \end{array} \left(\begin{array}{l} 10 \\ 0 \end{array} \right) \end{array}$$

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

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

$$\begin{array}{r} \downarrow \\ 5 \cancel{|} \begin{array}{l} 7 \\ 5 \end{array} \left(\begin{array}{l} 1 \\ 2 \end{array} \right) \end{array}$$

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

$$\begin{array}{r} \downarrow \\ 5 \cancel{|} \begin{array}{l} 2 \\ 0 \end{array} \left(\begin{array}{l} 0 \\ 2 \end{array} \right) \end{array}$$

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

$$7 \cancel{|} \begin{array}{l} 5 \\ 0 \end{array} \left(\begin{array}{l} 0 \\ 5 \end{array} \right)$$

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

$$5 \cancel{|} \begin{array}{l} 7 \\ 5 \end{array} \left(\begin{array}{l} 1 \\ 2 \end{array} \right)$$

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

$$2 \cancel{|} \begin{array}{l} 5 \\ 2 \end{array} \left(\begin{array}{l} 1 \\ 2 \end{array} \right)$$

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

$$1 \cancel{|} \begin{array}{l} 2 \\ 0 \end{array} \left(\begin{array}{l} 0 \\ 2 \end{array} \right)$$

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

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

$$\begin{array}{r} \text{↓} \\ 100) 5 \quad (0 \\ \text{---} \\ \quad 0 \\ \cdot \frac{0}{5} \end{array}$$

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

$$\begin{array}{r} \text{↓} \\ 5) 100 \quad (20 \\ \text{---} \\ \quad 100 \\ \cdot \frac{100}{100} \end{array}$$

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

$$\begin{array}{r} \text{↓} \\ 5 \end{array}$$

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

$$\begin{array}{r} \text{↓} \\ 200) 100 \quad (0 \\ \text{---} \\ \quad 0 \\ \cdot \frac{0}{100} \end{array}$$

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

$$\begin{array}{r} \text{↓} \\ 100) 200 \quad (2 \\ \text{---} \\ \quad 200 \\ \cdot \frac{200}{200} \end{array}$$

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

$$\begin{array}{r} \text{↓} \\ 100 \end{array}$$

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

$$\begin{array}{r} \text{↓} \\ b \end{array}$$

$$\begin{array}{r} b) a \mid c \\ \text{---} \\ \quad 0 \\ \cdot \frac{0}{?} \end{array}$$

#

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 * fact (n-1)

return (f)

}

}

Termination Condition another name is Initial Condition

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

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 Rel n is

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

✓ $\text{fib}(\text{int } n)$

```

{
    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);

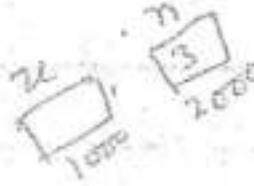
}

{

```

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

```



13

fact (int n)



```

{
    (1) int f;
    if (n==0 || n==1)
        return 1;
}

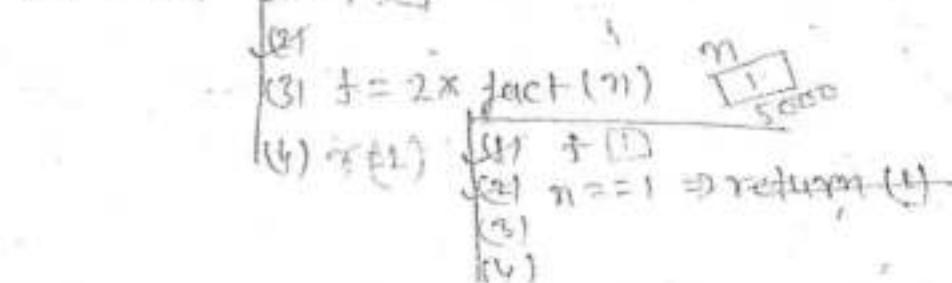
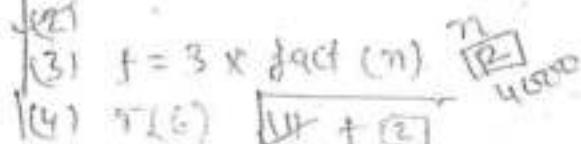
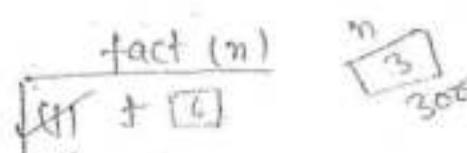
```

else

```

    {
        (3) f = n * fact(n-1),
        (4) return f;
    }
}

```



More space is Drawback of the Recursive Program.

Simple C Program of factorial n.

fact (int n)

{

 int t=1, i;

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

 t = t * i;

 return(t);

}

Solving Recurrence Relation

① Substitution Method.

(2) Recursive Tree Method.

(3) Master Method.

1 Substitution Method

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

eg

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

(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 .
 |
 |
 ↓
 50

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

|
 ↓
 50(n-1)

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

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

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

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

$$= O(n^2)$$

solve the following recursion Relation

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

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
 $n-1$

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

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

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

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

* $T(n) = T(n/2) + c$ if $n > 1$
 = 1 if $n = 1$

Soln

$$\begin{aligned}
 T(n) &= T(n/2) + 1.c \\
 &= \boxed{T(n/2) + c} + c \\
 &= T(n/2^2) + 2c \\
 &= \boxed{T(n/2^2) + c} + 2c \\
 &= T(n/2^3) + 3c \\
 &\quad \downarrow \text{K times} \\
 &= T(n/2^K) + KC \\
 &= T(1) + KC \\
 &= 1 + c * \log_2 n \\
 &= \boxed{O(\log n)}
 \end{aligned}$$

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

Soln

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

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

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

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

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

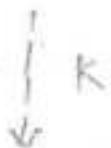
$$= \underline{\underline{O(n)}}$$

~~K~~

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

Sd^n

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



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

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

Sol

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

$$\begin{aligned} T(n) &= T\left(\frac{n}{2^K}\right) + kn \left[\frac{1}{2^0} + \frac{1}{2^1} + \dots + \frac{1}{2^{K-1}} \right] \\ &= T(1) + n \left[\underbrace{\frac{1 - (1/2)^K}{1 - 1/2}}_{1 - \gamma_2} \right] \\ &= 1 + n \left[1 - \frac{1}{n} \right] \\ &= n \left[\frac{n-1}{n} \right] \\ \Rightarrow & O(n) \end{aligned}$$

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

Solve

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

=

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

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

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

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

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

$$= 8^2 T\left(\frac{n}{2^2}\right) + 2^1 n^2 + 2^0 n^2$$

$$= 8^2 \left[8T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)^2 \right] + 2^2 n^2 + 2^0 n^2$$

$$= 8^3 T\left(\frac{n}{2^3}\right) + 2^2 n^2 + 2^1 n^2 + 2^0 n^2$$

1
1
1
K-1

$$T(n) = 8^{K-1} T\left(\frac{n}{2^{K-1}}\right) + n^2 \left[2^0 + 2^1 + 2^2 + \dots + 2^{K-2} \right]$$

$$= \frac{8^K}{8} T(2) + n^2 \left[\frac{1(2^{K-1} - 1)}{2^1 - 1} \right]$$

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

$$= n^3 + n^2 [n-1]$$

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

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

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

Solⁿ

$$\begin{aligned} T(n) &= 7T\left(\frac{n}{2}\right) + n^2 \\ &= 7 \left[T\left(\frac{n}{2}\right) + \left(\frac{n^2}{4}\right) \right] + n^2 \\ &= 7T\left(\frac{n}{2^2}\right) + 7\left(\frac{n^2}{4}\right)^2 + (n^2) \\ &= 7T\left(\frac{n}{2^4}\right) + \left(\frac{7}{4}\right)^1 n^2 + \left(\frac{7}{4}\right)^2 n^2 \\ &= 7T\left(\frac{n}{2^{12}}\right) + \left(\frac{7}{4}\right)^2 n^2 + \left(\frac{7}{4}\right)^1 n^2 + \left(\frac{7}{4}\right)^0 n^2 \end{aligned}$$

K-1

$$\begin{aligned} T(n) &= 7^{K-1} T\left(\frac{n}{2^{K-1}}\right) + n^2 \left[\left(\frac{7}{4}\right)^6 + \left(\frac{7}{4}\right)^1 + \dots + \left(\frac{7}{4}\right)^{K-2} \right] \\ &= \frac{7^K}{7} T(2) + n^2 \left[1 \left(\left(\frac{7}{4}\right)^{K-1} - 1 \right) \right] \\ &= n^{\log_2 7} + n^2 \left[\left(\frac{7}{4}\right)^K - 1 \right] \\ &= n^{\log_2 7} + n^2 \times \left(\frac{7}{4}\right)^K - n^2 \end{aligned}$$

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

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

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

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

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



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

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

$$\log_2 n = k$$

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

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

$$\log_2 n = k$$

$$T\log_2 n = k$$

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

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

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

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

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

$$1 + \log n!$$

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

$$= 1 + n \log n$$

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

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

S_n

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

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

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

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

\downarrow
n-1 time

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

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

$\log n$

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

$\log_2 n = k$

$$\begin{aligned}
 T(n) &= 2 \left[2T\left(\frac{n}{2^2}\right) + \frac{n}{2} \log \frac{n}{2} \right] + n \log n \\
 &= 2^2 \left[2T\left(\frac{n}{2^3}\right) + \frac{n}{2^2} \log \frac{n}{2^2} \right] + n \log \frac{n}{2} + n \log n \\
 &= 2^3 T\left(\frac{n}{2^4}\right) + n \log \frac{n}{2^3} + n \log n \\
 &\quad \vdots \text{k times} \\
 &= 2^k T\left(\frac{n}{2^{k+1}}\right) + n \left[\log \frac{n}{2^0} + \log \frac{n}{2^1} + \log \frac{n}{2^2} + \dots + \log \frac{n}{2^k} \right]
 \end{aligned}$$

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

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

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

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

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

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

 $n = 5^K$ $\log_5 n = K$ S_1^n

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

$$= S^2 T\left(\frac{n}{5^2}\right) + \frac{n}{\log_5 n/5} + \frac{n}{\log_5 n}$$

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

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

↓ K times.

$$= S^K T\left(\frac{n}{5^K}\right) + n \left[\frac{1}{\log_5 \frac{n}{5^0}} + \frac{1}{\log_5 \frac{n}{5^1}} + \dots + \frac{1}{\log_5 \frac{n}{5^{K-1}}} \right]$$

$$= n T(1) + n \left[\underbrace{\frac{1}{\log_5 \frac{n}{5^0}}}_{K}, \underbrace{\frac{1}{\log_5 \frac{n}{5^1}}, \dots, \frac{1}{\log_5 \frac{n}{5^{K-1}}}}_{K} \right]$$

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

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

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

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

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



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

$$\begin{array}{l} n=2^k \\ n=2 \times k \\ k=n/2 \end{array}$$

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

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

\downarrow
K times

$$\begin{aligned} T(n) &= T(n-2 \times K) + 2\log(n(2^{K-1})) + 2\log(n-(2^{K-1})) \\ &\quad + \dots + 2\log(n-2^K) \end{aligned}$$

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

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

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

$$\begin{aligned} &= \left(\log_2 2 + \log_2 1 \right) + \left(\log_2 2 + \log_2 2 \right) + \left(\log_2 2 + \log_2 3 \right) \\ &\quad + \dots + \left(\log_2 2 + \log_2 K \right) \end{aligned}$$

$$= K \cdot \log_2^2 + \left[\log_2 1 + \log_2 2 + \log_2 3 - \log_2 K \right]$$

$$= K + \log K!$$

$$= K + K \log K.$$

$$= n_2 + n_2 \log n_2$$

$$= O(\log n_2)$$

$\frac{1}{2}$ $T(n) = T(n-2) + n^2 \quad \text{if } n > 0$

 $= 1 \quad \text{if } n = 0$

Solⁿ $T(n) = T(n-2) + n^2$

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

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

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

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

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

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

=

|
 | $n-1$ times
 |
 |
 |
 |

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

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

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

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

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

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

$$\begin{aligned} &+ 2^{n-3}(3) + 2^{n-2}(2) \\ &\quad \text{+ } \cancel{2^{n-3}(1)} \quad \text{+ } \cancel{2^{n-2}(1)} \end{aligned}$$

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

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

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

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

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

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

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

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

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

$n = 2^K$

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

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

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

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

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

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

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

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

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

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

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

$$= O(n)$$

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

$$\int_{0}^{1} x^n dx = \frac{1}{n+1} x^{n+1} \Big|_0^1 = \frac{1}{n+1}$$

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

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

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

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

$$T(n) = \dots$$

↓
K

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

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

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

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

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

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

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

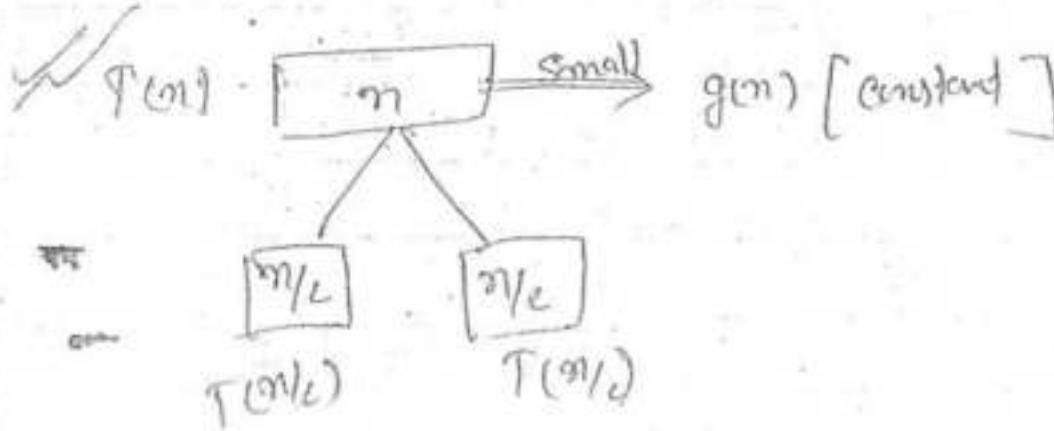
$$\frac{1}{2^k} \log \frac{n}{2} = 1$$

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

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

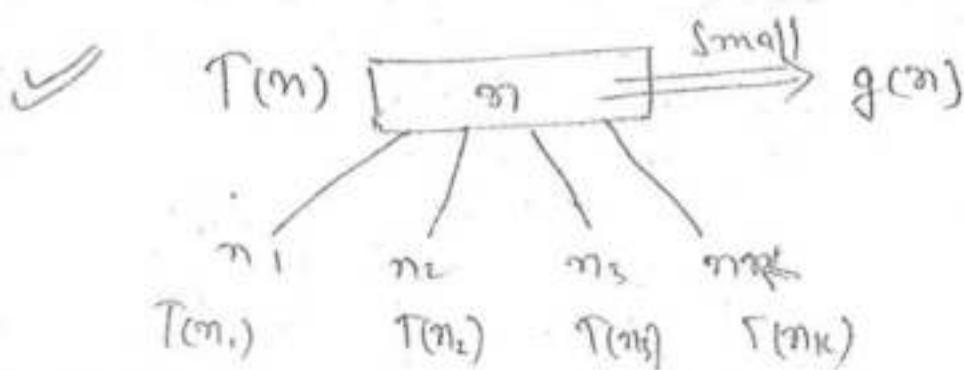
$$\log(\log n) = K$$

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



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

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



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

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 DACDAC(P, Q) {

{
if ($\text{small}(P, Q)$)

return ($\text{solution}(P, Q)$)

$m = \text{Divide}(p, q)$
 Returns
 Comk. bnd $(\text{DAC}(p, m), \text{DAC}(m+1, q))$
 }
 }.

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~~

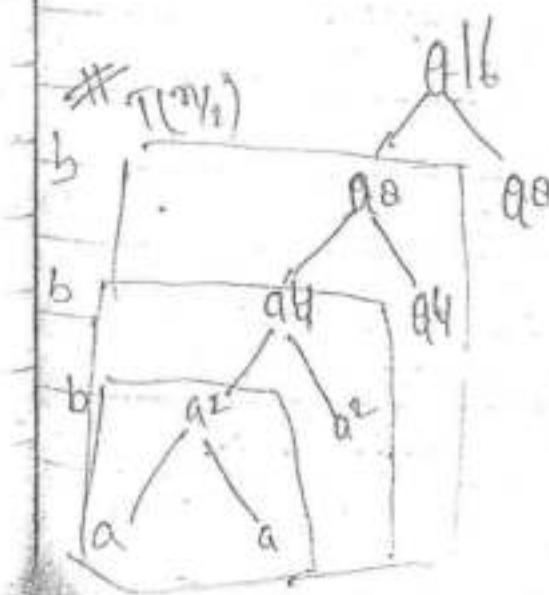
Power (a, n)

```
{
    int f=1, i;
    for (i=1, i<=n, i++) {
        f = f*a;
    }
    return (f);
}
```

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

$$\Downarrow$$

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



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);
    }
}
  
```

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

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

\downarrow $k \in \mathbb{N}$

$$= \Theta(1) + c \cdot \log n$$

$$\Rightarrow \Theta(\log n)$$

KC
 ↓
 $c \cdot \log n$
 ↓
 $\Theta(\log n)$

↳ 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]
}

```

ComplexityBest Case: $\Theta(n-1)$ Time Complexity
 $\Theta(n)$ Worst case: $\Theta(n^2)$

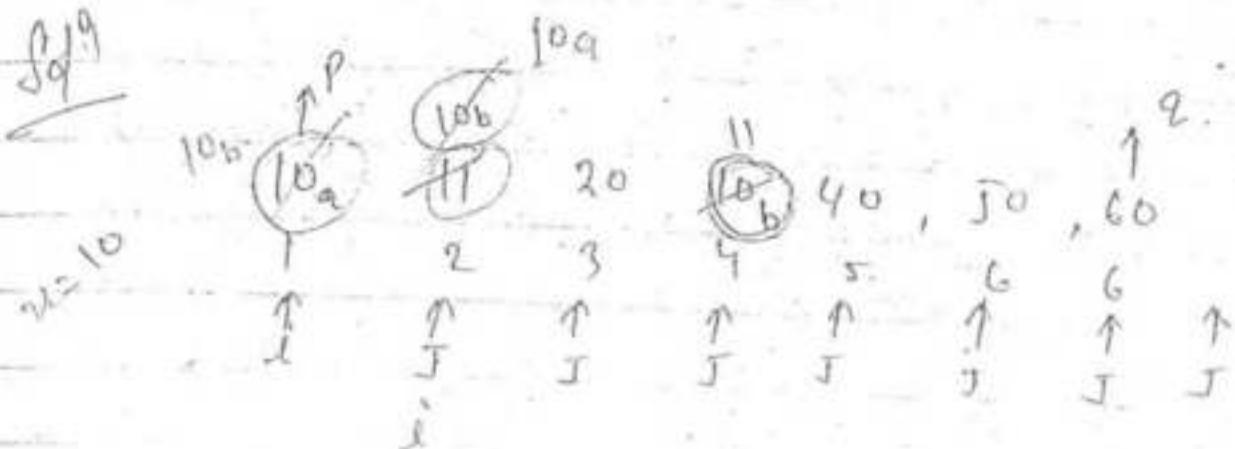
Avg case: $\frac{n-1}{2} + \left(\frac{n-1}{2}\right) * 2$

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

$= 1.5(n-1)$

Q Apply partition algo on following algo.

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



come $(10_b) 10_a (20, 11, 40, 50, 60)$

stable sorting Tech

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

Stable sorting technique,

\Downarrow
Q.S is not stable sorting.

Selection procedure

i/p: An array of m 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 = $10 \quad K=4$ | $11 \quad K=5$ | $31 \quad K=9$

A. ① $1, 2, 3, \textcircled{10}, 11, 20, 21, 30, 31 \Rightarrow O(n \log n)$

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

$O(n \log n)$

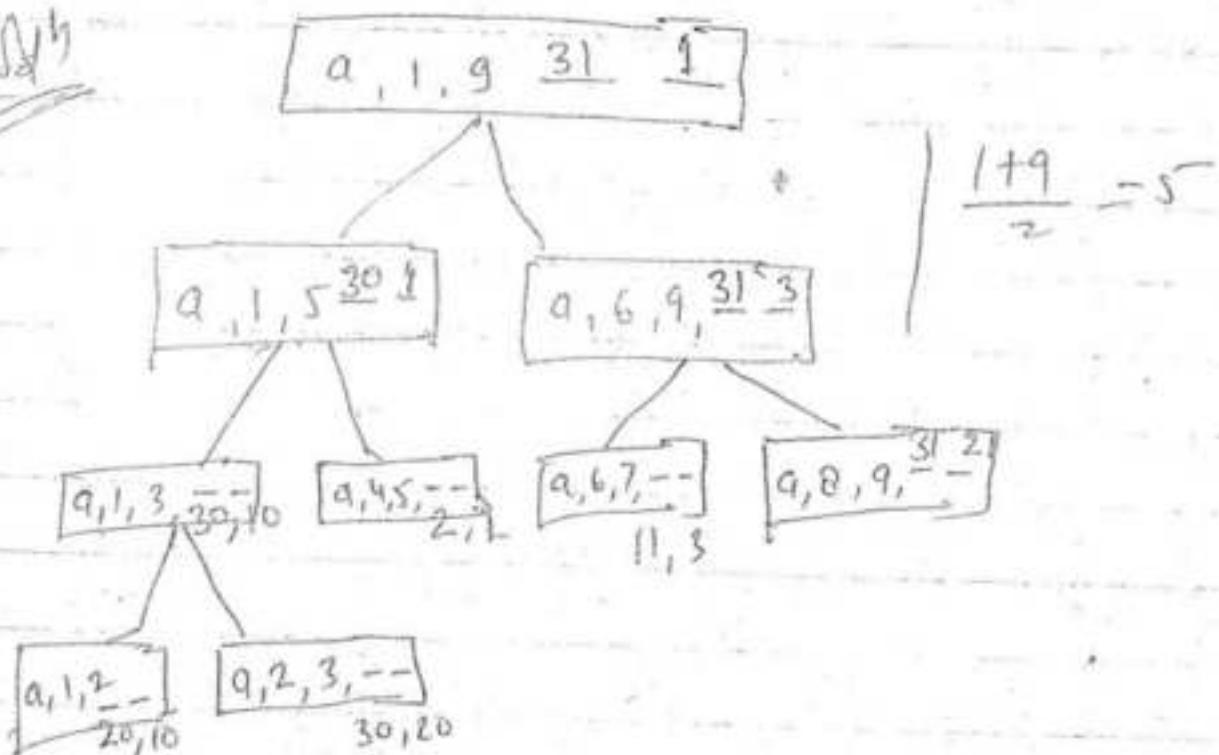
Using DAC Max & min.

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

Max : 51

min : 1

~~SAh~~



DACmaxmin (a, i, j, \max, \min)

Small

{ int mid,

if ($i == j$)

~~return~~ { $\max = \min = a[i]$;

else 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 = $(i+j)/2$ \Rightarrow C-constant
 $\Rightarrow T(n/2)$

DACmaxmin (a, i, mid, \max, \min)

DACmaxmin ($a, mid+1, j, \max, \min$)

$\Rightarrow T(n/2)$

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.

$$T(n) = \begin{cases} 0 & \text{if } n=1 \\ 1 & \text{if } n=2 \\ T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + 2 & \text{if } n>2 \\ & \quad \downarrow \\ & 2T\left(\frac{n}{2}\right) + 2 \end{cases}$$

small

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

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

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

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

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

↓

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

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

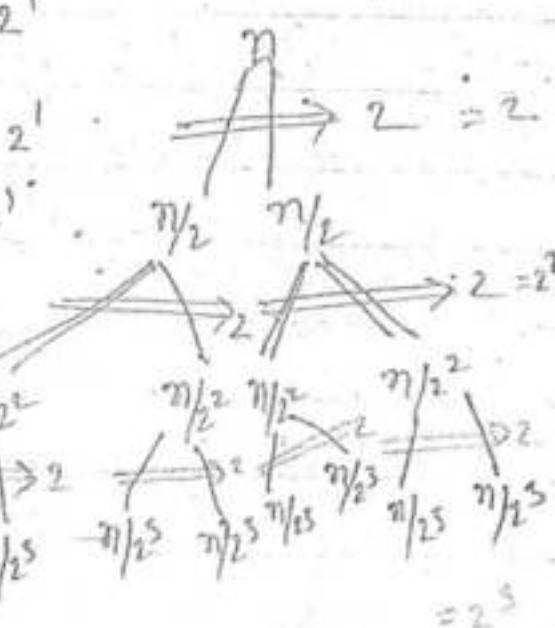
$$= \frac{n}{2} + n - 2$$

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

$$= 1.5n - 2$$

$$= O(n)$$

\Downarrow It is best, worst & avg case.



Note - Using DAC and without DAC Taking same time . So we have to consider space more, $\Theta(n^2)$

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 (DAAG) Directed acyclic graph

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

- (iv) Connected

atmost 2



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

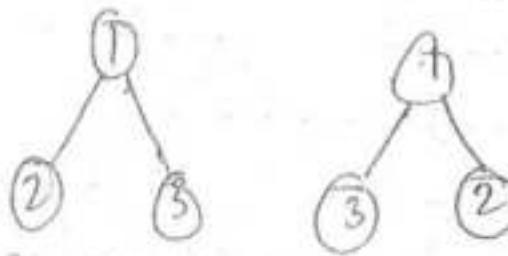
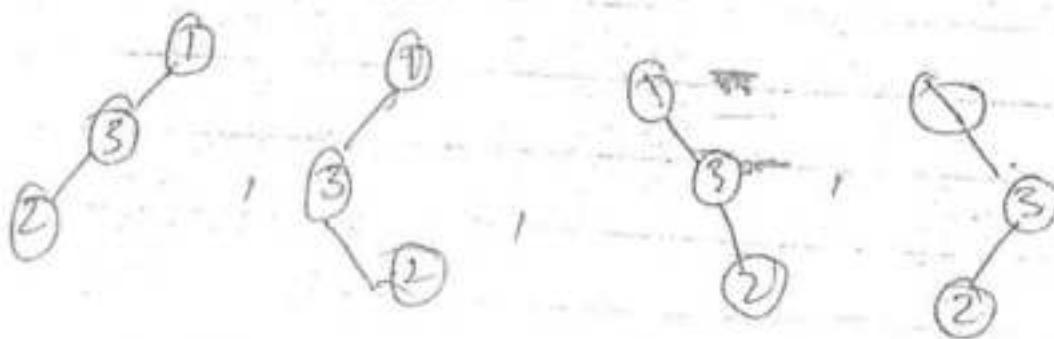
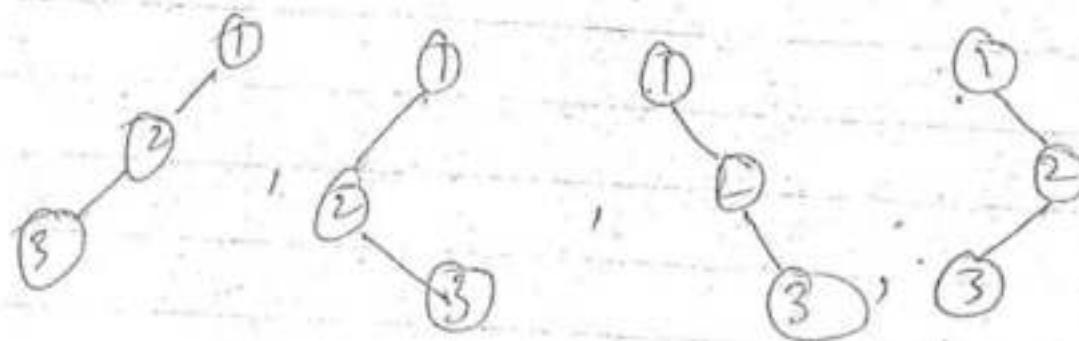
Q. Find no of binary trees with $n=3$ (1, 2, 5)

Solⁿ $n=1$
 ① \Rightarrow ①

$n=2$ (1, 2)



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



$$\begin{aligned} 1 & - 10 \\ 2 & - 10 \\ 3 & - 30 \end{aligned}$$

$$\Rightarrow \underline{\underline{30}}$$

the no of binary trees with n nodes

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

By formula

if $n=3$

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

$$= \underline{\underline{30}}$$

Binary Search Tree

(Repetition not allowed)

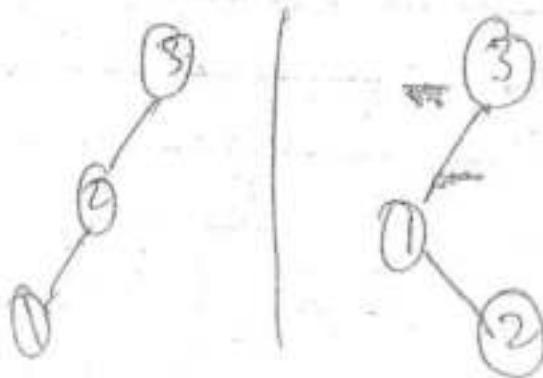
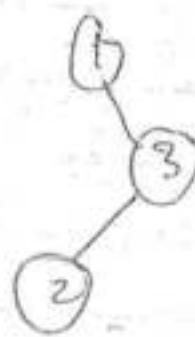
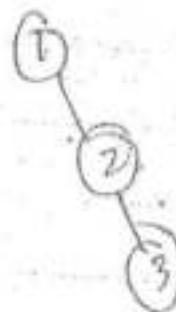
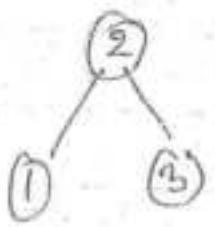
(a) at every node Root is greater than all the elements present left sub tree (LST)

(b) At every node Root is smaller than all the elements present in RST.

eg $n=2 (1,2)$



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



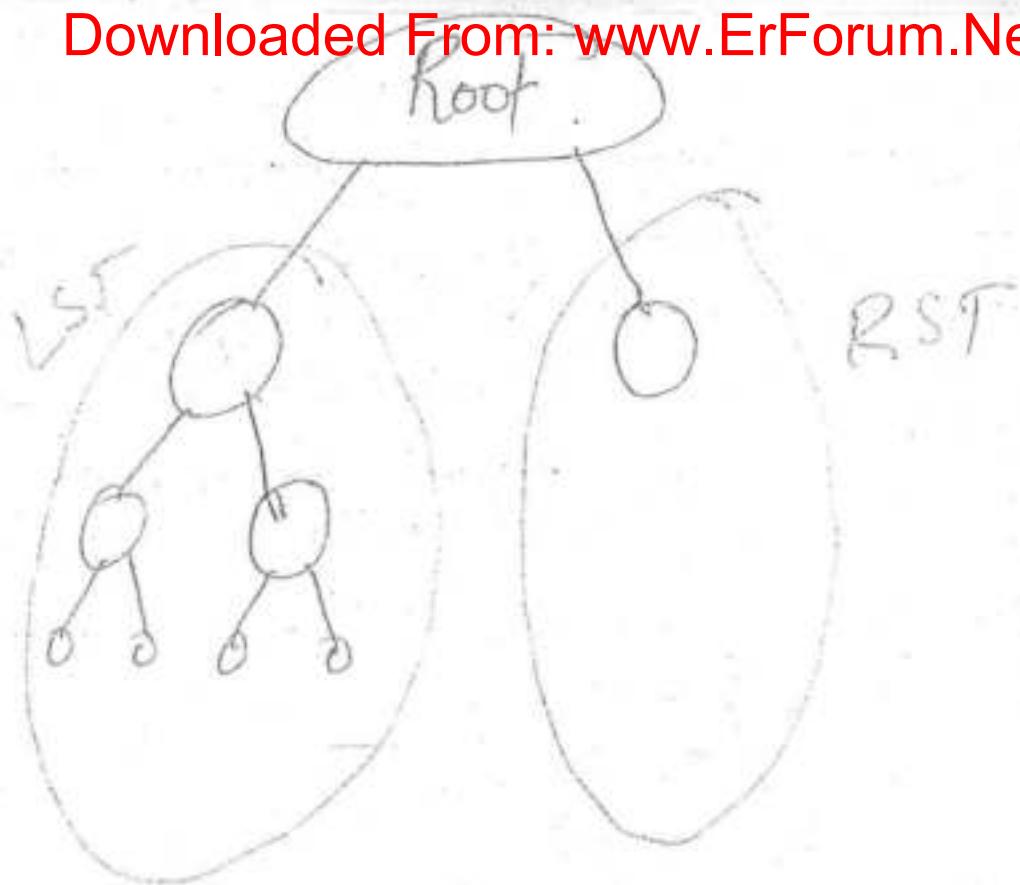
$$\text{binary Search Tree} = \frac{(2^n)n}{n+1}$$

$$n=1 \Rightarrow 1$$

$$n=2 \Rightarrow 2$$

$$n=3 \Rightarrow 5$$

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



!
Simplifying

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 $A[10, 20, 30, 1, 2, 3, 41, 21, 31] \quad u=11$

o/p : 7 | $u=11$ ✓

9 | $u=31$ ✓

84
out : 1 | $u=10$ ✓

-1 | $u=90$ ✓

Without DAC

Linear Search

linearSearch(a, n, u)

```

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

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

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

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

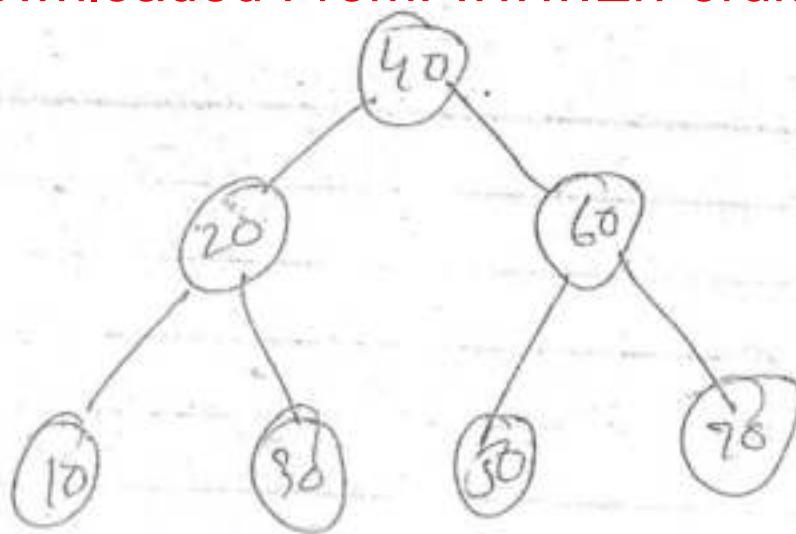
Successful search

Unsuccessful SearchBest case - $\Omega(n)$ Worst - $O(n)$ Avg. " - $O(n)$

$$\frac{1}{n}$$

$$O(n)$$

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)egi/p: $A[10, 20, 30, 40, 50, 60, 70]$ $x=40$
 $1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7$ sol.



$$\log_2 8 = 3 \text{ Comprasion.}$$

Binary Search (a, i, j, u)

```

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

Binary

```

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)
    }
}

```

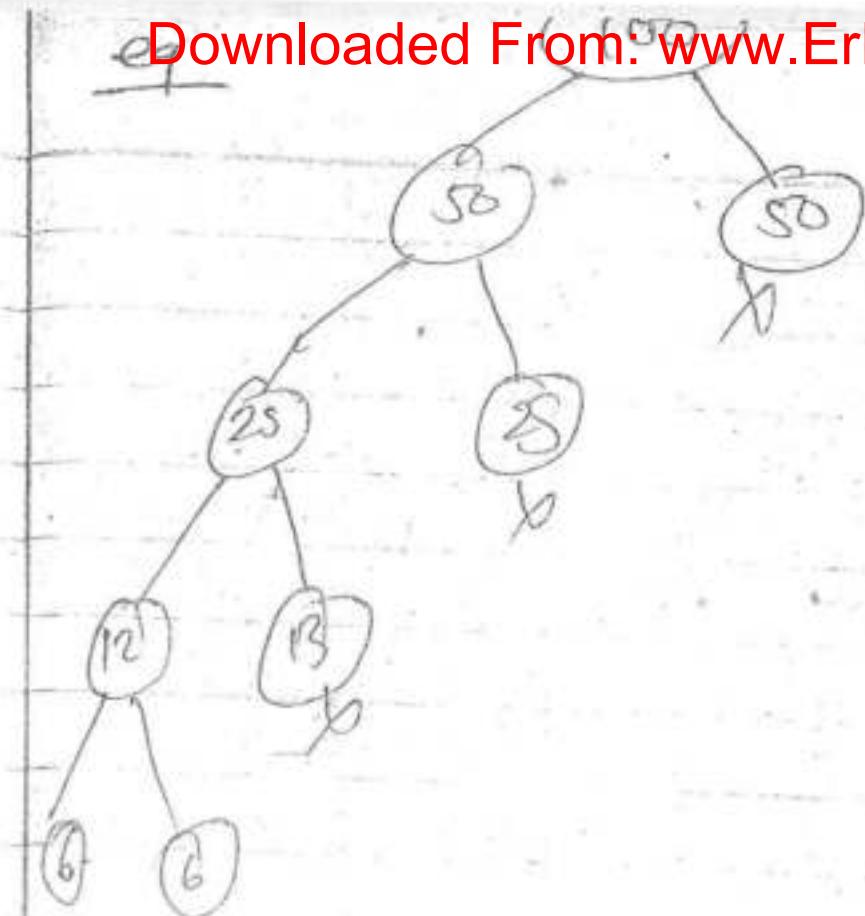
{ }

Recurrence Relation

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

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

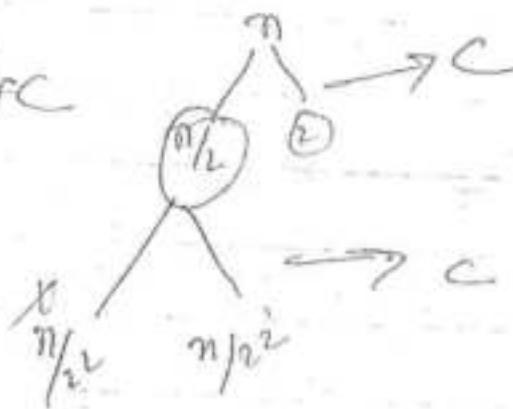
\downarrow
 $O(\log n)$



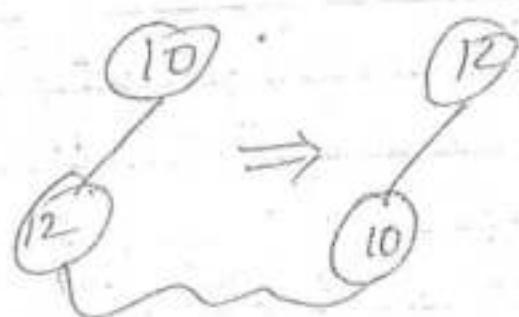
$$T(n) = T(n/2) + C$$

$$\approx T(n/2) + C \quad [fc]$$

\downarrow

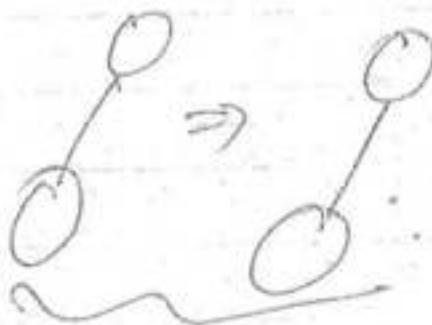


Unlabeled Binary tree



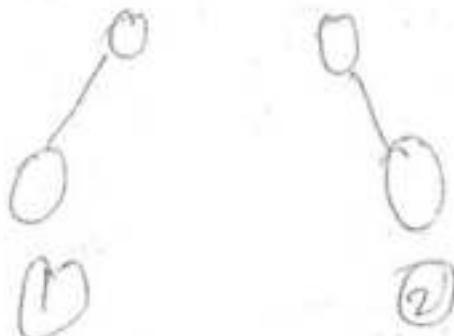
2 - different unlabeled tree

Labeled B.P

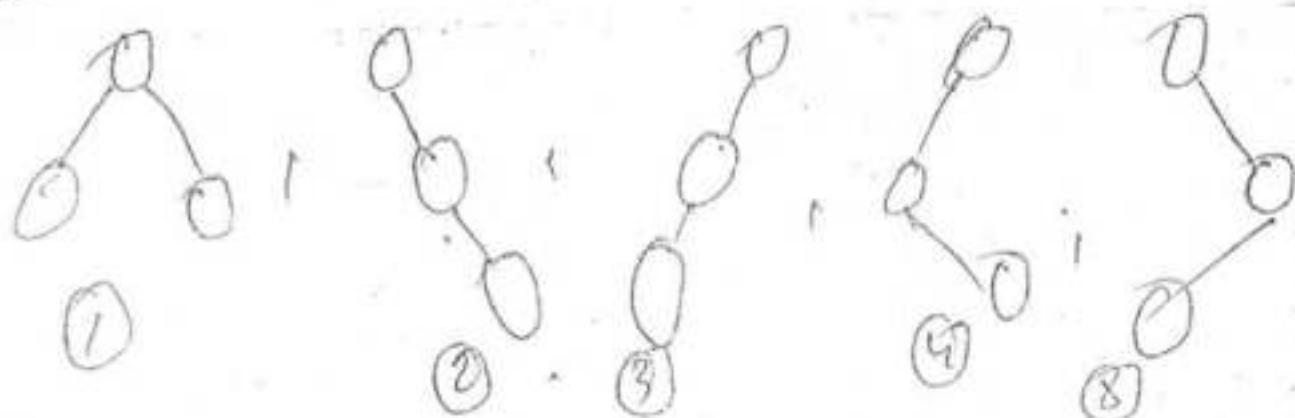


both are same.

Q) $n=2$ has many different unlabeled B.P.



$n=5$



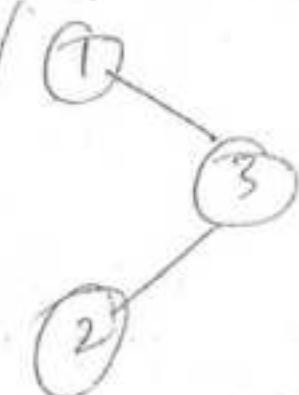
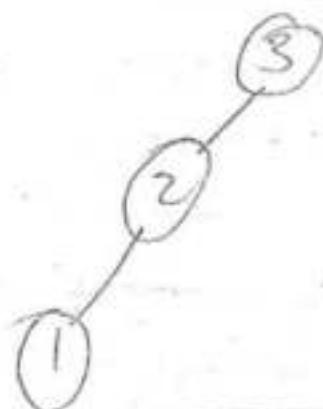
=5

Q. 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) $\frac{(2n)!}{n+1}$

Sol:

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



A binary search tree is used to locate no 43, which one of the following sequences are possible.

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

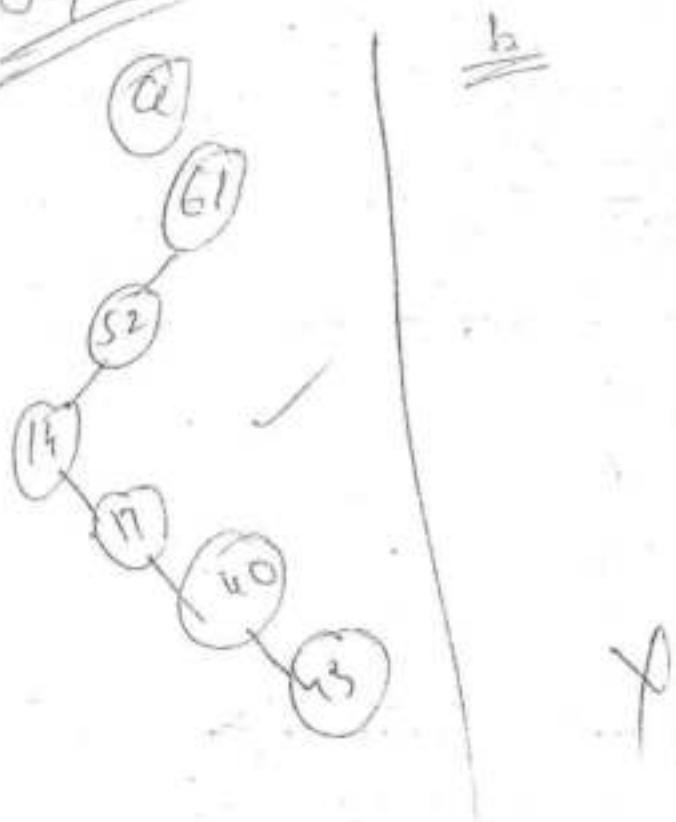
b) 2, 3, 50, 48, 60, 43

c) 10, 65, 31, 48, 37, 43

d) 31, 61, 52, 14, 41, 43

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

Soln.



(Linear
Binary Search)

$\Omega(1)$ - Best case

$O(n)$ - Worst case

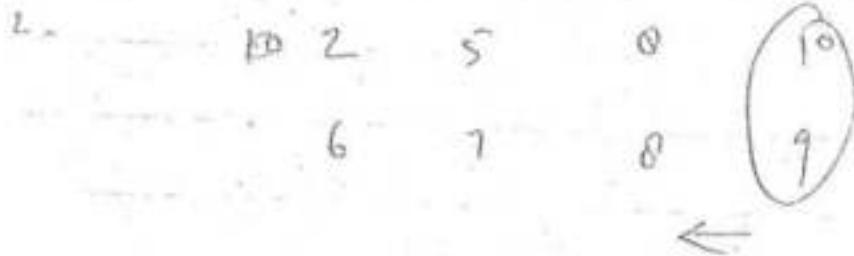
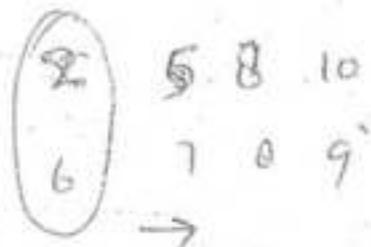
$O(n)$ - Avg case

Binary Search

means always goes left or right.

If Postion is greater than element we go right side.
and if it 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.

Ex

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 x to find x
 $\Rightarrow \mathcal{O}(n^2)$

(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

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

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

$$n = 1000$$

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

$$n = 1000$$

$$n \log n = 1000 \times 3 = 3000$$

Ans

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

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

flag:

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

②

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

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

flag [a[i]] = 1

else
Print + (a[i]) } $\Rightarrow O(n)$

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

#

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

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

eg

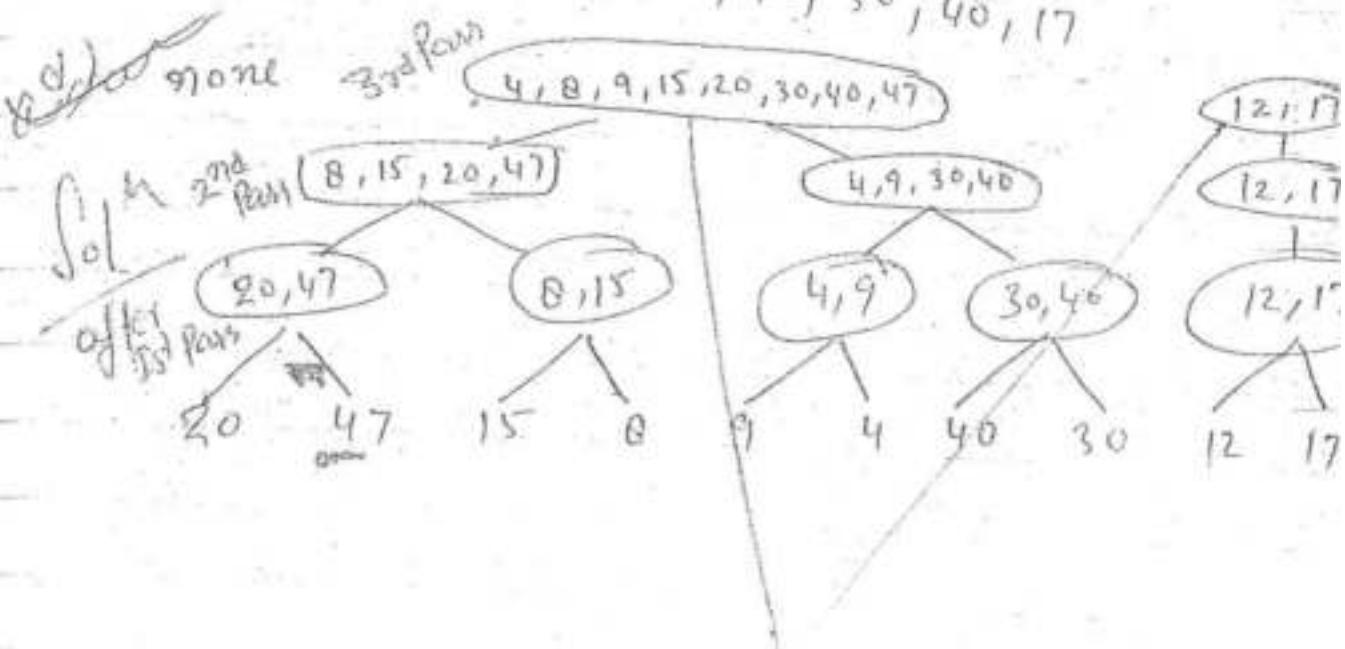
i/p B : b₁, b₂, b₃, b₄, ..., b_n
 N : n₁, n₂, n₃, n₄, ..., n_m

(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, 8, 9, 12, 30, 40, 17



after fourth round

$$\text{order} = \log_2 16 = 4$$

Quick Sort (In-place)

- (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.

\hookrightarrow Partial Divide and Conquer approach.

Partition (a, p, q)

```

    {
        n = a[p];
        i = p;
        for (j = p+1; j <= n; j++)
            {
                if (a[j] <= n)
                    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 lead. = 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.

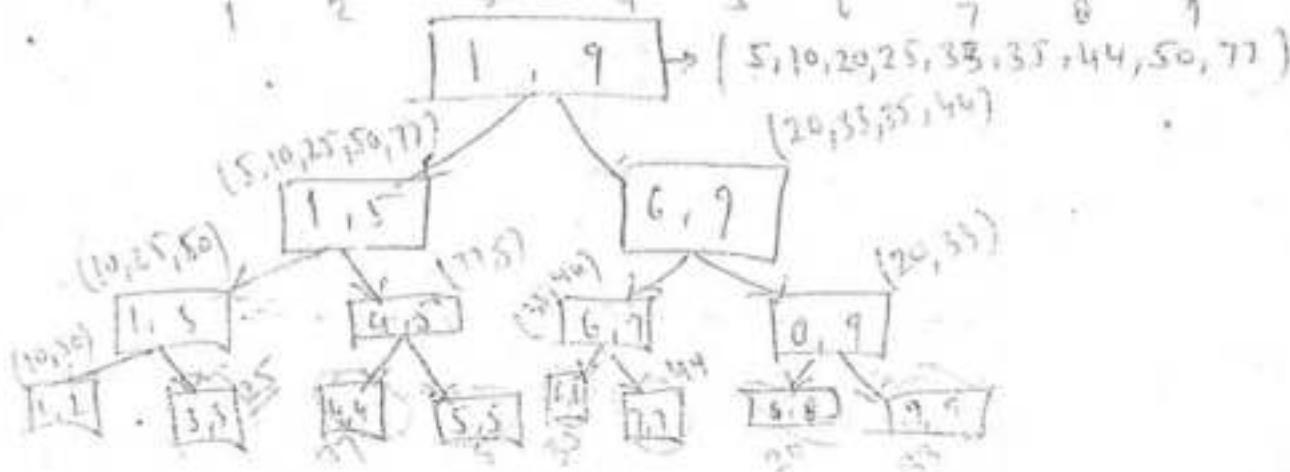
~~Ex:~~ 100, 70, 50, 40, 30, 10

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

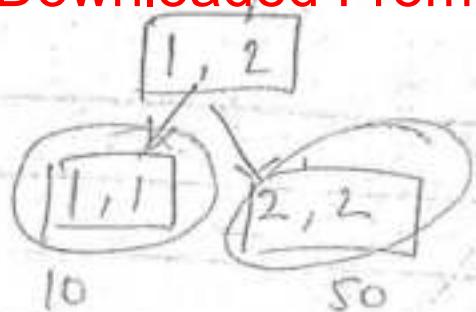
Merge Sort

Divide and Conquer. Merge sort.

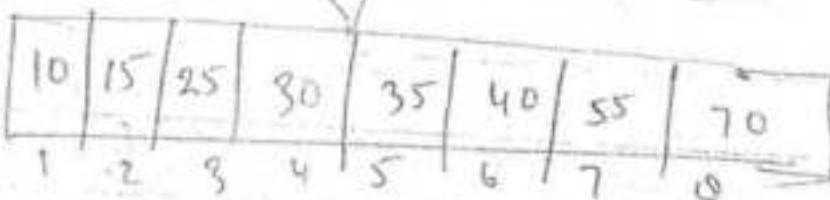
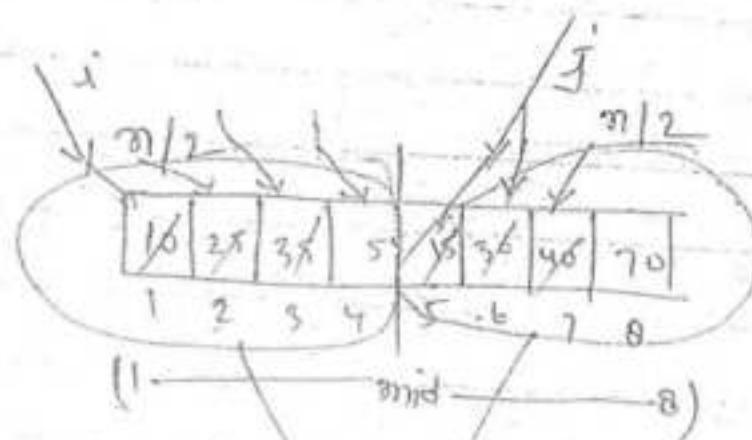
i/p 10, 50, 25, 77, 5, 35, 44, 20, 33



lement
next



~~eq~~ - eq



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

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

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

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

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

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

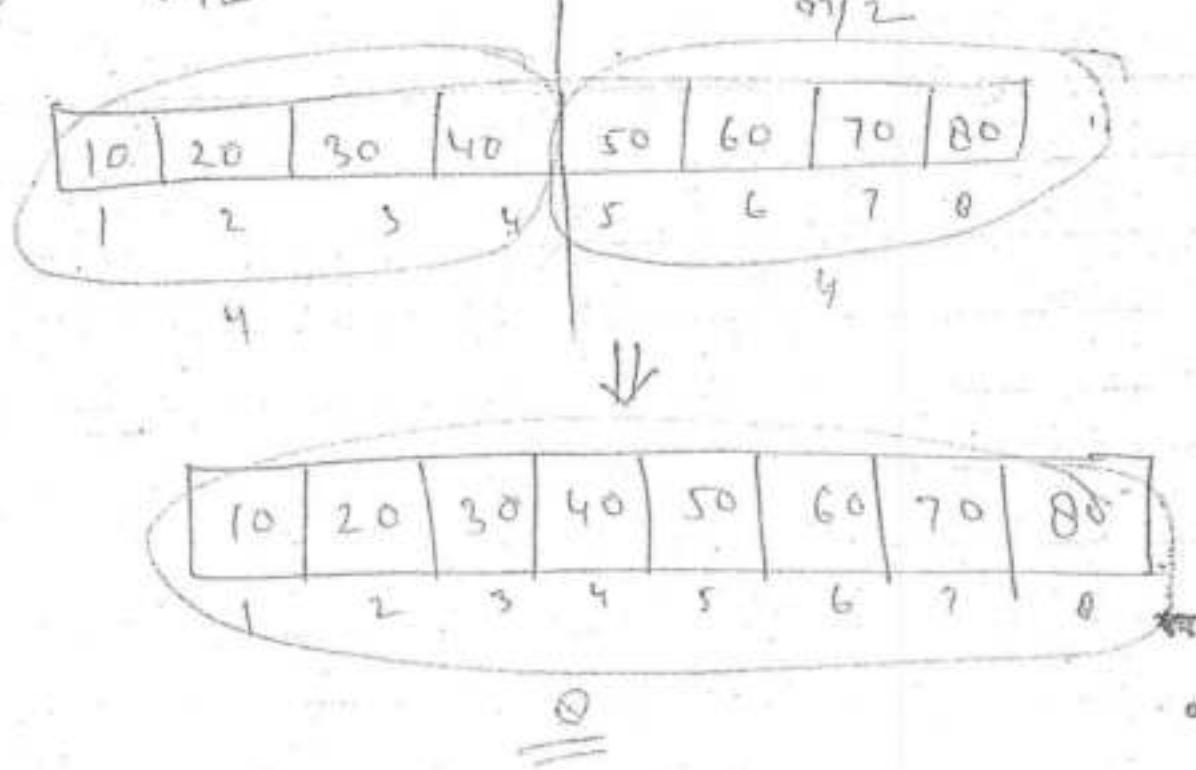
$$(55, 70) \Rightarrow 55$$

$$70 \Rightarrow 70$$

$$\text{total Qemponen} = \frac{4+4-1}{2} = 7$$

$$= \frac{n_1 + n_2 - 1}{2}$$

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



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

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

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

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

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

Worst Case $\Rightarrow n/2 + nh^{-1} \Rightarrow O(n)$

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

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) $\rightarrow T^{(m|l)}$

\Rightarrow Mergesort($a, mid+1, j$) $\rightarrow T^{(n|m)}$

Merge(a, i, mid, j)

return (a)

[1, 8]
↓

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

(1, 4)

(4, 8)

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

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

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

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

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

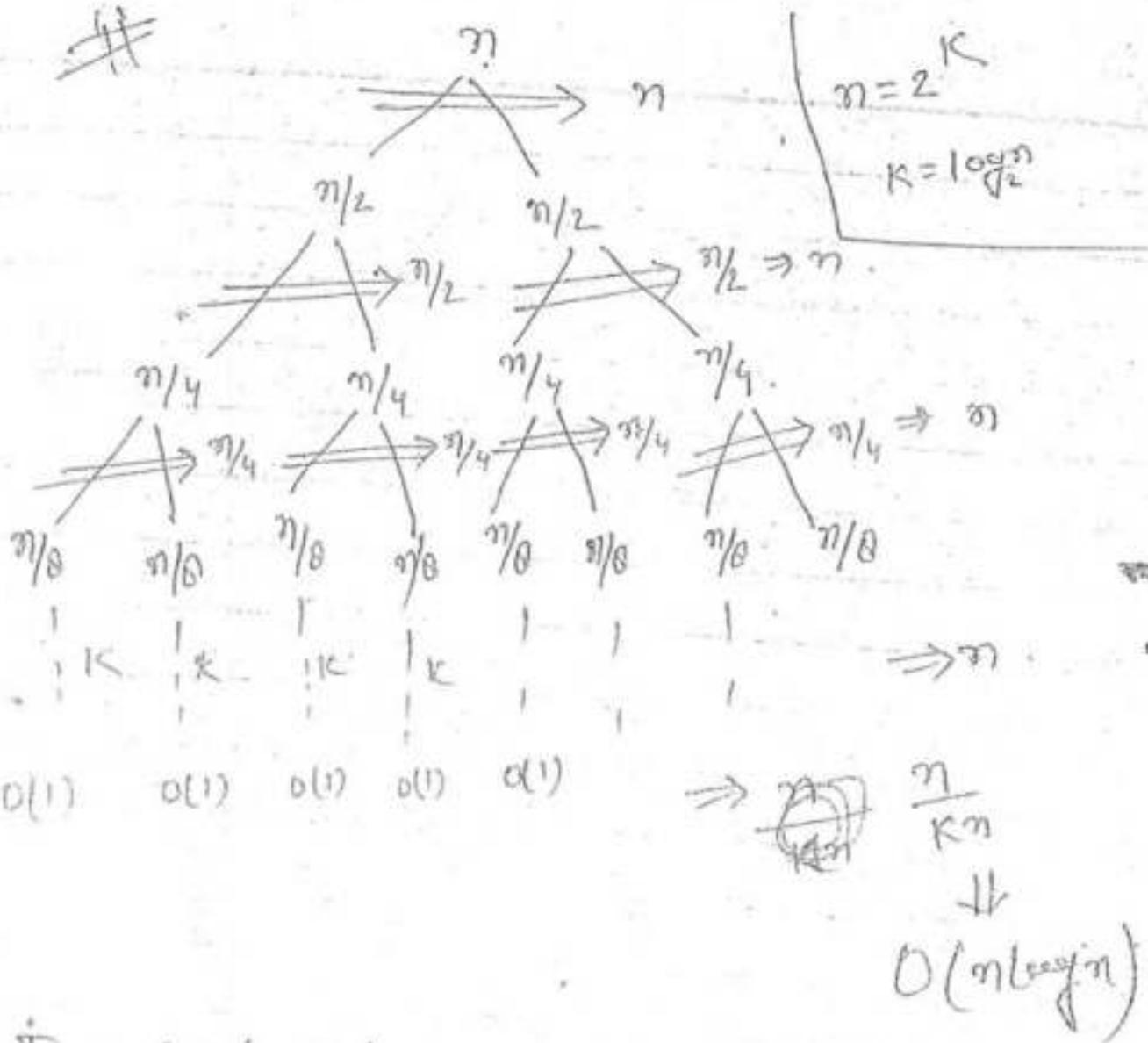
$$= 2^3T(\frac{n}{2^3}) + 3n$$

\vdots
K

$$= 2^K T\left(\frac{n}{2^K}\right) + Kn \rightarrow n + \log_2 n \cdot n$$

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

(1)



Time Complexity of Merge sort.

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

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

In Best case = Worst case.

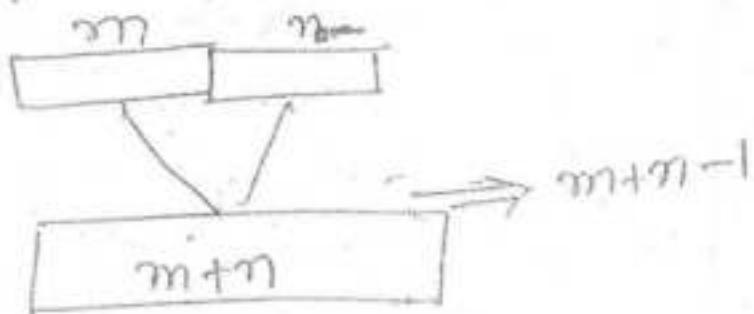
① Merge Sort is preferable for larger size arrays
for smaller size, smaller size can be
preferable by insertion sort

(2)

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

(3) ~~Time~~

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



(P)

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

20, 47, 15, 0, 9, 4, 40, 50, 12, 17

then

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

- (A) 0, 9, 15, 20, 47, 4, 12, 17, 30, 40

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

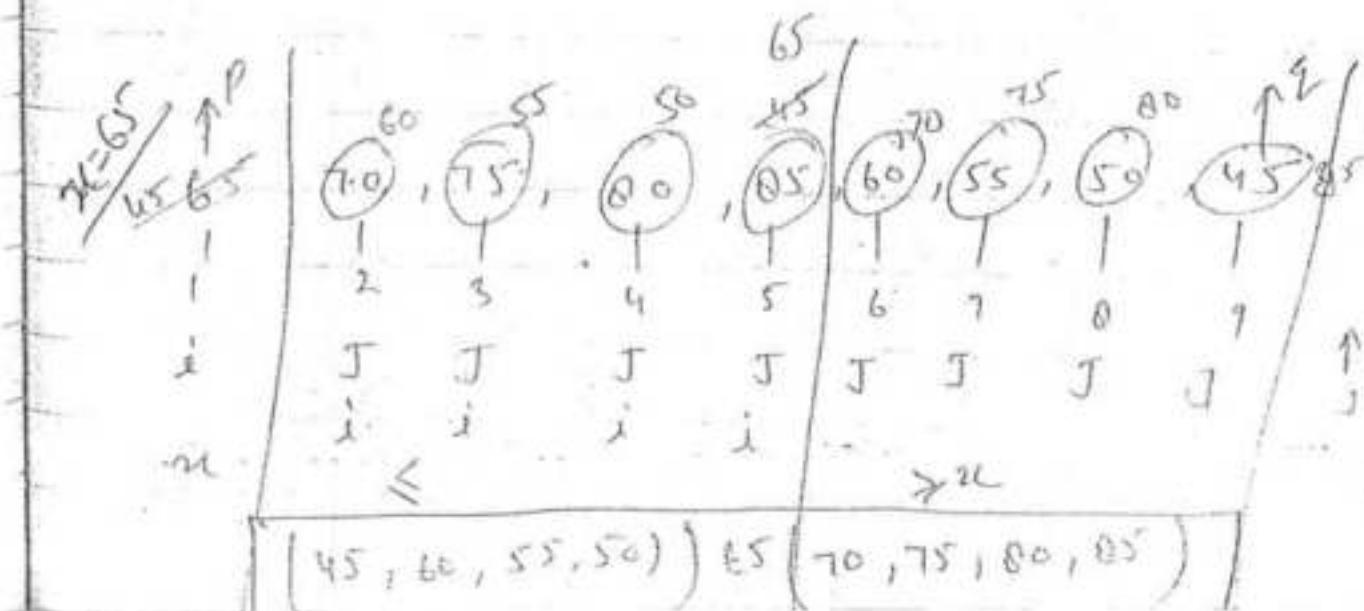
return(i);

}

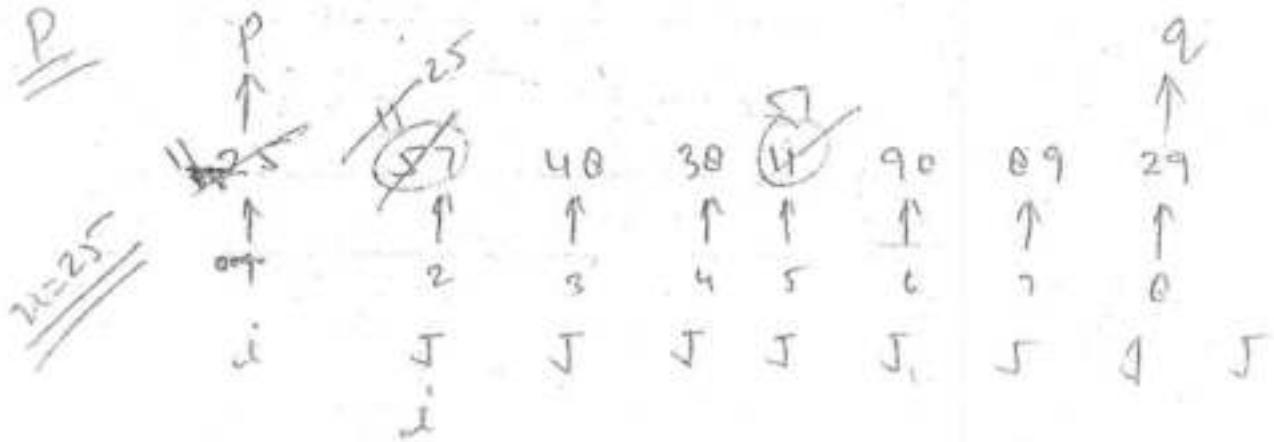


(2 5 3) 6 (8 13 10 11)

eg Apply partition algo to the following input



\leq_u	\geq_u	\neq_u
i	J	

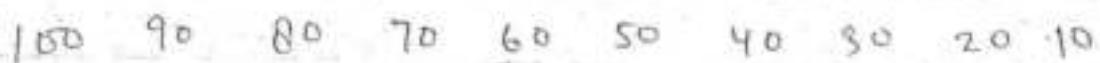


(ii) $25(40, 38, 57, 90, 89, 2^9)$

P(1)

10 20 30 40 50 60 70 80 90 100

P(2) -

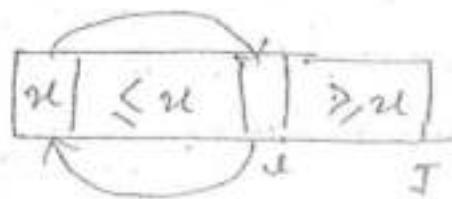


三

200

100

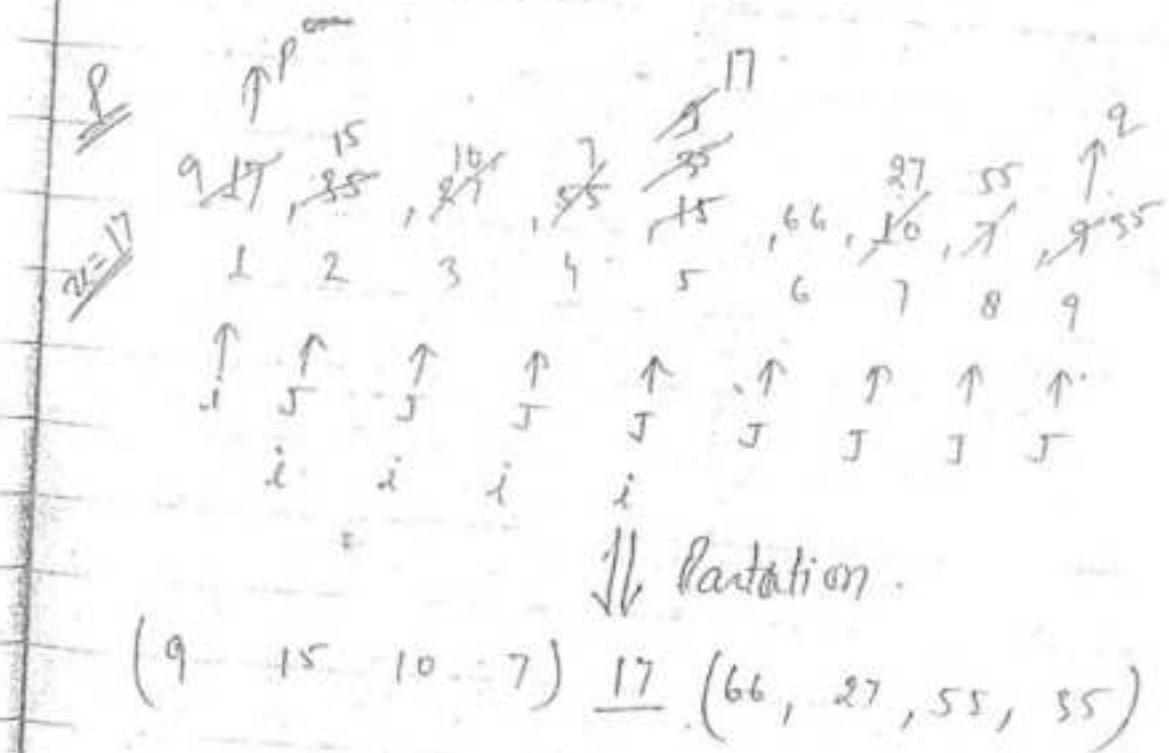
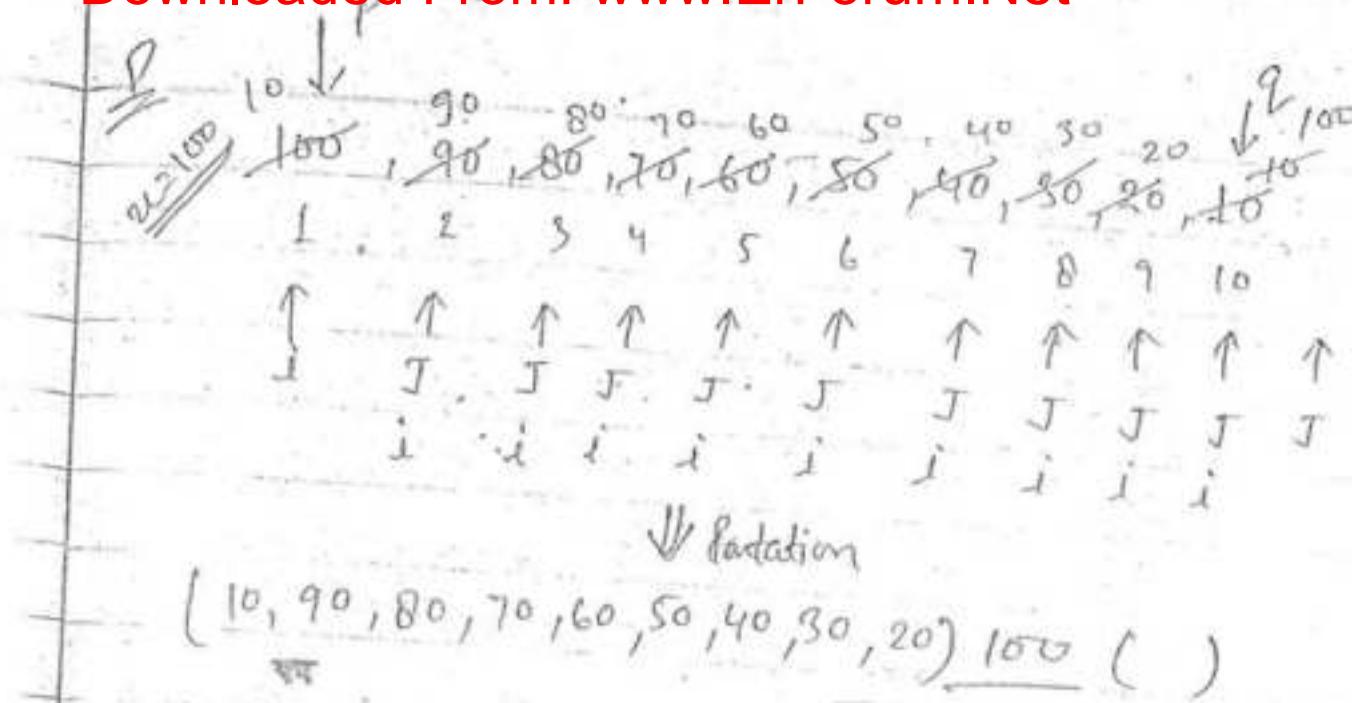
Partition Algo



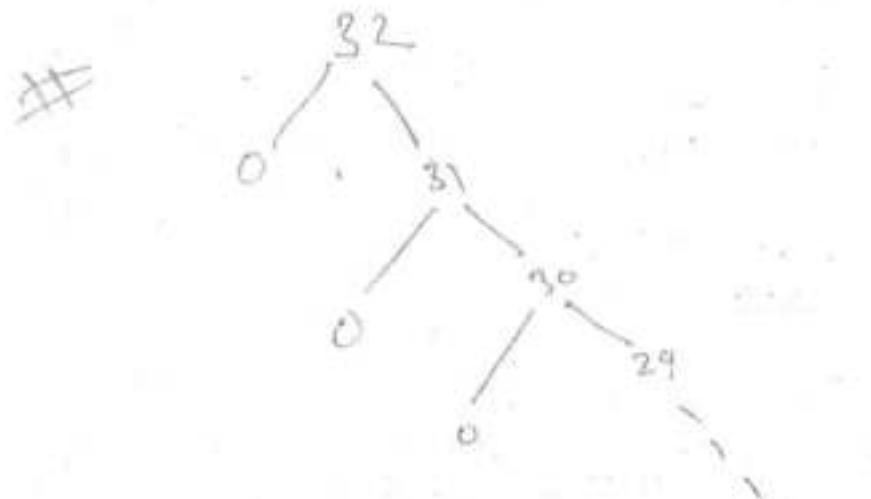
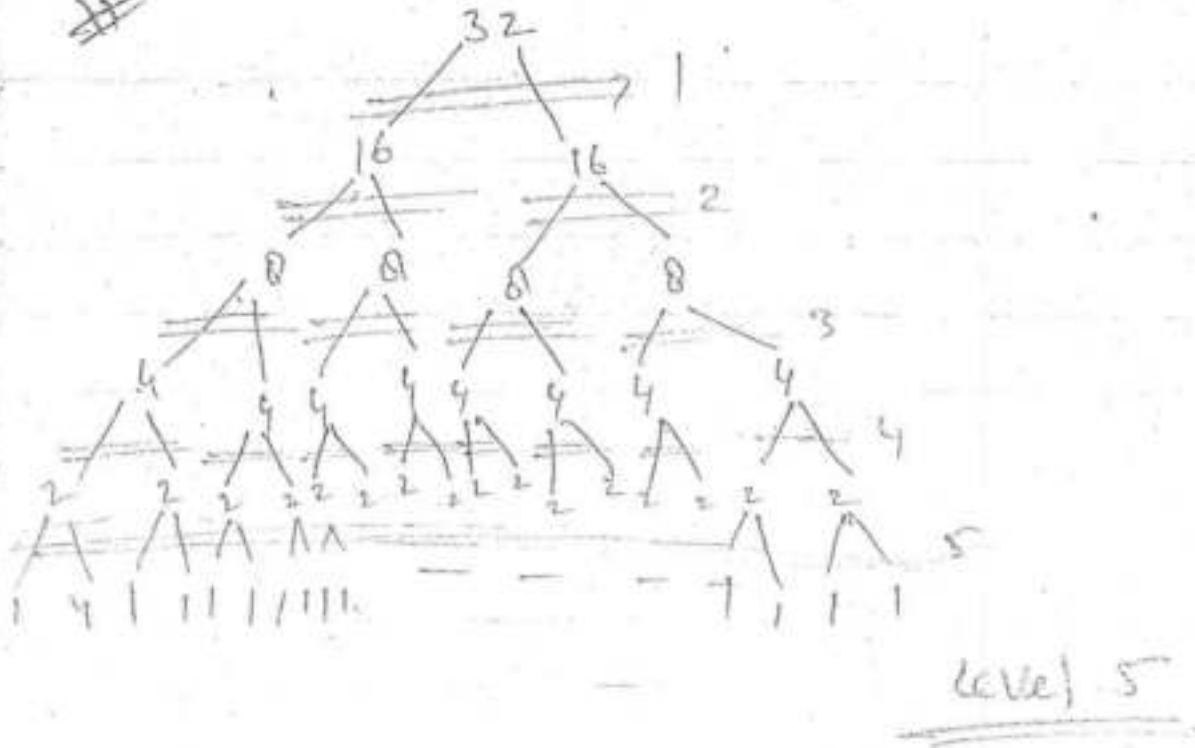
$i \leftarrow 1$	$j \leftarrow n$	\uparrow^P	\uparrow^Q
10	20	30	40
50	60	70	80
90	100		

\downarrow Partition

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



No of Level is $\log_2 n$



$$\text{No of level} = n$$



Quicksort (a, p, q)

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

{
 $m = \text{Partition}(a, p, q)$ }
} $\Rightarrow n$

Quicksort ($a, p, m-1$) $\Rightarrow \frac{n}{2}$
Quicksort ($a, m+1, q$) $\Rightarrow \frac{n}{2}$

return (a)

{
}
}

Best Case T.C of Quicksort Algo.

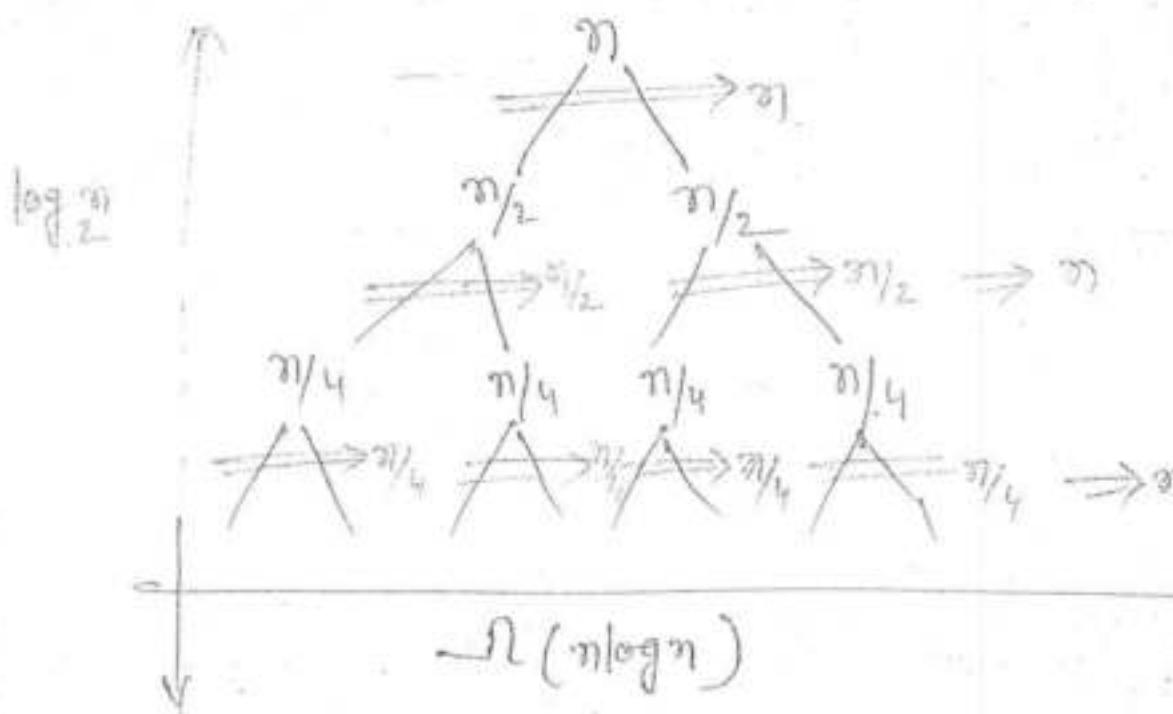
n elements.



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

\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

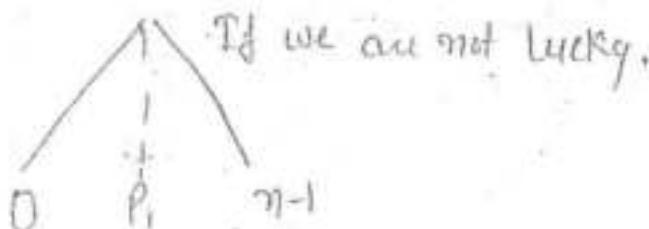
() 50 ()

#

50 40 30 20 10

↓ Partition \Rightarrow 5

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

Worst Case n - elements \Downarrow Partition $\Rightarrow n$ 

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

Average Case

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

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

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

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

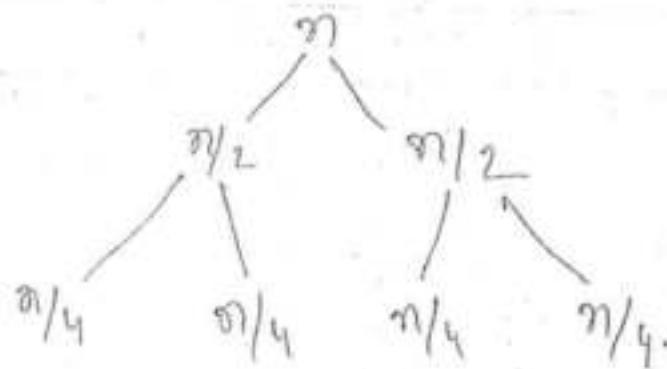
$$\Downarrow \\ = 2 \left[L(n/2-1) + n/2 \right] + n \\ = 2L(n/2) + O(n)$$

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

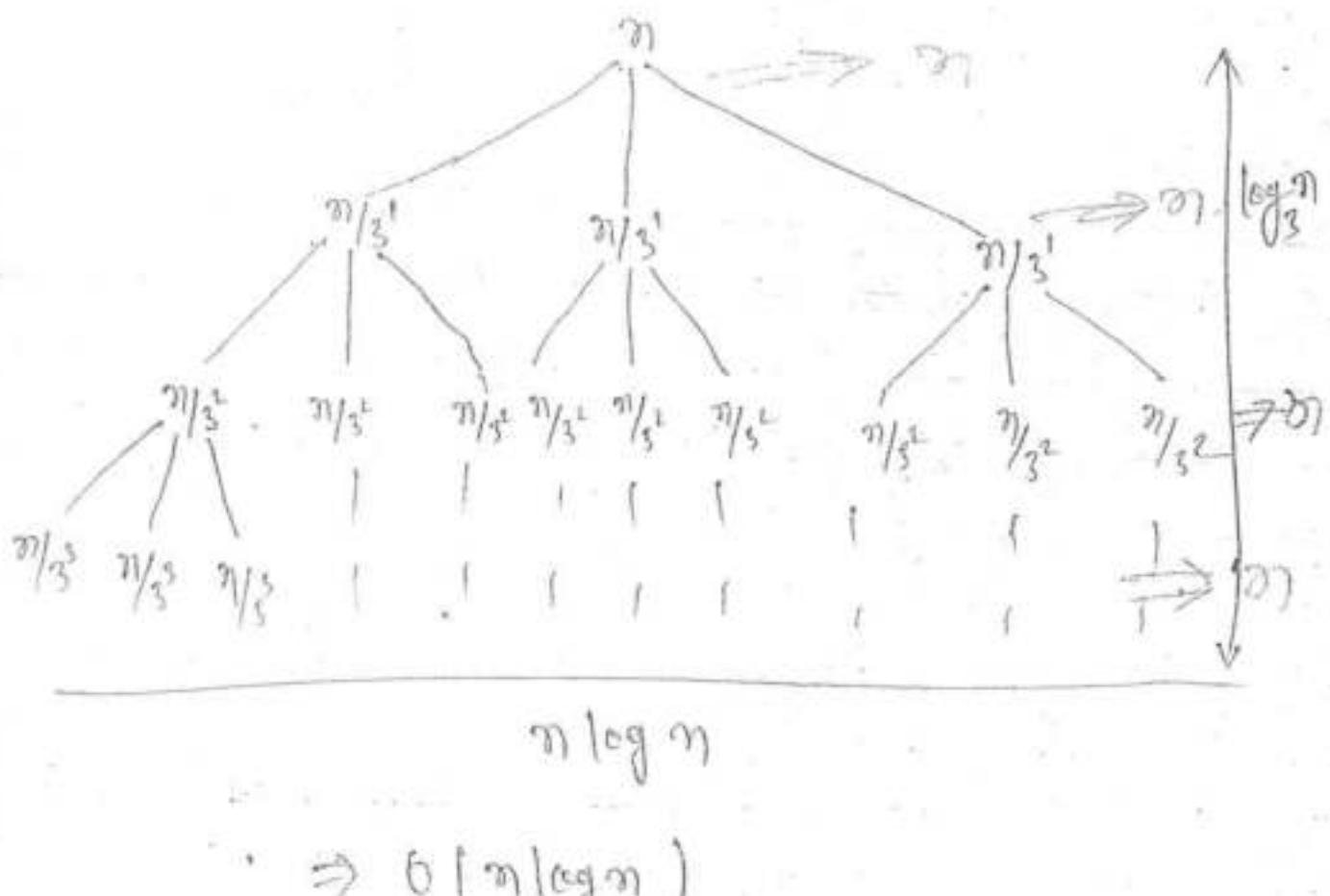
Average Case \Rightarrow $O(n \log n)$

Recursive Tree Method

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



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

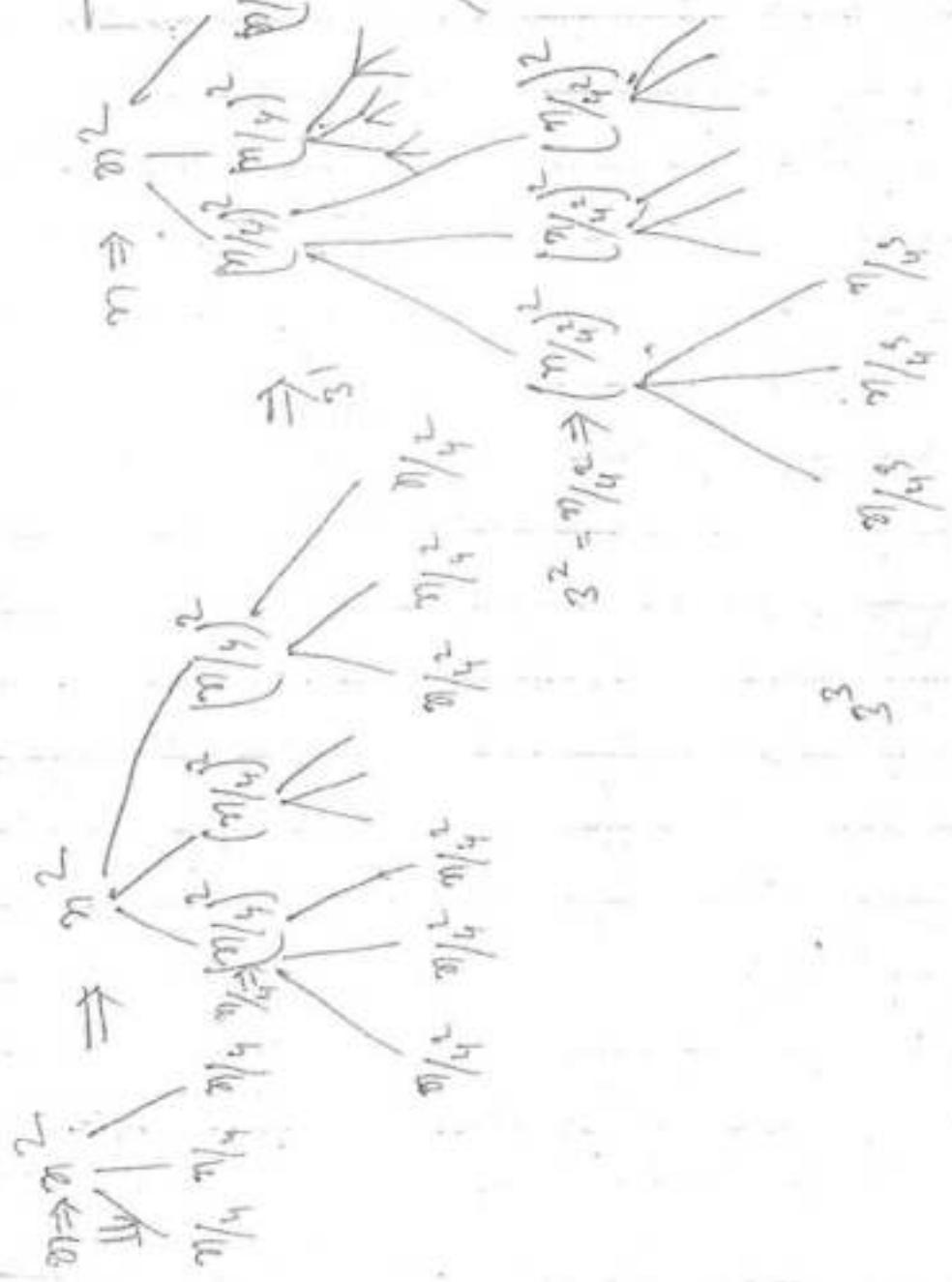


$$\Rightarrow O(n \log n)$$

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

$$\Rightarrow 27 \times \frac{n^2}{16} = \left(\frac{3}{16}\right)^3 n^2$$

$$\Rightarrow \left(\frac{3}{16}\right)^0 n^2$$



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

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

$$= 3^2 T\left(\frac{n}{4^2}\right) + \frac{3}{16} n^2 \cdot \left(\frac{3}{16}\right)^0 n^2$$

$$= \underbrace{\left(\frac{3}{16}\right)^2 n^2}_{n^2} + \underbrace{\left(\frac{3}{16}\right)^1 n^2}_{n^2} + \underbrace{\left(\frac{3}{16}\right)^0 n^2}_{n^2}$$

$$\Rightarrow n^2 \left[\left(\frac{3}{16}\right)^0 + \left(\frac{3}{16}\right)^1 + \dots + \left(\frac{3}{16}\right)^{\log_4 n} \right]$$

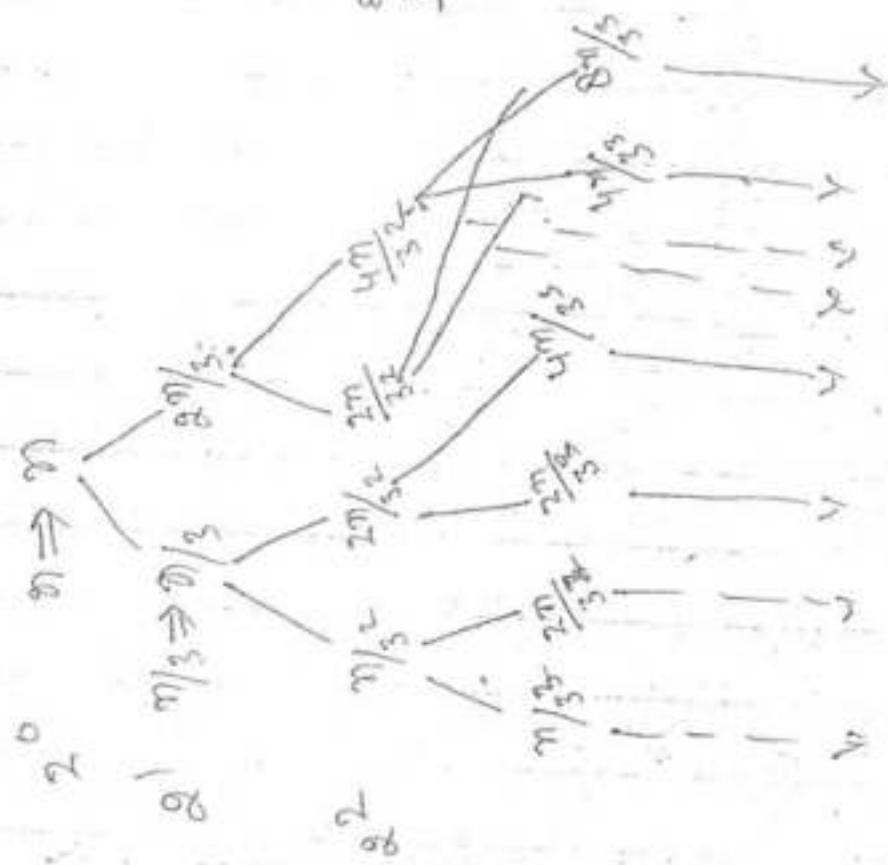
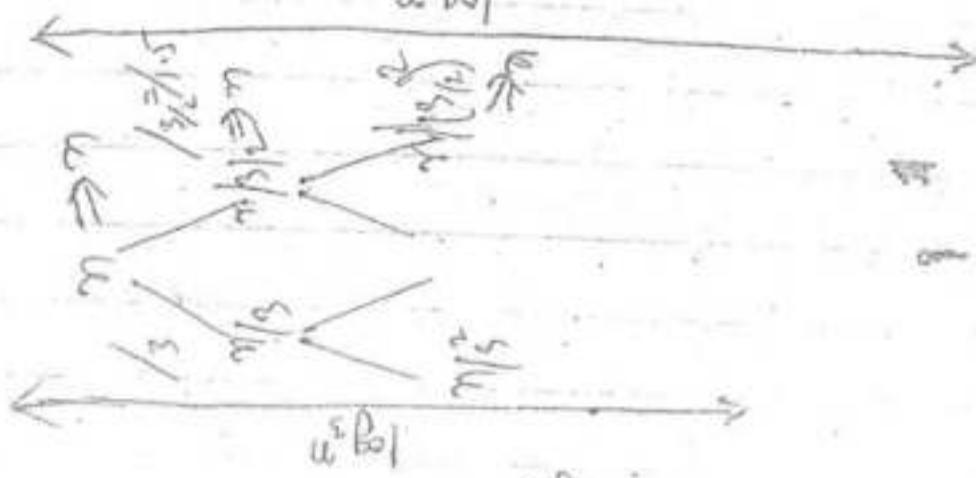
$$\Rightarrow n^2 [?]$$

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

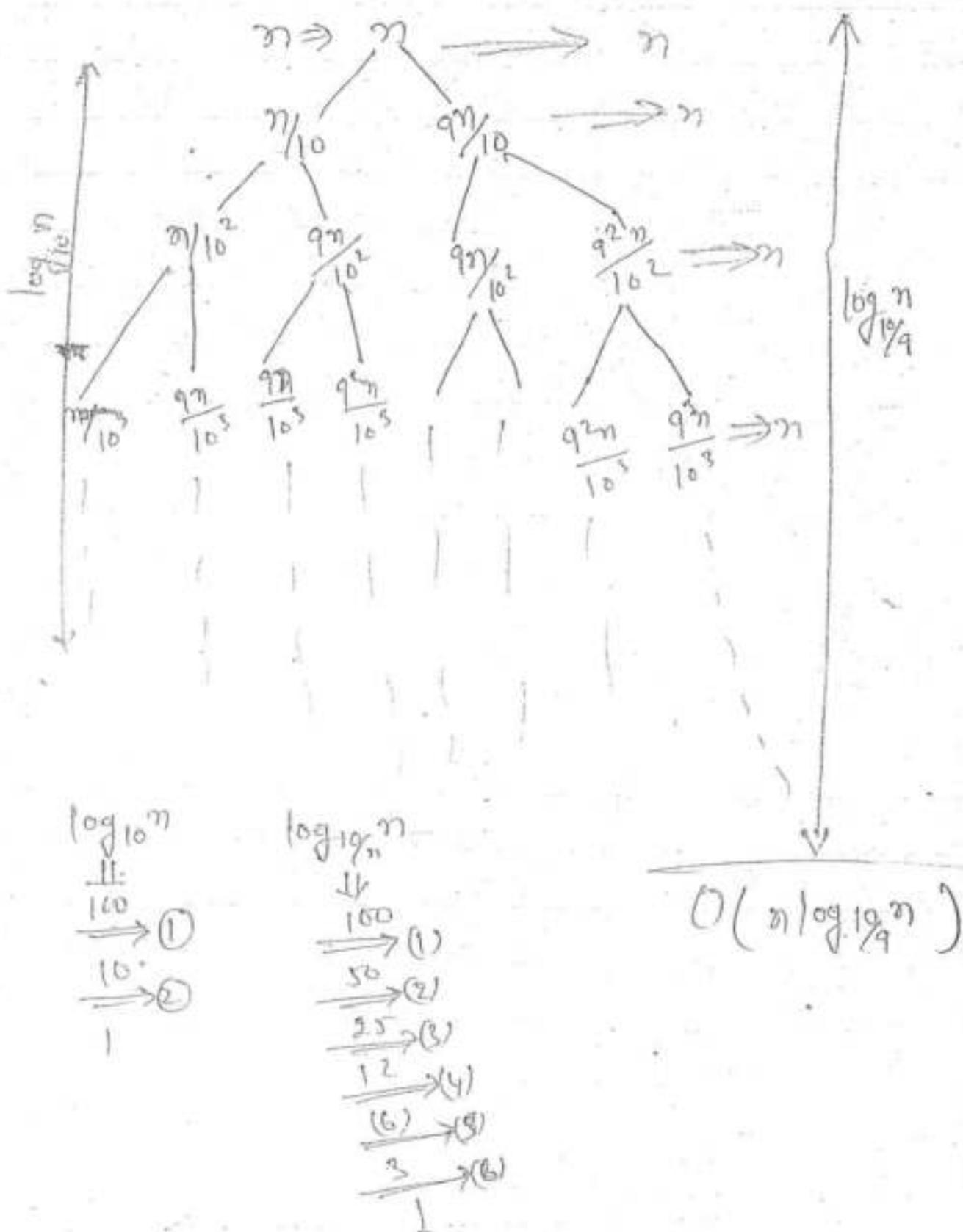
$$\Phi(\gamma) = T(\gamma_3) \cdot F(\gamma_3) + \gamma$$



O(n log m).



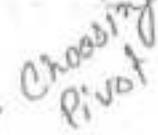
$$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + n$$



Qsort(a, p, q).

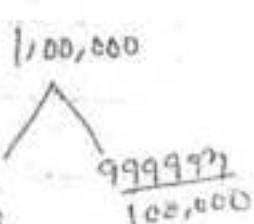
```

    {
        If ( p == q )
        else
            {
                m = Partition( a, p, q )
                Q.s( a, p, m-1 )
                Q.s( a, m+1, q )
            }
    }
  
```

Best Case  

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

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

$$T(n) = O(1) + O(n) + T(6) + T(n-1) \quad \left| \begin{array}{l} \text{eg} \\ \frac{n}{100,000} \\ \downarrow \\ T(n-1) + n \end{array} \right.$$


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 applying 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 $(\frac{n}{4})^{th}$ 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.

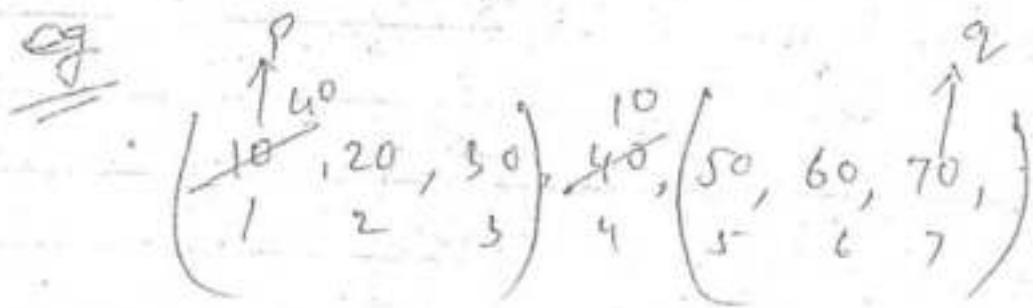
- a) $O(n^2)$
- b) $O(n \log n)$
- c) $O(n)$
- d) none.

Sol^{ans}

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

Randomized Q.Sort:

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

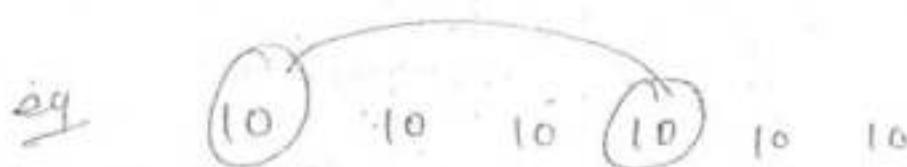


$q = \text{random generator } (1, 2, 3, 4, 5, 6, 7)$

swap $[a[1], a[4]]$

Q.S (a, p, q)

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



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

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

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 $\Rightarrow n$ $O(Kn)$ AlgoSelection procedure (a, i, j, K)

```

    {
        if (i == j)           → b
            {
                if (K == i)
                    return [a[i]]
                else
                    return (-1);
            }
        }
    }
}

```

else

```

    {
        m = partition(a, i, j) → n
        if (m == K) return (a[m]) → 2x C
        else
            {
                if (m < K) Selection Procedure(a, m+1, i, K)
                else Selection Procedure(a, i, m-1, K)
            }
    }
}

```

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

Matrix Multiplication

Without DAE

Matrix addition -

~~A~~
A_{m x n}

~~B~~
B_{n x n}

$$C_{m n} = A_{m x n} + B_{n x n}$$

C contains n^2 elements

for every element required one element
addition.

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

$$\left\{ \begin{array}{l} \text{for } (i = 1 \text{ to } n) \\ \quad \quad \quad \text{for } (j = 1 \text{ to } n) \\ \quad \quad \quad C(i, j) = A(i, j) + B(j, i) \end{array} \right.$$

Matrix multiplication

eg

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2 \times 3}, \quad B = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}_{3 \times 3}$$

$$C = A * B$$

2×3 2×3 3×3

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

Pf

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

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

$$I \cdot C \Rightarrow n^2 \times n$$

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

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

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

$$C[i][j] = 0 \quad \overbrace{\hspace{1cm}}^{n^2}$$

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

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

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

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

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

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

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

Diagram showing matrix A divided into four 2x2 sub-matrices labeled A₁₁, A₁₂, A₂₁, and A₂₂. Each sub-matrix is enclosed in a circle.

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

Diagram showing matrix B divided into four 2x2 sub-matrices labeled B₁₁, B₁₂, B₂₁, and B₂₂. Each sub-matrix is enclosed in a circle.

$$C = A * B$$

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

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

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

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

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

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

∴

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

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

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

$$T(n) = \Theta T(\frac{n}{2})$$

$$T(n) = \begin{cases} c & \text{if } n \leq 2 \times 2 \\ \Theta T(\frac{n}{2}) + \Theta (\frac{n}{2})^2 \\ \quad \vdots \\ \Theta T(\frac{n}{2^k}) + n^2 \end{cases}$$

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

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

\Downarrow
 $8T(\frac{n}{2}) + n^2$
 \Downarrow
 $\underline{\underline{O(n^3)}}$

Multiply two matrices of size $m \times n$ 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 relation.

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

Master Theorem

This method will be applicable only those recurrence relation 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 :-

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

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

Case 2 :-

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

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

Case 3 :-

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

and

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

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

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

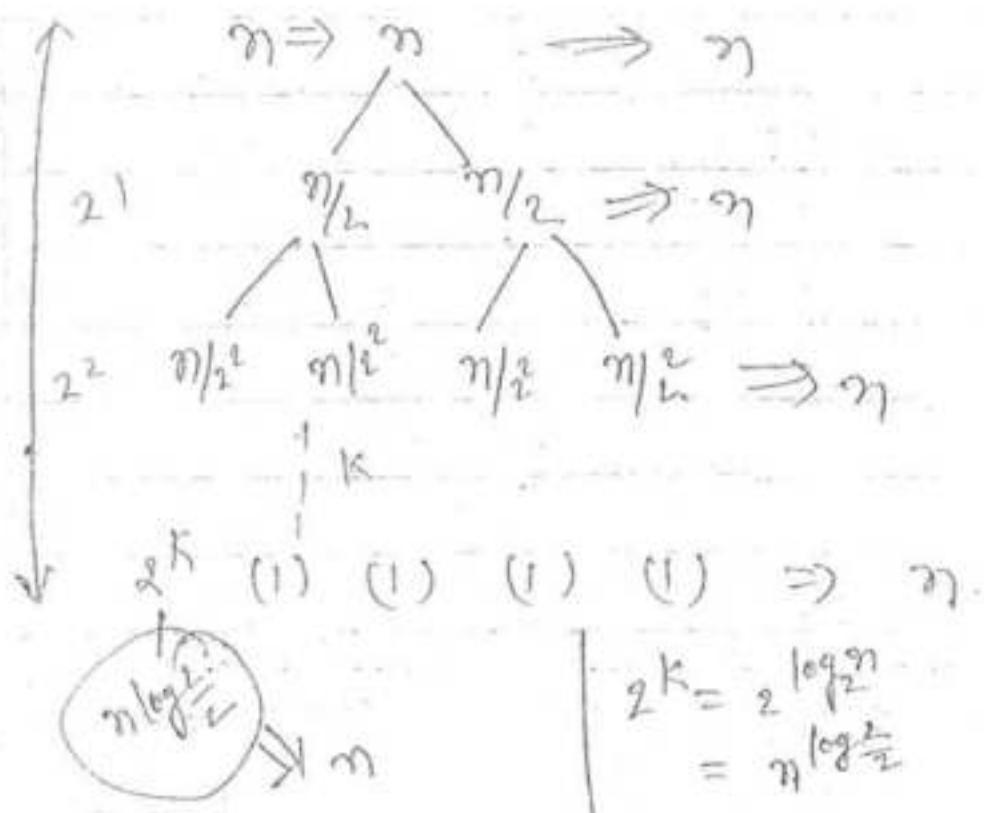
$$a = 2, \quad b = 2, \quad f(n) = n.$$

$$f(n) = n.$$

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

Case 2:

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



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

$$a=2, b=2$$

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

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

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

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

~~$$a=2, b=2$$~~

~~$$f(n) = 1$$~~

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

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

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

~~$$a=4, b=2$$~~

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

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

Case 3:-

~~$$T(n) = \underline{\Theta(n^5)}$$~~

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

$$a=2, b=2$$

$$n^{\log_2^2} = n \quad f(n) = n \log n.$$

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

here $f(n)$ function greater than $n^{\log_2^2}$
 by \log , which is not polynomial
 so master theorem will not work.

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

$$a=1, b=2$$

$$n^{\log_2^2} = n^{\log_2^2} \Rightarrow n^0 \Rightarrow 1$$

$$f(n) \Rightarrow n$$

Case 3 :-

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

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

$$a=2, b=1$$

it is not in own required format.

Conversion

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

$$(1) \text{ Assume } n = 2^k$$

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

$$(2) \text{ Assume } T(2^k) = S(k)$$

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

$$a=2, b=2$$

$$\begin{aligned} f(k) &= k \\ k^{\log_b a} &= k \end{aligned}$$

$$k \cdot \log k$$

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

$$T(n) = T(\sqrt{n}) + c$$

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

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

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

$$S(K) = S(\frac{K}{2}) + c$$

$$a=1$$

$$b=2$$

$$f(K) = 1$$

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

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

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

* Find Time Complexity for the following C Program.

```
A(n)
{
    if (n≤1) return 1;      { b=√n ⇒ c
    else
        return (A(√n)) → c. A(b) ⇒ T(√n)
    }                         return (c) ⇒ c
}
```

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

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

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

↓
 $S(k) = S(k/2) + c$
 ↓
 $\log k$
 $\underline{\underline{O(\log(\log n))}}$

Find time complexity for following C-Program.

DAA (n)

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

do^n

$$a = DAA (\frac{n}{2})$$

$$b = DAP (\frac{n}{2})$$

$$c = a+b+n$$

return (c)

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

\Downarrow
 $O(n)$

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

$$= 2[2T(\frac{n}{2^2}) + c] + c$$

$$= 2^2 T(\frac{n}{2^3}) + 2^1 + 2^0 c$$

$$= 2^3 T(\frac{n}{2^4}) + 2^2 c + 2^1 c + 2^0 c$$

↓
k

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

$A(n)$

```

    {
        if (n ≤ 1) return;
        else
            return (n * A(√n))
    }
  
```

$a = \sqrt{n}$
 $b = A(a)$
 $c = n \# b$
 $\text{return}(c);$

3.

Sol^{by}

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

↓
 $O(\log(\log n))$

~~→~~

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

↓
 $O(n^2)$

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

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

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

$$\Downarrow O(n)$$

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

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

* $\text{fact}(n)$

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

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

Cube Root

for (i = 1 to 1000)

```

  {
    if ( i3 ≤ 1000 )
      em
      return ( i-1 )
  }
  ⇒ 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 $\Rightarrow T(n/l) + n \Rightarrow O(n)$

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

7) Matrix multiplication \Rightarrow

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

(Strassen)

Insertion Sort



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

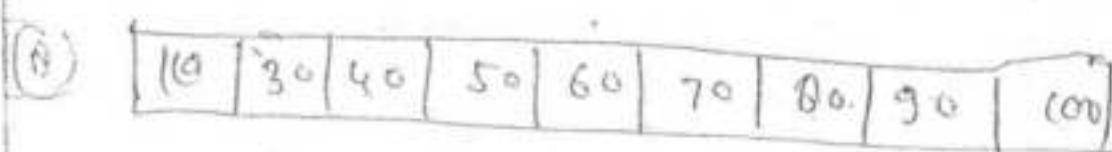
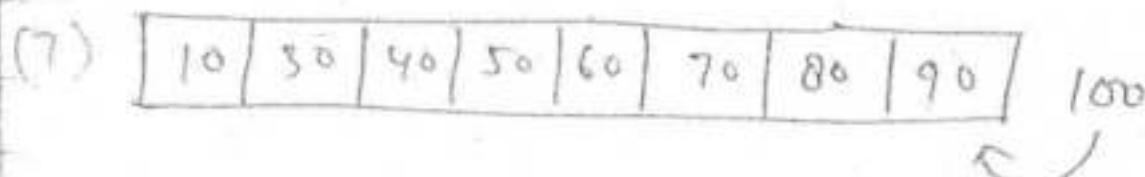
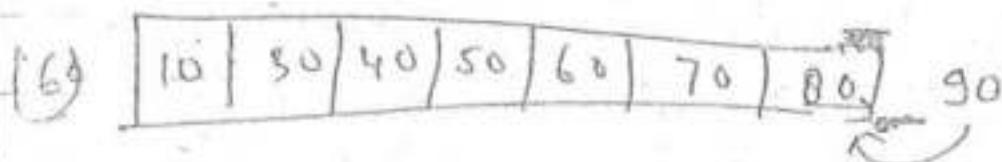
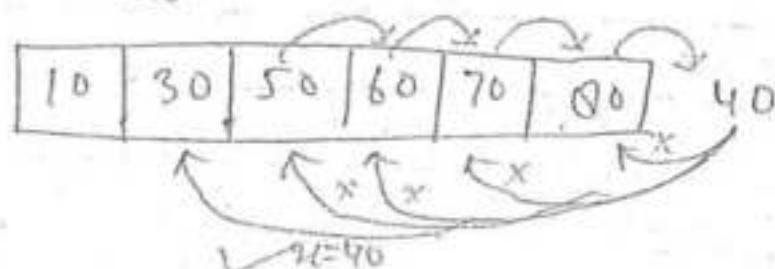
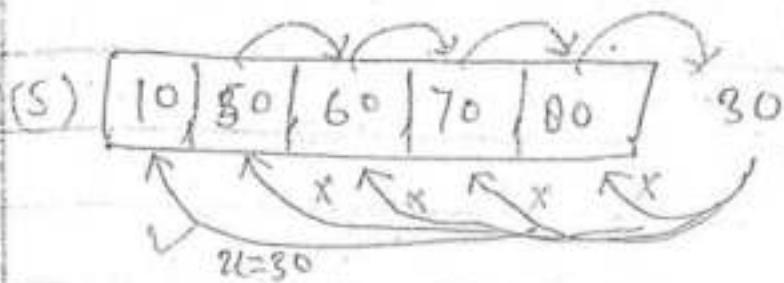


(1) [10] 50

(2) [10 | 50] 60

(3) [10 | 50 | 60] 70

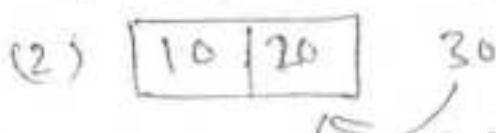
(4) [10 | 50 | 60 | 70] 80

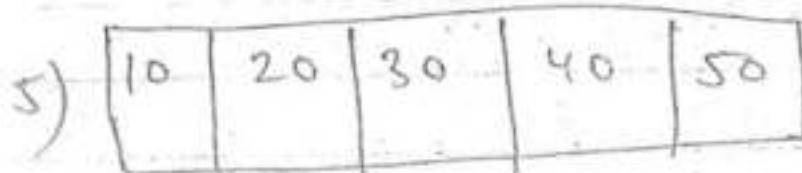
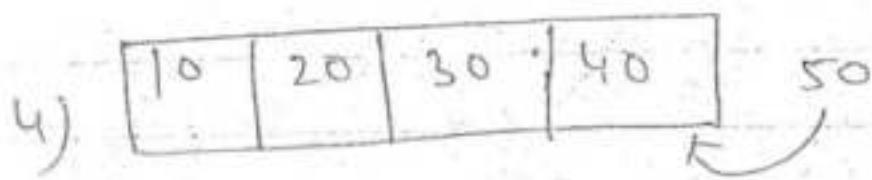
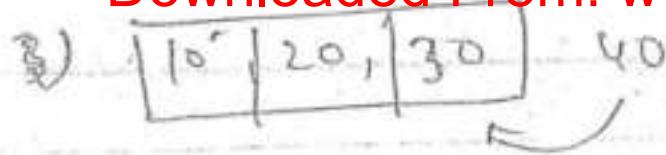


Best Case

Increasing Order

10, 20, 30, 40, 50





Best case of T.C of Insertion sort is

$O(n)$ - in $(n-1)$ time (Θn^2)

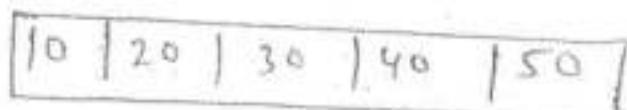
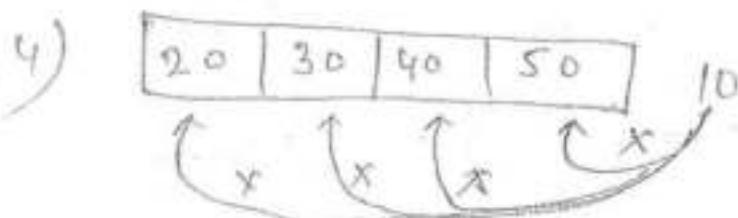
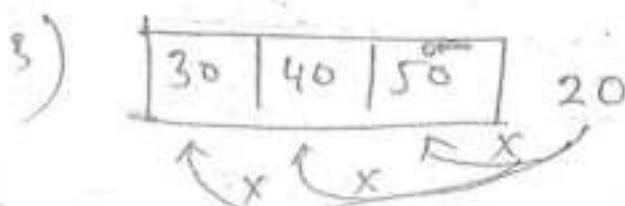
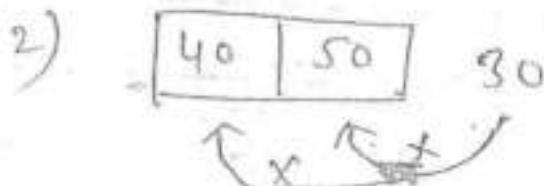
Note For smaller size of input insertion sort is Better.

for greater " " " Menger

" " Better

worst case

50, 40, 30, 20, 10



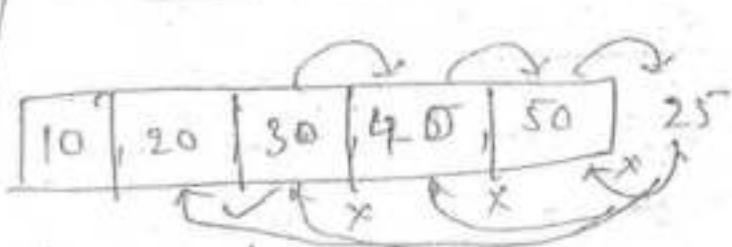
Comparison	swap
1 $\Rightarrow n$	$1 \Rightarrow 2$
2 $\Rightarrow n$	$2 \Rightarrow 3$
3 $\Rightarrow n$	$3 \Rightarrow 4$
4 $\Rightarrow n$	$4 \Rightarrow 5$
1	!
1	!
$n-1$	$n-1$
$n-1$	$n-1$

Total Comparison $O(n)^2 \Rightarrow n^2 + n^2$
 Total swap $O(n)^2 \Rightarrow O(n^2)$

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 P.R. of Insertion sort, If you replace linear search at place of Binary search.

a) Search

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



Soln

Comparison

$$1 \Rightarrow \log n$$

$$2 \Rightarrow \log n$$

$$3 \Rightarrow \log n$$

1

1

1

$$n-1 = \log n$$

swap

$$1 \Rightarrow \log n$$

$$2 \Rightarrow \log n$$

$$3 \Rightarrow \log n$$

1

1

1

$$n-1 = \log n$$

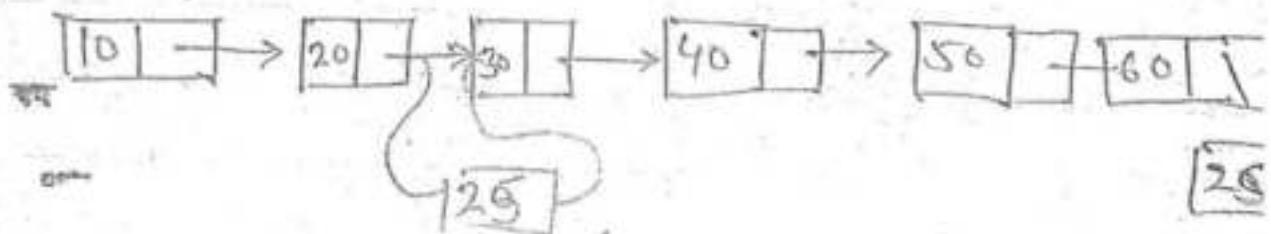
rect

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

chart

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

#



Comparison

$$1 \Rightarrow n$$

$$2 \Rightarrow n$$

$$3 \Rightarrow n$$

$$n-1 \Rightarrow n$$

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

$$n^2 + 0$$

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

Swap

$$0$$

$$0$$

$$0$$

$$0$$

$$0$$

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

$$0$$

Average case also

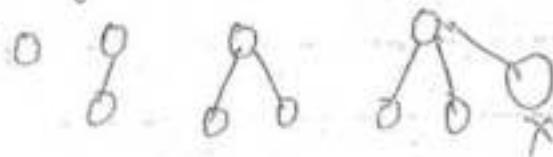
leads to

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

~~done~~

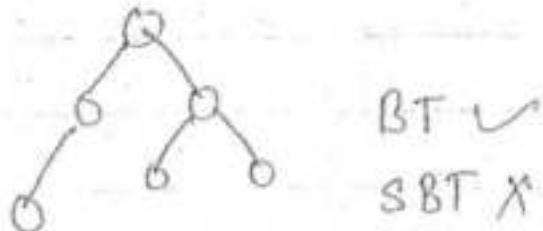
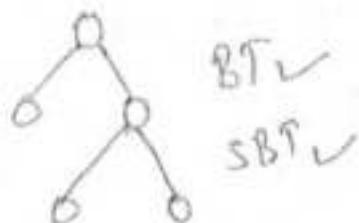
Heap Sort

1) Binary Tree



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

binary tree at every node exactly zero
or two children.

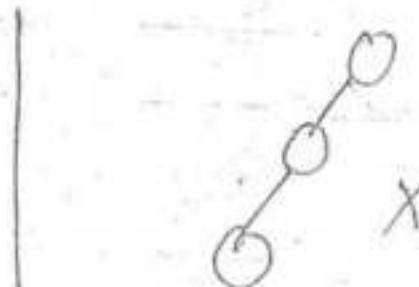
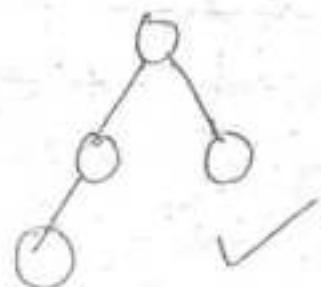
eg

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

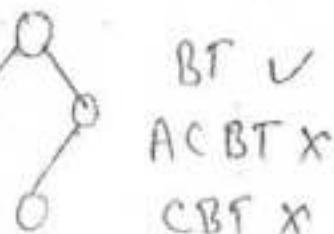
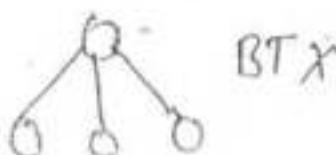
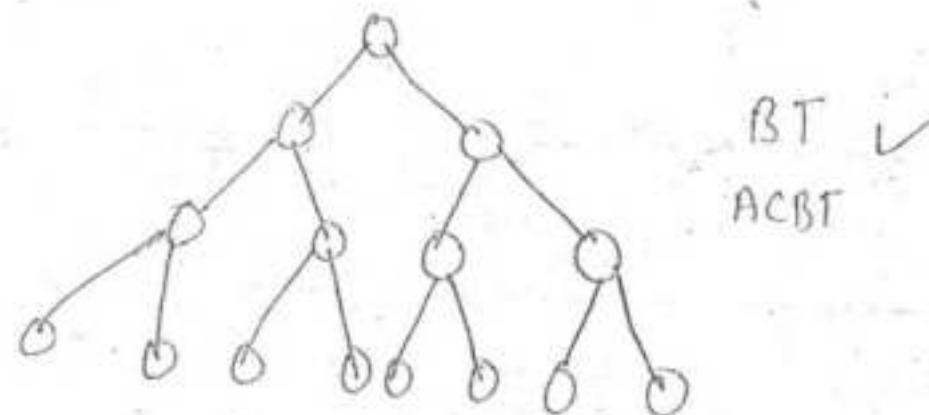


2) at every node after completing present level completely then only go to next level

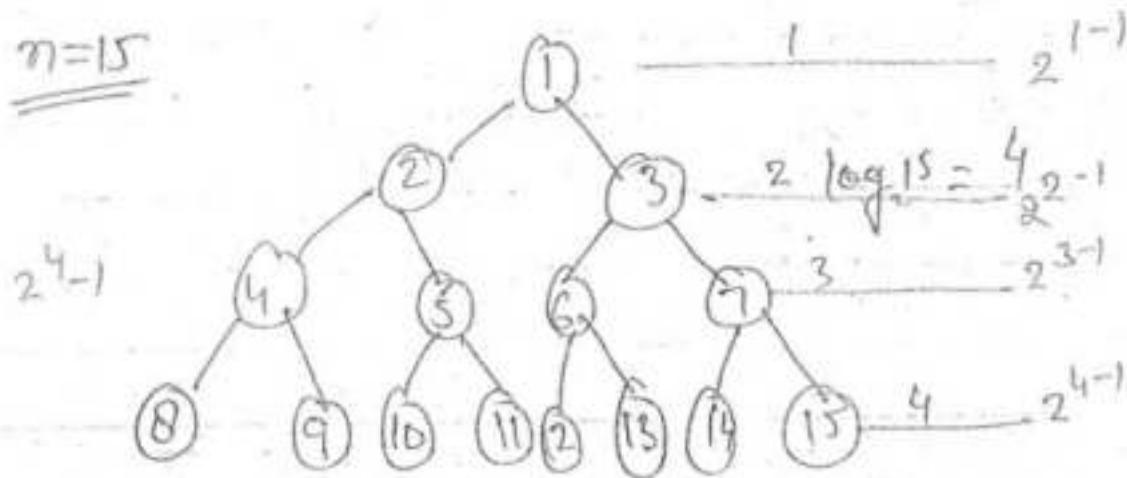
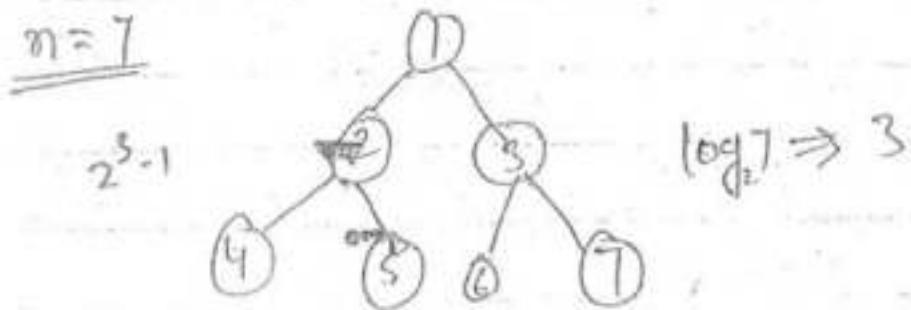
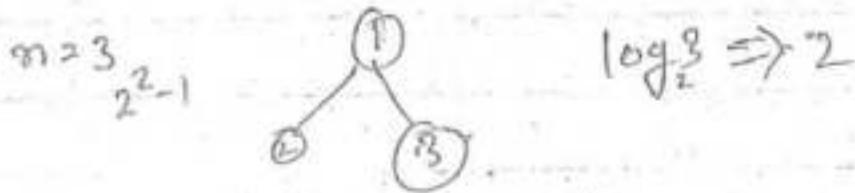


4) Complete binary tree:-

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 no's 1, 2, 3.



all are BT, CBT, ACBT

The maxth 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) = $2^i - 1$

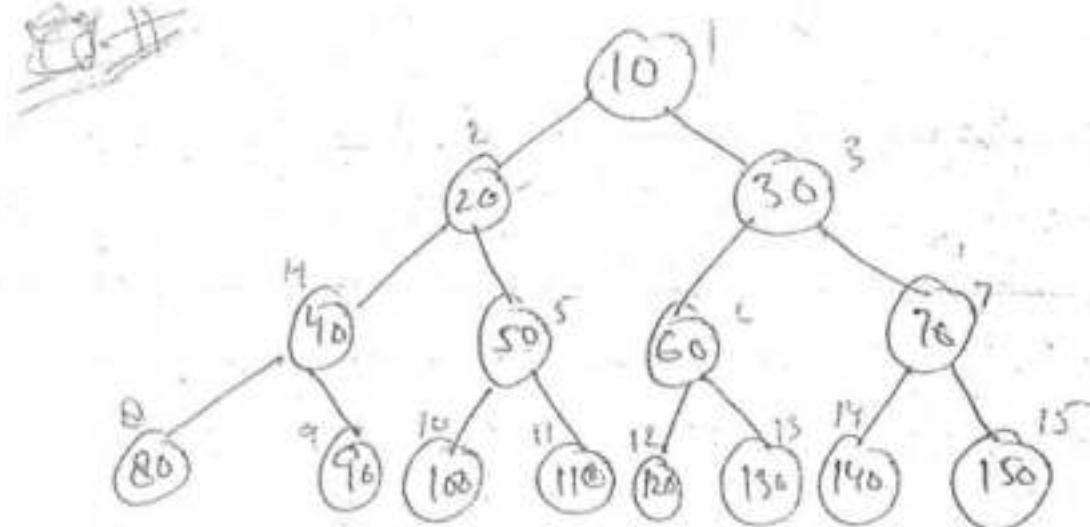
No of levels in almost complete binary tree with n nodes is

$$n = 2^i - 1$$

$$2^i = n + 1$$

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

where i is level

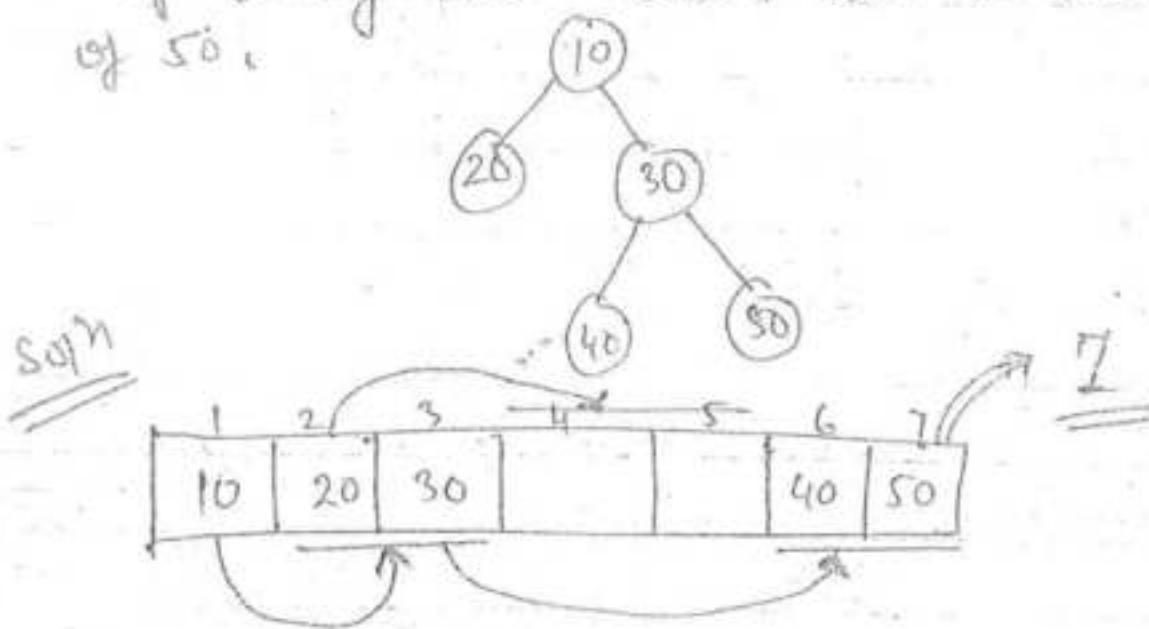


10	20	30	40	50	60	70	80	90	100	110	120	130	140	150
-1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

the node present at i th replace

$$\begin{aligned}\text{left}(i) &= 2 \times i \\ \text{right}(i) &= (2 \times i) + 1 \\ \text{Parent}(i) &= \left\lfloor \frac{i}{2} \right\rfloor\end{aligned}$$

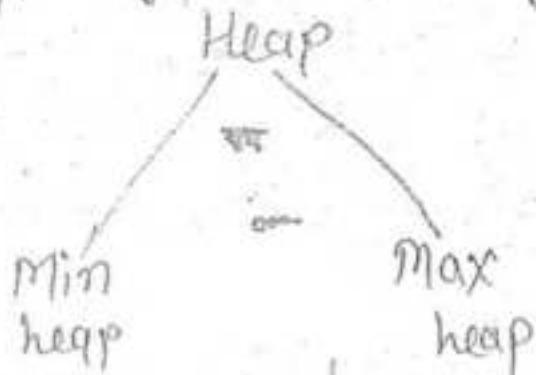
Consider the following binary tree in the form of array then what will be the position of 50.



~~Heap Sort~~Heap tree →

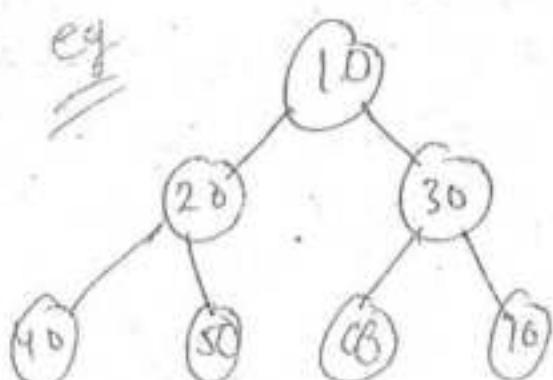
An almost complete binary tree is said to be Heap tree iff.

It should satisfy heap property.
two types of heap property.



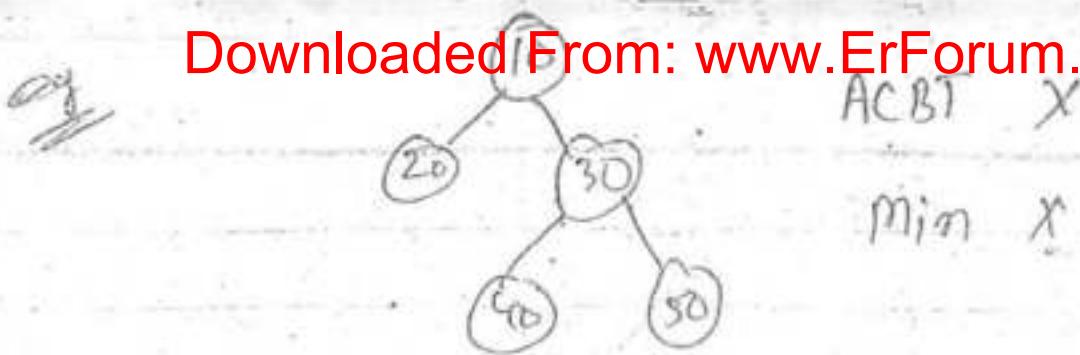
In the almost complete binary tree at every node present is minⁿ comparing their children is called min heap tree

In the almost complete binary tree at every node present is maximum comparing their children is called max heap tree.



ACBT ✓

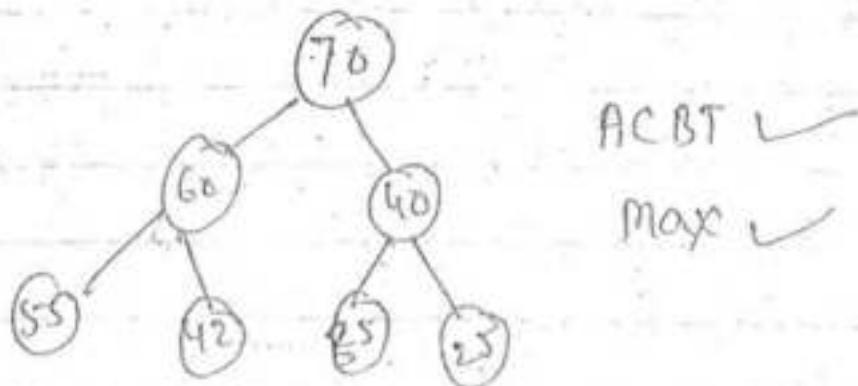
min ✓



ACBT X

min X

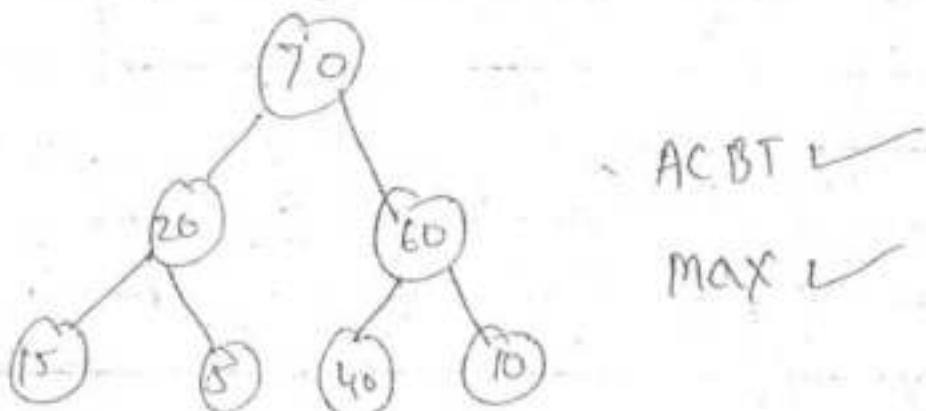
eg



ACBT ✓

max ✓

eg

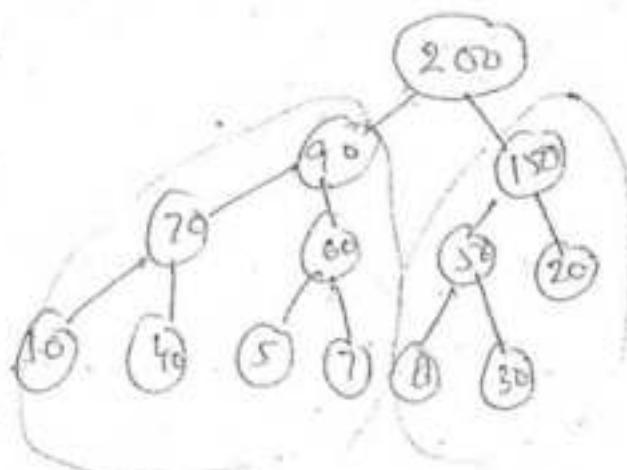
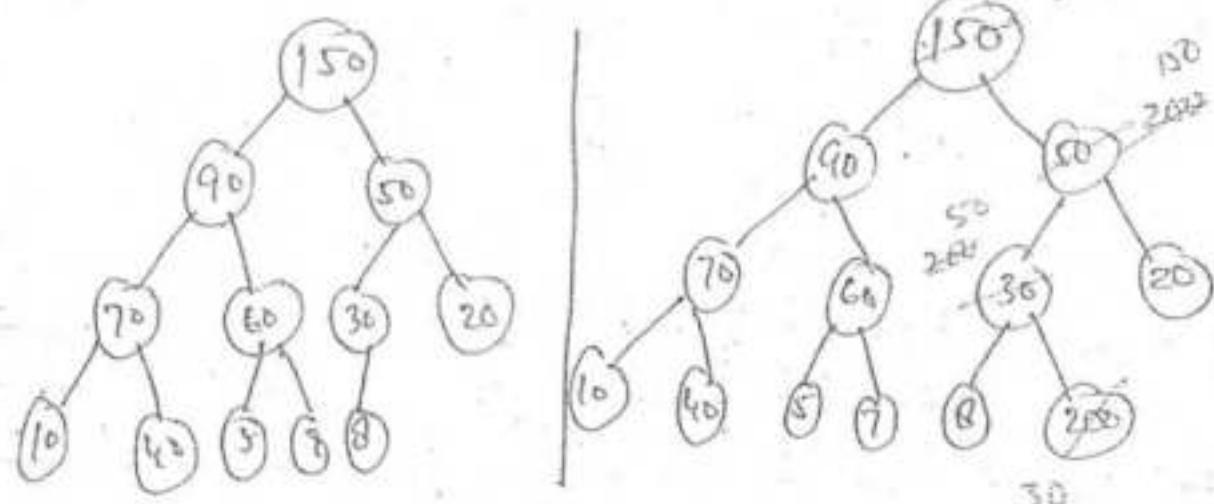
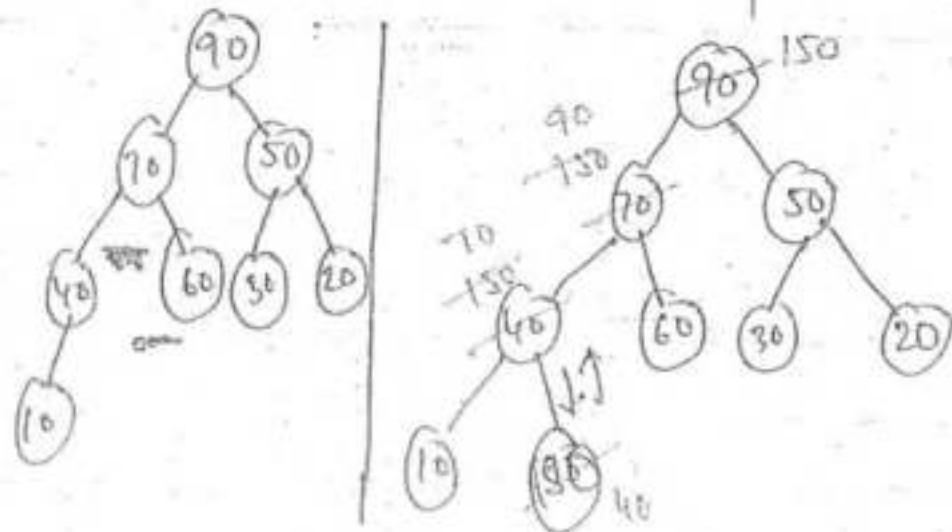
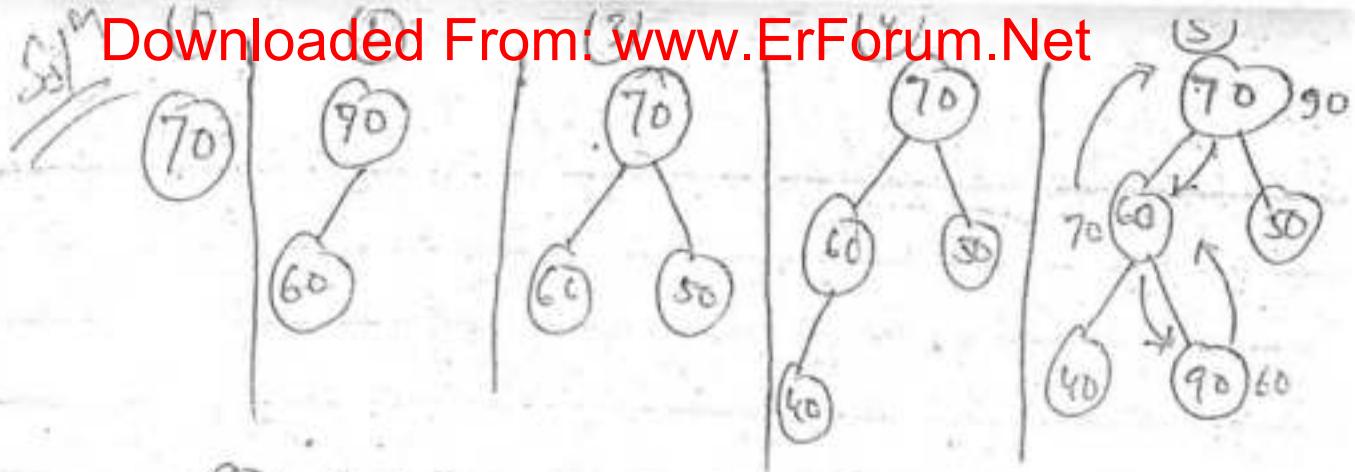


ACBT ✓

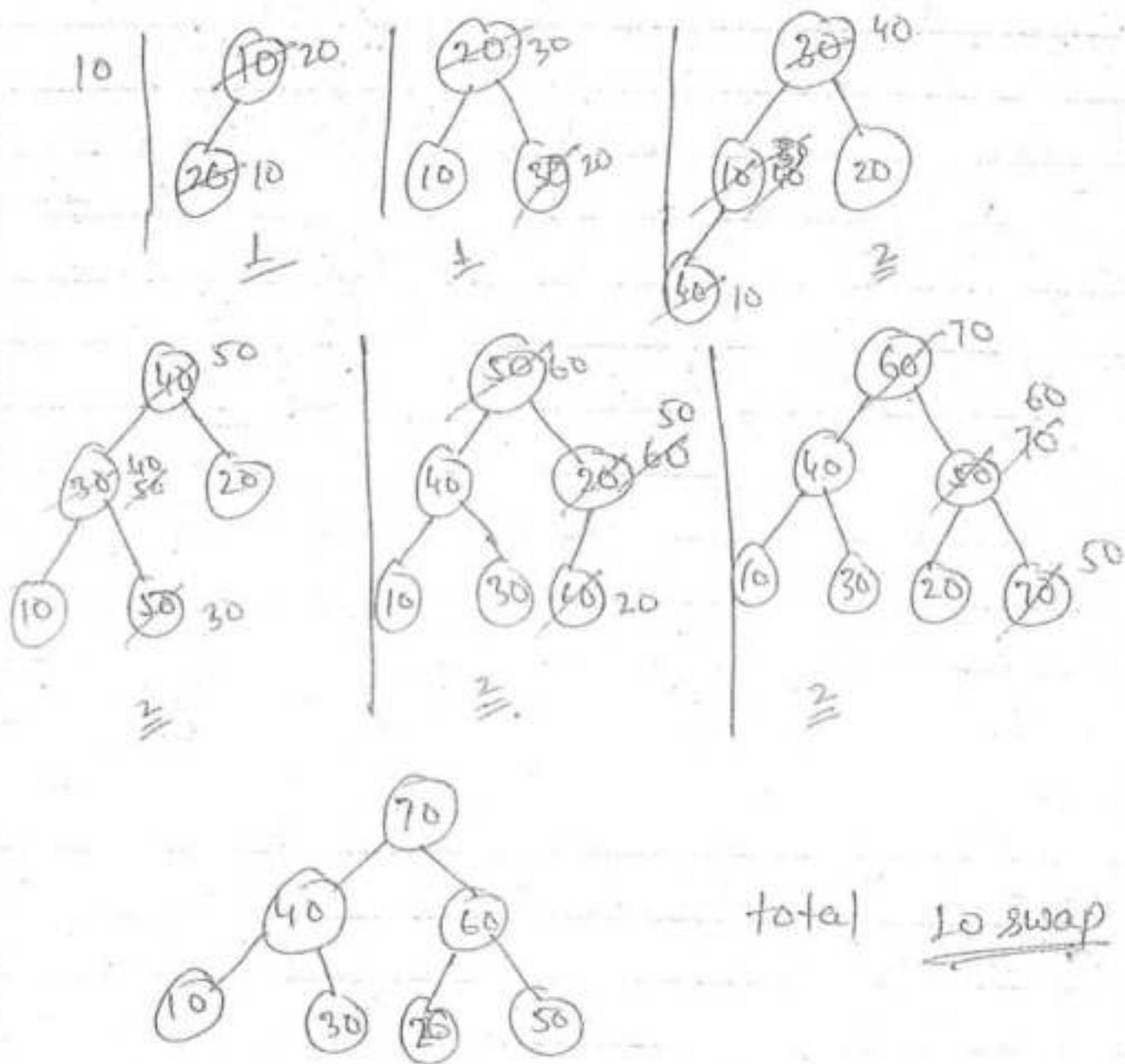
max ✓

P

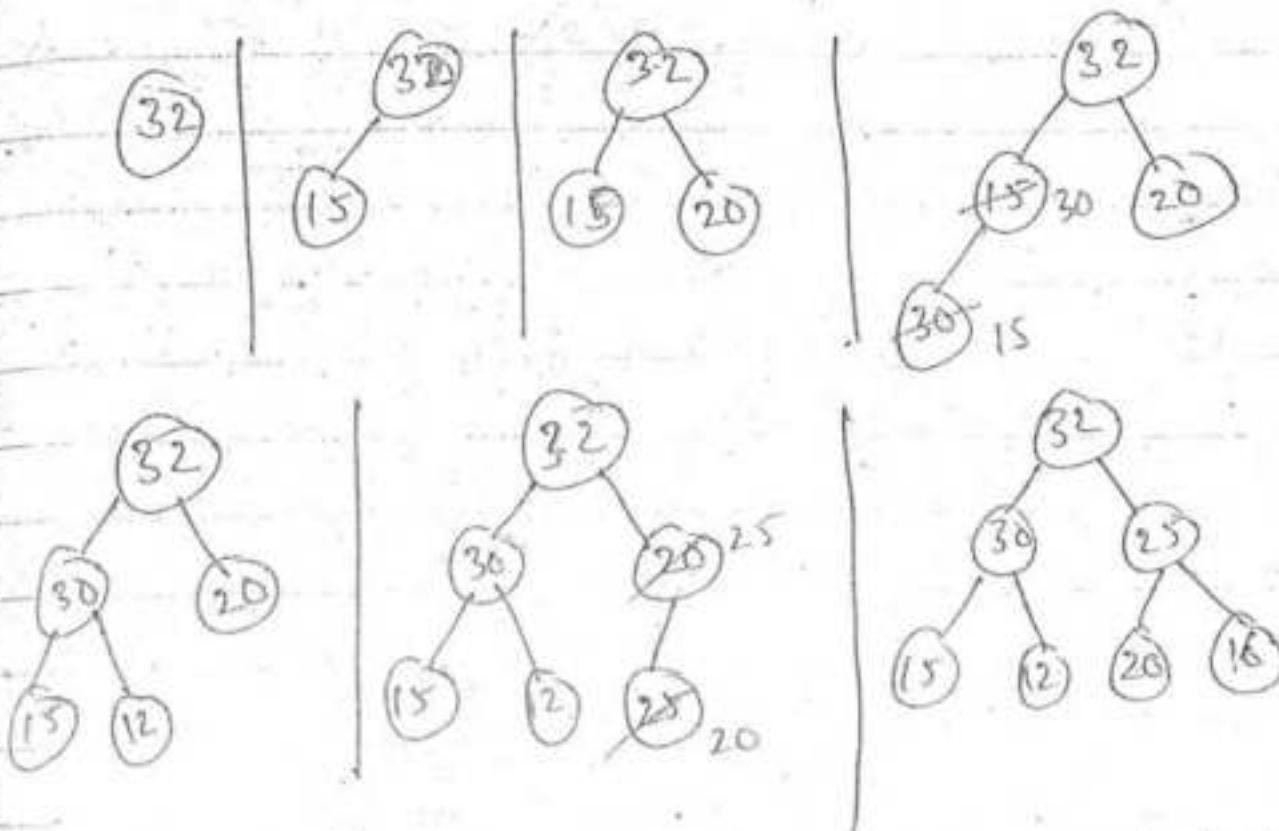
Consider the max heap tree for the following elements: 70, 60, 50, 40, 90, 30, 20, 10, 150, 5, 7, 8, 200.



construct max heap tree for the following element 10, 20, 30, 40, 50, 60, 70

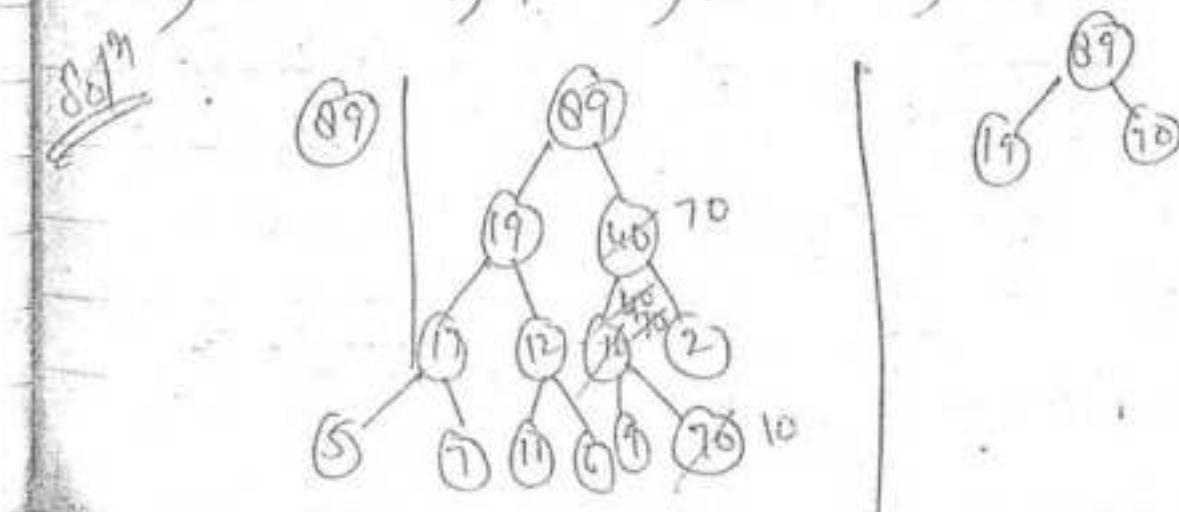


* The elements: 82, 15, 20, 30, 12, 25, 16 are inserted one after another into a max heap. The resultant max heap tree is.

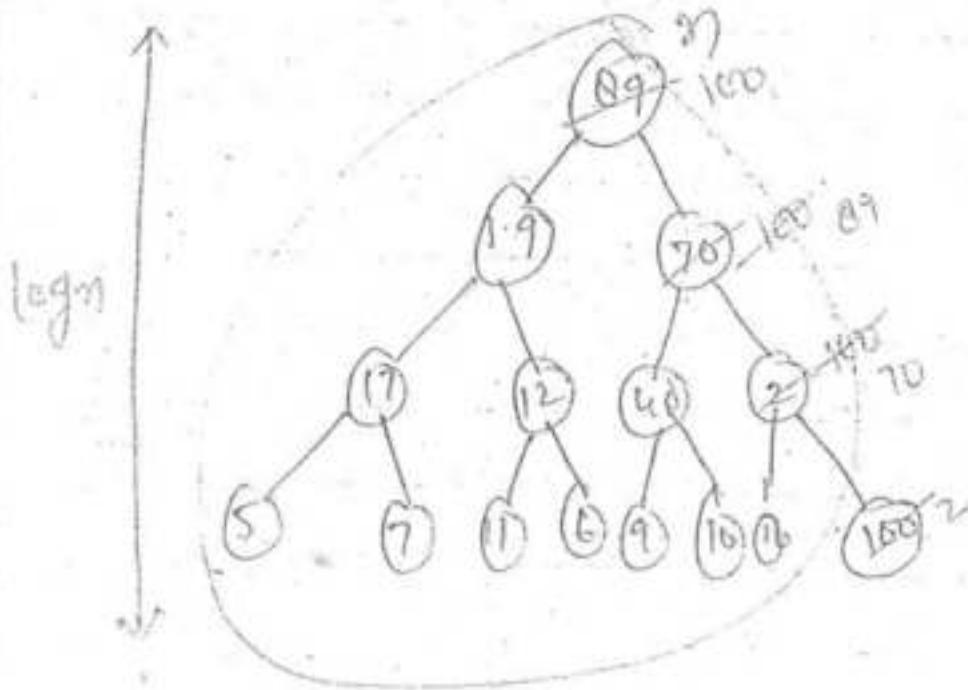


* The minⁿ no of interchanges needed to convert the array: 89, 19, 40, 17, 12, 10, 25, 7, 11, 6, 9, 70 into heap. with max^m elem. at the root is

- a) 0 b) 1 c) 2 d) 3



Q. 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 $\Rightarrow \mathcal{O}(1)$

e.g. let we have to insert 1

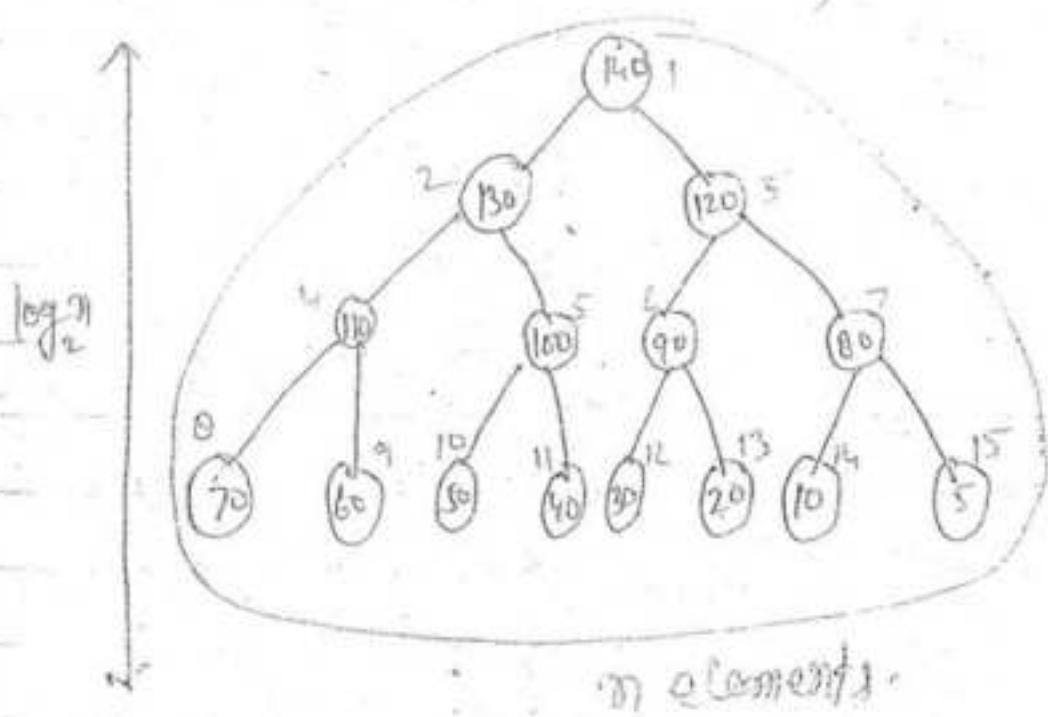
Worst case $\Rightarrow \mathcal{O}(\log n)$

Let we have to insert 100 then all parent will be affected. Hence total no of mapping = 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

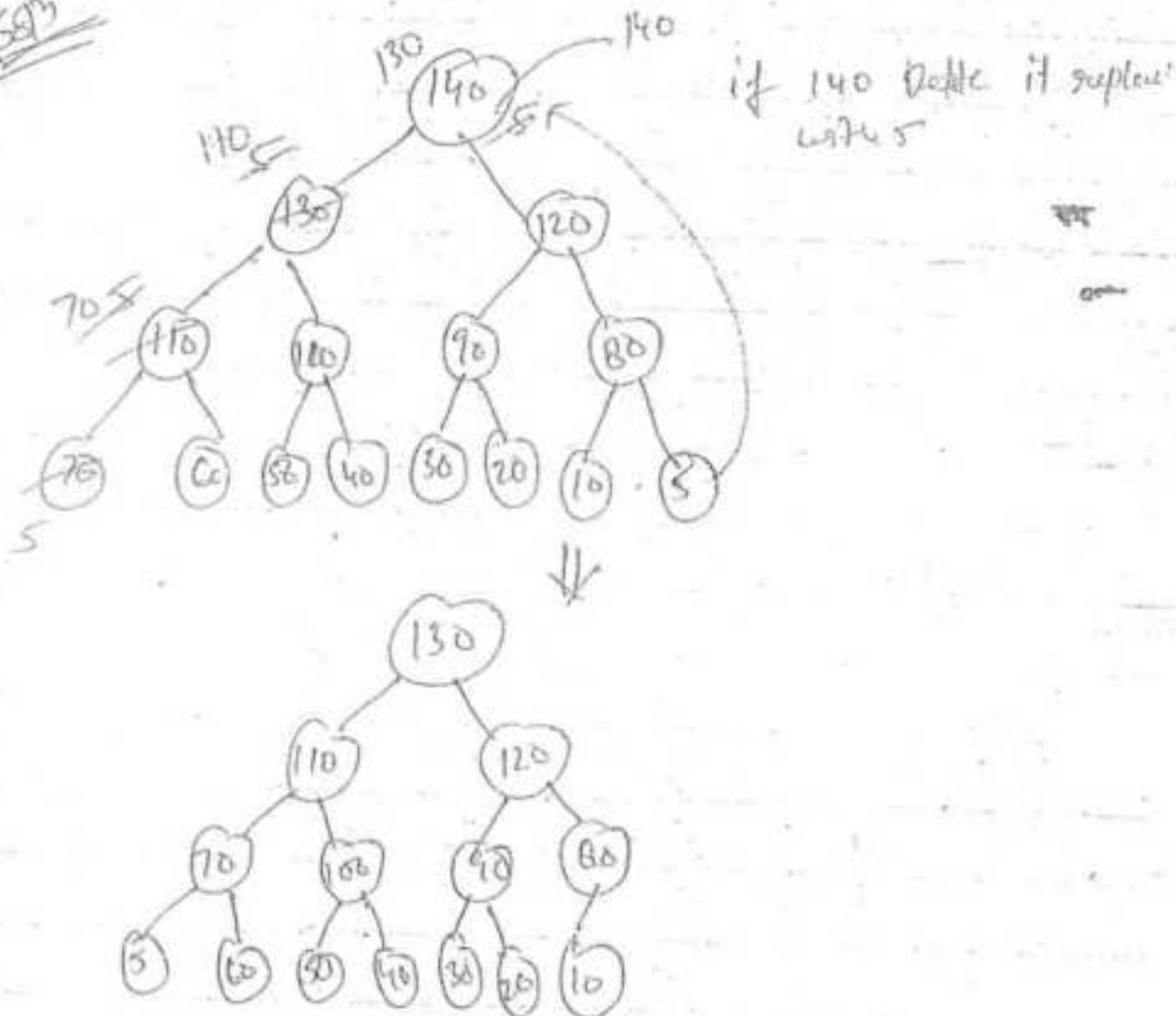
Q How much time it will take to delete an element from min heap or max heap which contain already n elements.



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	

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~~



140	130	120	110	100	90	80	70	60	50	40	30	20	10	5
10	2	3	4	5	6	7	8	9	10	11	12	13	14	15

replace

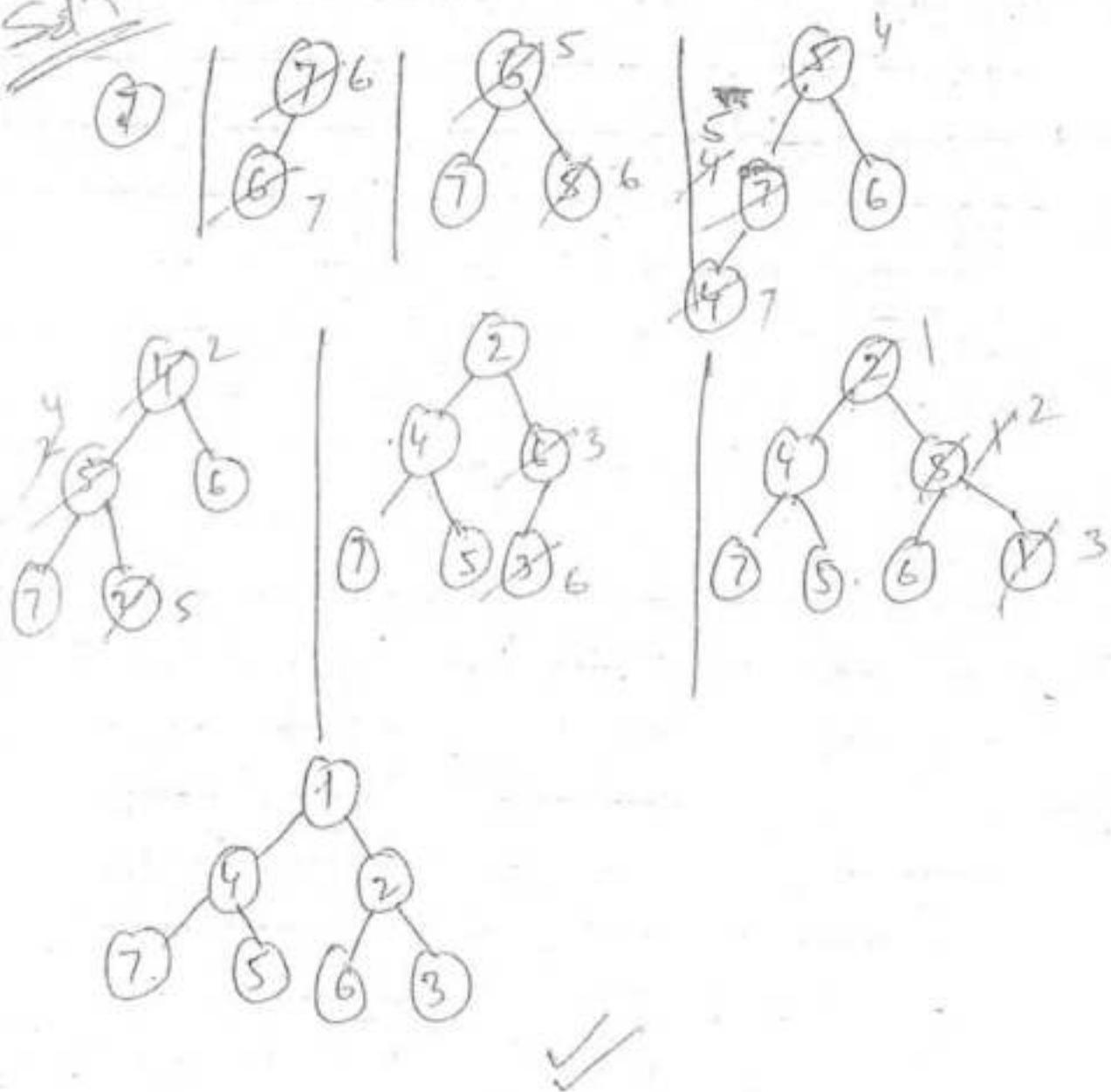
To delete 140 replace it with 5

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.

~~Soln~~



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

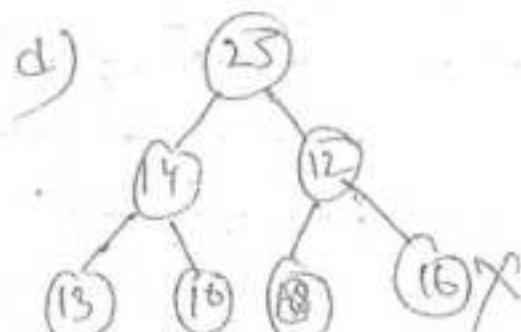
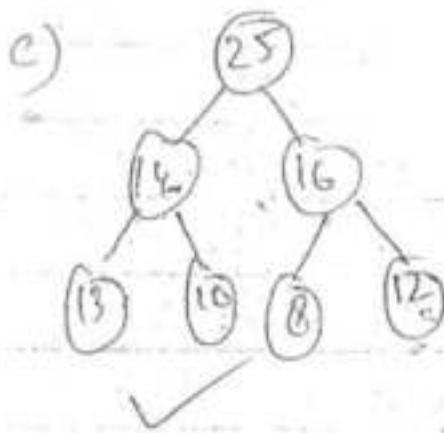
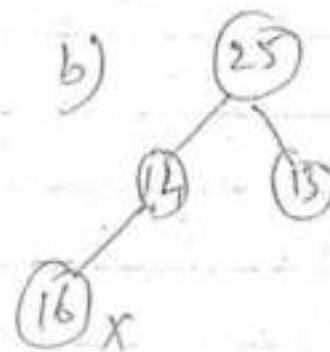
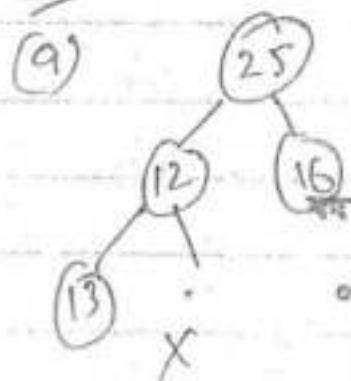
Q2 What are the contents of the array after deleting on the correct answer to the above problem.

- 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

A 3-Array max-heap is like a binary max heap but insert of 2-children nodes have 3-children.

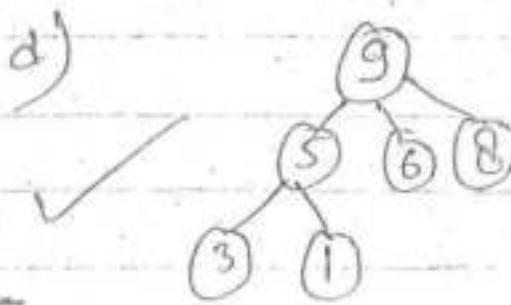
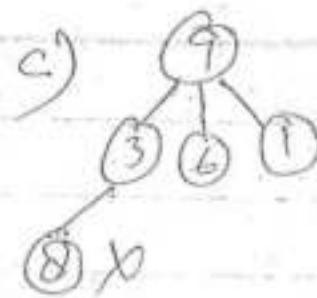
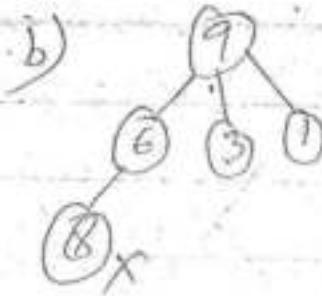
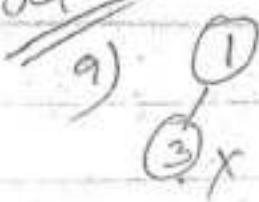
1) which one of the following is a valid sequence of elements in array representation of 3-array max-heap.

- 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

2) 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.

~~SD^n~~



Heap Sort (Assending order)

(1) Create ^{heap tree.} ~~maxⁿ~~ heap tree by inserting one after another(n) $\Rightarrow O(n \log n)$

(2) ^{log n} Delete one by one element and store in the array from right to left(n)
 $O(n \log n)$

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

Min Heap

(1) Finding minⁿ $\Rightarrow O(1)$

(2) Deleting minⁿ $\Rightarrow O(\log n)$

~~(3)~~ Getting 100th smallest element
 Getting $\Rightarrow (100 \log n) \Rightarrow O(\log n)$

(4) Getting maxⁿ element $\Rightarrow O(n)$

Max heap

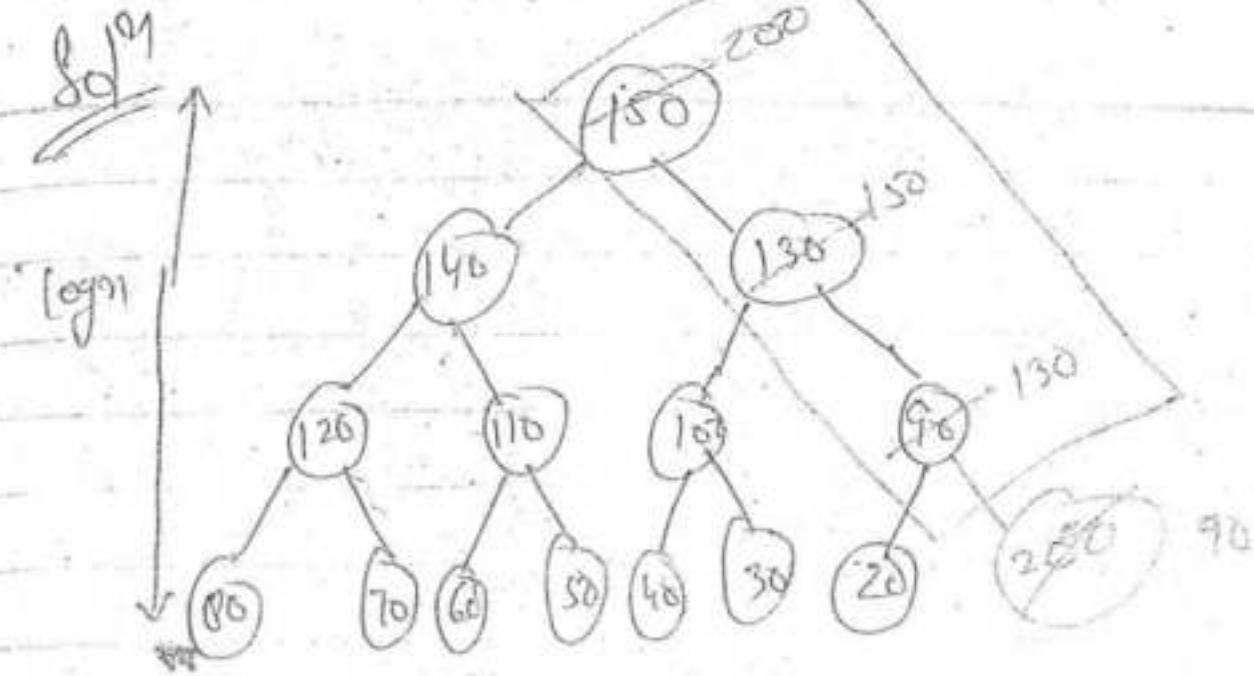
- 1) finding $\text{Max}^m \Rightarrow O(1)$
- 2) Deleting $\Rightarrow O(\log n)$
- 3) Getting $100^{\text{th}} \text{ max}^m \text{ element}$
 $100 \log n \Rightarrow O(\log n)$
- 4) Getting $\text{min}^m \text{ element} \Rightarrow O(n)$

Q 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) $O(n \log n)$



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

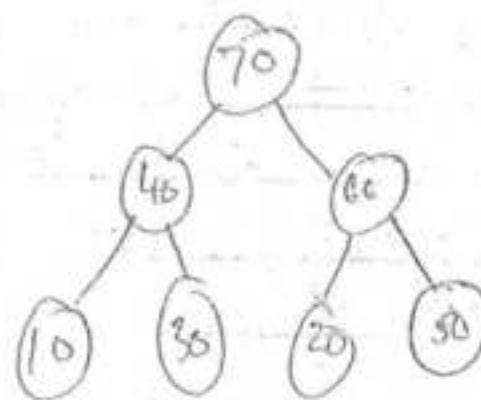
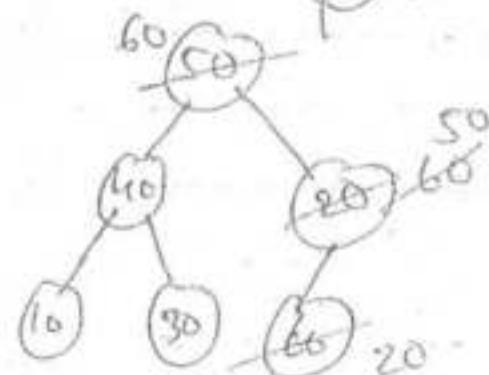
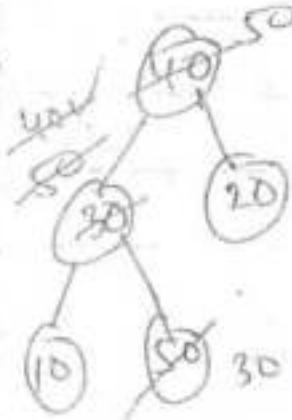
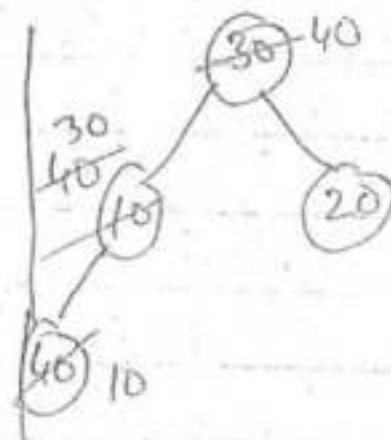
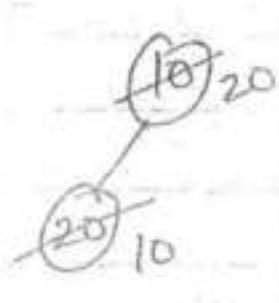
$\Rightarrow O(\log n)$

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.

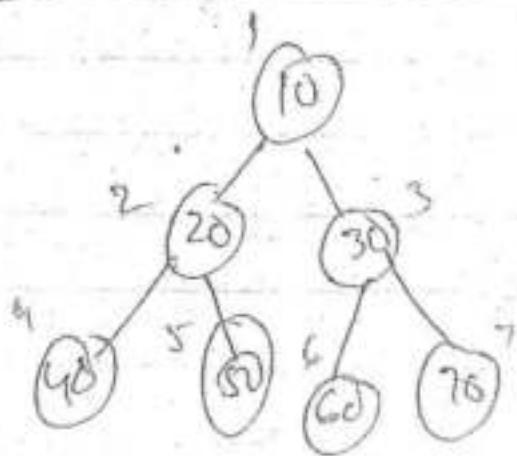
B) Create max heap tree for the following elements using build heap method.
 10, 20, 30, 40, 50, 60, 70



$$2 + 2 + 2 + 2 + 1 + 1$$

$$= \underline{\underline{10}}$$

Build heap metho



Build heap(a, n)

{

for ($i \leftarrow \lfloor \frac{n}{2} \rfloor$ down to 1)

max-heapify(a, i)

max-heapify(a, i)

{

$l = left(i)$

$r = right(i)$

~~max~~,

max = i ;

if ($a[l] > a[i]$

 max = l

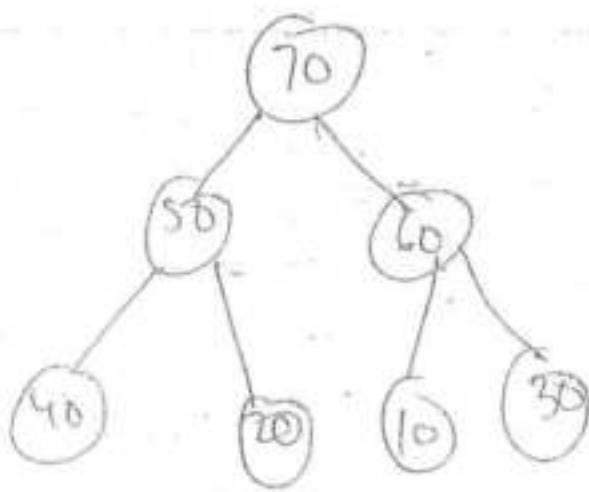
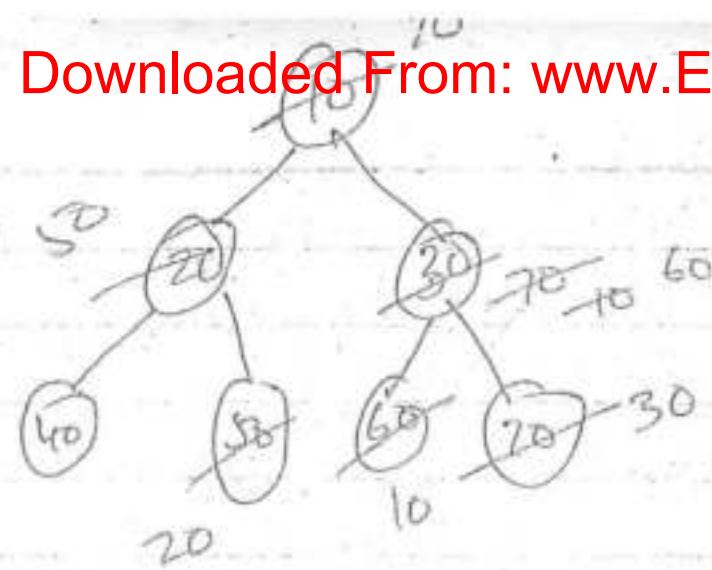
if ($a[r] > a[max]$)

 max = r ;

swap ($a[i], a[max]$)

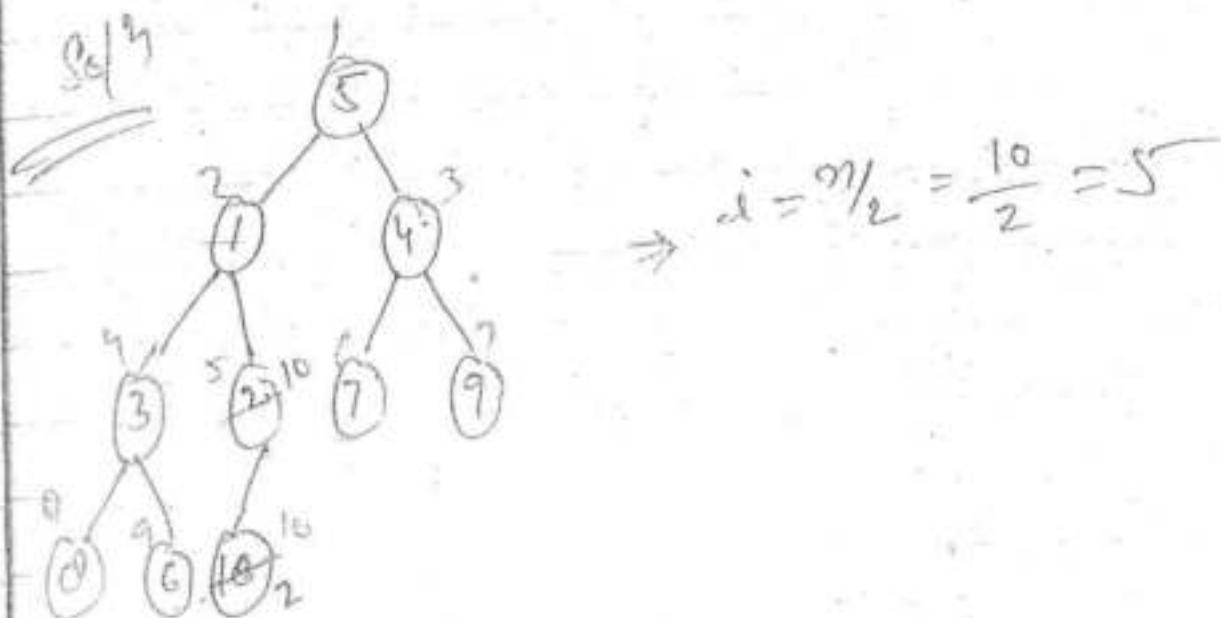
max-heapify(a, max)

3



Construct max heap tree for the following elements using Build heap method.

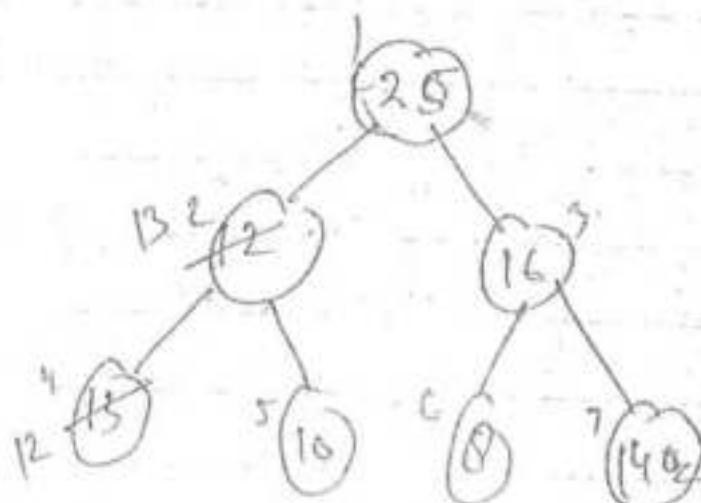
5, 1, 4, 3, 2, 7, 9, 8, 6, 10



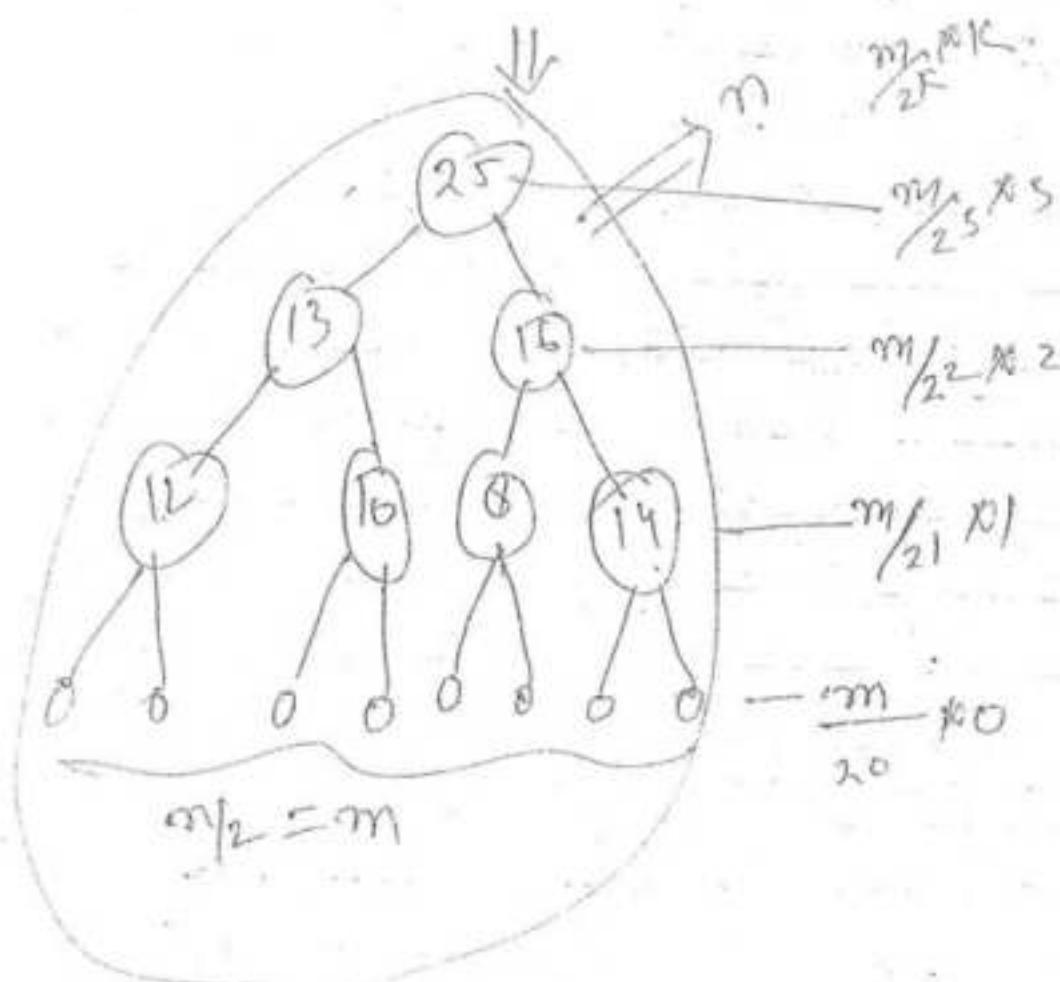
P

Construct a max heap tree for the following using build heap method.

25, 12, 16, 13, 10, 8, 14



$$\lambda = \frac{I}{2} = \frac{3}{2}$$



$$\sum_{n=0}^K \frac{m}{2^n} \cdot h$$

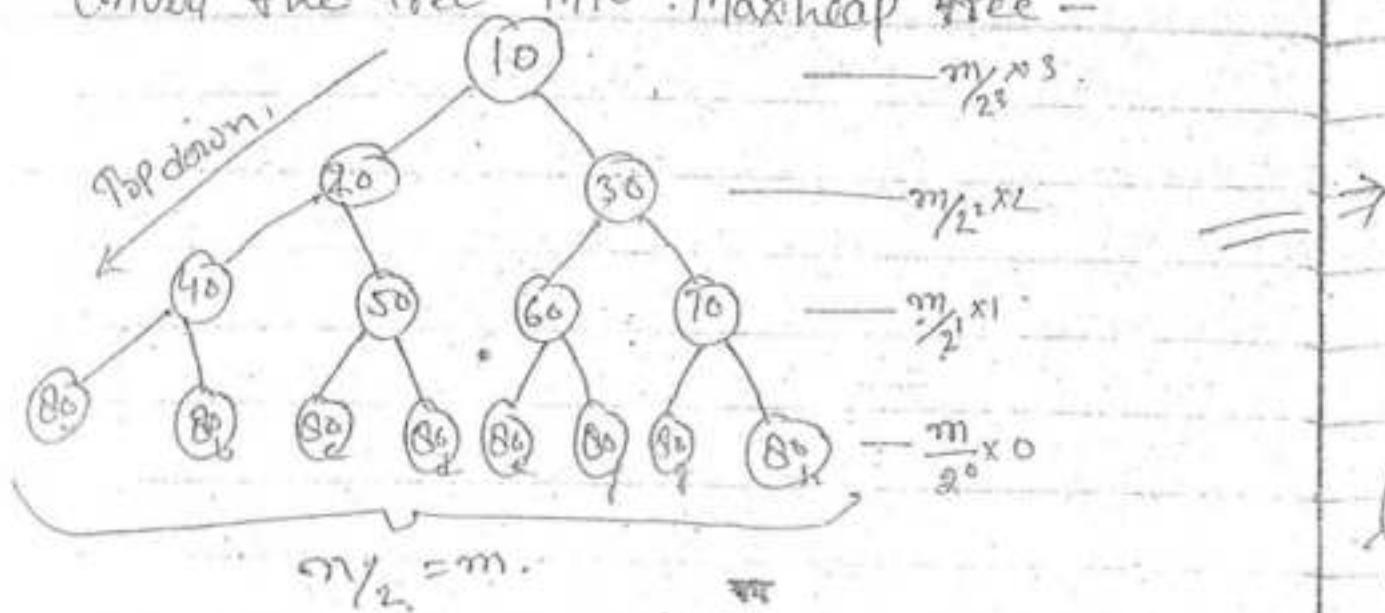
$$= m \left[\frac{0}{2^0} + \frac{1}{2^1} + \frac{2}{2^2} + \frac{3}{2^3} + \frac{4}{2^4} + \dots + \frac{K}{2^K} \right]$$

= m [Constant]

= $\frac{m}{2}$ + Constant

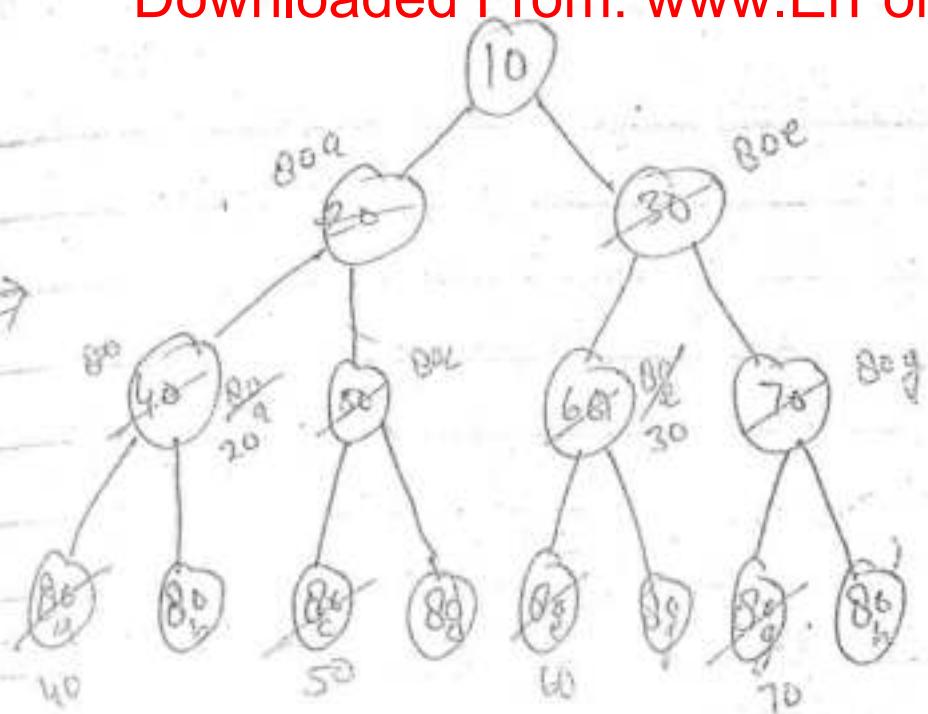
= $O(n)$

P Convert the tree into Maxheap tree -



$$\sum_{h=0}^K \frac{m}{2^h} \cdot h$$

$$\Rightarrow m \left[\frac{0}{2^0} + \frac{1}{2^1} + \frac{2}{2^2} + \frac{3}{2^3} + \dots \right]$$



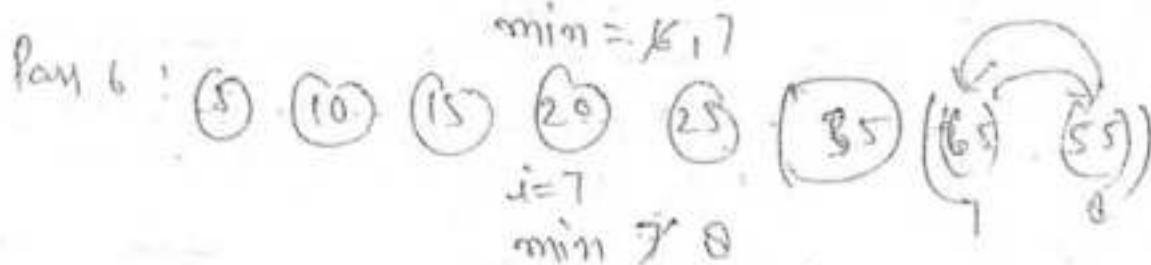
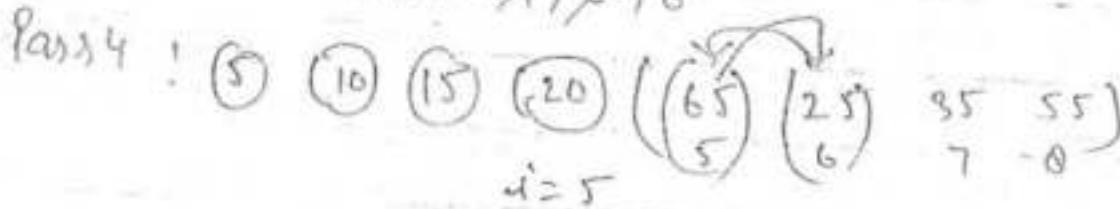
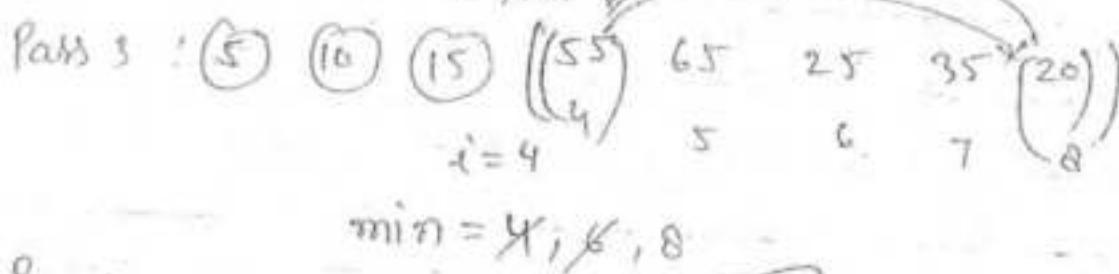
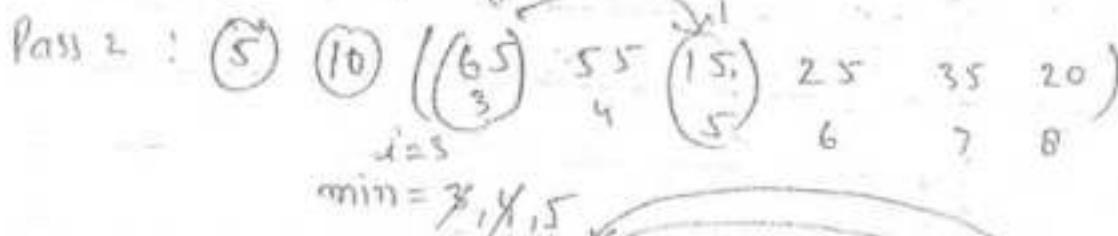
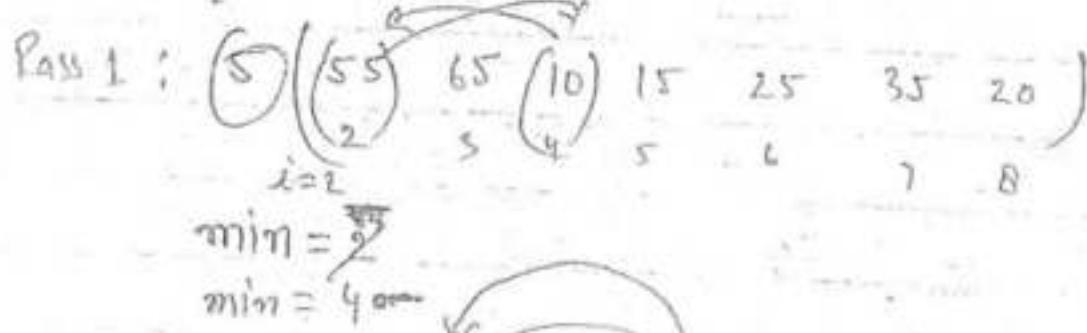
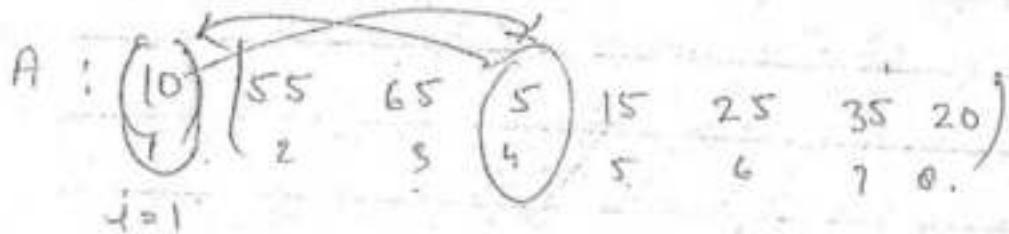
B_{c_1}	(B_{c_2})	B_{c_3}	B_{c_4}	B_{c_5}
-----------	-------------	-----------	-----------	-----------

Heap sort not ~~stable~~ stable sorting technique.

④ In-place

⑤ Construct

Selection Sort



Part 7! $(5)(10)(15)(20)(25)(30)(55)(65)$

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 = 0
                        }
                }
            }
        }
    }
}

```

modification taken

~~eg~~

10 15 25 40 50
 1 2 3 4 5

Pass 1

10 15 25 40 | 50

Pass 2

10 15 25 | 40

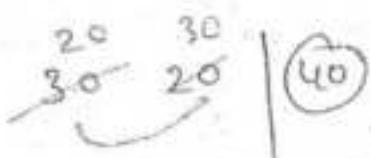
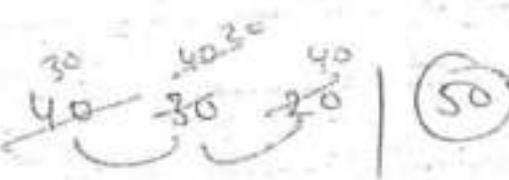
Pass 3

10 15 | 25

Pass 4

10 | 15

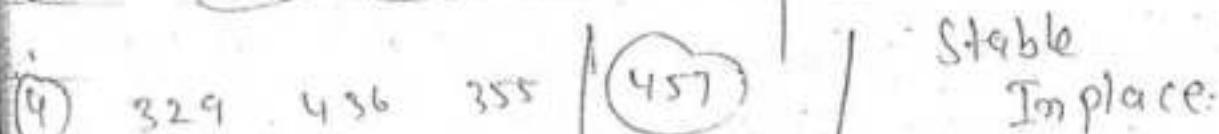
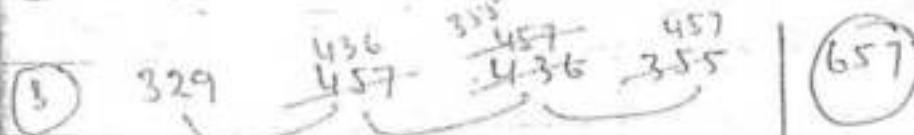
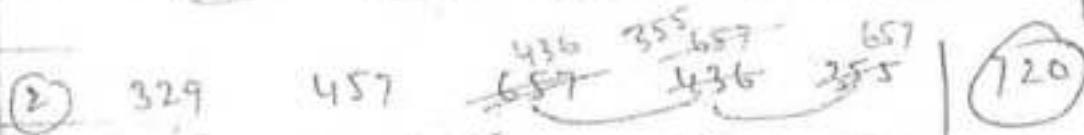
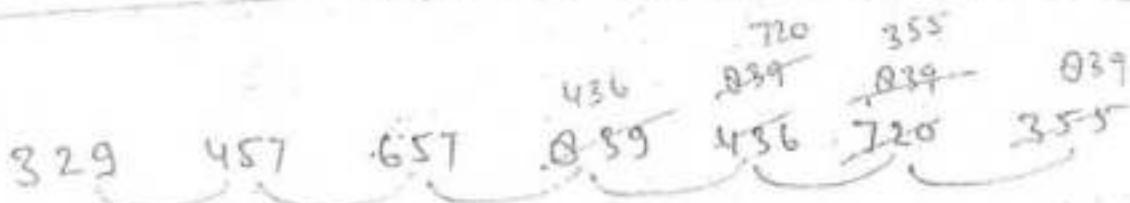
Q1



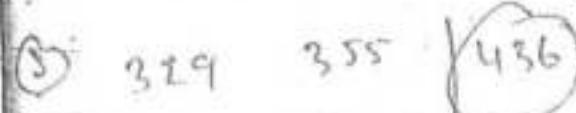
Ans

0000

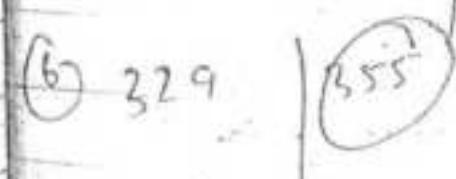
Q2



Stable
In place:



$\min = 0$
 $\max = O(n^2)$



eg

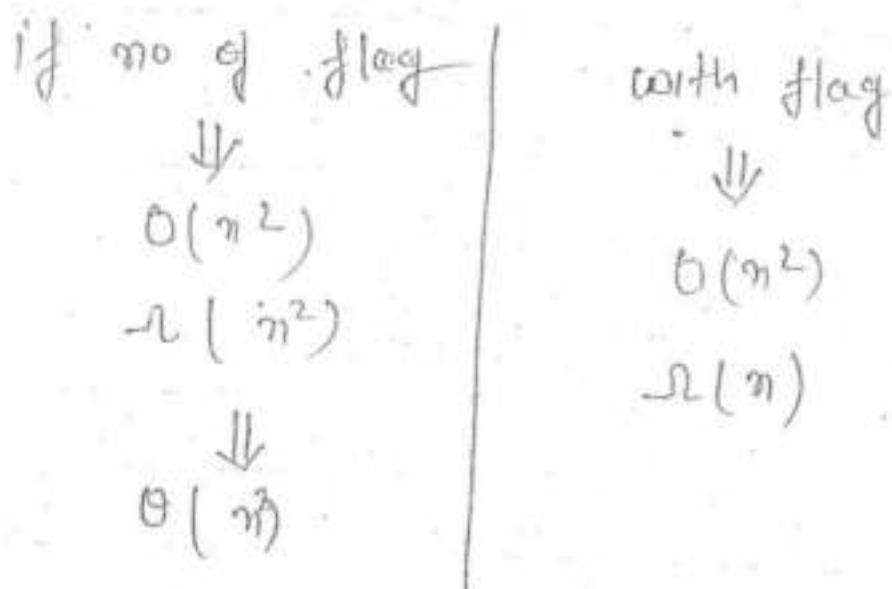
10, 20, 30 | 40 | 50

10 20 30 | 40 | 50

10 20 | 30 | 40 | 50

10 | 20 30 40 50

10 20 30 40 50



No:

(
C
C
)

(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)

L3

return (Solution)

3

Flow of Control

Select } It is the flow of control
Feasible } But each & every step is iterative
↓ Add } If 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)

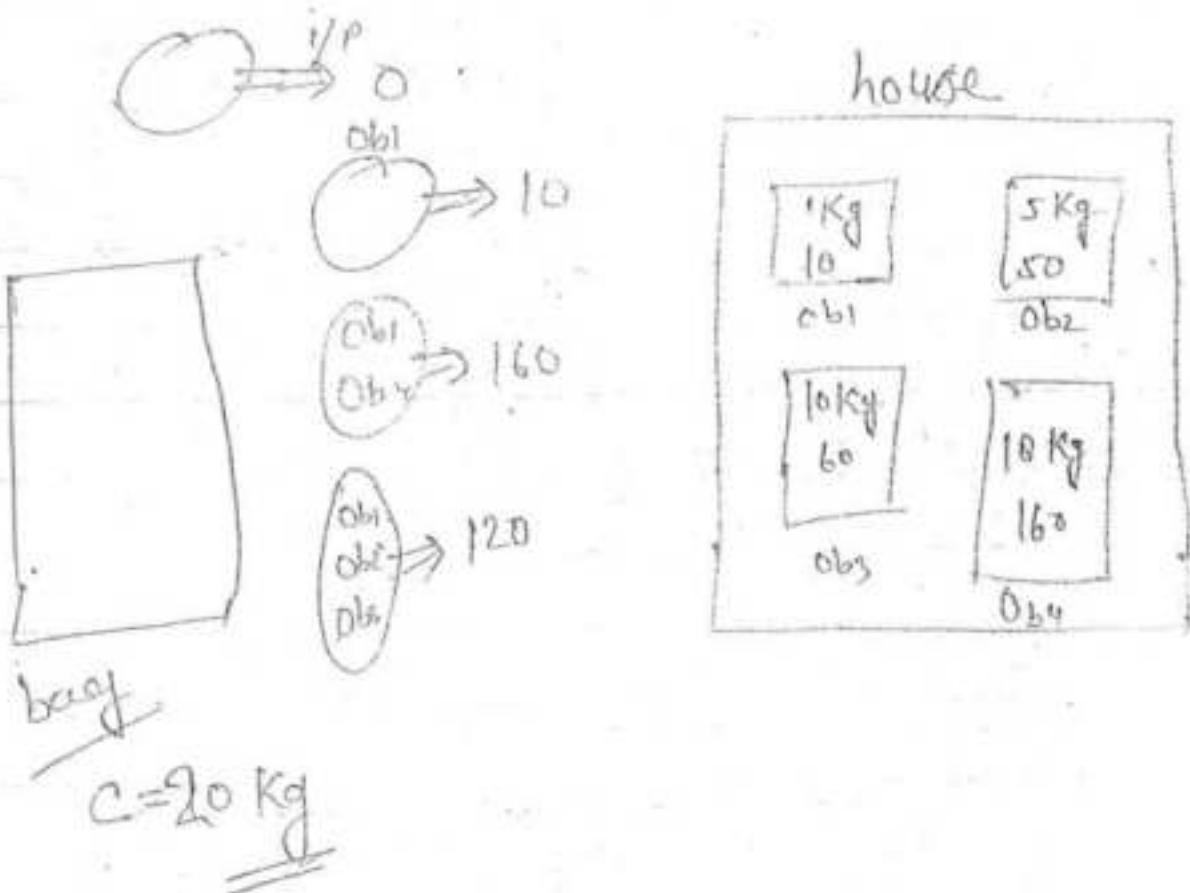
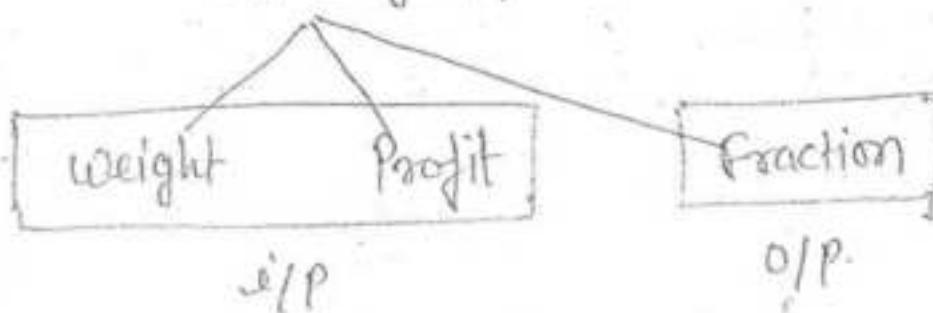
Real Knapsack

(OR)

Fractional Knapsack



n-object.



4 - object
(n) $c = 20$

Object	Ob ₁	Ob ₂	Ob ₃	Ob ₄
weight	w ₁	w ₂	w ₃	w ₄
Profit	P ₁	P ₂	P ₃	P ₄

Problem

$$\sum_{j=1}^n w_j > c$$

Feasible

$$\sum_{j=1}^n w_j \times u_j \leq c$$

Optimal

$$\sum_{j=1}^n p_j u_j \rightarrow \max$$

$n = 3$	$c = 20$		
object	Ob ₁	Ob ₂	
Profit	25	24	15
Weight	10	15	10

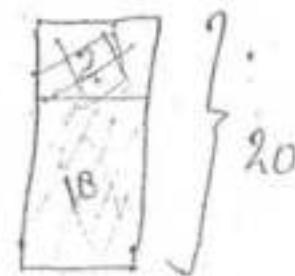
Greedy about profit

$$(1 \quad 2/5 \quad 0)$$

$$= 1 \times 25 + \frac{2}{5} \times 24 + 0 \times 15$$

$$= 25 + 32$$

$$= 57$$



~~Maximize~~

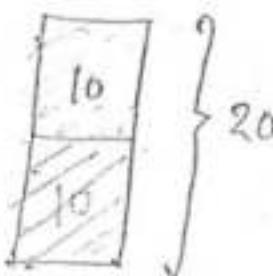
Greedy about weight

$$(0 \quad 10/15 \quad 1)$$

$$= 0 \times 25 + \frac{10}{15} \times 24 + 1 \times 15$$

$$= 0 + 16 + 15$$

$$= 31$$



~~Profit~~

Greedy about both (P & W)Obj₁: $18 \Rightarrow 25$

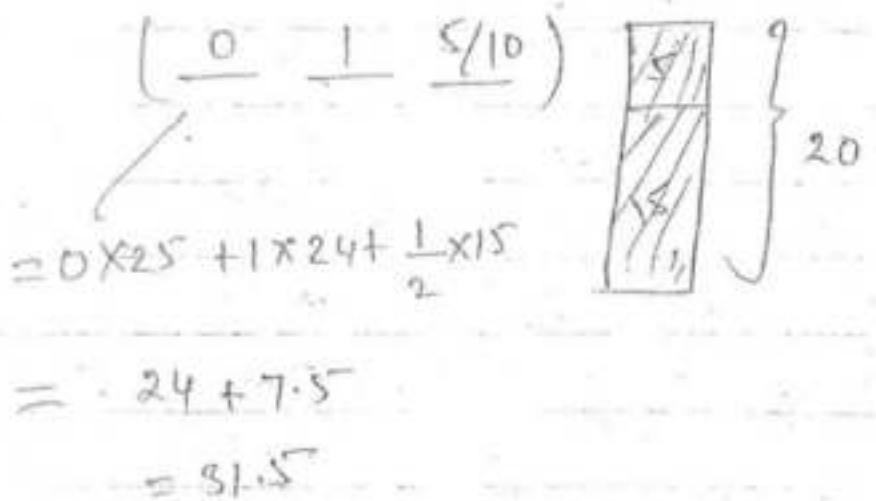
$$1 \Rightarrow \frac{25}{18} = 1.3$$

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.

$$n=5 \quad C=12$$

Object	Ob ₁	Ob ₂	Ob ₃	Ob ₄	Ob ₅
Profit	5	2	2	4	5
Weight	5	4	6	2	1

$$\text{Ob}_1: 5 \Rightarrow 5$$

$$1 = 5/5 = 1 \quad (1)$$

$$\text{Ob}_2: 4 \Rightarrow 2$$

$$1 \Rightarrow 2/4 = .5 \quad (2)$$

$$\text{Ob}_3: 6 \Rightarrow 2$$

$$1 \Rightarrow 2/6 = .33 \quad (3)$$

$$\text{Ob}_4: 2 \Rightarrow 4$$

$$1 \Rightarrow 4/2 = 2 \quad (4)$$

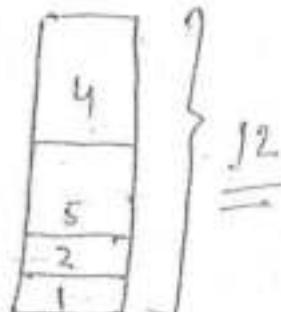
$$\text{Ob}_5: 1 \Rightarrow 5$$

$$1 = 5/1 = 5 \quad (5)$$

(1 1 0 1 1)

$$= 1 \times 5 + 1 \times 2 + 0 \times 2 + 1 \times 5 + 1 \times 4$$

$$= 16$$



$$\frac{1}{2} n = 7 \quad C = 15.$$

Object	Obj ₁	Obj ₂	Obj ₃	Obj ₄	Obj ₅	Obj ₆	Obj ₇
Profit	10	5	15	7	6	10	3
Weight	2	3	5	7	1	4	1

then find out Optimal Solⁿ

Solⁿ

$$1 \rightarrow \frac{10}{2} = 5 \rightarrow \textcircled{1}$$

$$2 \rightarrow \frac{5}{3} = 1.6 \rightarrow \textcircled{2}$$

$$3 \rightarrow \frac{15}{5} = 3 \rightarrow \textcircled{3}$$

$$4 \rightarrow \frac{7}{7} = 1 \rightarrow \textcircled{4}$$

$$5 \rightarrow \frac{7}{1} = 7 \rightarrow \textcircled{5}$$

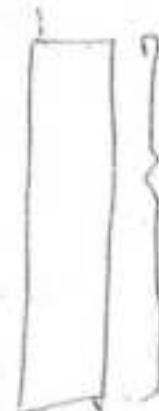
$$6 \rightarrow \frac{10}{4} = 4.5 \rightarrow \textcircled{6}$$

$$7 \rightarrow \frac{3}{1} = 3 \rightarrow \textcircled{7}$$

$$\left(\frac{1}{1}, \frac{2}{2}, \frac{1}{3}, \frac{0}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7} \right)$$

$$= 1 \times 10 + \frac{2}{3} \times 5 + 1 \times 15 + 0 \times 7 + 1 \times 6 + 1 \times 10 \\ + 1 \times 3$$

$$= \underline{\underline{55.33}}$$



15.4

10
3
2

Algo

(1) $\text{for } (i=1; i \leq n, i++)$ $\Rightarrow O(n)$
 $a[i] = p_i/w_i$

(2) Arranging Array in A in decreasing order of
 p_i/w_i ratio. $\Rightarrow O(n \log n)$

(3) $\text{for } (i=1, i \leq n, i++)$
 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.
 $\Rightarrow O(n)$
 $\underline{\underline{O(n \log n)}}$

(3) $\text{for } (j=1; j \leq n; j++)$

{ if ($c > 0$)
 { if ($c > w_j$)
 { $P = P + p_j, C = C - w_j$
 }
 { $P = P + p_j, C = 0$
 }

$\frac{1}{z} \rightarrow$ Solution space for greedy
 y_{10} Knapack is infinite

14/7/11

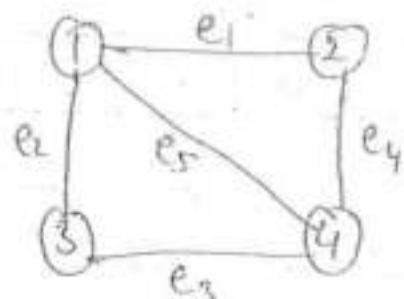
Minimum Cost Spanning Tree

$G(V, E)$

V = Set of vertices.

E = Set of edges.

1/2

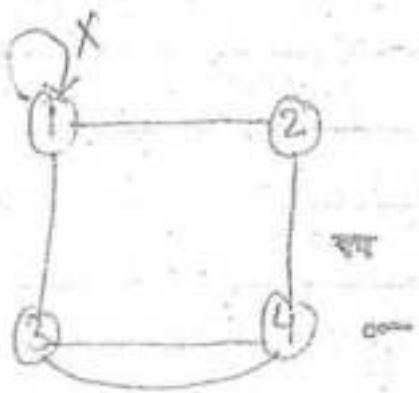


$$V = \{1, 2, 3, 4\}$$

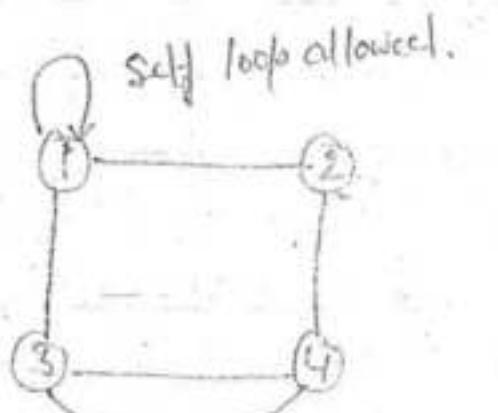
$$E = \{e_1, e_2, e_3, e_4, e_5\}$$

Types of Graph

Simple graph Multi Graph.



no-self loop
parallel edge.



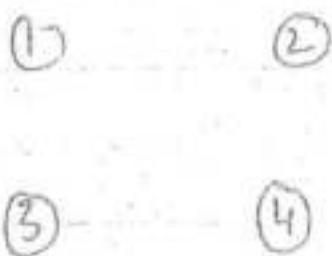
self loop allowed.
parallel edge

Simple Graph

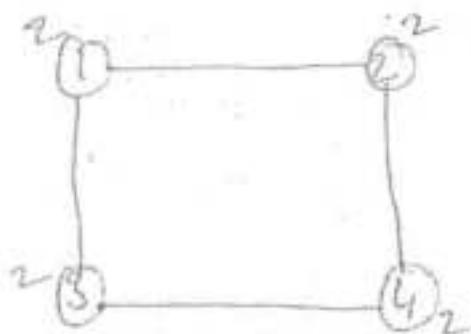
Types of Graph

- (i) Null Graph.
 - (ii) Regular "
 - (iii) Complete "
 - (iv) "
- } edges.

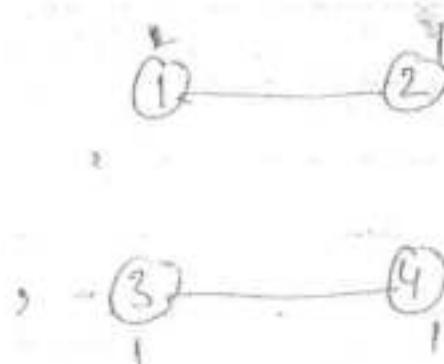
Null Graph



Regular Graph

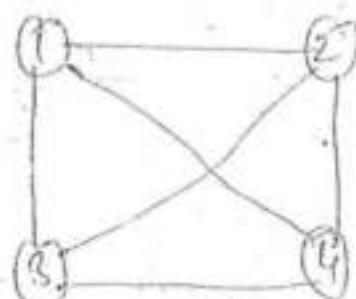


2-regular graph

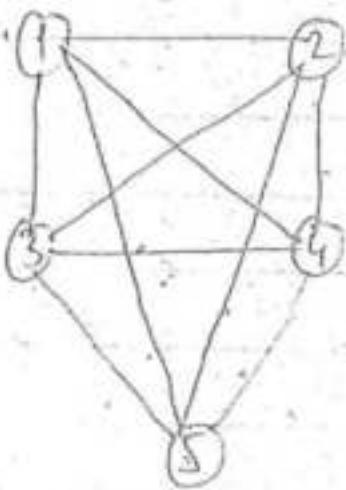


1 regular graph

Complete Graph



K_4 Complete graph



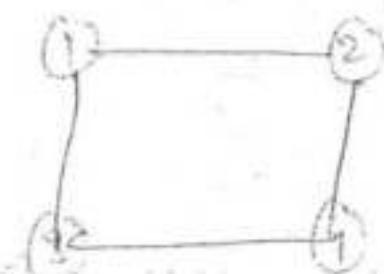
K5 Graph.

* Maxⁿ no of edges in n vertex simple graph.
~~Total edge~~ is equal to no of edges in complete graph.

$$E(K_n) = \frac{n(n-1)}{2}$$

* The no of edges in a n vertex K regular

$$= \frac{n \times k}{2}$$

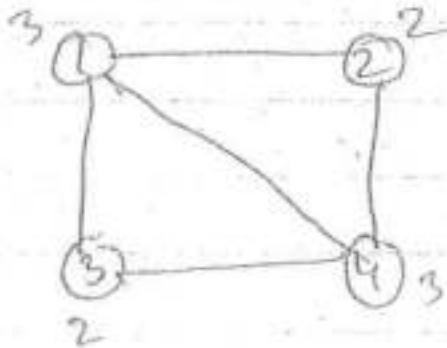


K4 graph

$(n-1)$

every complete 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.

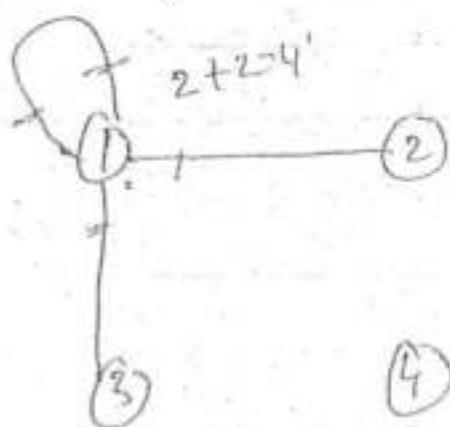


- (1,2) ✓
- (2,1) ✓

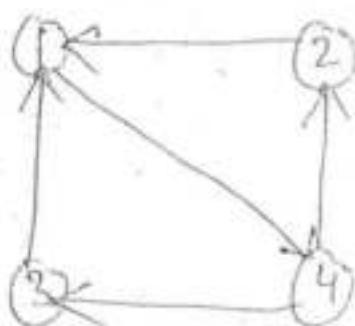


- (1,2) ✓
- (2,1) ✗

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 (-)

$$1^- = 1$$

$$2^- = 1$$

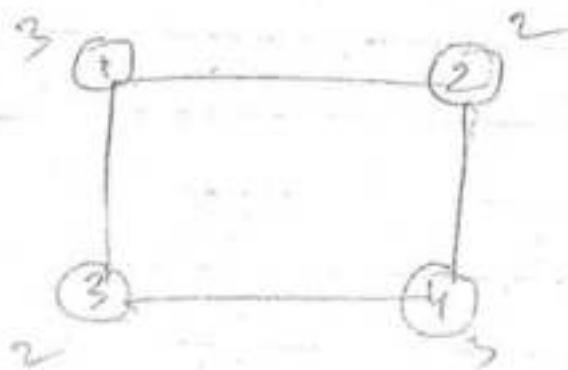
$$3^- = 1$$

$$4^- = 2$$

5

$$\sum_{i=1}^n (v_i)^+ = \sum_{i=1}^n (v_i)^- = |E|$$

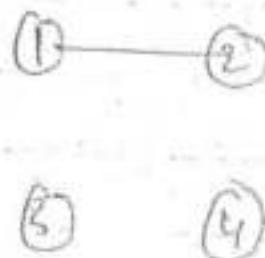
Hand Shaking Lemma



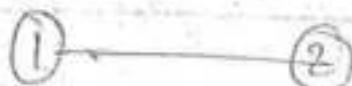
In a given simple graph no of vertices
the with odd degree is always even.



0 Vertices with
odd degree



2 vertices with odd
degree.



4- vertices with odd degree.

~~How many different simple graphs are possible with n -vertices ?~~

Soln

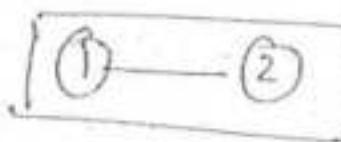
$n=1$



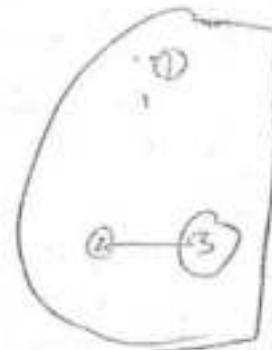
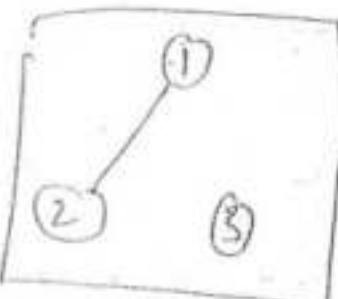
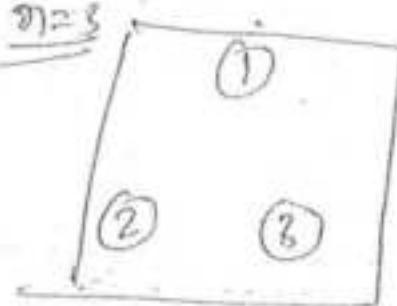
$n=2$

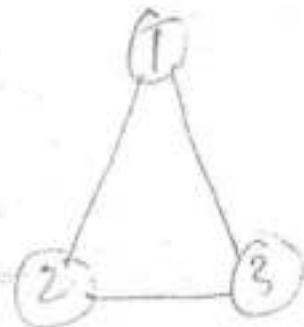
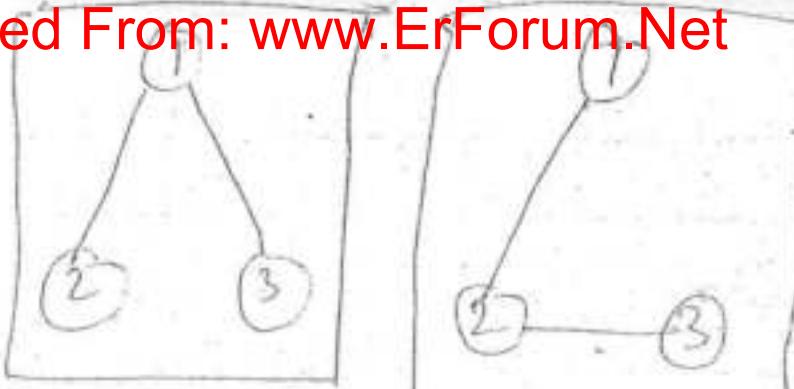
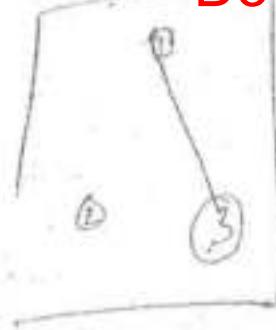


$\Rightarrow 2$



$n=3$





$\Rightarrow 0$

total simple graph if mode is

a) n

b) 2^n

c) 2^{n^2}

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 no of edge
written without repetition.

DESIGN
ANALYSIS
&
ALGORITHM

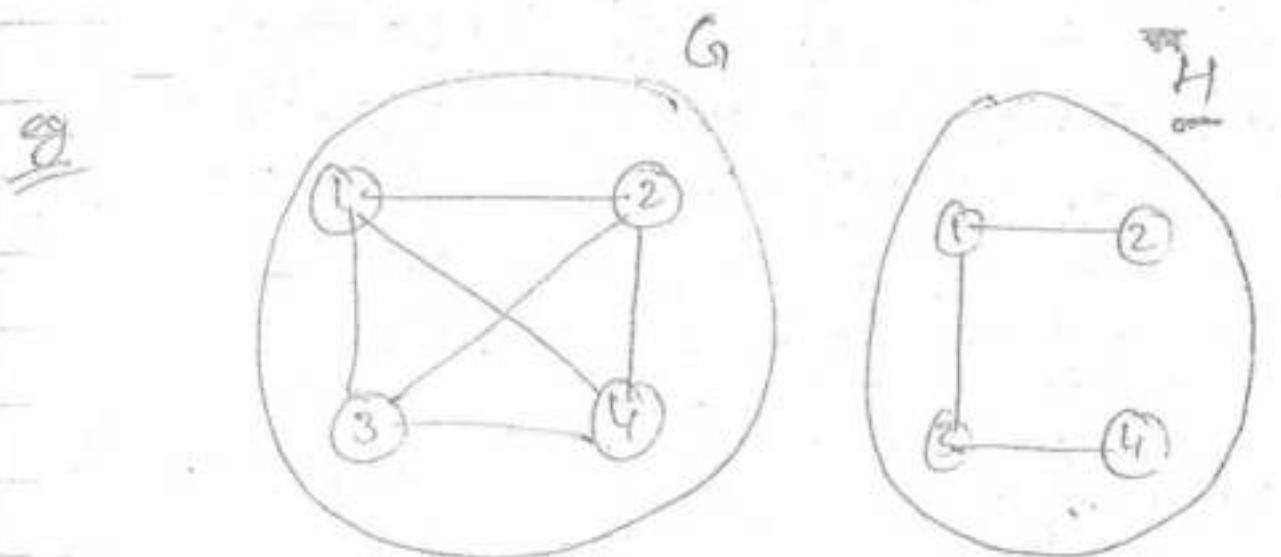
PART-II

(1)

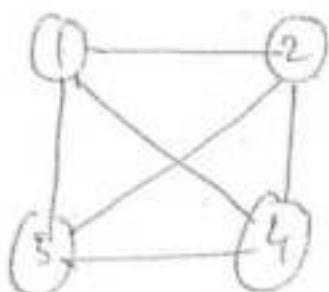
Spanning Tree

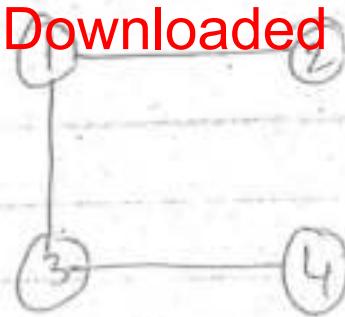
A connected subgraph H of a graph G is said to be spanning tree iff:

- i) It should contain all the vertices of G .
- ② " " " $(n-1)$ edges if G contains n vertices.



Find all possible different spanning trees for the following graph.

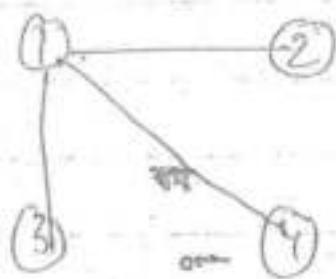




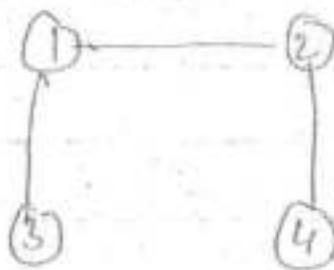
(I).



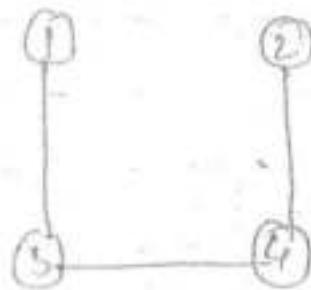
(II)



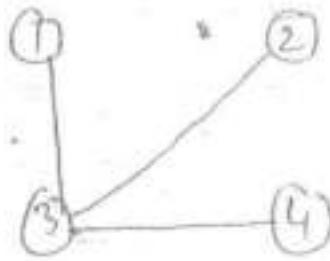
(III)



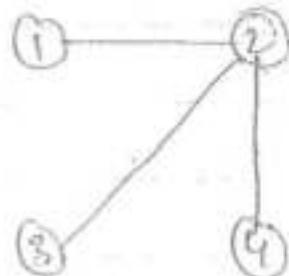
(IV)



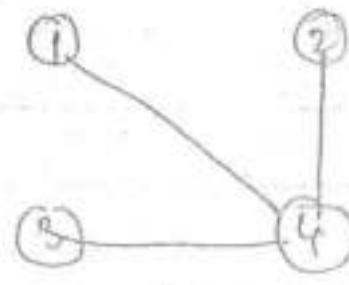
(V)



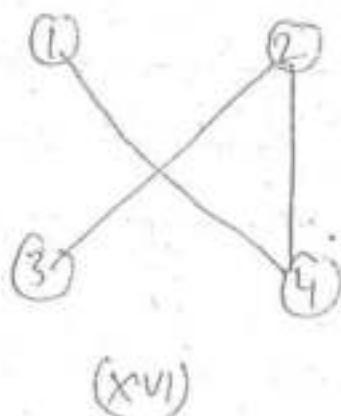
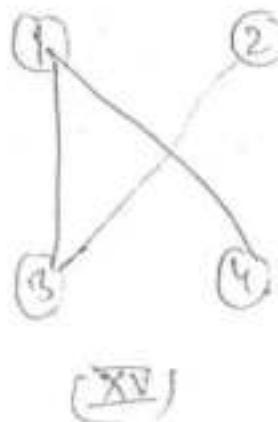
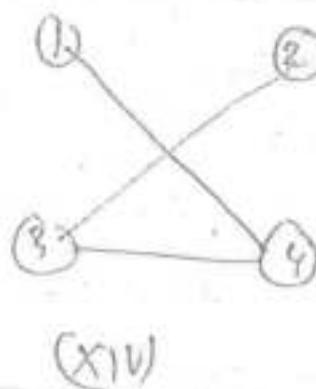
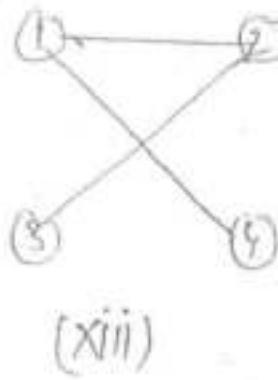
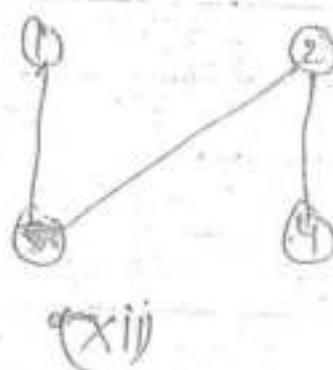
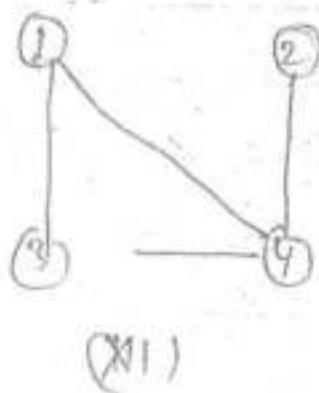
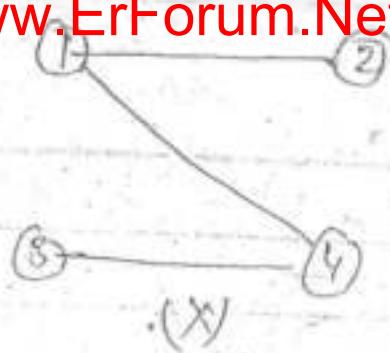
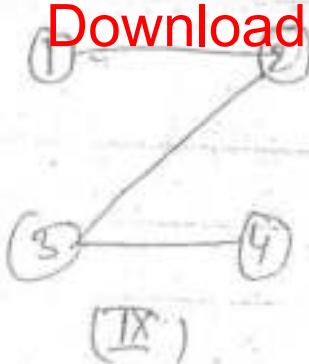
(VI)



(VII)

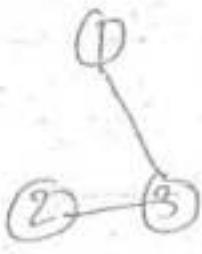
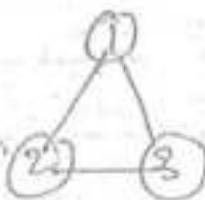


(VIII)



"Find all

K_3

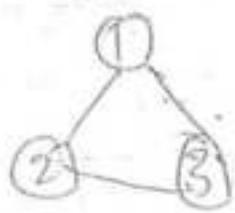


$$\begin{aligned} ST(K_3) &= s \cdot \binom{3^{3-2}}{} & ST(K_5) &= 125 \cdot \binom{5^{5-2}}{} \\ ST(K_4) &= 16 \cdot \binom{4^{4-2}}{} & ST(K_n) &= n^{n-2} \end{aligned}$$

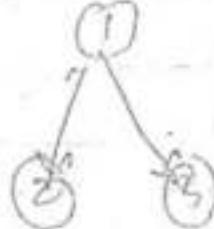
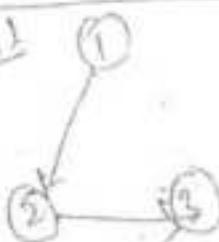
The no. of possible spanning trees in complete graph with n vertices

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

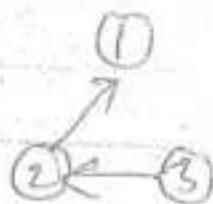
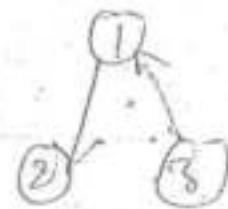
K_3 =



root 1



root 3

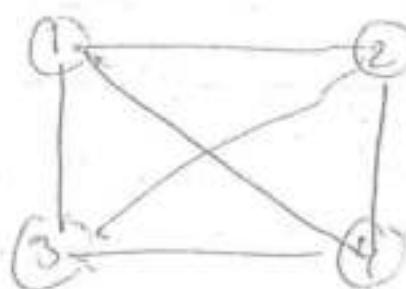


Note \Rightarrow The no. of root a spanning tree in a with n vertices complete graph.

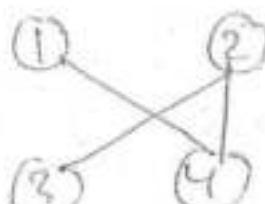
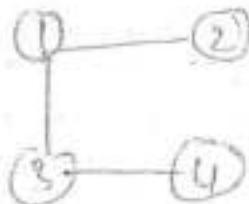
$$= n \times (n^{n-2})$$

K₄

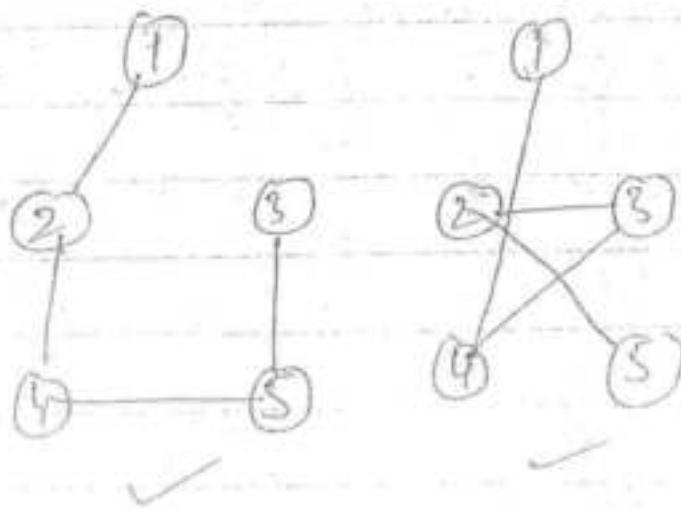
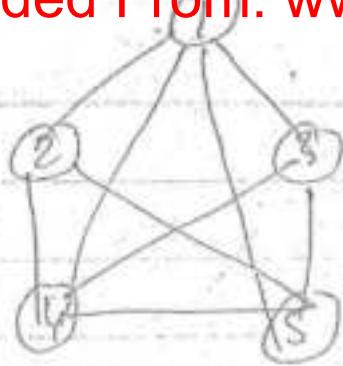
K₄



edge disjoint ST



K_5



$$K_3 = 1 \Rightarrow \left\lfloor \frac{3}{2} \right\rfloor = 1$$

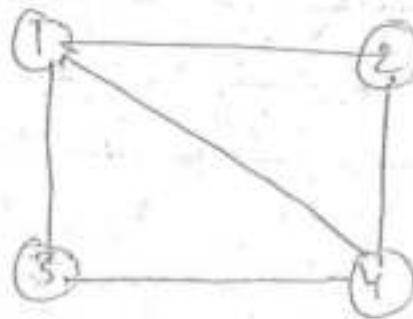
$$K_4 = 2 \Rightarrow \left\lfloor \frac{4}{2} \right\rfloor = 2$$

$$K_5 = 2 \Rightarrow \left\lfloor \frac{5}{2} \right\rfloor = 2$$

$$K_6 = 3 \Rightarrow \left\lfloor \frac{6}{2} \right\rfloor = 3$$

$$\text{EDST}(Kn) = \left\lfloor \frac{n}{2} \right\rfloor$$

Q. How many spanning tree are there for the following graph.

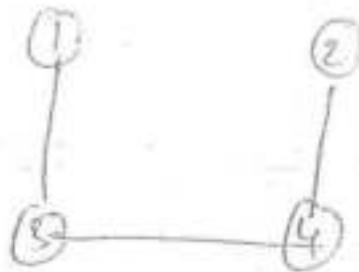
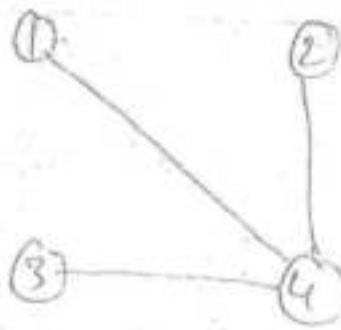
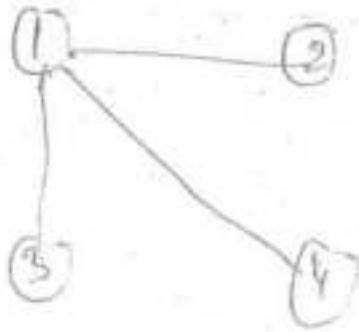


a) 16

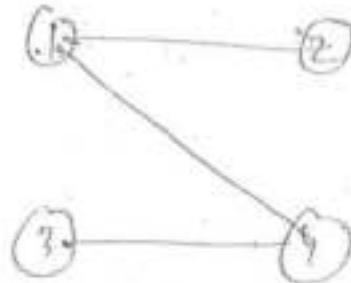
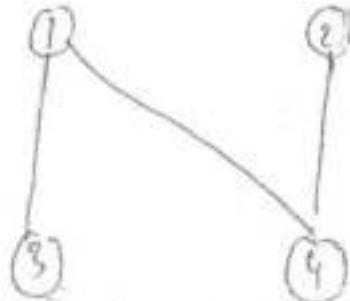
b) 10.

c) 8 ✓

d) 12



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 Dense graph
(more edge)

Adjacency Matrix $\Rightarrow O(V^2)$

Adjacency Graph Matrix M is good for
Those Graph which has more edges. Such
graph is called Dense Graph (more edge).

Those Graph which has less edges
such graph is called Sparse graph (less edge)

(2)

(i) replace all non diagonal 1's by 1's.

(ii) replace all diagonal 0's by its equal degree.

$$M = \begin{bmatrix} 1 & 1 & 2 & 3 & 4 \\ 3 & -1 & -1 & -1 & -1 \\ -1 & 2 & 0 & -1 & -1 \\ -1 & 0 & 2 & -1 & -1 \\ -1 & -1 & -1 & 3 & -1 \end{bmatrix}$$

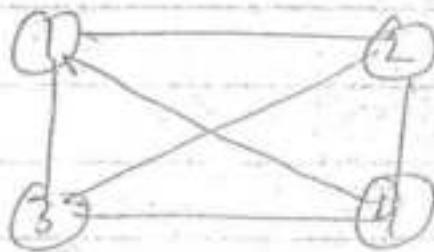
(3) Find co-factor of any element which will give no of spanning tree.

$$\left| \begin{array}{ccc} 2 & 0 & -1 \\ 0 & 2 & -1 \\ -1 & -1 & 3 \end{array} \right|$$

$$2(6-1) - 0(\cdot) - 1(0+2)$$

$$= 10 - 2 = 8$$

Find no of spanning tree for the following graph using Kirchhoff Theorem



~~84ⁿ~~

$$\textcircled{1} \quad M_2 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

$$\textcircled{2} \quad M = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

\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 SCT.

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 CST must contain e_{\min} .

(b) If e_{\max} is in MST, then its removal
must disconnect G .

(c) No min^m CST contain e_{\max}

(d) G ~~contains~~ has unique MST

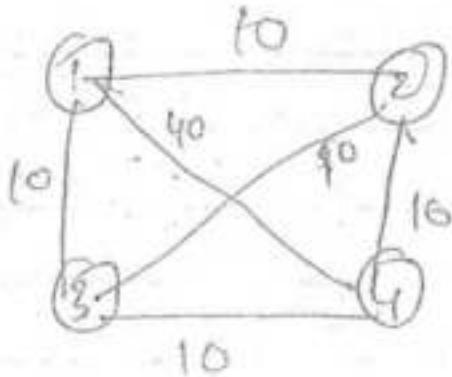
P Let G be an undirected connected graph with n vertices.

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 δ min^m est should contain an edge with w .
- X d) Every msct should contain edge e .

Sol^m



$$w = 10$$

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.

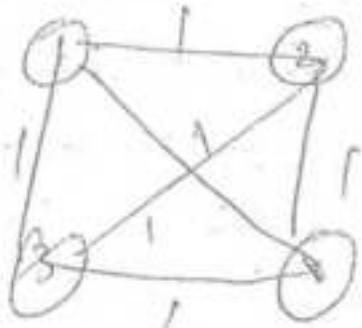
- i) all diagonal elements are 0's. {means}
- ii) all non-diagonal elements are 1's. } complete graph.

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 $n-1$

soln

K_4



16 \Rightarrow MST

each of cost $\underline{\underline{=3}}$

$K_n \Rightarrow n^{n-2}$ MST

each of cost $\underline{\underline{n-1}}$

\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.

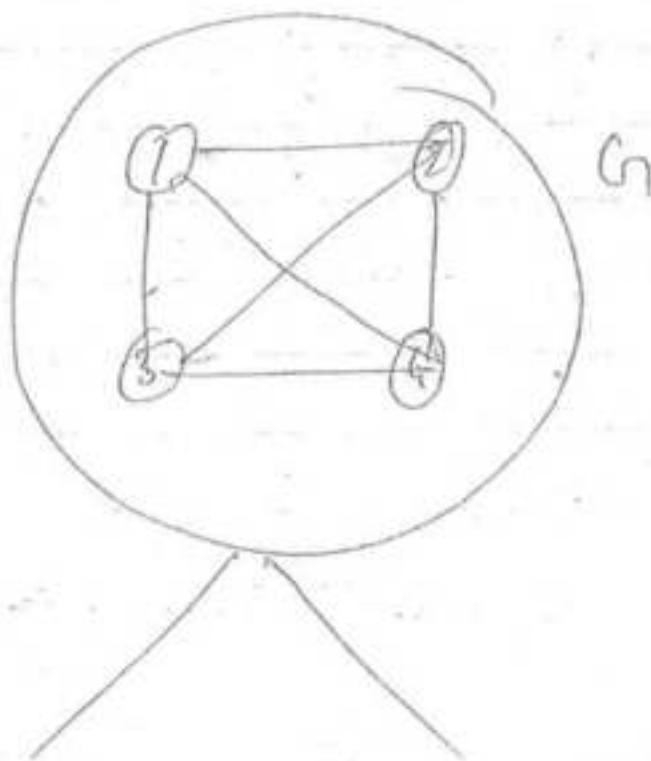
(i) $(T - e) \cup e'$

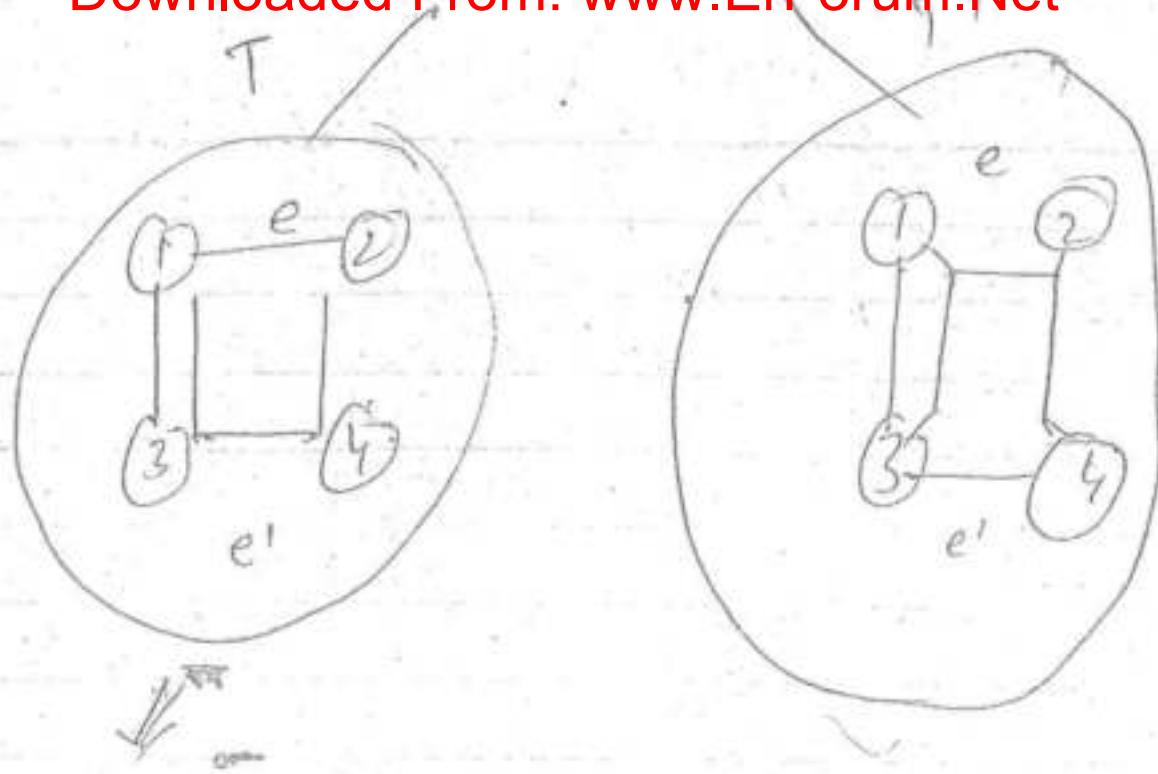
(ii) $(T' - e') \cup e$

~~(iii)~~ both T and T' are still ST

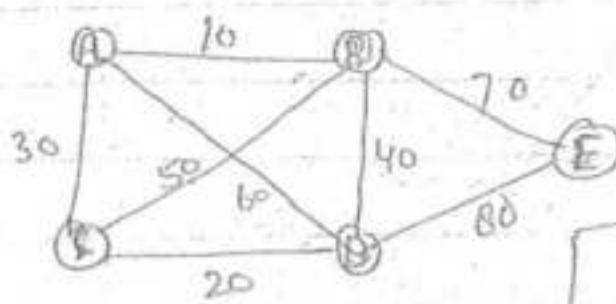
- a) only T is ST
- b) only T' is ST
- c) both are not.

Solⁿ

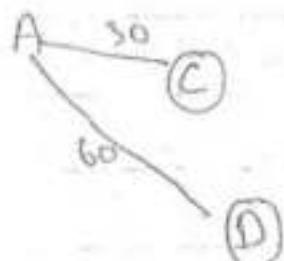
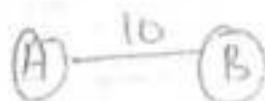
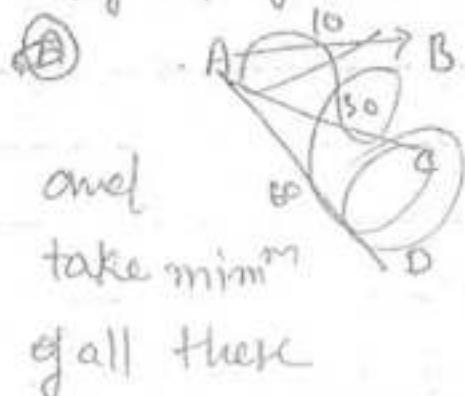




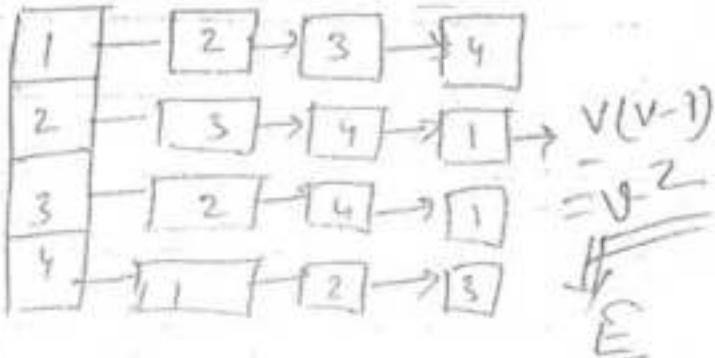
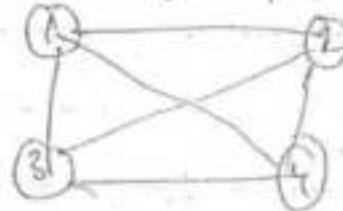
Prim's Algo



(i) Take any Vertex & find adjacent of that vertex



Adjacency list representation



① ② no s edge

③ ④

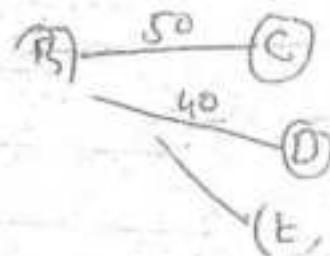
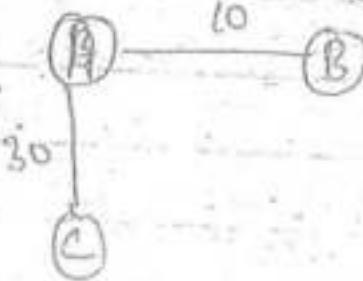


$O(v)$

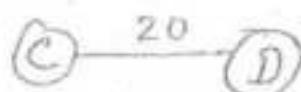
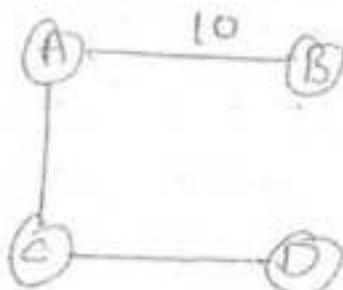
$O(v+e)$

No

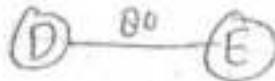
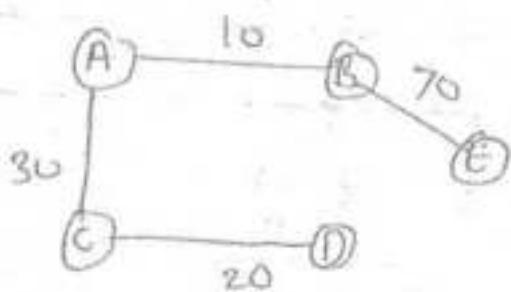
of the new vertex find adjacent and min^m
among vertices & previous



- ③ for the new vertex find adjacent and take min^m
of three adjacent and previous all adjacent.



④ for the



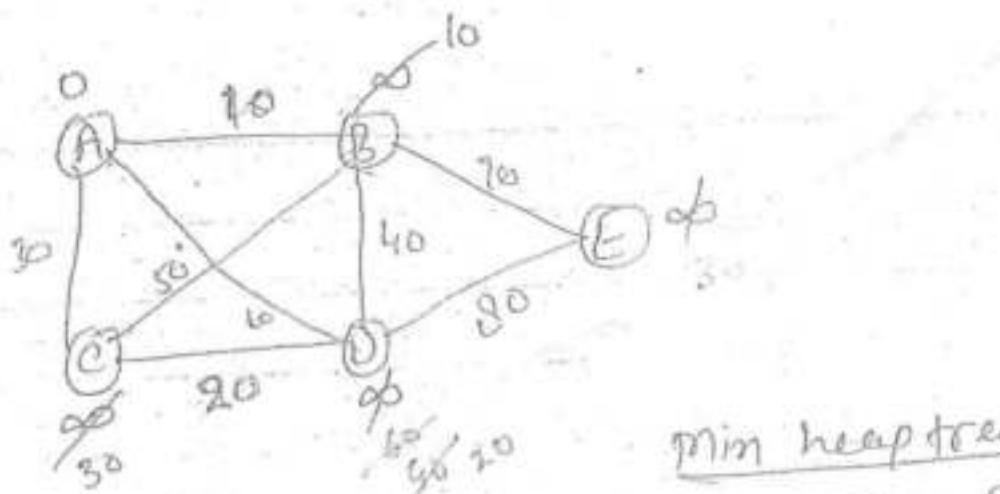
BD X.

BC X

AD X

Note

Prim's & Kruskal algo Both generate same cost
Can generate diff. MST.
But in B/w algo Prim's doesn't have any
any forest and Kruskal have forest

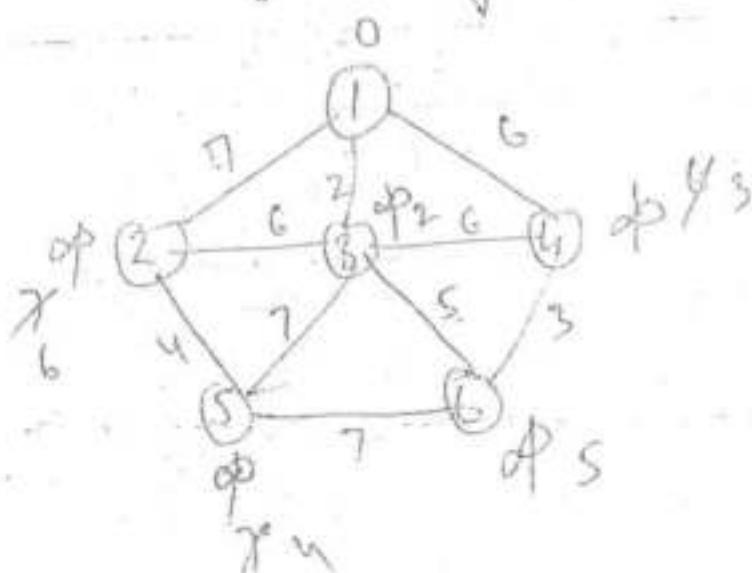


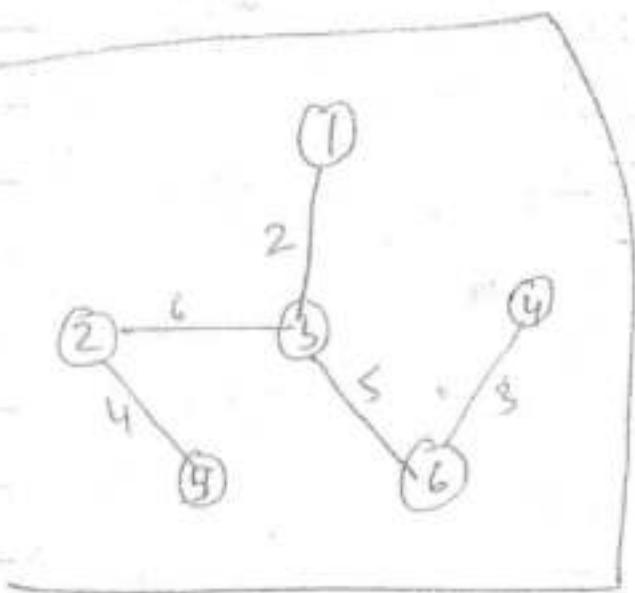
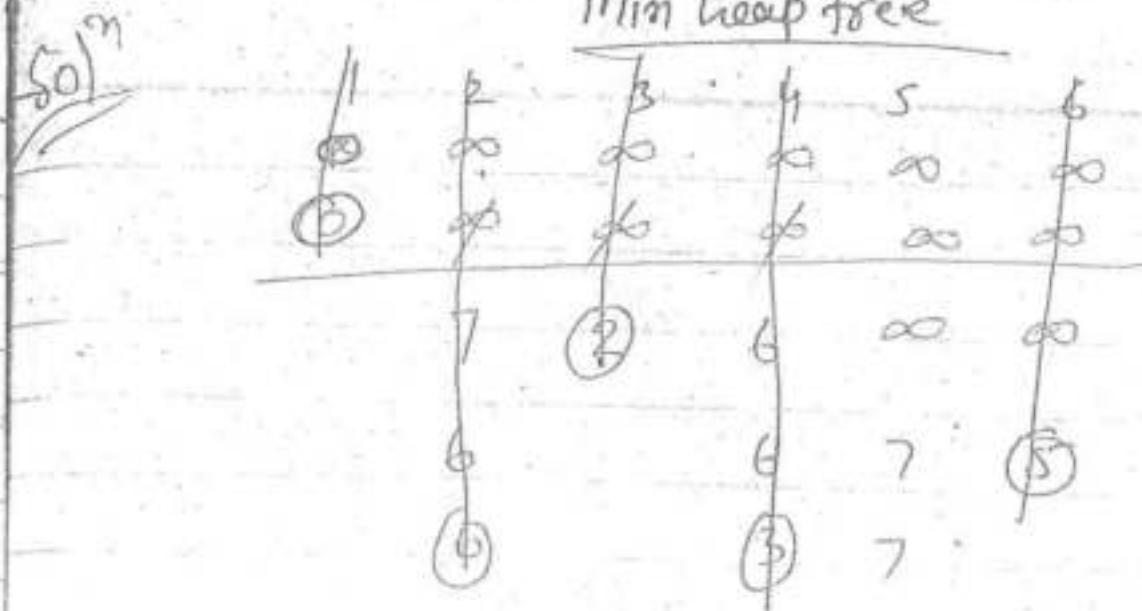
min heap tree

A	B	C	D	E
∞	∞	∞	∞	∞
0	∞	∞	∞	∞
10	30	60	∞	70

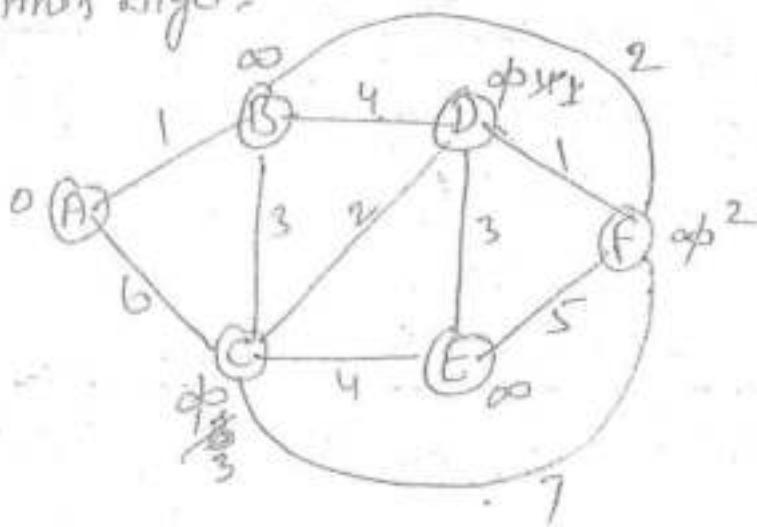


Consider the following graph.



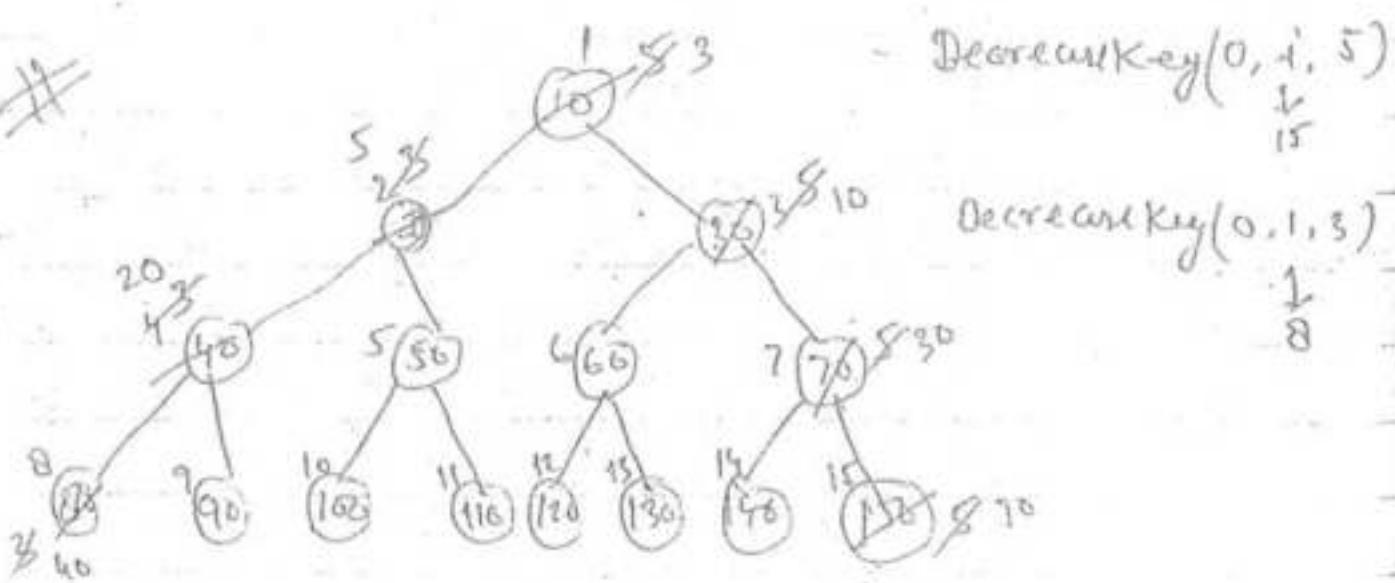
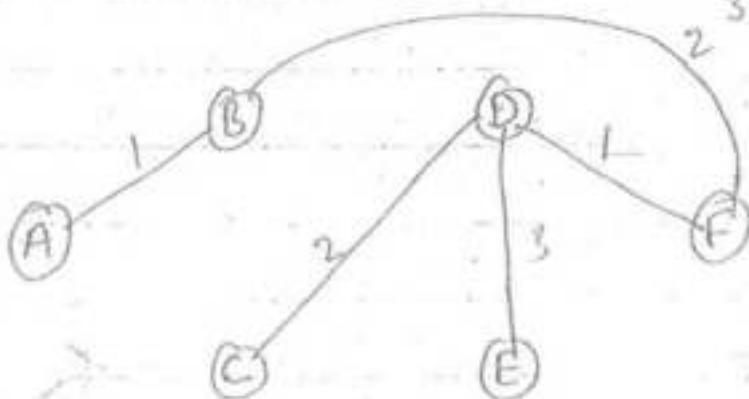


Q. Construct an m-CST for the following Graph using Prim's Algo.



SFM

A	B	C	D	E	F
∞	∞	∞	∞	∞	∞
0	1	0	6	0	0
			2	0	0
			3	5	0
			4	∞	2
			5	3	3
			6	3	3



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)

- {
- ① for each vertex $v \in V$
 $v_{min} = \infty$ $\Rightarrow O(V)$
 - ② $S, min = 0$ $\Rightarrow O(1)$
 - ③ Create min heap tree (Q) for all vertices. $\Rightarrow O(V)$

④ while ($Q \neq \emptyset$)

- {
- $u = extract\min(Q)$
- for all v adjacent u
- if ($d(u, v) < v_{min}$)
- $v_{min} = d(u, v)$
- $\Rightarrow O(\log V)$
- decrease key of
 $(\log V)$
- }

$$= \cancel{O(V)} + \cancel{O(E)} + O(V) + V(\log V + (V-1)\log V)$$

$$= V \log V + V^2 \log V$$

$$= V \log V + E \log V$$

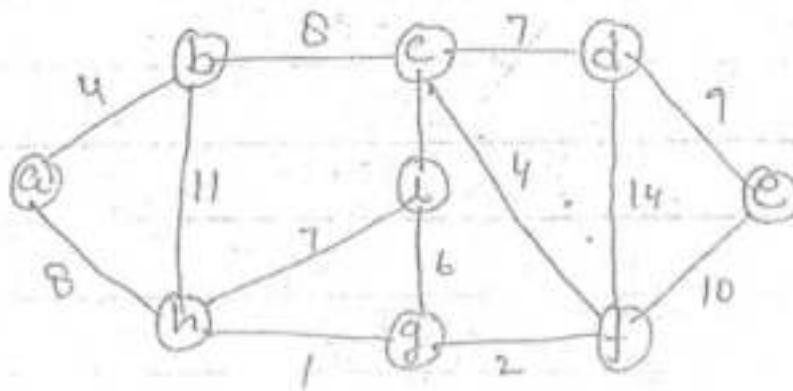
$$= \boxed{O(V+E) \log V}$$

using min heap

$$= \boxed{O(E + V \log V)}$$

using Fibonacci min heap

Consider the following Graph.

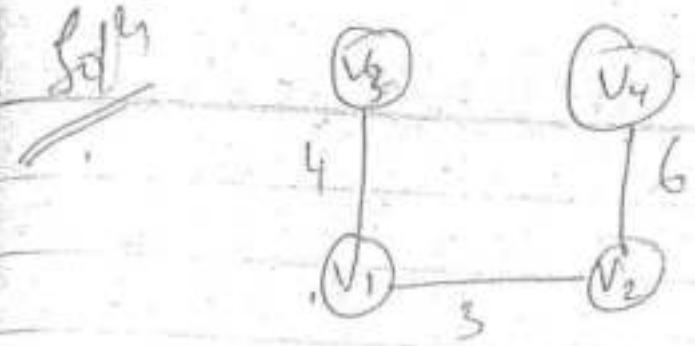


Which one of the following sequence of edges of mCST is not true when the algo is started.

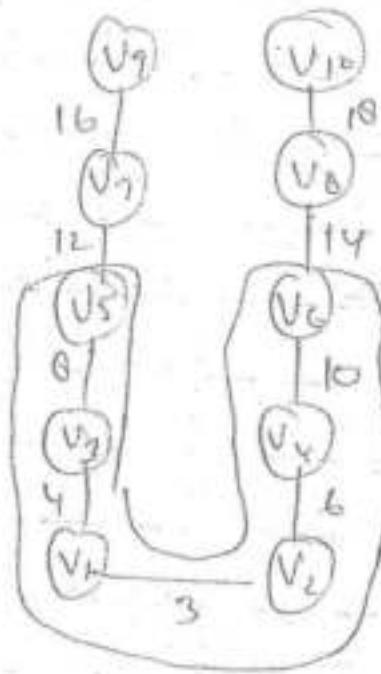
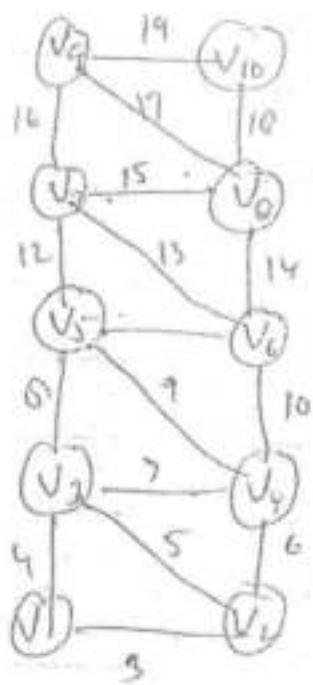
- a) (a,b) (b,c) (c,i) (c,f) (f,g) (g,h) (c,d) (d,e)
- b) (a,b) (a,h) (h,g) (g,f) (f,c) (c,i) (c,d) (d,e)
- c) (a,b) (b,c) (f,h) (c,i) (f,g) (g,h) (c,d) (d,e)
- d) (a,b) (b,c) (h,g) (f,g) (c,i) (c,f) (c,d) (d,e)

disconnected

Graph.
so. Prim's not possible



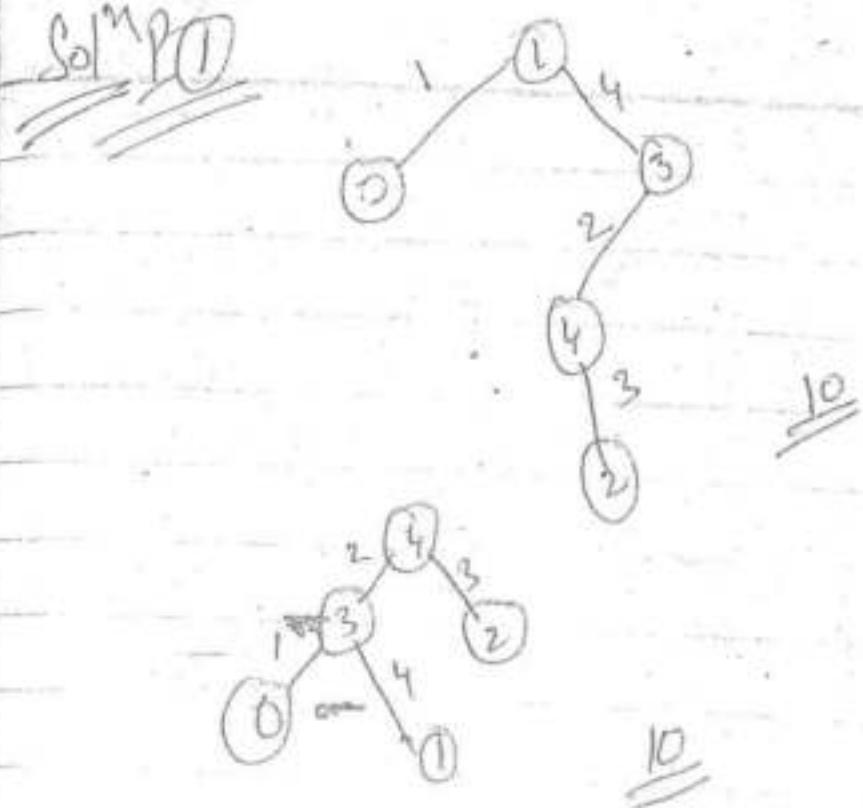
13



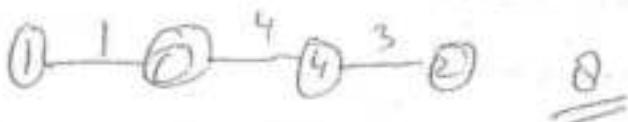
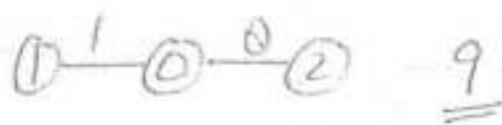
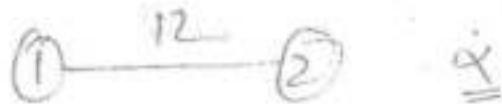
P Consider a ^{complete} undirected graph $G(V, E)$ with vertex set $\{0, 1, 2, 3, 4\}$. Entry w_{ij} in the matrix W below is the weight of the edge (i, j) .

$$W = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \left[\begin{matrix} 0 & 1 & 0 & 1 & 4 \\ 1 & 0 & 12 & 4 & 9 \\ 0 & 12 & 0 & 7 & 3 \\ 1 & 4 & 7 & 0 & 2 \\ 4 & 9 & 3 & 2 & 0 \end{matrix} \right] \end{matrix}$$

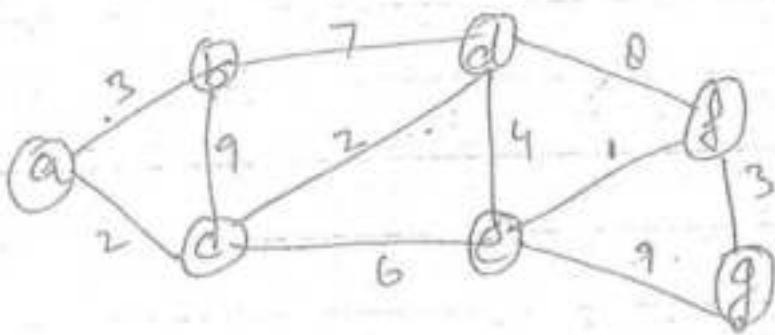
- ① What is the min^m possible weight of MST of SF T for the above graph, such that vertex 0 is leaf node in T
 a) 7, b) 8, c) 9, d) 10
- ② What is the min^m possible weight of a path P from vertex 1 to vertex 2 for the above graph G such that P contains atmost 3 edges.
 a) 7, b) 8, c) 9, d) 10



P(2)



= Consider the following graph.

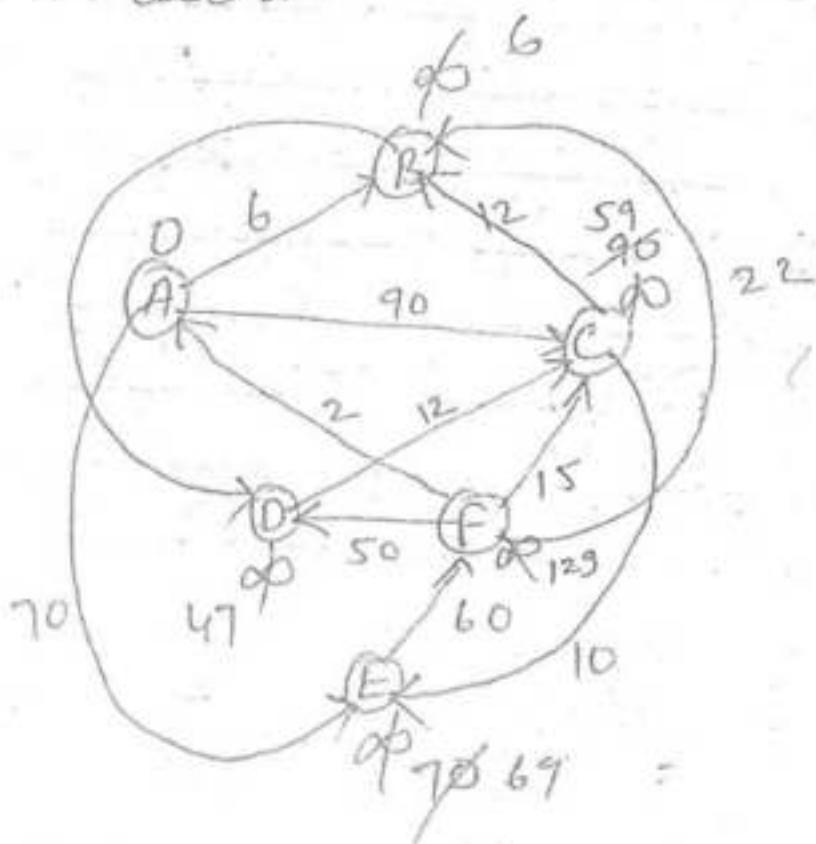


If Prim's & Kruskal algo is used to generate MST of above graph, then which one of the following is true :

- a) Prim's is 25% decrease comparing with Kruskal,
- b) Kruskal is 10% increase comparing with prim's.
- c) Kruskal is 7.3% decrease with Prim's algo
~~100%~~
- d) Prim's decrease 10% increase with Kruskal.

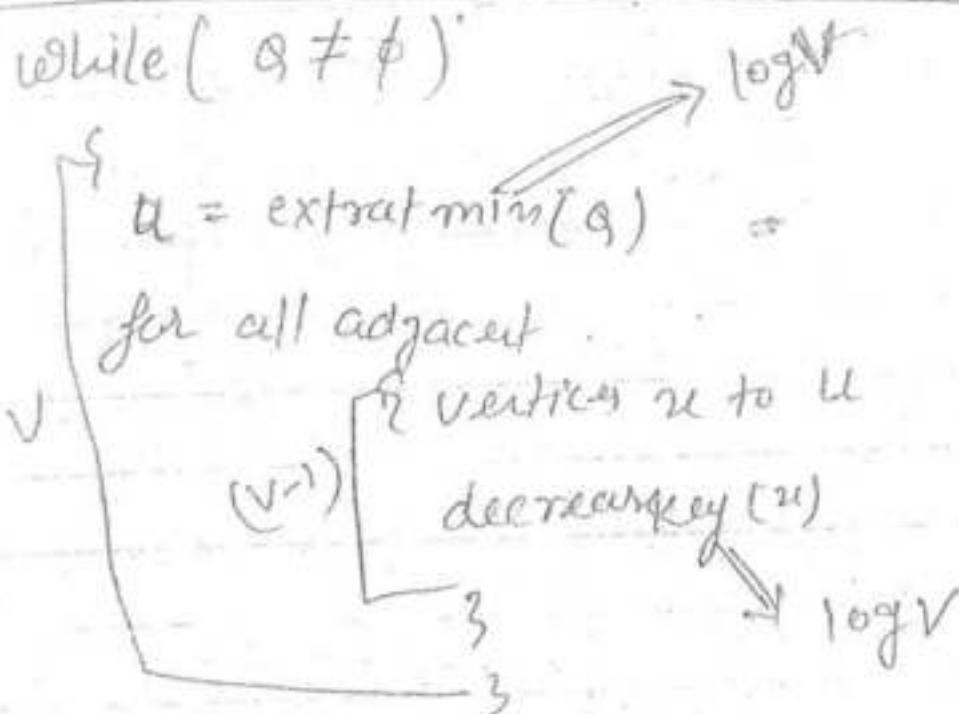
~~Ans~~ Both Prim's and Kruskal will give same answer

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	0	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						





apply $|V|-1 = 5-1 = 4$ time ($i=4$ time).

Step

Wt

	A	B	C	D	E	
0	0	∞	∞	∞	∞	9
0	0	-1	∞	∞	∞	$i=1$
0	0	-1	4	∞	∞	
0	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 found~~ exists.

weighted cycle not form source to every vertex)

$O(VE)$

$O(VV^2) = O(V^3)$

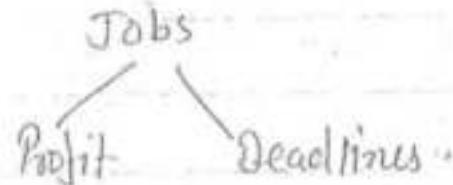
$O(V \cdot V) = O(V^2)$

eq

Job Sequencing with Deadline

- ① Single process is available. (at a time for each job)
- ② Interleaving is not allowed.
- ③ Arrived time all over the job are same.
- ④ Every job requires 1-unit of time.

$n=4$



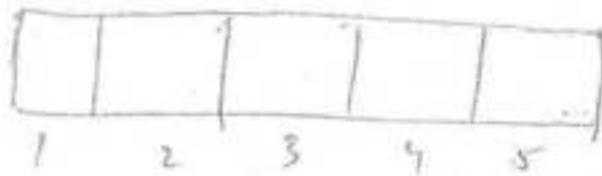
Jobs	J_1	J_2	J_3	J_4
Profit	P_1	P_2	P_3	P_4
Deadline	d_1	d_2	d_3	d_4

$n=6$

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

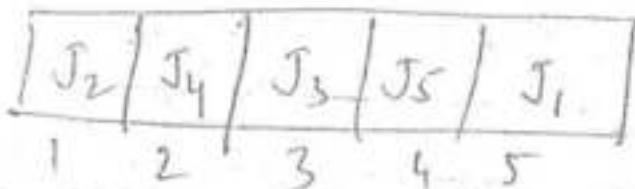
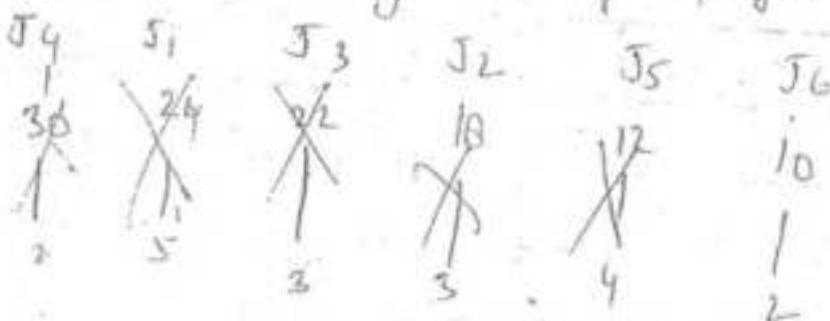
Solⁿ
① { }

② { {J₃, J₂}, {J₁} } { {J₃} }, { {J₄} }, { {J₅} }, { {J₆} }.



Solⁿ

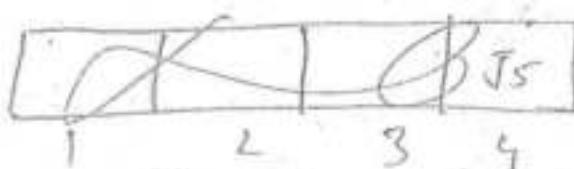
↓ Decreasing order of profit



$$= 10 + 30 +$$

$$= 106$$

Profit	10	20	15	5	80
Deadline	3	3	3	4	4



J ₅	J ₂	J ₃	J ₁	J ₄
80	20	15	10	5
1	1			
4	3	3	3	4

J ₁	J ₃	J ₂	J ₅
1	2	3	4

$$= 10 + 15 + 20 + 80 = 125 \quad //$$

3) $n=9$

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.

c) J_5, J_2, J_8 " left only.

~~d) J_4, J_6~~ " "

② max^m Profit

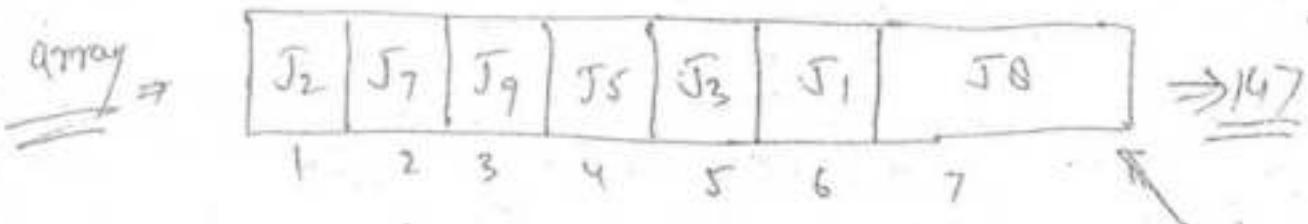
~~a) 147~~ b) 165

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	15	10
1	1	1	1	3	4	7	X	X
5	3	2	2	5	7	7	5	7



Algo

① Sort all the jobs in decreasing order of profits. $\Rightarrow O(n \log n)$ $O(n)$

② Find max^m deadlines in the given array of ~~jobs~~ n -deadline and take the same size array and start from RHS of the array.

For every job which contain $\{ \text{deadline} \geq d \}$

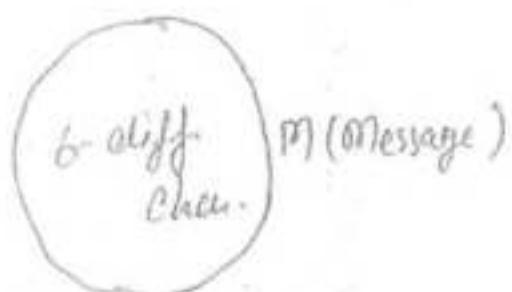
$$O(n^2)$$

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

Huffman Coding:

- (1) Binary tree
- (2) Encoding Technique
- (3) Data compression mechanism.

eg



100-character A (ASCII) 3-bit (not ASCII)

a - 50	M
b - 5	
c - 4	
d - 10	
e - 30	
f - 1	

D/I destinat.

a - 50×7

b - 5×7

c - 4×7

d - 10×7 compress

e - 30×7

f - 1×7

700 bits.

a - 000 = 50×3

b - 001 = 5×3

c - 010 = 4×3

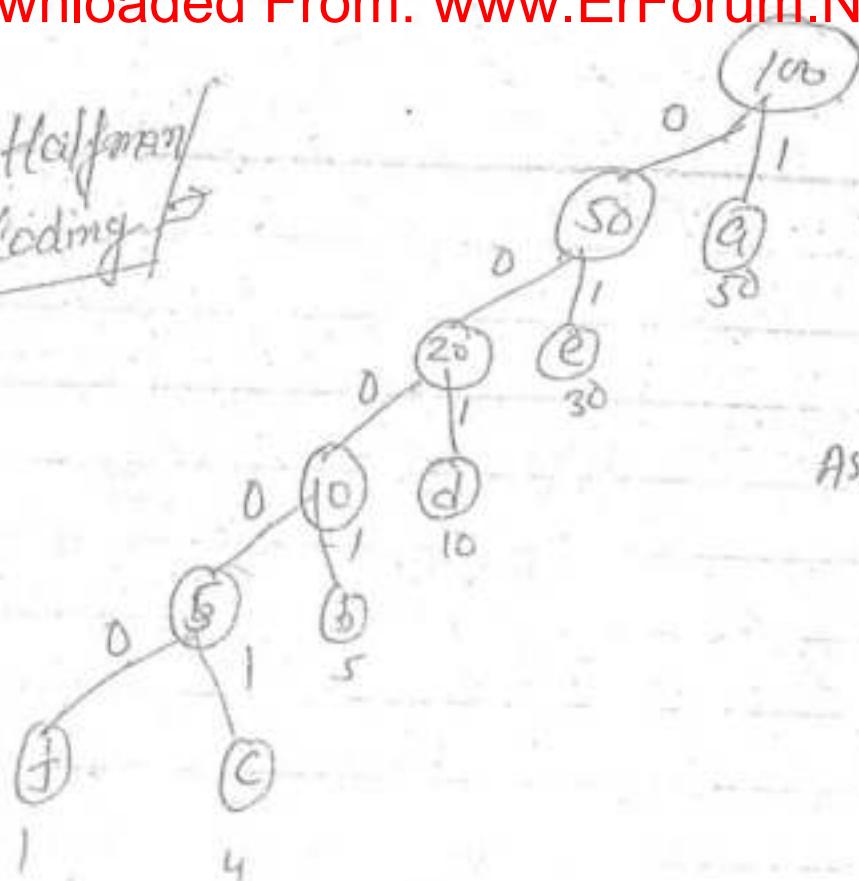
d - 011 = 10×3

e - 100 = 30×3

f - 101 = 1×3

300 bits

Q. Huffman Coding



Assume Left = 0

Right = 1

$$a = 1$$

$$b = 0001$$

$$c = 00001$$

$$d = 001$$

$$e = 01$$

$$f = 00000$$

↑ 50

105 bit to perform for
100 char

$$100 \text{ char} = 105 \text{ bits}$$

$$1 = \frac{105}{100} = 1.05 \text{ bits/char}$$

after 2ⁿ compression

$$a = 1 \times 50 = 50$$

$$b = 5 \times 4 = 20$$

$$c = 4 \times 5 = 20$$

$$d = 10 \times 3 = 30$$

$$e = 30 \times 2 = 60$$

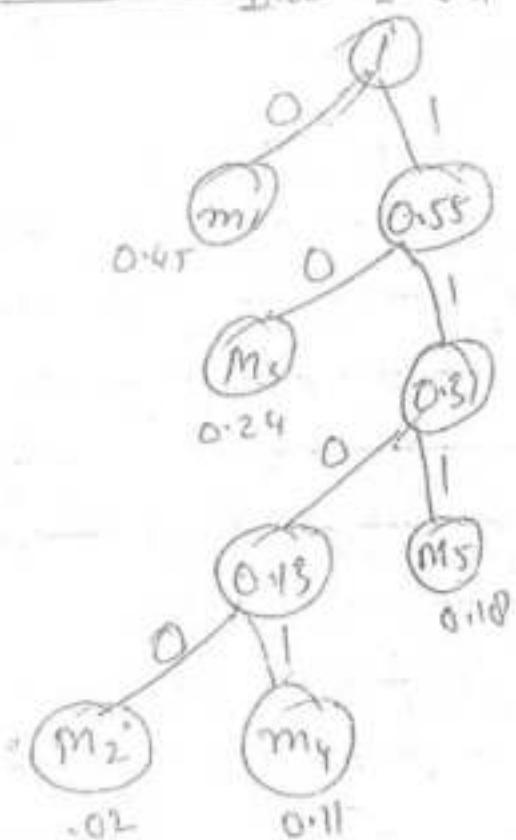
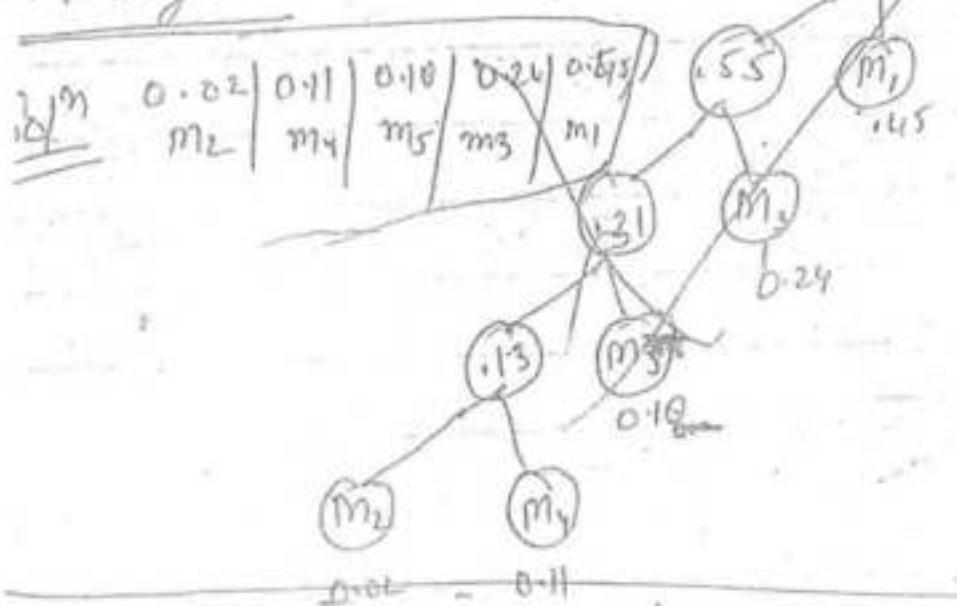
$$f = 1 \times 5 = 5$$

$$\underline{185}$$

This is Called Huffman Coding

0.45 0.02 0.11 0.18

Using Huffman coding find Encoded form of message.



$$\begin{aligned}
 m_1 &= 0 \\
 m_2 &= 1100 \\
 m_3 &= 10 \\
 m_4 &= 1101 \\
 m_5 &= 111
 \end{aligned}$$

$$\begin{aligned}
 \text{Total no of bit to transfer} \\
 &= 0.45 \times 1 + 0.02 \times 4 + 0.24 \times 2 \\
 &\quad + 0.11 \times 4 + 0.18 \times 3 \\
 &= 1.99 \text{ bits}
 \end{aligned}$$

Avg no of bit to transfer

$$\frac{1.99}{5} = 1.99 \text{ bits/msg}$$

Probability $\rightarrow \frac{1}{2}, \frac{1}{2^2}, \frac{1}{2^3}, \frac{1}{2^4}$ and $\frac{1}{2^5}$ respectively

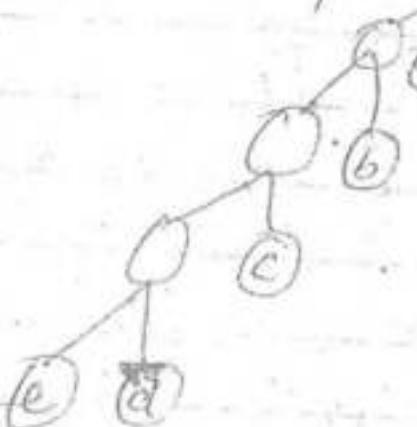
Q) Which one of the following is Huffman coding for the letters a, b, c, d & e respectively?

a) 11, 10, 011, 010, 001

b) 0, 10, 110, 1110, 1111

c) 1, 10, 01, 001, 0001

d) 110, 100, 010, 000, 001



Q) What is the avg no of bits $\overset{\text{avg}}{\text{avg}}$ required per message.

a) 3

b) 3.75

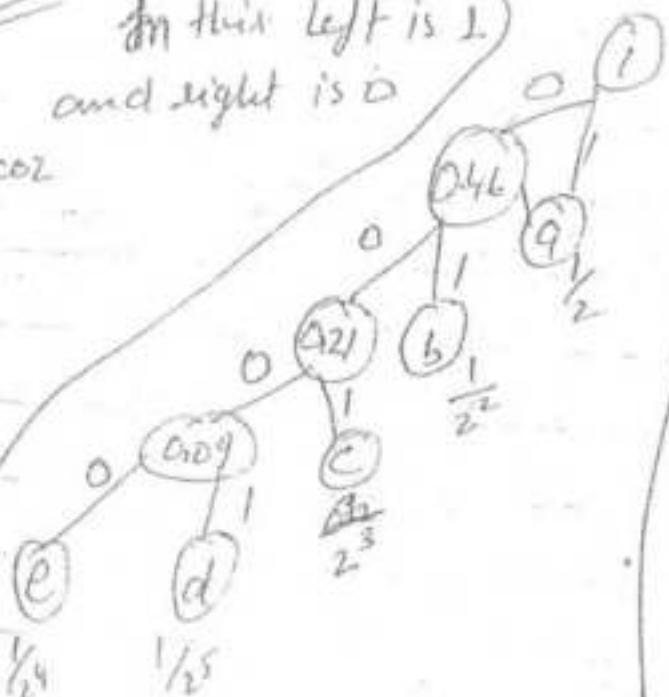
c) 2.10

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

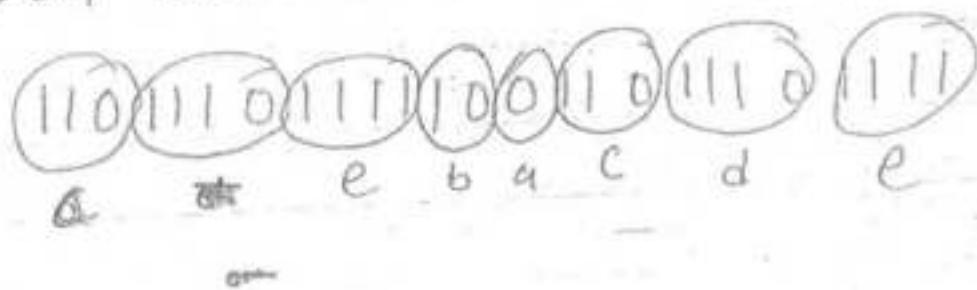
b_i = # of bits to represent msg M_i

represent msg one

$$= \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^5}$$

$$= 1.78 \text{ bits/msg}$$

Q3 If the encoded msg is 110110111100110110111
what will be the equivalent Decoded msg.

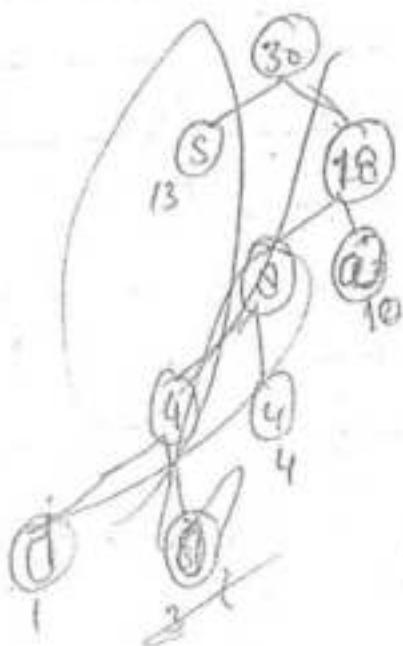


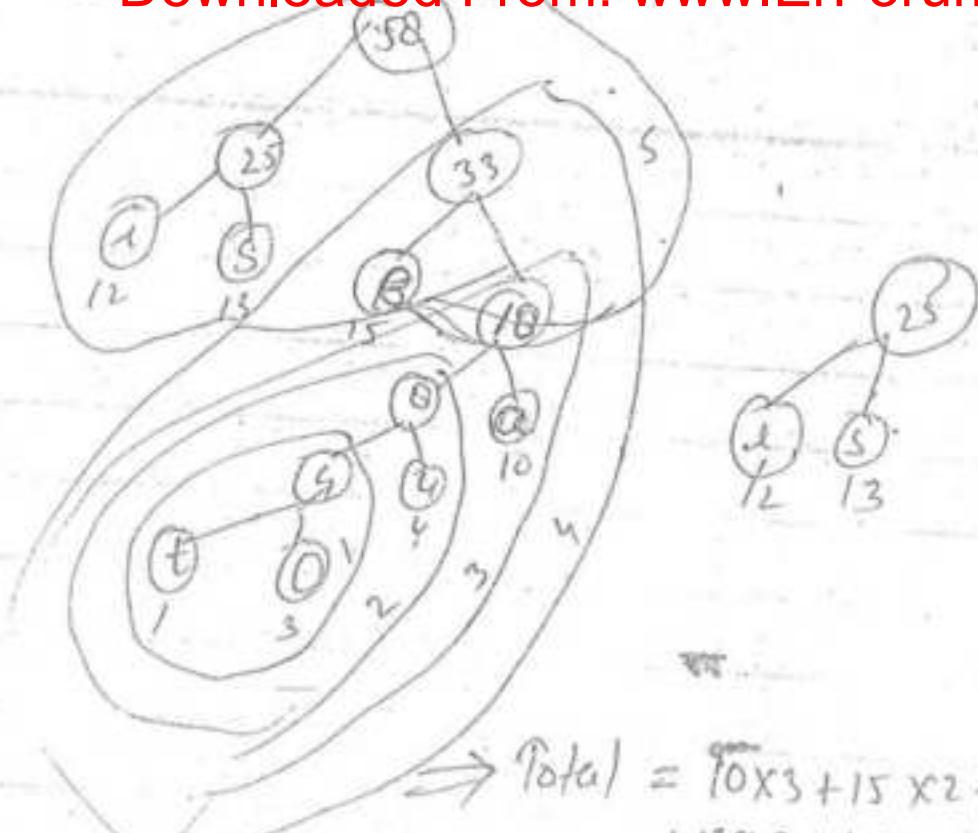
Q4 A file contain char a, e, i, o, u, s and d.
and frequency 10 15 ~~18~~ 3 4 ~~15~~ 1

If we use Huffman coding for data coding
what will be the avg code length?

- a) ~~146/58~~ b) $\frac{149}{58}$ c) $\frac{140}{58}$ d) none

Soln





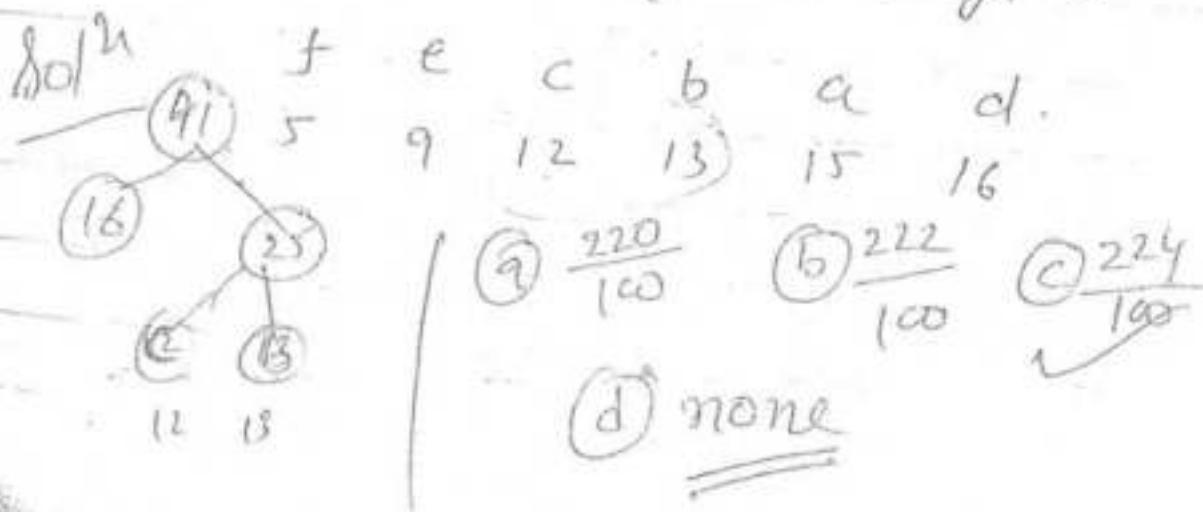
$$\rightarrow \text{Total} = 10 \times 3 + 15 \times 2 + 12 \times 2 + 3 \times 5 \\ + 13 \times 2 + 1 \times 5 \\ = 146$$

$$\text{Avg} = 146 / 58$$

P Consider the frequencies for the chars. a, b, c, d, e, 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.

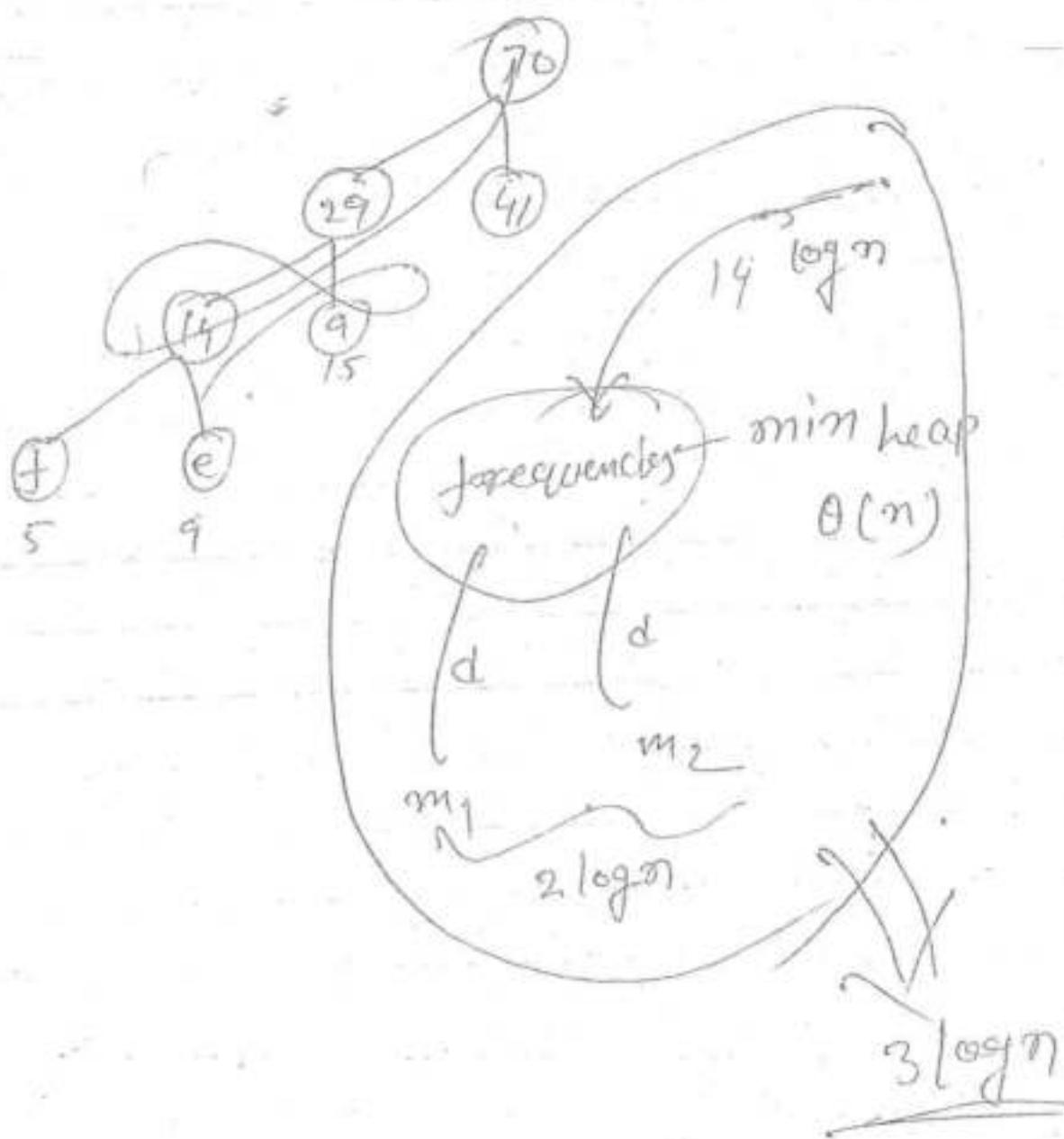


~~Service no 7017~~

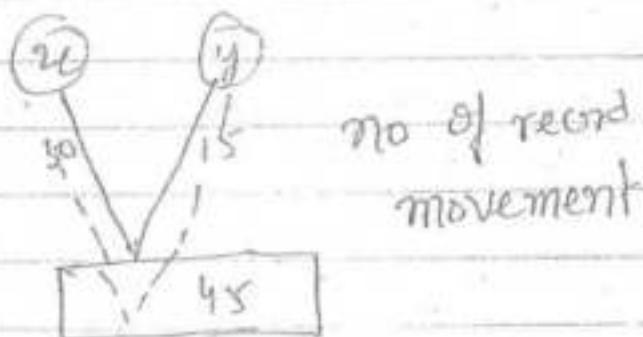
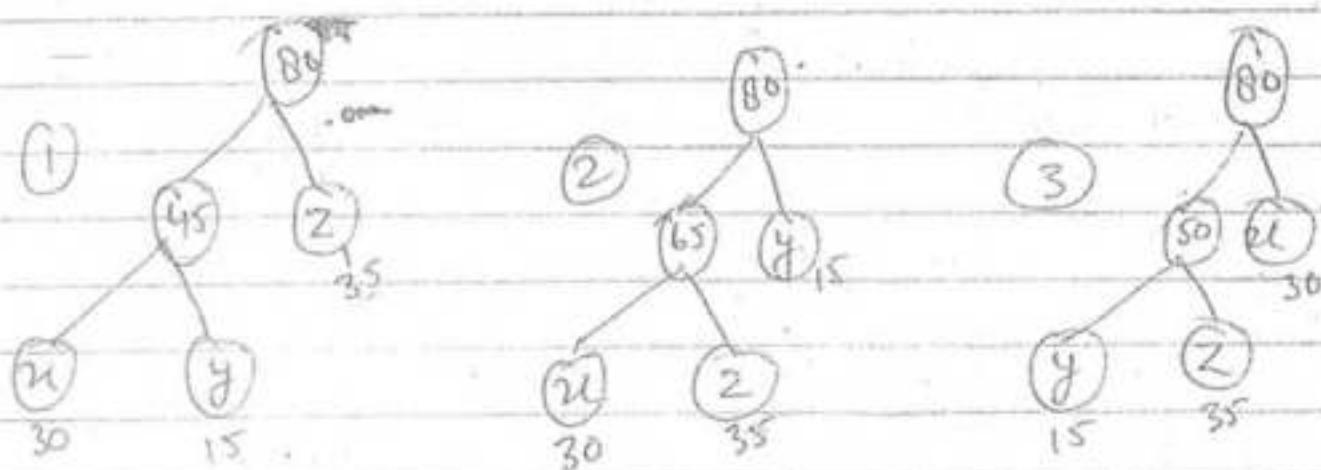
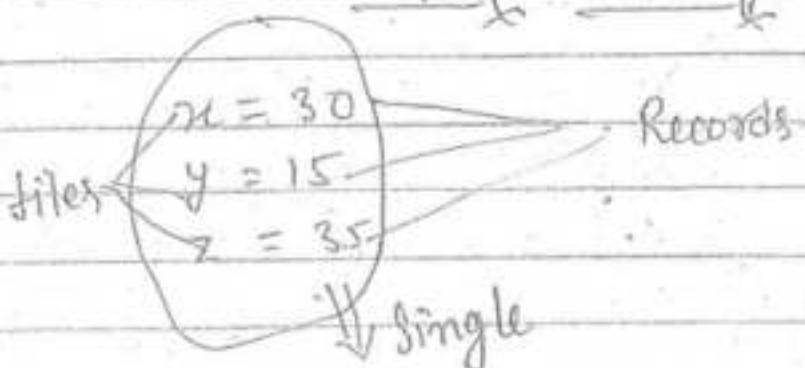
~~11 Name DAI-LKC~~

~~Type Sleeper Coach~~

~~7:30 - 8:15~~

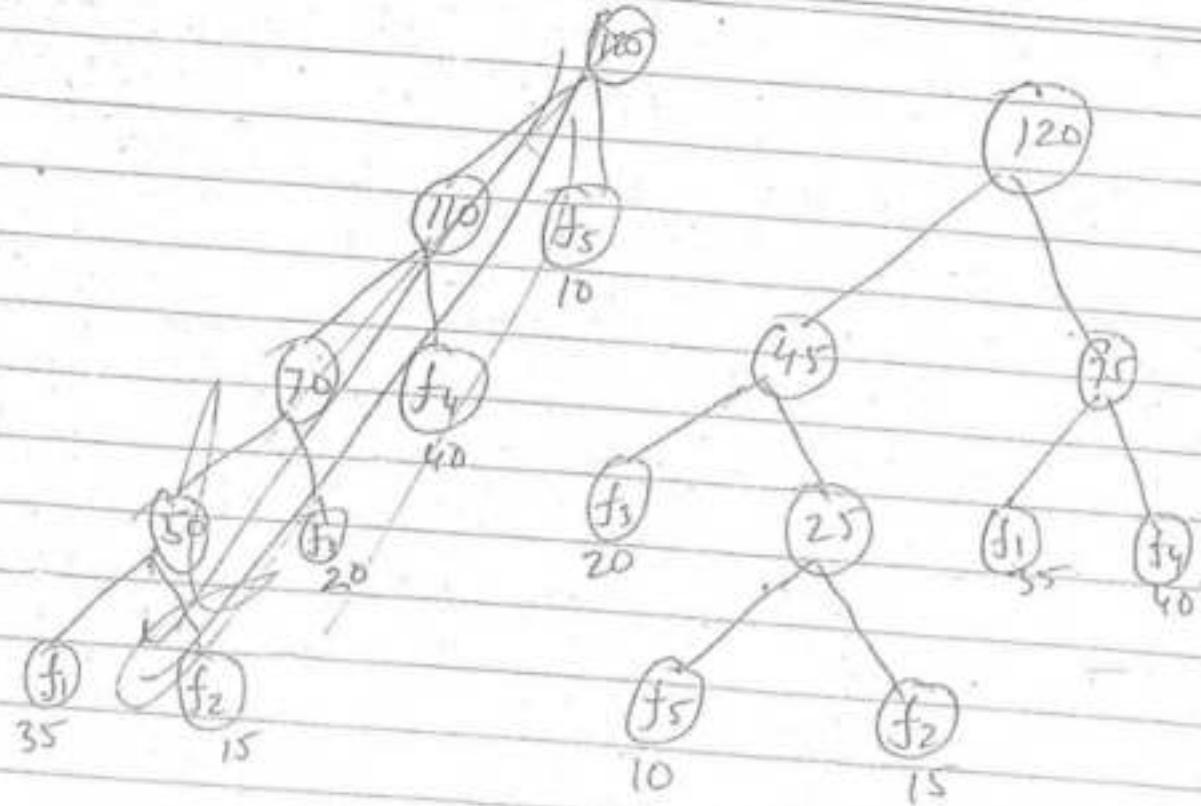


Optimal Merge Pattern



The min^m no of record movement to merge 5 files

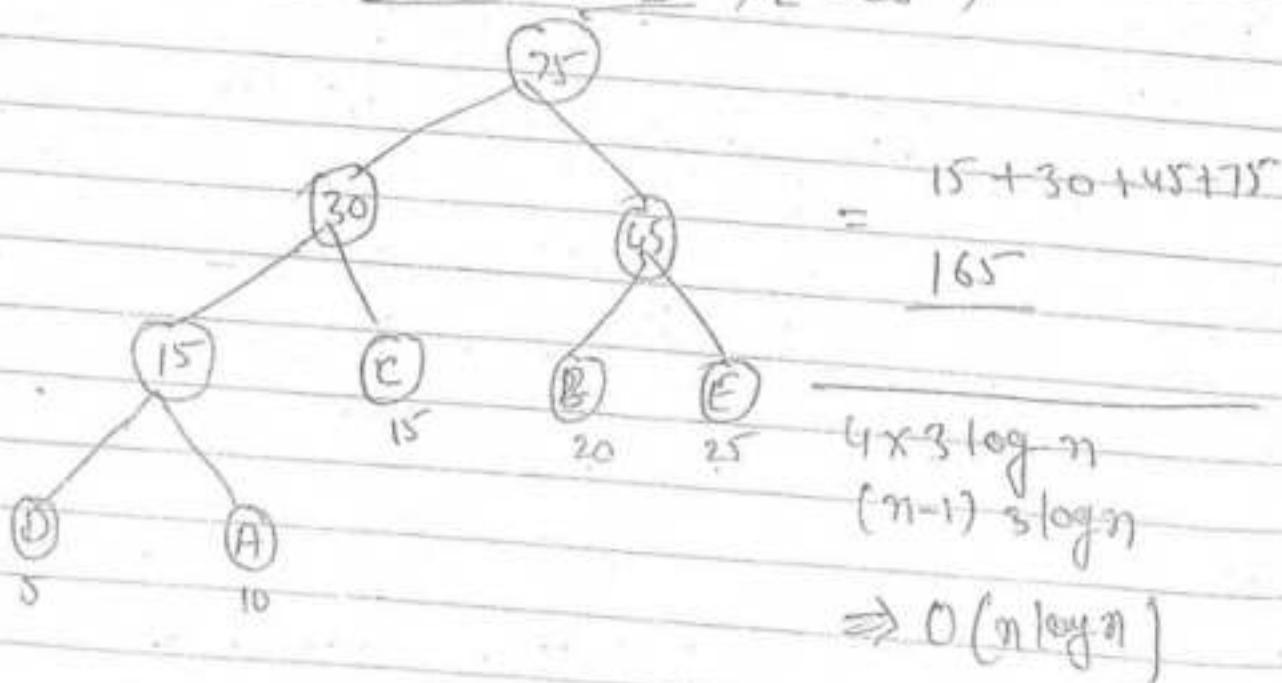
$$f_1 = 35, f_2 = 15, f_3 = 20, f_4 = 40, f_5 = 10$$



$$25 + 45 + 75 + 120 = 265$$

Min no of Root1 not required to merge files.

$$A=10, B=20, C=15, D=35, E=25$$



$$15 + 30 + 45 + 75 =$$

$$\underline{165}$$

$$4 \times 3 \log n$$

$$(n-1) \log n$$

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

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 $\Rightarrow O(n \log n)$

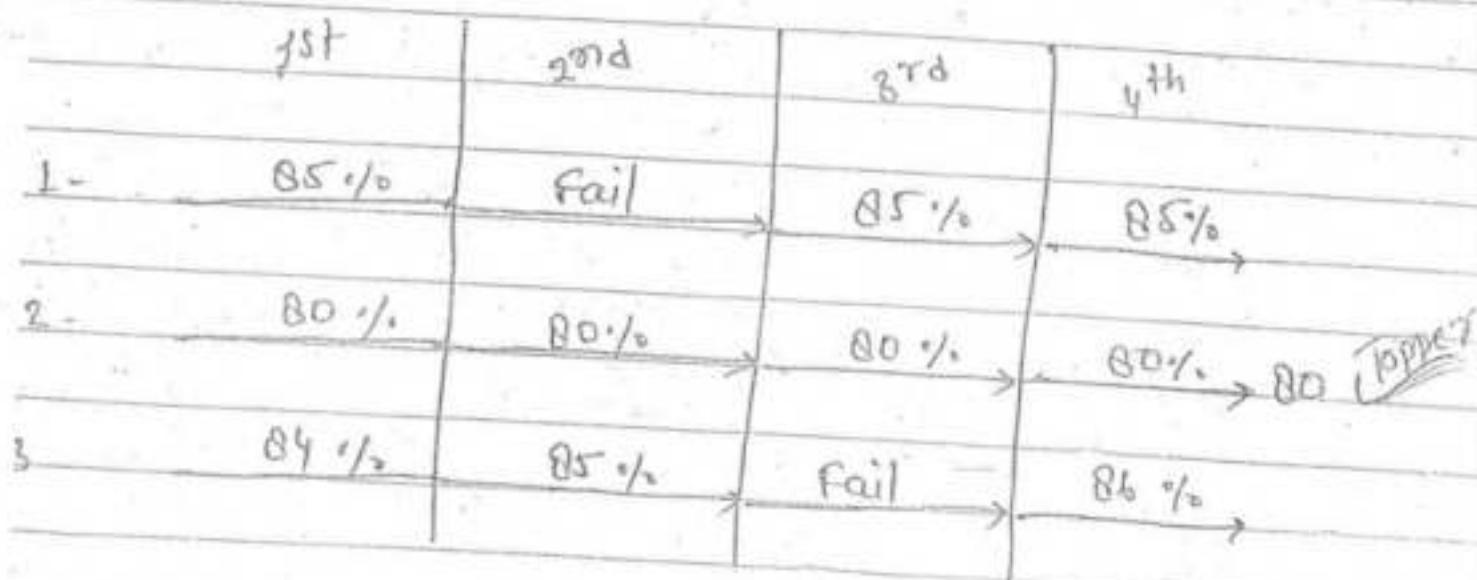
(5) Optimal merge Pattern $\Rightarrow O(n \log n)$

(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 \Rightarrow Optimal Solution
 Greed 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.

e.g. $\text{u} = (\text{A, B, C, B, D, A, B})$

$$\begin{matrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \text{u} = & \text{A} & \text{B} & \text{C} & \text{B} & \text{D} & \text{A} & \text{B} \end{matrix}$$

$$S_1 = (\text{A, A, B}) \quad \begin{matrix} 1 & 6 & 7 \end{matrix} \Rightarrow \text{Subsequence.}$$

$$S_2 = (\text{A, B, B}) \quad \begin{matrix} 1 & 4 & 7 \end{matrix} \Rightarrow \text{Subsequence}$$

$$S_3 = (\text{C, D, B}) \quad \begin{matrix} 3 & 5 & 7 \end{matrix} \Rightarrow \text{Subsequence.}$$

$$S_4 = (\text{A, A, B, C}) \quad \begin{matrix} 1 & 6 & 7 & 3 \end{matrix} \times$$

$$S_5 = (\text{A, B, C, B, D}) \quad \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \Rightarrow \text{Subsequence}$$

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 .

Q $x = (A, B, B, A, B, D)$

$y = (A, A, B, B, D, D)$

$CS_1 = (A, A, D)$ — 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 .

Q $x = (A, B, C, B, D, A, B)$

$y = (B, D, C, A, B, A)$

$$CS_1 = (A, B, A, B)$$

longest common subsequence

$$CS_2 = \{A\}$$

$$CS_3 = \{A, B\}$$

$$CS_4 = \{B, C, A\}$$

$$CS_5 = \{\emptyset\}$$

$$P - W = (A, B, C, B, D, A, B,)$$

$$Y = (P$$

$$\begin{array}{l} X \\ Y \end{array} \quad \begin{array}{l} u = (a_1, a_2, a_3, a_4, \dots, a_n) \\ v = (b_1, b_2, b_3, \dots, b_n) \end{array}$$

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) 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

↓
 (In technical term means
 algo is very slow)

$$x = (A, A, A, A, A)$$

$$y = (\underset{1}{A}, \underset{2}{A}, \underset{3}{A}, \underset{4}{A}, \underset{5}{A})$$

longest common subsequence (5,5) = 1 + LCS(4,4)

$$LCS(4,4) = 1 + LCS(3,3)$$

$$LCS(3,3) = 1 + LCS(2,2)$$

$$LCS(2,2) = 1 + LCS(1,1)$$

$$LCS(1,1) = 1 + \underbrace{LCS(0,0)}_0$$

$$LCS(n,n) = 1 + LCS(n-1, n-1) \text{ if } x[n] = y[n]$$

eg $x = (A, A A B A A A)$
 $y = (\overset{A}{\underset{A}{\cdot}} A A A A A A)$

soln $LCS(7,7) = 1 + LCS(6,6)$

\Downarrow

$1 + LCS(5,5)$

\Downarrow

$1 + LCS(4,4)$

\Downarrow

$\begin{matrix} 6 \\ \diagup \end{matrix} \leftarrow LCS(3,4)$

\Downarrow

$1 + LCS(2,3)$

\Downarrow

$1 + LCS(1,2)$

\Downarrow

$1 + LCS(0,1)$

eg $x = (B, B, B, B, A, A, A)$

$x = (B, B, B, A, A, A, A)$

$LCS(7,7) = 1 + LCS(6,6)$

\Downarrow

$1 + LCS(5,5)$

$\begin{matrix} 6 \\ \diagup \end{matrix} \leftarrow$

$1 + LCS(4,4)$

\Downarrow
 $LCS(4,3)$

\Downarrow
 $1 + LCS(3,2)$

\Downarrow
 $1 + LCS(2,1)$

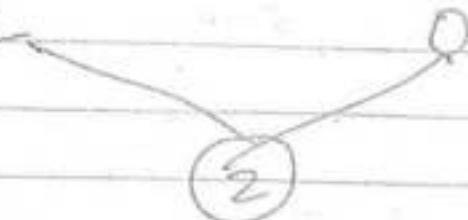
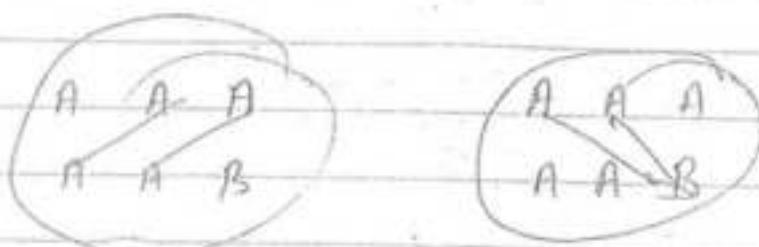
\Downarrow
 $1 + LCS(1,0)$

eg $u = \text{A B A B}$
 $y = \text{B A B A}$

if match is not found then
 don't decrement both.

$$\text{LCS}(i, j) = \begin{cases} 0 & \text{if } i=0 \text{ (or) } j=0 \\ 1 + \text{LCS}(i-1, j-1) & \text{if } u[i] == y[j] \\ \max(\text{LCS}(i-1, j), \text{LCS}(i, j-1)) & \text{if } u[i] \neq y[j] \end{cases}$$

eg



Optimal Solution

(i) Optimal Solution of original problem contains optimal soln of subproblem.

Write a recursive "c" Program to lcs(i,j)

LCS(i,j)

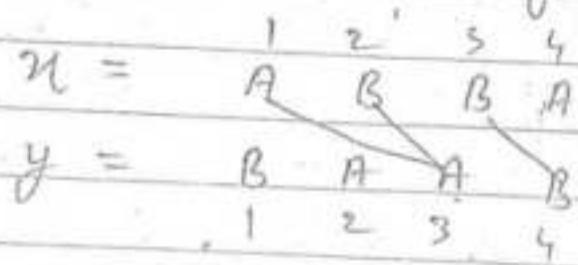
```

    {
        if (i == 0 || j == 0)           // exponential
            return(0);                // O(2^n)
        else
            {
                if (x[i] == y[j])
                    return (1 + LCS(i-1, j-1));
                else
                    {
                        {
                            a = LCS(i-1, j);
                            b = LCS(i, j-1);
                            c = max(a, b);
                            return(c);
                        }
                    }
            }
    }

```

worst

~~eg~~ Find LCS for the following Problem.



~~Soln~~

$$\text{LCS}(4,4) = \max \left\{ \begin{array}{l} \text{LCS}(3,3) \\ \text{LCS}(4,3) \end{array} \right\}$$

$$\text{LCS}(3,4) = 1 + \text{LCS}(2,3)$$

$$\text{LCS}(2,3) = \max \left\{ \begin{array}{l} \text{LCS}(1,3) \\ \text{LCS}(2,2) \end{array} \right\}$$

$$\text{LCS}(1,3) = 1 + \text{LCS}(0,2)$$

$$\text{LCS}(0,2) = 0$$

$$\text{LCS}(2,2) = \max \left\{ \begin{array}{l} \text{LCS}(1,2) \\ \text{LCS}(2,1) \end{array} \right\}$$

$$\text{LCS}(1,2) = 1 + \text{LCS}(0,1)$$

$$\text{LCS}(2,1) = 1 + \text{LCS}(1,0)$$

(0,1)

$$LCS(4, 3) = 1 + LCS(3, 2)$$

$$LCS(3, 2) = \max \left\{ \begin{array}{l} LCS(2, 2) \\ LCS(3, 1) \end{array} \right\}$$

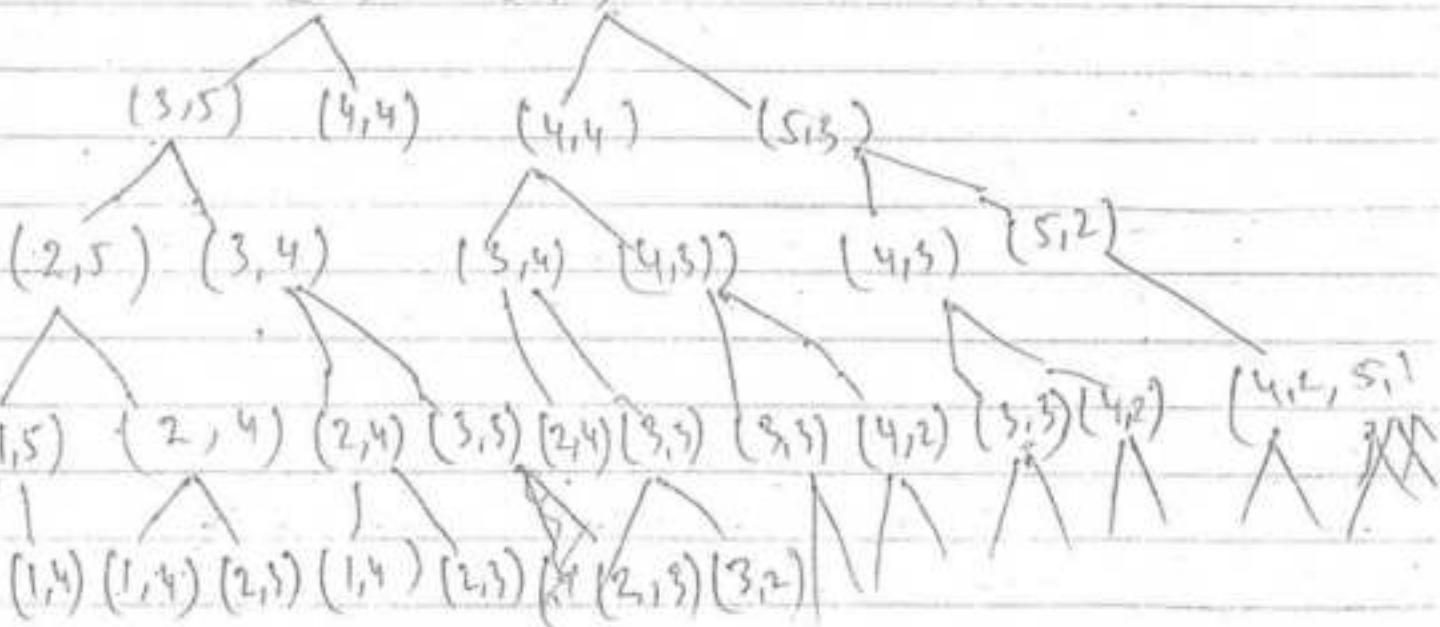
$$LCS(3, 1) = 1 + \cancel{LCS(2, 0)}$$

eg $x = (A, A, A, A, A)$

$y = (B, B, B, B, B)$

(5, 5)

$\Theta(m+n)$



$$3 \Rightarrow 2^3 - 1$$

$$2^{m+n} = 2^m \cdot 2^n$$

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 out.

$\Rightarrow 2^{m+n}$



Using recursion (With DP)

$$m=4, n=5$$

$$\begin{array}{ccc} 15 & 3,5 & 2,5 \\ 14 & 3,4 & 2,4 \\ 13 & 3,3 & 2,3 \\ 12 & 3,2 & 2,2 \\ 11 & 3,1 & 2,1 \end{array} \Rightarrow m \times n$$

$$\begin{array}{ccc} 15 & 3,5 & 2,5 \\ 14 & 3,4 & 2,4 \\ 13 & 3,3 & 2,3 \\ 12 & 3,2 & 2,2 \\ 11 & 3,1 & 2,1 \end{array} \Rightarrow D(m,n) D(1)$$



$$O(m,n) \Rightarrow O(n^2)$$

Quadratic

Drawback

Dynamic Programming take more space to store table for every subproblem.

I saw answer
this previously.

Memoization - 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] == 0)
                            Table[i, j] = 1 + Table[i-1, j-1];
                        else
                            Table[i, j] = Table[i-1, j-1];
                    }
                else
                    Table[i, j] = max(Table[i-1, j], Table[i, j-1]);
            }
    }
}

```

Table

1	2	3	4	5
1				
2				
3				
4				
5				

Content of Table

initially all values of table are nil

Contents Table

```

else
{
    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] = LCS(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
→

$$\text{LCS}(4,4) = \max \left\{ \begin{array}{l} \text{LCS}(3,4) \\ \text{LCS}(4,3) \end{array} \right\}$$

$$\text{LCS}(3,4) = 1 + \underline{\text{LCS}(2,3)}$$

$$\text{LCS}(2,3) = 1 + \underline{\text{LCS}(1,2)}$$

$$\text{LCS}(1,2) = 1 + \underline{\text{LCS}(0,1)}$$

$$\text{LCS}(4,3) = 1 + \underline{\text{LCS}(3,2)}$$

$$\text{LCS}(3,2) = 1 + \underline{\text{LCS}(2,1)}$$

$$\text{LCS}(2,1) = 1 + \underline{\text{LCS}(1,0)}$$

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 (1 0 1) manually

9

↓

~~eg 2~~

$$n = 3$$

objects	obj ₁	obj ₂	obj ₃
---------	------------------	------------------	------------------

Profit	50	29	33
--------	----	----	----

Weight	10	6	7
--------	----	---	---

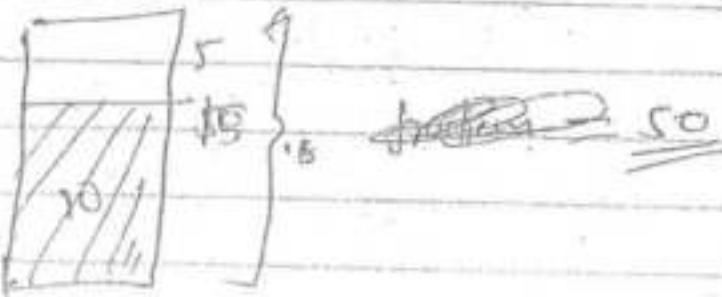
$$M = 15$$

$$\frac{50}{10} = 5$$

10

$$\frac{29}{6} = 4.8$$

$$\frac{33}{7} = 4.7$$

~~manually~~

$$M = 15$$

In DP there is no shortcut becz it is not giving us a optimal soln

P
 $m = 5$

Object	obj ₁	obj ₂	obj ₃	obj ₄	obj ₅
Weight	3	1	2	4	6
Profit	7	2	1	6	12

 $m = 10$ ~~Optimal Knapsack Solution~~

$$\text{OKS}(m, n) = \begin{cases} 0 & \text{if } m=0 \text{ (or) } n=0 \\ \text{OKS}(m, n-1) & \text{if } m < w[n] \\ \max \left\{ \begin{array}{l} \text{OKS}(m, n-1) + p \\ \text{OKS}(m - w[n], n-1) + p(n) \end{array} \right\} & \text{otherwise} \end{cases}$$

 m = Capacity of Knapsack n = no. of object m, n

OKS

10, 5

4, 4

12

4, 3

7

4, 2

2

3, 1

21

Recursive Program for 0/1 Knapsack

OKS(m, n)

```
if ( $m == 0$  (or)  $n == 0$ )
```

```
    return (0);
```

```
else
```

```
if ( $m \leq w[n]$ )
```

```
    return (OKS( $m - w[n], n-1$ ))
```

```
else
```

```
    return (max(OKS( $m - w[n], n-1$ ) +  $p[n]$ ), OKS( $m, n-1$ ))
```

$O(n^2)$

(A)

(M, N)

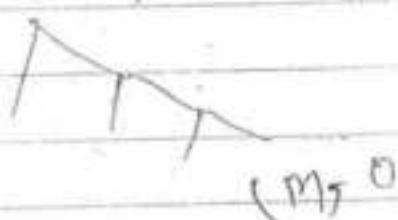
$(M - w_n, N - 1)$

$(M, N - 1)$

$(M - w_n, N - 2)$ $(M, N - 2)$

In worst case

Level = n



Note \Rightarrow In the 0/1 Knapsack Program (recursive)
Step (A) gives worst case behavior i.e.
every time two possibilities like them in time
So binary tree 'n' levels.
so time complexity of 0/1 Knapsack $\approx O(2^n)$

S

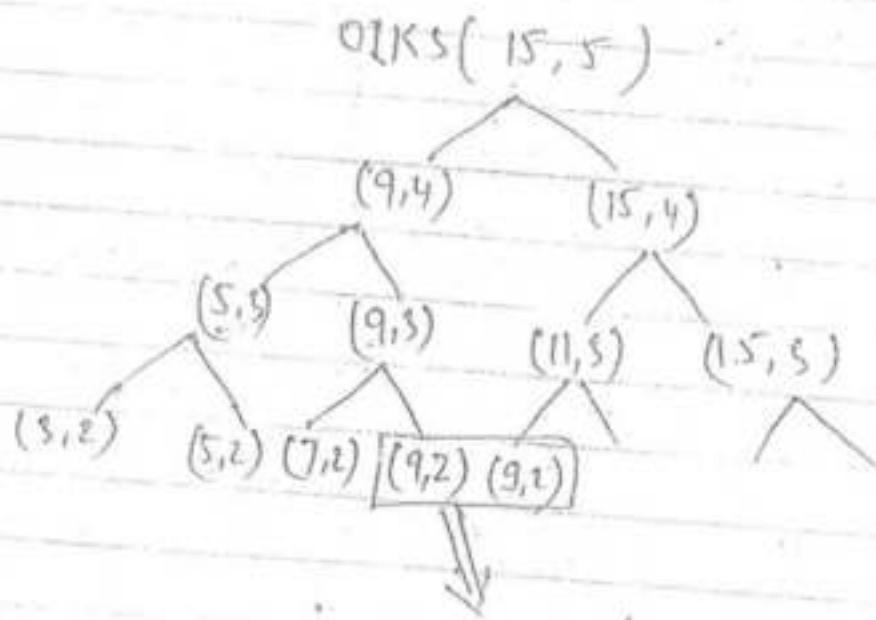
$M=15$

$n=5$

वा

Object	Ob ₁	Ob ₂	Ob ₃	Ob ₄	Ob ₅
Weight	3	1	2	4	6
Profit	7	2	1	6	12

Sol⁰¹



Overlapping Subproblem

(Can be solved only one time in DP)

Q) How many distinct subproblems OIKS(M,N) having.

$$M = 15$$

$$N = 5$$

$$(15, 5) \quad (\cancel{14}, \cancel{4})$$

$$(15, 4) \quad (14, 4) \quad (13, 4) \quad 12, 11, \dots, 10 \dots 0$$

$$(15, 3) \quad (14, 3) \quad (13, 3)$$

$$\cancel{(15, 2)} \quad (14, 2) \quad (13, 2)$$

$$\cancel{(15, 1)} \quad (14, 1) \quad (13, 1)$$

$$(15, 0) \quad (14, 0) \quad (13, 0)$$

the no of ^{distinct} subproblem are.

$$OIKS(M, N) = (M+1) * (N+1)$$

$$= M * N$$



larger

= Exponential

= NP Complete

Dynamic Programming OIKS

Using Dynamic Programming

Planarization - CLKS (M, N)

```
{  
if ( m == 0 (or) n == 0 )  
    { return (0); }
```

Table [M, N-1] = 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 [N, 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)

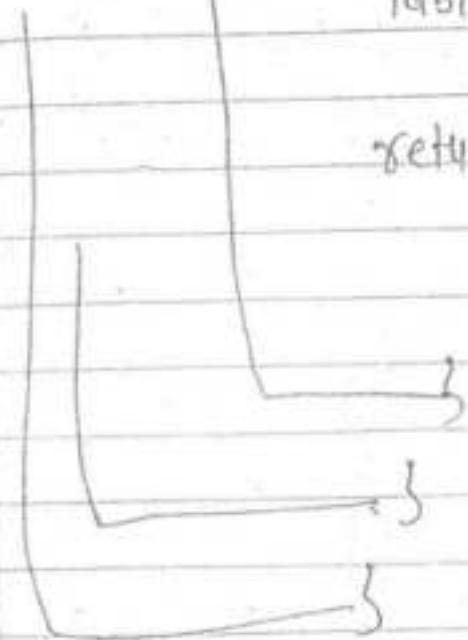
If (Table [M-W[N], N-1] == Nil)

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

$$M = 210$$

$$S_1 = \{30, 80, 100\}$$

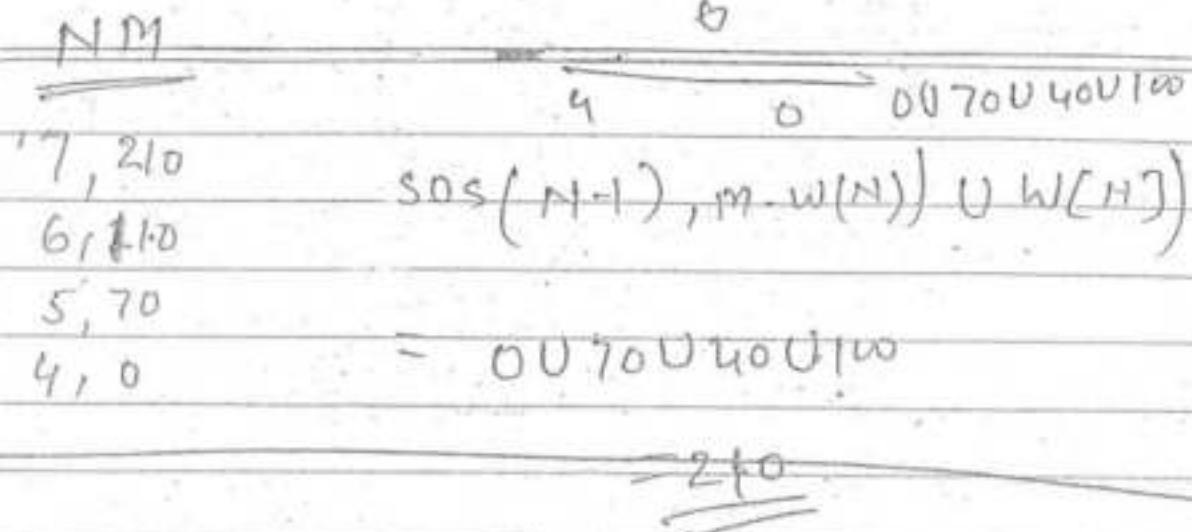
$$S_2 = \{70, 40, 100\}$$

$$S_3 = \{10, 30, 70, 100\}$$

Recurrence Reln

$$\text{sos}(N, M) = \begin{cases} 0 & \text{if } M=0 \\ 1 & \text{if } N=0 \\ \text{sos}(N-1, M) & \text{if } w[N] > M \\ \text{max} \left(\text{sos}(N-1, M-w[N]) \cup w[N] \right) & \text{if } w[N] \leq M \end{cases}$$

complete binary tree with
 n-levels $\Rightarrow O(2^n)$

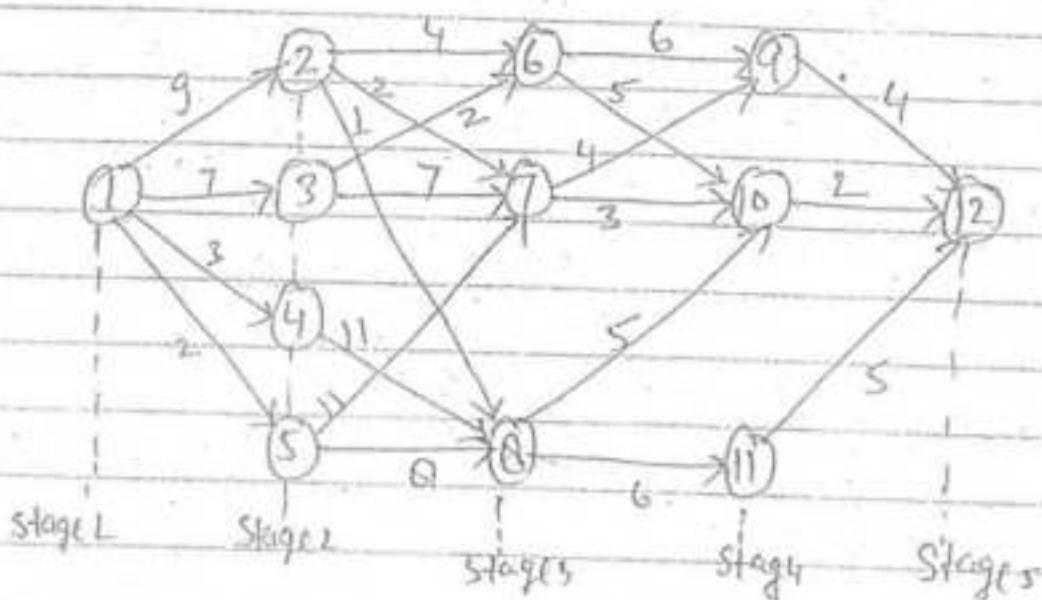


Using Dynamic Programming

$M \times N \Rightarrow O(MN)$

\downarrow
longer
 \hookrightarrow Exponential.

Multi Stage Graph

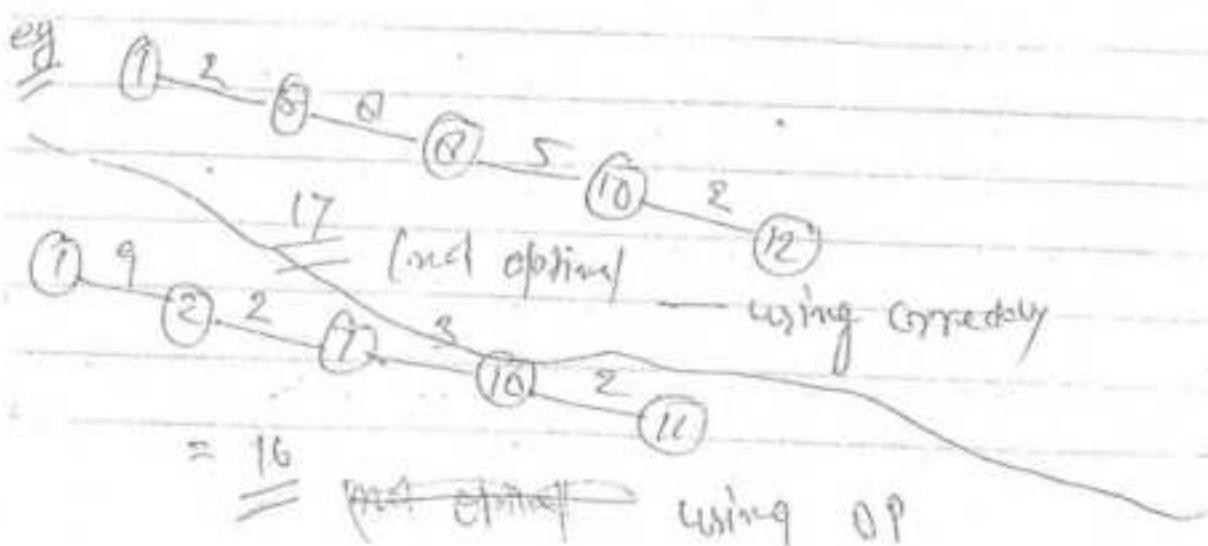


$MSG(1, 1)$

$MSG(2, 5)$

$MSG(3, 7)$

$MSG(s_i, v_i) = \text{The min}^m \text{ cost required from stage } s_i \text{ to destination } v_i$



$$MSG(1,1) = \min_{\substack{1 \\ 2 \\ 3 \\ 4}} \left\{ \begin{array}{l} C(1,2) + MSG(2,2) \\ C(1,3) + MSG(2,5) \\ C(1,4) + MSG(2,4) \\ C(1,5) + MSG(2,5) \end{array} \right\}$$

$$MSG(2,2) = \min_{\substack{1 \\ 2 \\ 3}} \left\{ \begin{array}{l} C(2,3) + MSG(3,3) \\ C(2,4) + MSG(3,7) \\ C(2,8) + MSG(3,8) \end{array} \right\}$$

$$MSG(3,3) = \min_{\substack{1 \\ 2 \\ 3}} \left\{ \begin{array}{l} C(3,4) + MSG(4,4) \\ C(3,5) + MSG(4,10) \end{array} \right\}$$

$$MSG(4,4) = \min \left\{ C(9,12) \right\}$$

$$MSG(4,10) = \min \left\{ C(10,12) \right\}$$

$$MSG(3,7) = \min_{\substack{1 \\ 2}} \left\{ \begin{array}{l} C(7,9) + MSG(4,9) \\ C(7,10) + MSG(4,10) \end{array} \right\}$$

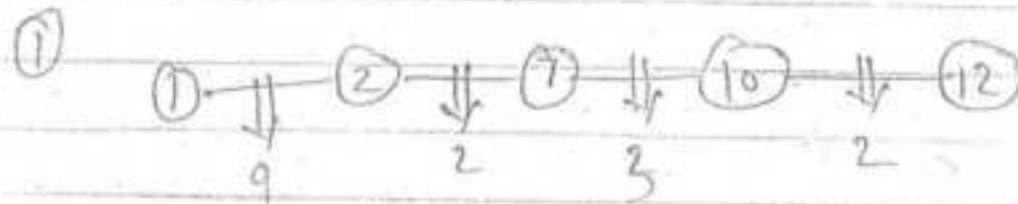
$$MSG(3,0) = \min \left\{ \begin{array}{l} c(\overbrace{0,10}^5) + MSG(\overbrace{4,10}^2) \\ c(\overbrace{0,11}^6) + MSG(\overbrace{4,11}^5) \end{array} \right.$$

$$MSG(4,11) = c(\overbrace{11,12}^5)$$

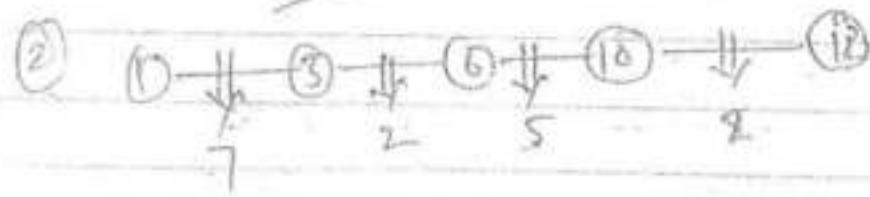
$$MSG(2,3) = \min \left\{ \begin{array}{l} c(\overbrace{3,6}^2) + MSG(\overbrace{3,6}^7) \\ c(\overbrace{3,7}^7) + MSG(\overbrace{3,7}^5) \end{array} \right.$$

$$MSG(2,4) = c(\overbrace{4,8}^{10}) + MSG(\overbrace{3,0}^7)$$

$$MSG(2,5) = \min \left\{ \begin{array}{l} c(\overbrace{5,7}^{11}) + MSG(\overbrace{3,7}^5) \\ c(\overbrace{5,0}^9) + MSG(\overbrace{3,0}^7) \end{array} \right.$$



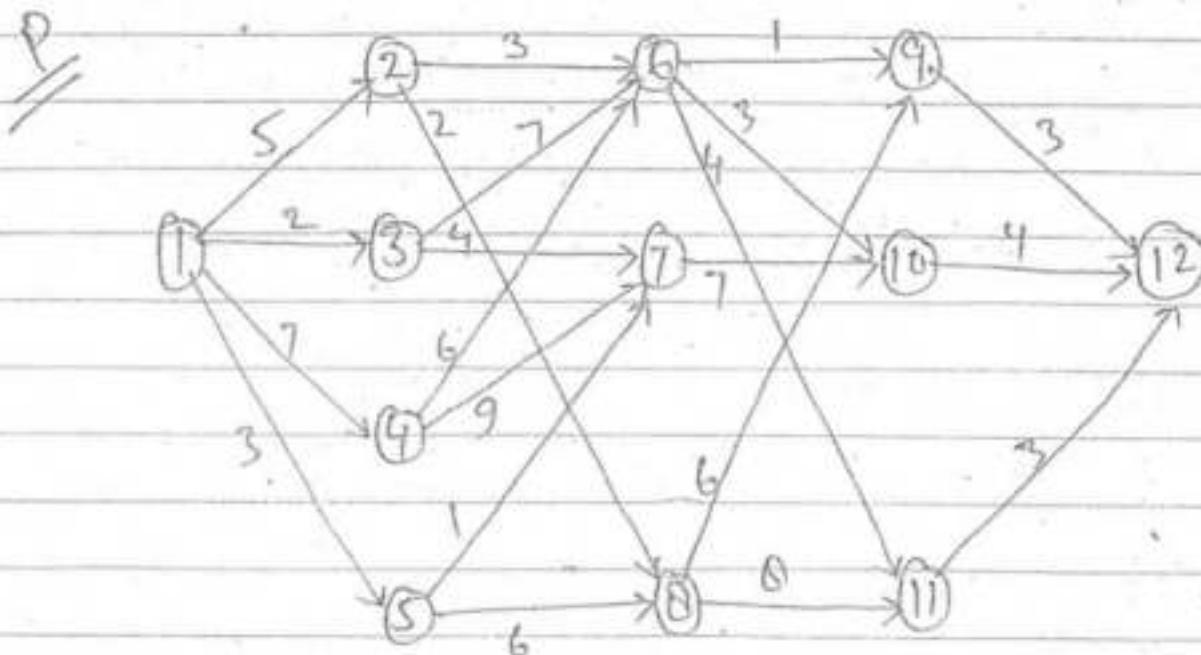
$$= \underline{\underline{16}}$$



$$\underline{\underline{16}}$$

Note - Every edge is covered only once.
At every vertex covered only once.

no of fun call = no of vertex



$$MSG(1,1) = \min \left\{ \begin{array}{l} C(1,2) + MSG(2,1) \\ C(1,3) + MSG(2,3) \\ C(1,4) + MSG(2,4) \\ C(1,5) + MSG(2,5) \end{array} \right\}$$

$$MSG(2,2) = \min \left\{ \begin{array}{l} c(\overline{2,6}) + MSG(\overline{\frac{3}{4},6}) \\ c(\overline{2,8}) + MSG(\overline{\frac{3}{9},8}) \end{array} \right\}$$

$$\begin{array}{l} \text{MSG}(3,6) = \min \left\{ \begin{array}{l} c(\overline{6,9}) + MSG(\overline{\frac{5}{4},9}) \\ c(\overline{6,10}) + MSG(\overline{\frac{4}{3},10}) \\ c(\overline{6,11}) + MSG(\overline{\frac{4}{3},11}) \end{array} \right\} \\ \Downarrow \end{array}$$

$$MSG(3,8) = \min \left\{ \begin{array}{l} c(\overline{8,9}) + MSG(\overline{\frac{3}{4},9}) \\ c(\overline{8,11}) + MSG(\overline{\frac{3}{3},11}) \end{array} \right\}$$

$$MSG(2,3) = \min \left\{ \begin{array}{l} c(\overline{3,6}) + MSG(\overline{\frac{4}{3},6}) \\ c(\overline{3,7}) + MSG(\overline{\frac{4}{11},7}) \end{array} \right\}$$

$$MSG(3,7) = c(\overline{7,16}) + MSG(\overline{4,16})$$

$$MSG(2,4) = \min \left\{ \begin{array}{l} c(\overline{4,6}) + MSG(\overline{\frac{4}{3},6}) \\ c(\overline{4,7}) + MSG(\overline{\frac{4}{11},7}) \end{array} \right\}$$

$$MSG(2,5) = \min \left\{ \begin{array}{l} c(5,7) + MSG(3,7) \\ c(5,8) + MSG(3,8) \end{array} \right\}$$

Recurrence Relation

$$MSG(s_i, v_j) = \min \left\{ \begin{array}{l} c(v_j, D) \text{ if } s_i = f_j \\ c(v_j, k) + MSG(s_{i+1}, k) \\ \forall k \in s_{i+1} \\ (v_j, k) \in E \end{array} \right.$$

D = Destination.

s_f = Final stage.

Euler Path & 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.

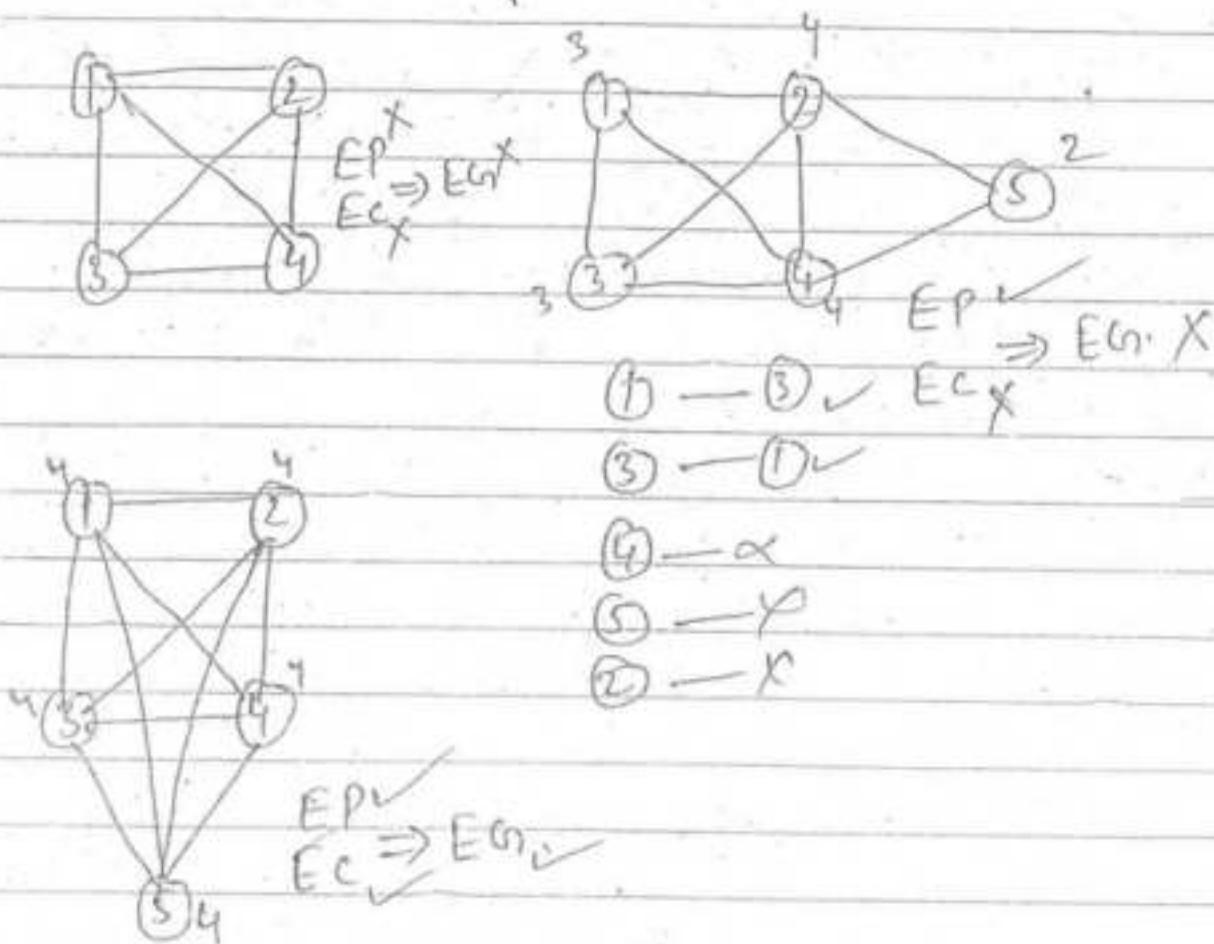
Euler Circuit: It is an Euler path in which $v_s = v_e$ is called Euler Circuit.

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 are possible then graph is called Euler Graph.

To check a graph is Euler or not $O(|E|)$

A class problem
 $= O(V^2)$

bcz
Polynomial
Time
Complexity

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

~~a) $G \Rightarrow |V|=10, |E|=10$ is E.G.~~

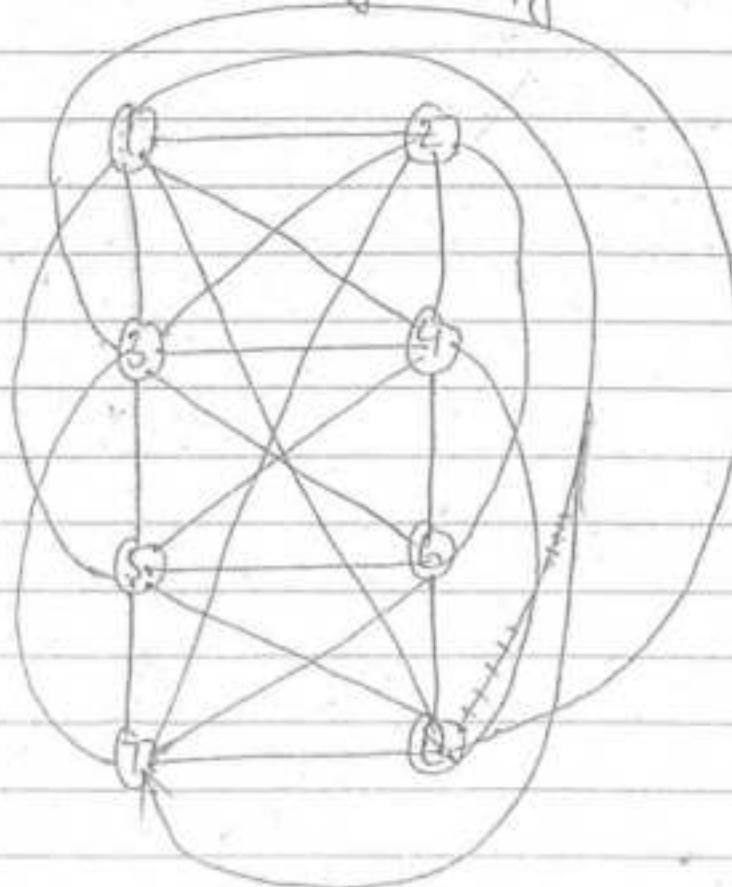
~~b) $G \Rightarrow |V|=50, |E|=100$ is E.G.~~

~~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 is E.G.~~

Every vertex degree
is even
degree.

Q Consider the following graph.



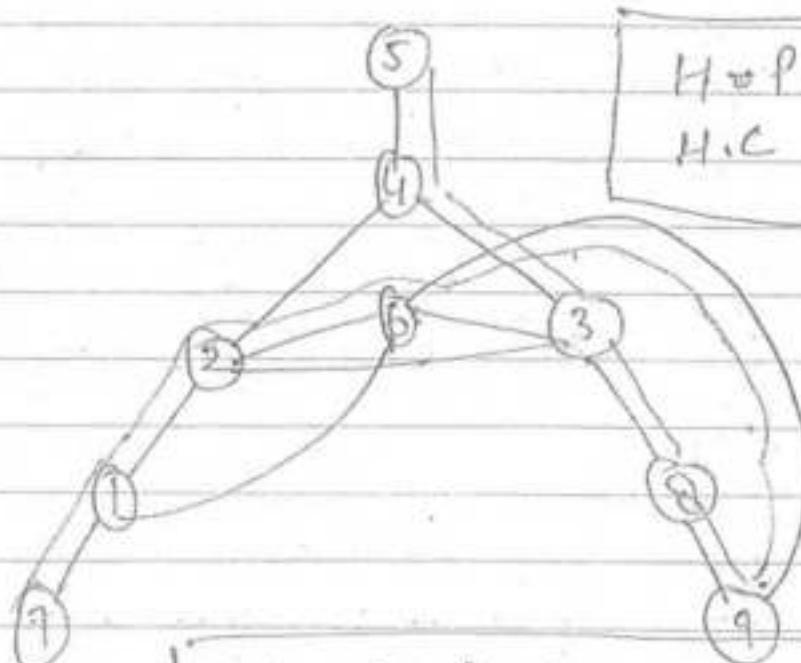
H.P ✓
H.G. ✓

H.C. ✓

1-2-4-6-8-7-5-3

even

Q The following graph is H.G or not



H.P ✓ H.G X
H.C X

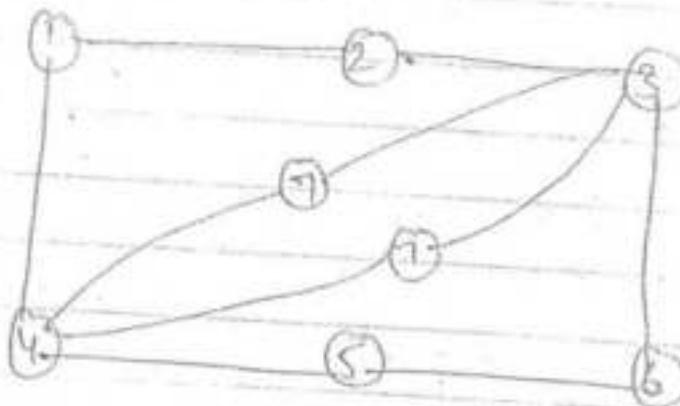
3Sat

3Sat
NP
Complete
P-Clean.

1-5-4-3-0-9-6-2-1-7

~~Note~~ To check given graph is H.G or not there is not polynomial time.

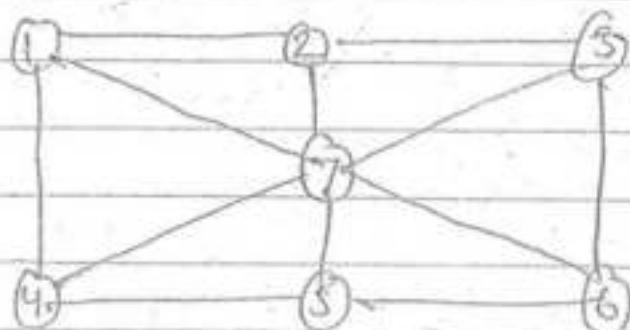
Q Which one of the following is true?



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

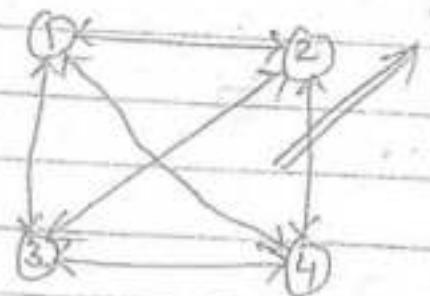
a) G - both E & H

or
EG X

b) G - E but not H

c) G - H but not E

d) both are not



$O(2^n) \Rightarrow$ 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	0	8	9	0

Cost adjacency matrix

$$\text{of } N = \{1, 2, 3, 4\}$$

initially source = 1

$$\text{TSP} = (1, (2, 3, 4))$$

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)

$$\text{# TSP}(1, (2, 3, 4)) = \min \left\{ \begin{array}{l} C(1, 2) + \text{TSP}(2, (3, 4)) \\ C(1, 3) + \text{TSP}(3, (2, 4)) \\ C(1, 4) + \text{TSP}(4, (2, 3)) \end{array} \right\}$$

Ans

35

$$\cancel{\text{#} \ TSP(2, \{3,4\}) = \min \left\{ \begin{array}{l} \underline{c(2,3)} + \cancel{TSP(\cancel{3}, \{4\})} \\ \underline{c(2,4)} + \cancel{TSP(\cancel{4}, \{3\})} \end{array} \right\}}$$

$$\cancel{\text{* } TSP(3, \{4\}) = c(\underline{3,4}) + \underline{c(4,1)}}$$

$$\cancel{\text{* } TSP(4, \{3\}) = c(\underline{4,3}) + c(\underline{3,1})}$$

$$\cancel{\text{* } TSP(3, \{2,4\}) = \min \left\{ \begin{array}{l} \underline{c(3,2)} + \cancel{TSP(2, \{4\})} \\ \underline{c(3,4)} + \cancel{TSP(4, \{2\})} \end{array} \right\}}$$

$$\cancel{\text{* } TSP(2, \{4\}) = c(\underline{2,4}) + c(\underline{4,1})}$$

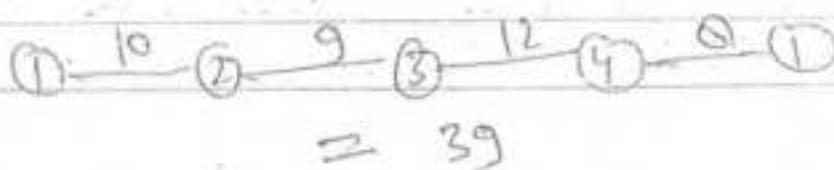
$$\cancel{\text{* } TSP(4, \{2\}) = c(\underline{4,2}) + c(\underline{2,1})}$$

$$\cancel{\text{* } TSP(4, \{2,3\}) = \min \left\{ \begin{array}{l} \underline{c(4,2)} + \cancel{TSP(\cancel{2}, \{3\})} \\ \underline{c(4,3)} + \cancel{TSP(\cancel{3}, \{2\})} \end{array} \right\}}$$

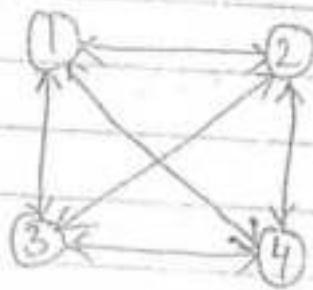
$$\cancel{\text{* } TSP(2, \{3\}) = c(\underline{2,3}) + c(\underline{3,1})}$$

$$\cancel{\text{* } TSP(3, \{2\}) = c(\underline{3,2}) + c(\underline{2,1})}$$

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	6	0

Source = 1

$$TSP(1, \{2, 3, 4\}) = \min_{\frac{24}{26}} \left\{ \begin{array}{l} C(1, 2) + TSP(2, \{3, 4\}) \\ C(1, 3) + TSP(3, \{2, 4\}) \\ C(1, 4) + TSP(4, \{2, 3\}) \end{array} \right\}$$

$$TSP(2, \{3, 4\}) = \min \left\{ \begin{array}{l} C(2, 3) + TSP(3, \{4\}) \\ C(2, 4) + TSP(4, \{3\}) \end{array} \right\}$$

$$TSP(\frac{15}{3, \{4\}}) = C(3, 4) + C(4, 1)$$

$$TSP(\frac{19}{4, \{3\}}) = C(4, 3) + C(3, 1)$$

$$\text{TSP}(3, \{2, 4\}) = \min \left\{ \begin{array}{l} c(3, 2) + \text{PSP}(\overline{2 \{4\}}) \\ c(3, 4) + \text{TSP}(\overline{4 \{2\}}) \end{array} \right\}$$

$$\text{TSP}(\overline{2 \{4\}}) = c(\overline{2, 4}) + c(\overline{4, 1})$$

$$\text{TSP}(\overline{4 \{2\}}) = c(\overline{4, 2}) + c(\overline{2, 1})$$

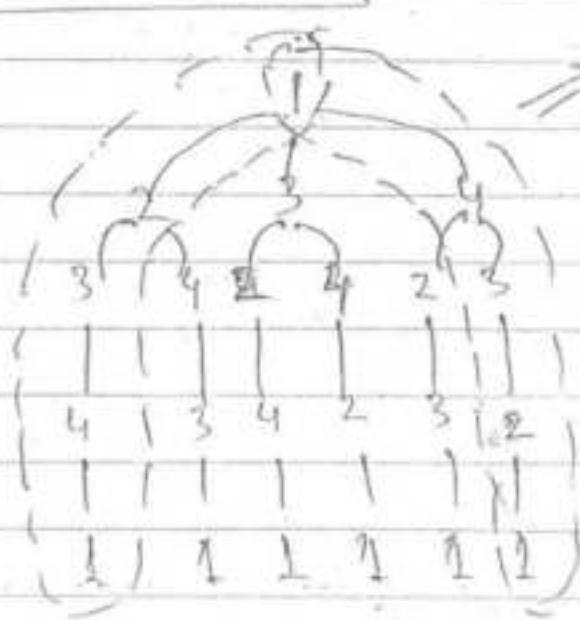
$$\text{TSP}(\overline{4 \{2, 3\}}) = \min \left\{ \begin{array}{l} c(\overline{4, 2}) + \text{PSP}(\overline{2 \{3\}}) \\ c(\overline{4, 3}) + \text{TSP}(\overline{3 \{2\}}) \end{array} \right\}$$

$$\text{TSP}(\overline{2 \{3\}}) = c(\overline{2, 3}) + c(\overline{3, 1})$$

$$\text{TSP}(\overline{3 \{2\}}) = c(\overline{3, 2}) + c(\overline{2, 1})$$

Hence DP = 26

$O(n^{2^n})$
by exponential.



Recursive Relation

$$TSP(S, V') = \min_{\substack{1, \{2, 3, 4\}}} \left\{ \begin{array}{l} c(S, \text{source if } V' = \emptyset) \\ c(S, K) + TSP(K, V' - K) \end{array} \right\}$$

\forall K \in V' \quad (S, K) \in E

Dry 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)

$A_{2 \times 3}, B_{3 \times 2}$

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



$$C = A \times B$$

2×2 2×3 3×2

$$= \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

2×2

A $A_{5 \times 10}$	B $B_{10 \times 20}$	$A_{m \times n} \cdot B_{n \times p}$ $C = AB$
$C = A \times B$ $C_{5 \times 20} \Rightarrow 100$ $5 \times 20 \times 10$ \Downarrow 1000		$C_{m \times p} \Rightarrow n$ $\Rightarrow mnp$

~~***~~ Let $A_{n \times n}$, $B_{n \times n}$ Matrix them

$$\text{if } C = A \times B$$

then C size is $C_{n \times n}$

to get every element of C , n -multiplied

total multiplication are $= n^2 \times n$

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

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 multiplications are
required.

$$50^4 (AB)C$$

$$A_{5 \times 10}, B_{5 \times 5}, C_{5 \times 20}$$

$$(AB)_{5 \times 5} C_{5 \times 20}$$

$$((AB)C)_{5 \times 20 \times 5}$$

$$= 250 \\ 500 \\ \hline 750$$

eg

$$A = \begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix}$$

$$B = \begin{Bmatrix} 2 & 3 \\ 4 & 5 \end{Bmatrix}, C = \begin{Bmatrix} 3 & 4 \\ 5 & 6 \end{Bmatrix}$$

$$(A(BC))$$

$$((AB)C)$$

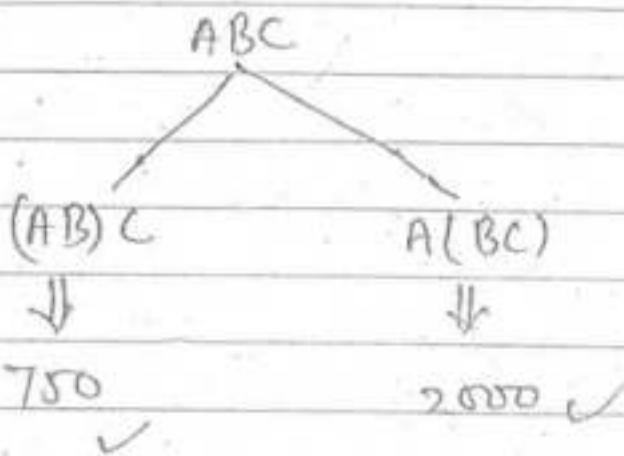
$$A = \begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix}, BC = \begin{Bmatrix} 21 & 26 \\ 37 & 46 \end{Bmatrix}$$

$$AB = \begin{Bmatrix} 10 & 13 \\ 22 & 29 \end{Bmatrix}, C = \begin{Bmatrix} 3 & 4 \\ 5 & 6 \end{Bmatrix}$$

$$A(BC) = \begin{Bmatrix} 95 & 110 \\ 211 & 262 \end{Bmatrix}$$

$$(AB)C = \begin{Bmatrix} 95 & 110 \\ 211 & 262 \end{Bmatrix}$$

Note - Matrix multiplication satisfied associative property

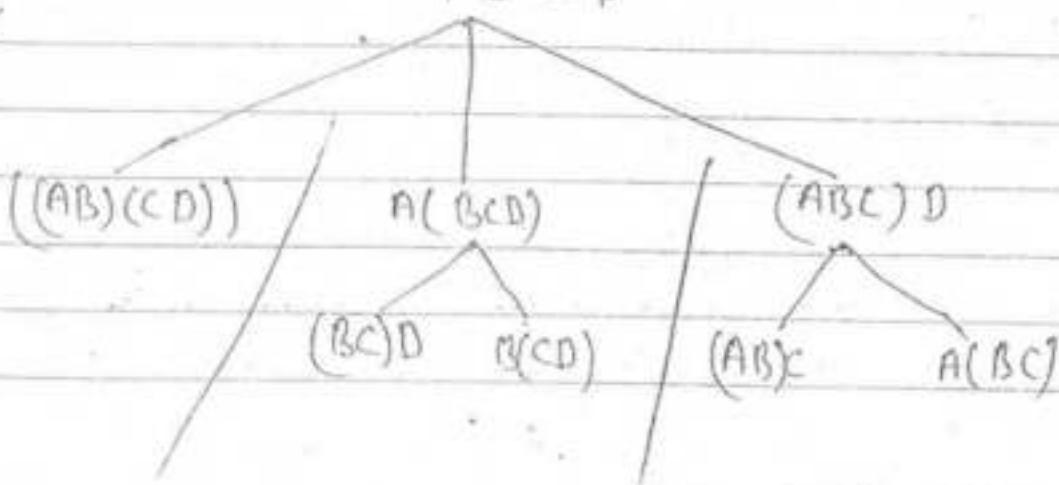


Note → In Matrix chain multiplication we don't want resultant matrix after multiplying n matrices.

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 minⁿ no of multiplication required?

~~Solⁿ~~



Ex 2

(AB)

$$2 \times 10 \times 5 = 100$$

(CD)

$$10 \times 2 \times 5 = 100 \rightarrow \begin{array}{r} 00 \\ 100 \\ \hline 40 \end{array}$$

(ABC)

$$2 \times 2 \times 10 = 40$$

$$\underline{240}$$

(BC)D

BC

$$5 \times 5 \times 10 = 250$$

$$(BC)5 \times 5, D5 \times 2 = 50$$

$$= 250$$

$$50$$

$$\underline{300}$$

B(CD)

$$CD10 \times 2 \times 5 = 100$$

$$B5 \times 10, (CD)10 \times 2 = 100$$

$$= 100$$

$$100$$

$$\underline{200}$$

(AB)C

$$AB2 \times 10 \times 5 = 100$$

$$(AB)2 \times 10, C10 \times 5 = 100$$

$$= 100$$

$$100$$

$$\underline{200}$$

A(BC)

$$BC5 \times 5 \times 10 = 250$$

$$A2 \times 5 (BC)5 \times 5 = 50$$

$$= 250$$

$$50$$

$$\underline{300}$$

$$A((BC)D) = A_{2 \times 5} \times (BC)_5 \times D_{5 \times 2} \Rightarrow (ABC)_2 \times 2 \times 5 = 20$$

$$\therefore 320$$

$$A(B(CD)) \Rightarrow 220$$

$$((AB)C)D = 220$$

$$(A(BC))D = 320$$

$$((AB)(CD)) = 240$$

$$(((AB)C)D) = 220$$

$$(A((BC)D)) = 320$$

$$((A(BC))D) = 320$$

$$A((B(CD))) = 220$$

* 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 multiplications.

$P=10, q=100, r=20, s=5, t=80$, Hence en min^m no of multiplication $M_1 + M_4$ is required

- a) 2,40,000
- b) 44,000
- c) 19,600
- d) 2,56,000

Sol

$M_1 M_2 M_3 M_4$

$(M_1 M_2)(M_3 M_4)$

$(M_1 M_2 M_3) M_4$

$M_1 (M_2 M_3 M_4)$

$(M_1 M_2) M_3 \quad M_1 (M_2 M_3)$

$M_2 (M_3 M_4) - (M_2 M_3) M_4$

$$(M_1 M_2) \quad (M_3 M_4) \\ 10 \times 100 \quad 100 \times 20 \quad 20 \times 5 \quad 5 \times 80$$

$$M_1 M_2 = 10 \times 20 \times 100 = 20,000$$

$$M_3 M_4 = 20 \times 80 \times 5 = 8000$$

$$(M_1 M_2) (M_3 M_4) = 10 \times 100 \times 20 = 16000 \\ 10 \times 20 \quad 20 \times 80$$

$$\begin{array}{r} 20000 \\ 8000 \\ 16000 \\ \hline 44000 \end{array}$$

$$m_2(m_3 m_4)$$

$$m_3 m_4 = 8000 \checkmark$$

$$m_2(m_3 m_4)$$

$$100 \times 20 \quad 20 \times 80$$

$$= 100 \times 80 \times 20$$

$$= 160000 \checkmark$$

$$m_1(m_2(m_3 m_4))$$

$$10 \times 100 \quad 100 \times 80$$

$$= 10 \times 80 \times 100$$

$$= 80000 \checkmark$$

$$= 2,40,000$$

$$(m_2 m_3) m_4$$

$$\frac{m_2 m_3}{100 \times 20} \frac{20 \times 5}{20 \times 5}$$

$$= 100 \times 5 \times 20 = 10000 \checkmark$$

$$(m_2 m_3) m_4$$

$$100 \times 5 \quad 5 \times 80$$

$$= 100 \times 80 \times 5$$

$$= 40000 \checkmark$$

$$m_1((m_2 m_3) m_4)$$

$$10 \times 100 \quad 100 \times 80$$

$$= 10 \times 80 \times 100$$

$$= 80000 \checkmark$$

$$= 1,30,000$$

$(M_1 M_2) M_3$

$$M_1 M_2 = 20,000 \quad \checkmark$$

$(M_1 M_2) M_3$

$$10 \times 20 \quad 20 \times 5$$

$$= 10 \times 5 \times 20$$

$$= 1000 \quad \checkmark$$

$((M_1 M_2) M_3) M_4$

$$10 \times 5 \quad 5 \times 00$$

$$= 10 \times 00 \times 5$$

$$= 4000 \quad \checkmark$$

$$= 25000$$

$M_1 (M_2 M_3)$

$$M_2 M_3 = 10,000$$

$M_1 (M_2 M_3)$

$$10 \times 100 \quad 100 \times 5$$

$$= 10 \times 5 \times 100$$

$$5000 \quad \checkmark$$

$(M_1 M_2 M_3) M_4$

$$10 \times 5 \quad 5 \times 00$$

$$= 10 \times 00 \times 5$$

$$= 4000 \quad \checkmark$$

$$= 19000$$

#

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

$$MCM(i, j) = \min_{\substack{1 \leq k \leq j \\ \text{or}}} \left\{ \begin{array}{l} 0 \quad \text{if } i=j \\ MCM(i, k) + MCM(k+1, j) \\ + NMRTPTM \end{array} \right\}$$

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 140.
matrix is equal to

~~Solⁿ~~

$$MCM(1, 4) = \min \left\{ \begin{array}{l} \cancel{0} \quad \cancel{20} \quad \cancel{20} \\ MCM(\cancel{1, 1}) + MCM(\cancel{2, 4}) + \cancel{20} \\ MCM(\cancel{1, 2}) + MCM(\cancel{3, 4}) + \cancel{20} \\ MCM(\cancel{1, 3}) + MCM(\cancel{4, 4}) + \cancel{20} \end{array} \right.$$

$\checkmark O(n)$

$$MCM(2,4) = \min_{O(n)} \left\{ \begin{array}{l} MCM(2,2) + MCM(3,4) + 20 \\ MCM(2,3) + MCM(4,4) + 50 \end{array} \right.$$

$$\cdot \underline{\text{2D}} \\ MCM(3,4) = MCM(3,3) + MCM(4,4) + 20$$

A B C D	(CD) $C_{2 \times 5} \quad D_{5 \times 2}$ $2 \times 2 \times 5 = 20$	$B(CD)$ $5 \times 2 \quad 2 \times 2$ $5 \times 2 \times 2 = 20$
---------------------------------	---	--

$$\underline{50} \\ MCM(2,3) = MCM(2,2) + MCM(3,3) + 50$$

(BC) $B_{5 \times 2} \quad C_{2 \times 5}$ $5 \times 5 \times 2 = 50$	$(BC)D$ $5 \times 5 \quad 5 \times 2$ $5 \times 2 \times 5 = 50$
---	--

$$A(BCD)$$

$$2 \times 5 \quad 5 \times 2$$

$$ABCD$$

$$2 \times 2 \times 5 \Rightarrow 20$$

$$\overline{mcm(1,2)} = \frac{\overline{mcm(1,1)}}{0} + \frac{\overline{mcm(2,2)}}{0} \quad 20$$

$$(AB)_{2 \times 2 \times 5} = 20$$

$$(A \bar{B}) \frac{(CD)}{2 \times 2} = 0$$

$$mcm(1,5) = \min \left\{ \begin{array}{l} \frac{mcm(1,1)}{0} + mcm(\frac{2,3}{2}, \frac{5}{3}) + \frac{50}{50} \\ \text{or} \\ \frac{mcm(1,2)}{20} + mcm(\frac{3,5}{0}) + \frac{20}{0} \end{array} \right.$$

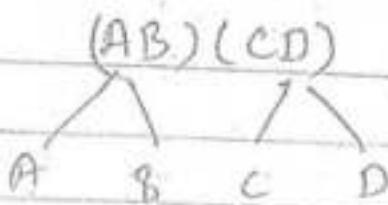
$A(BC)$	$(AB)(C)$
$2 \times 5 \quad 5 \times 2$	$2 \times 2 \quad 2 \times 5$
$10 \times 5 = 50$	$10 \times 2 = 20$

$$(ABC)(B)$$

$$2 \times 5 \quad 5 \times 2$$

$$= 2 \times 2 \times 5$$

$$= 20 \text{ ₹}$$



How many different funⁿ calls.

	1	2	3	4
1	X			
2		X		
3			X	
4				X

$$\frac{n \times n}{2} = \frac{n^2}{2} \Rightarrow n^2 \text{ fun }^n \text{ call}$$

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

① Dijkstra algo $\times |V|$



$V \times ((E + V) \log V)$

$O((V^2 E) \log V)$

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

{
Dijkstra algo(v_i)
}

② Bellman-Ford $\times |V|$

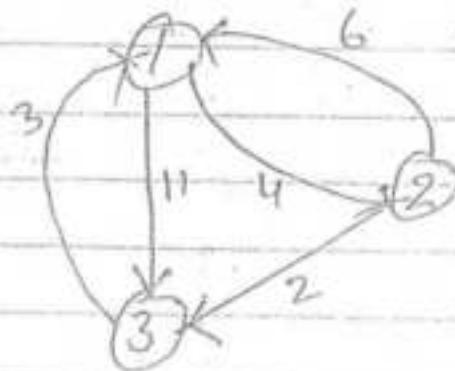


$|V| \times (VE)$



$O(V^2 E)$

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 |

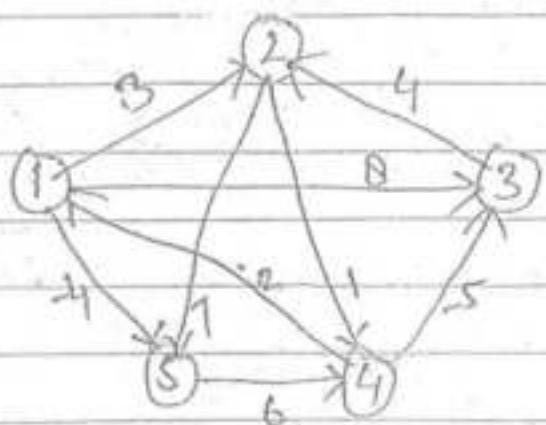
| ① - 11 - ③ |

(Floyd-Warshall's All-pairs shortest path algo

$$A^K(i, j) = \min \left\{ \begin{array}{l} \text{Previous value} \\ \frac{A^{K-1}(i, j), A^{K-1}(i, K) + A^{K-1}(K, j)}{\text{via } K} \end{array} \right\}$$

$$A^3(1, 3) = \min \{ A^2(1, 3), A^2(1, 3) + A^2(3, 2) \}$$

P



do^n

A^0	1	2	3	4	5
1	0	3	0	∞	-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	0	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

A^2

	1	2	3	4	5
1	0	3	0	4	-4
2	∞	0	∞	1	7
3	∞	4	4	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

A^3

	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

A^4

	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 ($s=1$, $s \leq |V|$, $s++$)

$$A^K(i, j) = \min \{ A^{K-1}(i, j), A^{K-1}(i, s) + A^{K-1}(s, j) \}$$

$\Theta(n^3)$

Graph Traversal

- (1) BFT (Breadth first traversal)
- (2) DFT (Depth first traversal)

Graph traversal is not unique & tree 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)

v

{

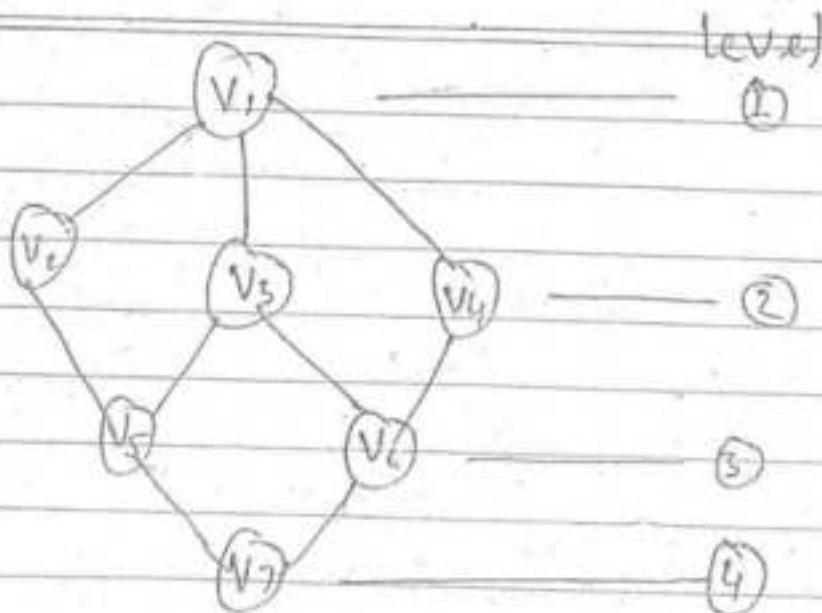
visited (w) = 1;

add (w, q);

}

}

}



w(V₂, V₃, V₄) adjacent to u(V₁)

w(V₁, V₅) adjacent to u(V₂)

- (1) The data structure we are using BFT is Queue.
- (2) BFT nothing but visited level by level because if random the BFT is called as level order traversal.

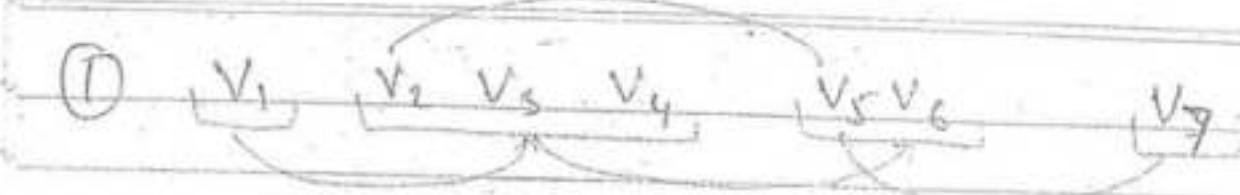
V₁, V₂, V₃, V₄, V₅, V₆, V₇,
 1 2 3 4

- (3) the time complexity of BFT

$$O(V + V^2)$$

$$O(V + E)$$

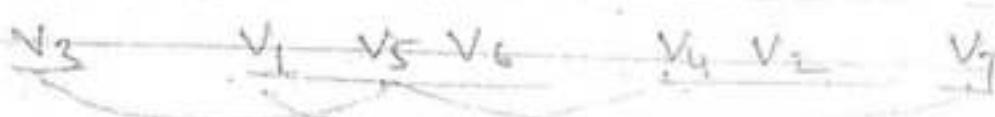
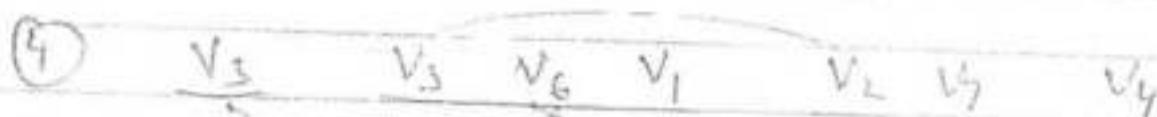
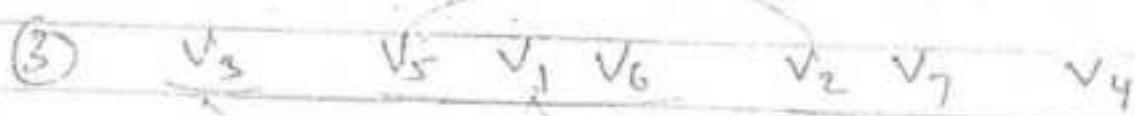
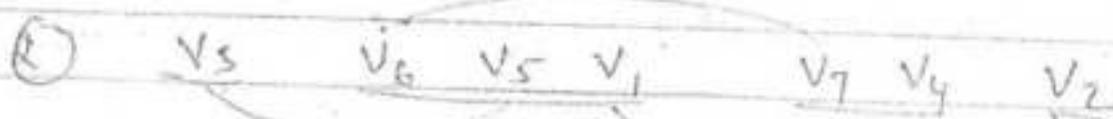
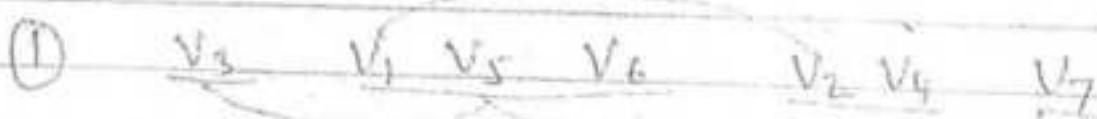
— ↓
 $\underline{O(V+E)}$



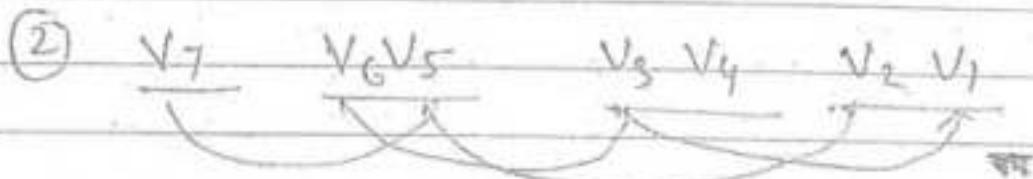
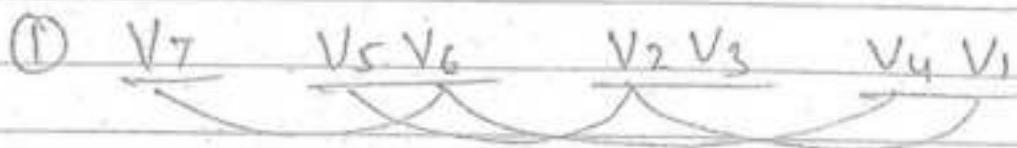
②

③

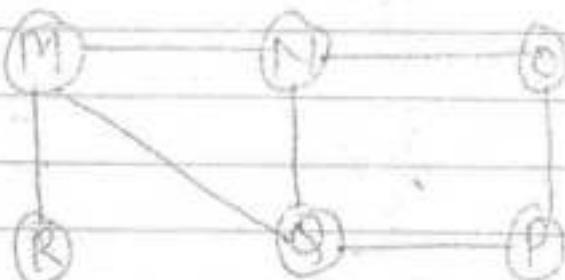
find 4 different answers if you start from
v₃ in BFT.



find 2-different answers if start from
V₇ in BFT.



Consider the following graph.



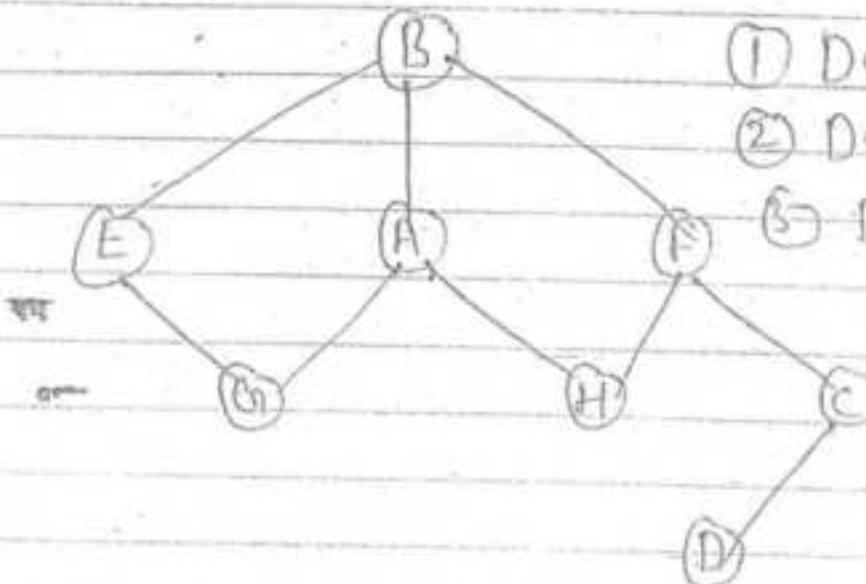
BFT applying on the above graph which one of the following is one possible order of visiting nodes in BFT.

a) MNOPQR b) NQMOPR

c) QMN PRO d) QMNPOR

Q Consider the following Graph.

Find 3-different BFT starting from D



① DCF-BHEAG

② DCF-HBAEG

③ DCF-BHAEG

D D C F B H E A G

E A C E F H B A E G

B D C F B H A E G

Q Consider undirected Graph G.

Let a BFT of G be done starting from a node v .

Let $d(v, u)$ and $d(v, v)$ be the length of the shortest paths from v to u and v to v respectively in G.

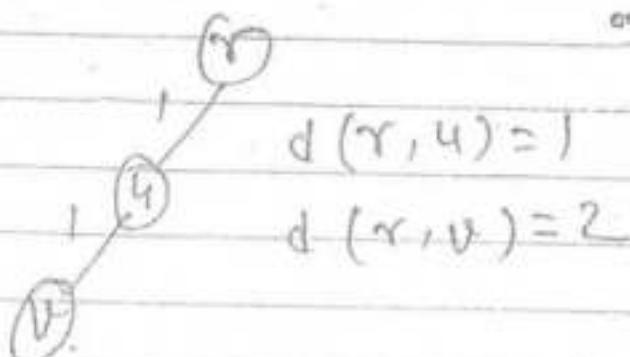
if u visited is visited before v during BFT,
which one of the following statements are
correct.

a) $d(\gamma, u) < d(\gamma, v)$

b) $d(\gamma, u) > d(\gamma, v)$

c) $d(\gamma, u) \leq d(\gamma, v)$

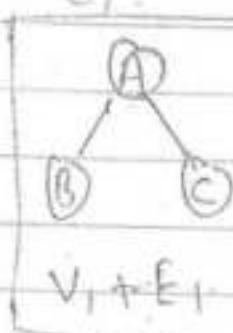
d) $d(\gamma, u) \geq d(\gamma, v)$



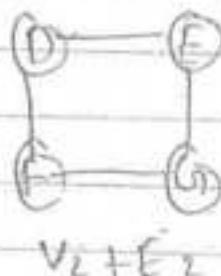
$\gamma \text{ U A V B}$



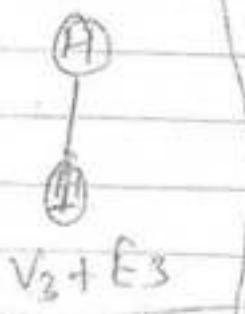
$d(\gamma, u) \leq d(\gamma, v)$



$V_1 + E_1$



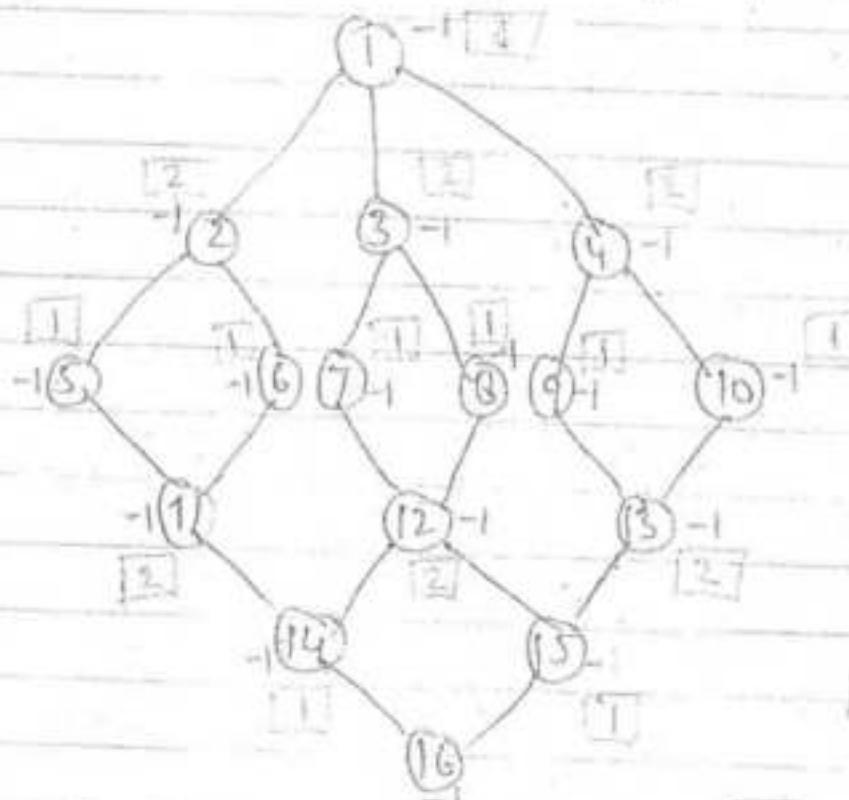
$V_2 + E_2$



$V_3 + E_3$

$C_1(A, B, C, D, E, F, G, H, I) \Rightarrow O(V+E)$

- 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.



$$V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$$

$$M = \{1, 5, 6, 7, 8, 9, 10, 14, 15\}$$

$$N = \{2, 3, 4, 11, 12, 13, 16\}$$



Fold boundary Method

Key = 123456789

M = 1000

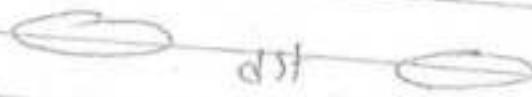
123

789

654321

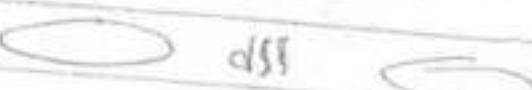
9n	12345678
	9

K₁



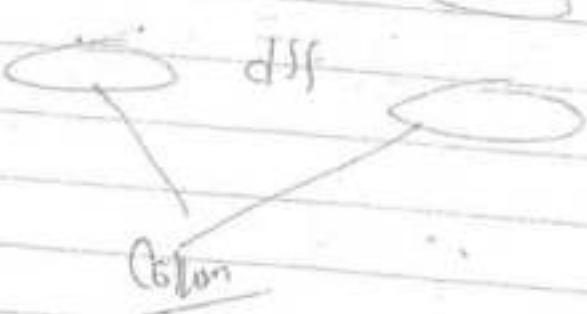
dss

K₂



dss

K₃



dss

C₆₁₀₀

Mid Square

Key

9452

M = 1000



(9452)²



8931034

403	9452 8931034

Digit Extraction Method (Truncation)

$$m = 1000$$

Ex. $379 \underline{4} 52 = 394$

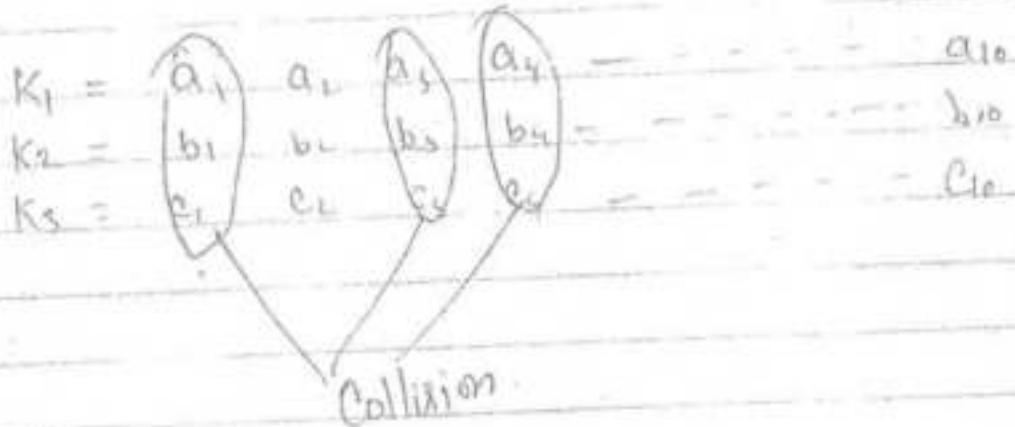
1 3 4

$\underline{1} \underline{2} \underline{1} 26 7 = 112$

$57 \underline{0} \underline{0} 45 = 388$

112	121267
394	379452

- Using digit extraction method selected digit are extracted from the key and use as the hash address.



Folding Method

① fold shift Method

Key = 123 456 789

$m = 1000$

123
456
789
1308

136
8

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

$$\textcircled{1} \quad m = 2^p \alpha$$

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 ↗

Hash Table Size = M

Key

Key Modulo M.

e.g.

Hash Table Size = 10

Key = 625

$$\textcircled{S} = 625 \bmod 10$$

e.g

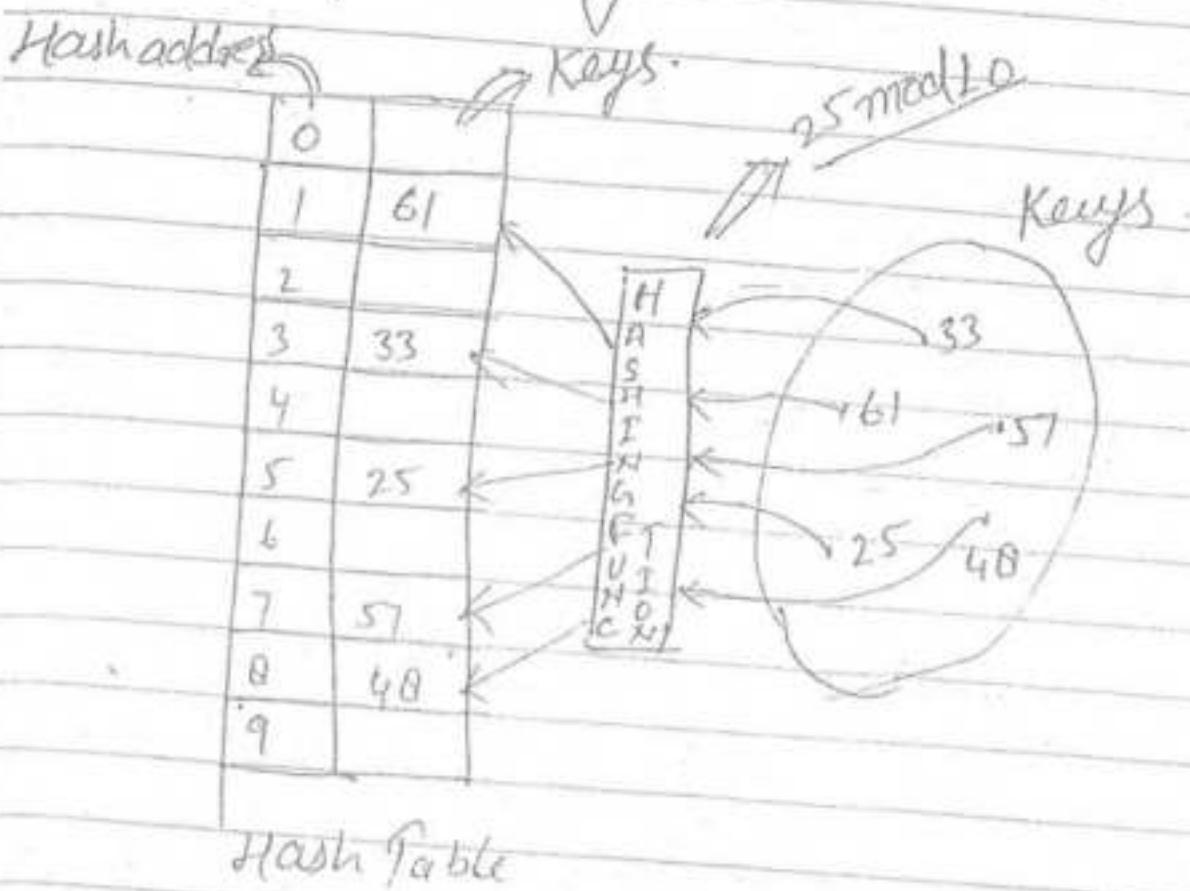
$$m = 2^4$$

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.



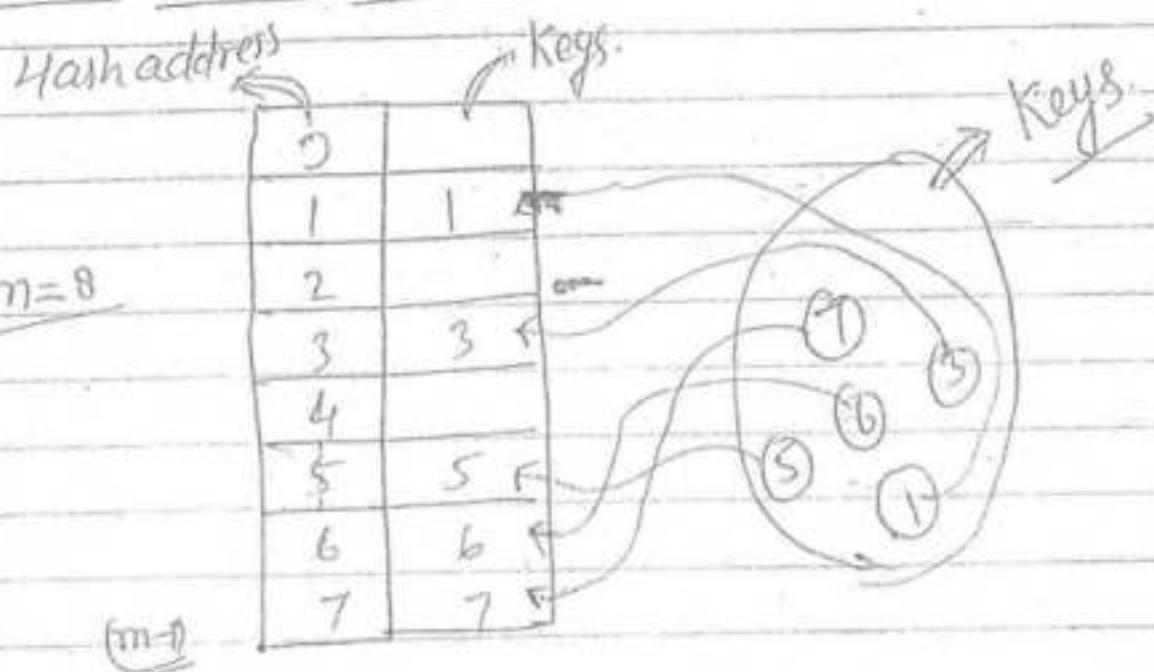
Producing the same Hash address that is already produced is called Collision.

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.

BOrder(root)

```

  {
    if (root != null)
      {
        ...
      }
  }
```

AOrder(root → left);

Point(root → data);

AOrder(root → right);

```

  {
  }
```

```

  }
```

AOrder(root)

```

  {
    if (root != null)
      {
        ...
      }
  }
```

Point(root → data),

BOrder(root → left),

BOrder(root → right);

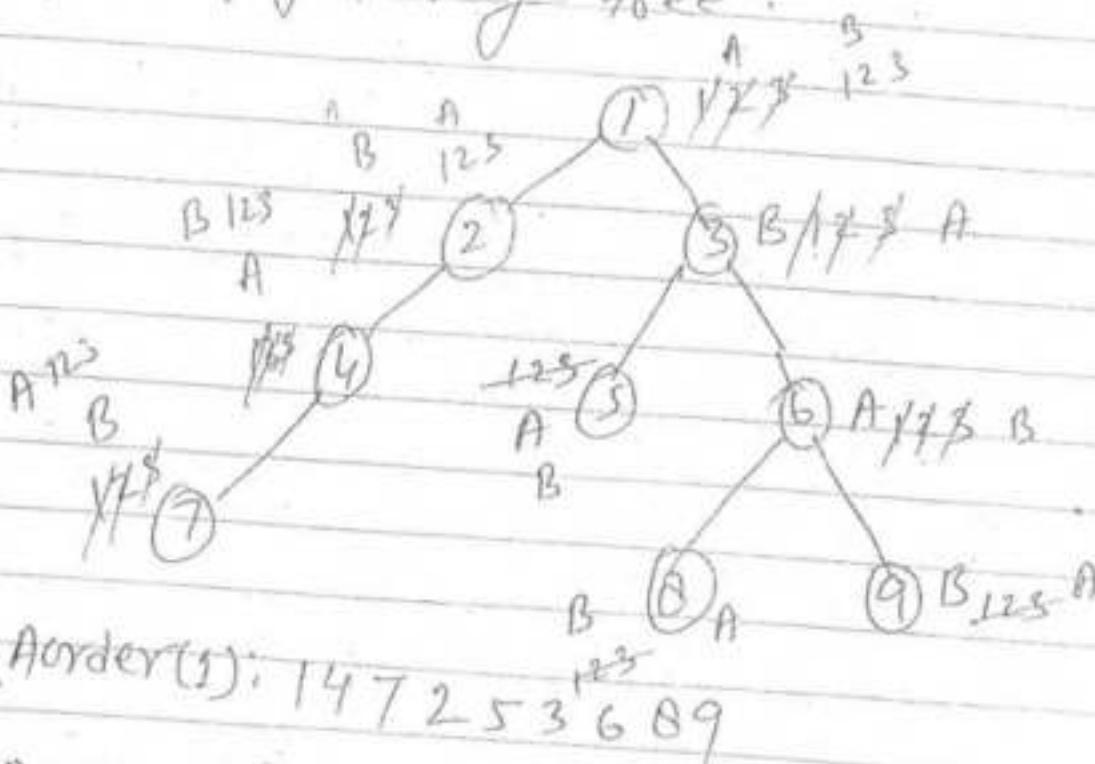
```

  }
```

```

  }
```

What will be the output of the above program
for the following tree.



AOrder(1): 147253689

BOrder(1): 274135069.

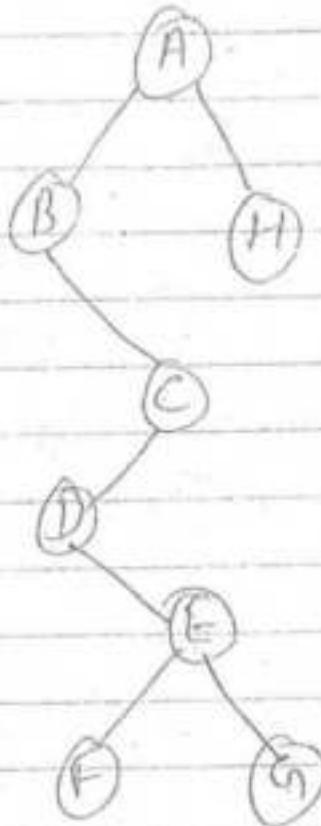
Q. Construct the following C program. What will be the O/P if we apply the given program on binary search tree.

```

    To (root)
    {
        if (root != null)
            {
                printf (root-data)
                To (root->left)
                printf (root->data)
                To (root->right)
                printf (root->data)
            }
    }
  
```

Sol:

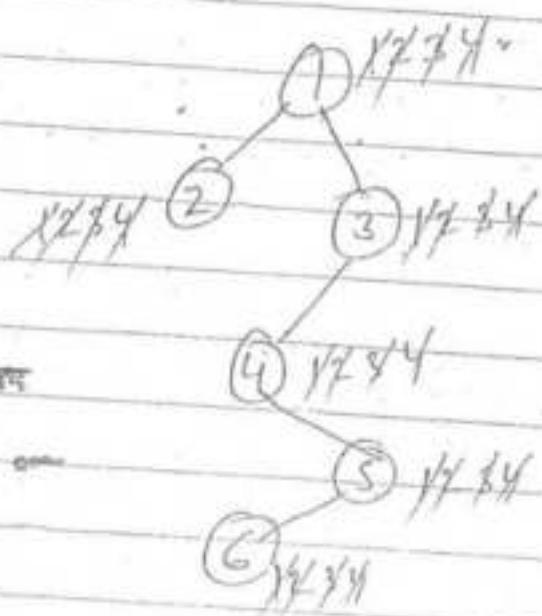
A B B C D D E F F F
 E G G G E D C C B A
 H H H A



A B B

- ③ Print(root.data)
- ④ Do [root → right]

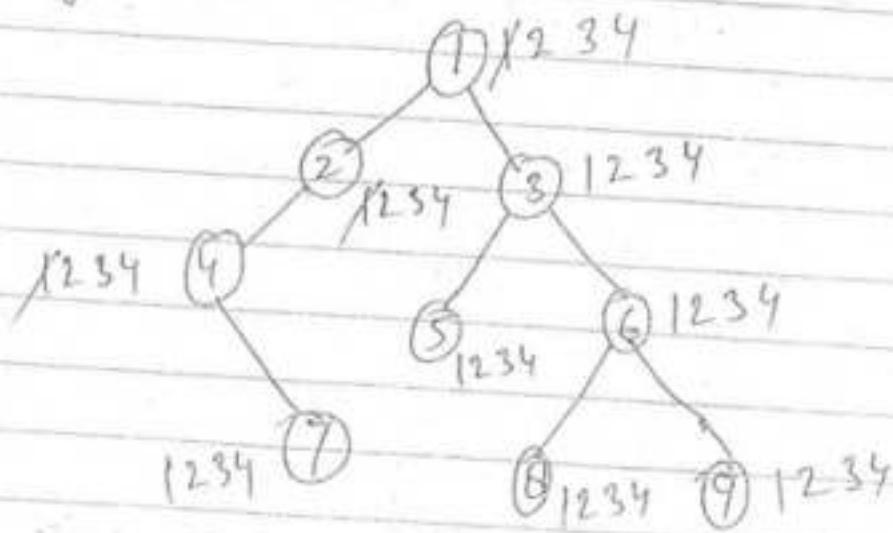
}



Sol:

~~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.



1 2 4 4 7 7 2 1 3 5 5 3 6 8 8 6 9 9

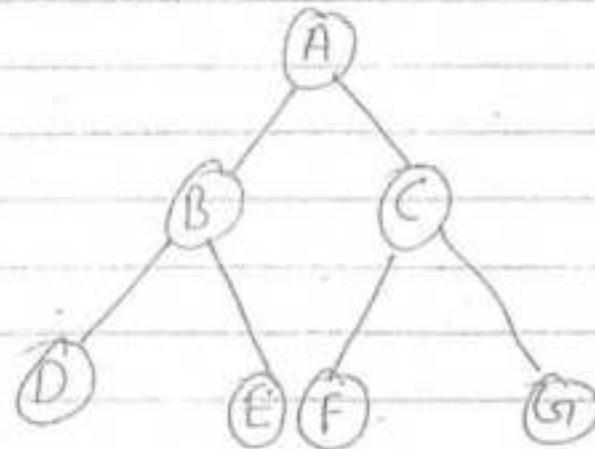
ABC (Root)

if (root != null)

① ABC (root → right)

② Printt (root → data)

③ ABC (root → left)



G, C F, A, E B, D

What will be the output for the following program .

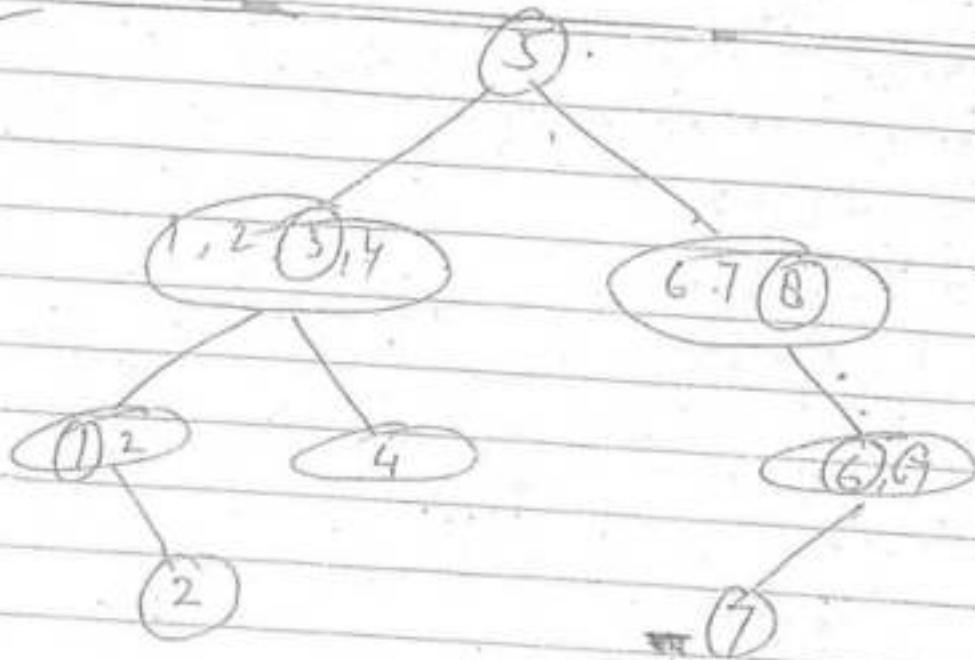
Do (root)

{

if (root != null)

① Printt (root - data)

② Do (root → left)



Post order 2, 1, 4, 3 7 0 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 RSR $\mathcal{O}(n)$
 $\mathcal{O}(n)$

\downarrow
 $\mathcal{O}(n^2)$

eg PreOrder - A B D E C F G

Postorder - D E B F G C A

H.

(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

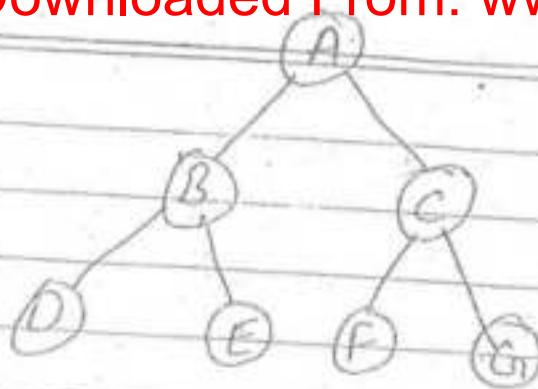
L

A binary search tree contains following PreOrder

PreOrder \Rightarrow 5, 3, 1, 2, 4, 6, 8, 7

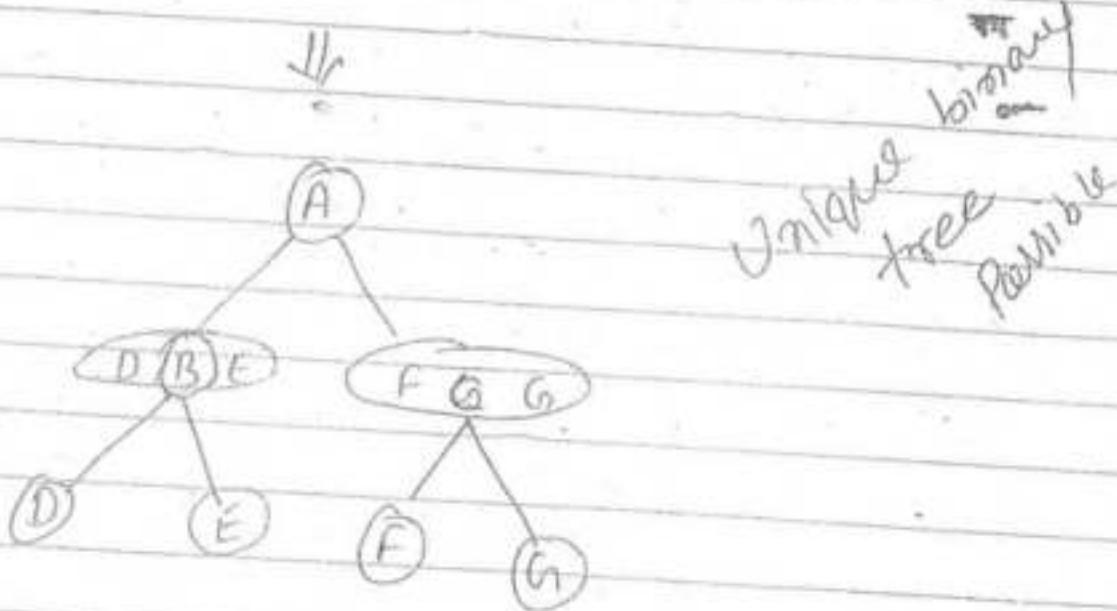
PostOrder \Rightarrow

- 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



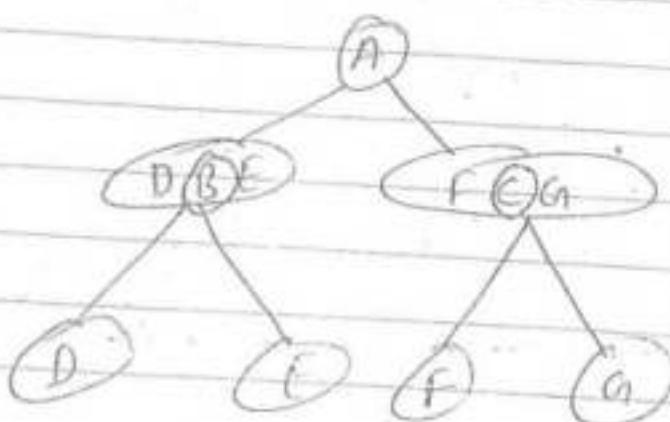
PreOrder - A B D E C F G

InOrder - D B E A F C G.

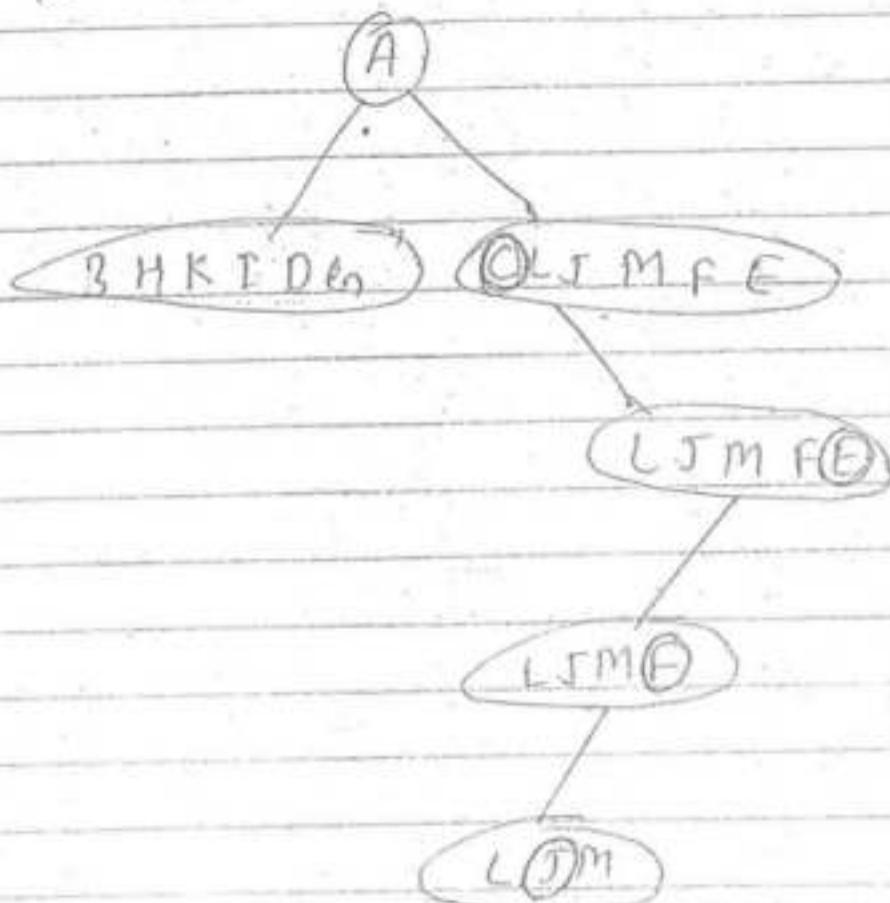


PostOrder - D E B F G C A

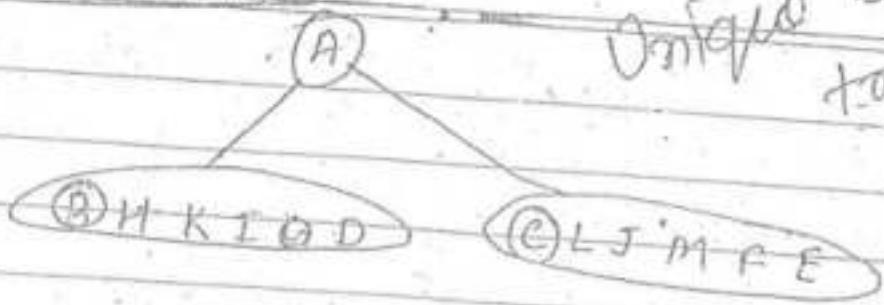
InOrder - D B E A F C G.

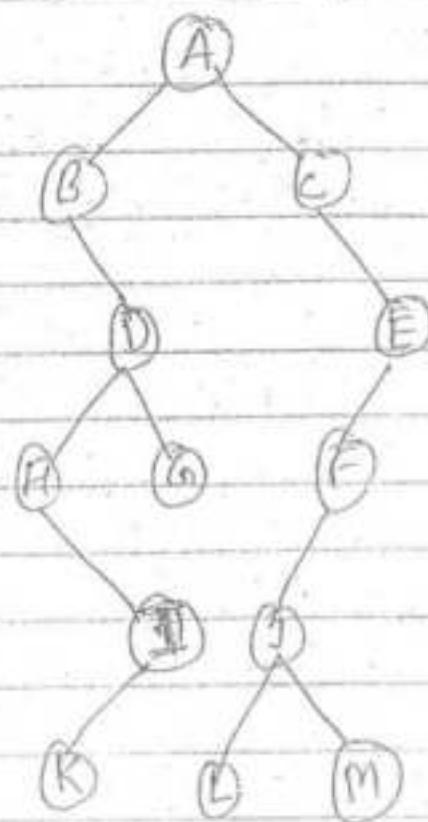


PostOrder - InOrder



Unique binary
tree possible





PreOrder - (R, LST, RST) \Rightarrow A, B, D, F, G, E, C, I, H, J, K, L, M

PostOrder - (LST, RST, Post) \Rightarrow K, J, I, H, G, D, B, L, M, G, F, E, C, A

InOrder - B, H, K, J, I, D, G, A, C, L, M, F, E

Postorder

Postorder (Root)

```
{  
    if (root != null)  $\Rightarrow$  O(n)  
    {  
        Postorder ( Root  $\rightarrow$  left )  
        Postorder ( Root  $\rightarrow$  right )  
        Pointt ( Root  $\rightarrow$  data )  
    }  
}
```

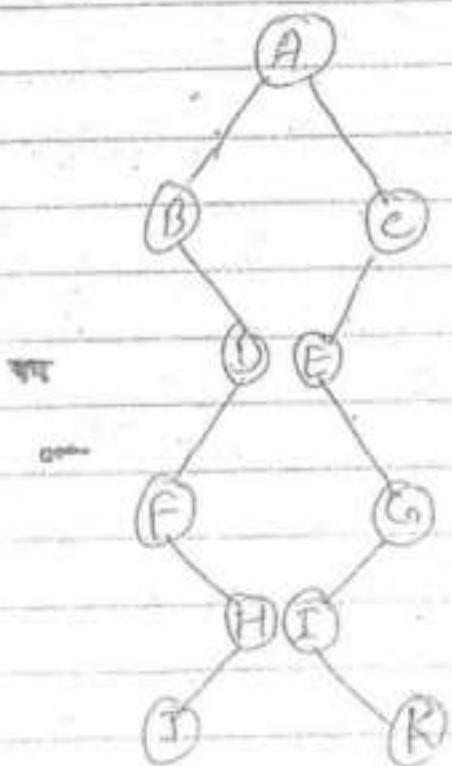
InOrder

InOrder (root)

```
{  
    if (root != null)  $\Rightarrow$  O(n)  
    {  
        InOrder ( Root  $\rightarrow$  left ) ;  
        Pointt ( root  $\rightarrow$  data ) ;  
        InOrder ( root  $\rightarrow$  right ) ;  
    }  
}
```

Monday

- 23/01
- 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, A,

InOrder - B, F, J, H, D, A, E, G, I, K, C,

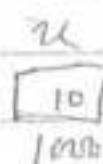
PreOrder

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

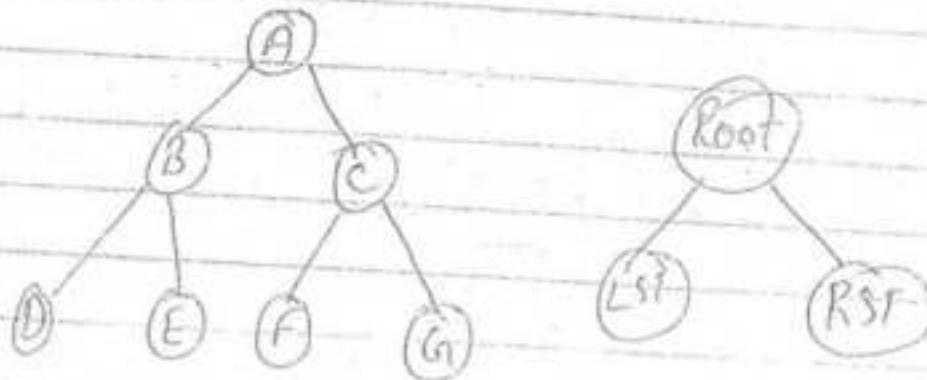


u	10
&u	1000
reference operator	
*(&u)	10
dereference operator	

Tree Traversal (Unique)

- ① InOrder - (LST , Root , Right Sub Tree (RST))
- ② PreOrder (Root , LST , RST)
- ③ PostOrder (LST , RST , Root)

Eg

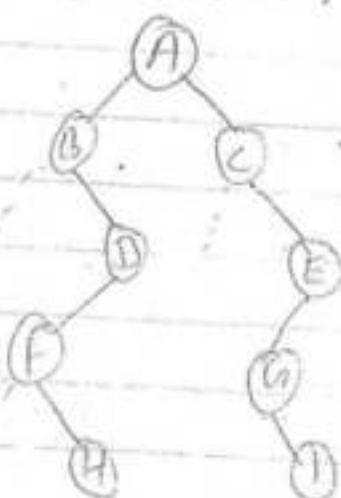


PreOrder - A, B, D, E, C, F, G

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, F, H, E, G, I

Postorder - H, F, D, B, I, G, E, C, A

InOrder - B, F, H, D, A, I, G, C, E

every strongly connected graph is
also weakly connected.
but every weakly connected graph
not be strongly connected.

ANS

Ques

graph :-

Directed

undirected.

Connected

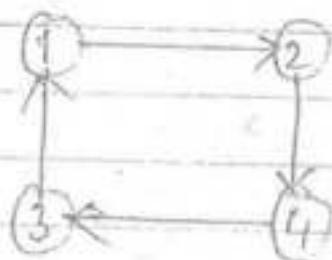
disconnected.

Strongly

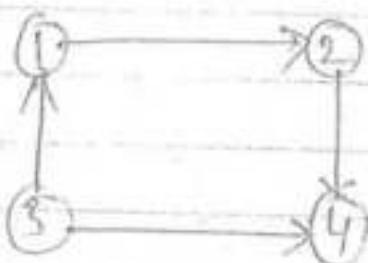
weakly

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.

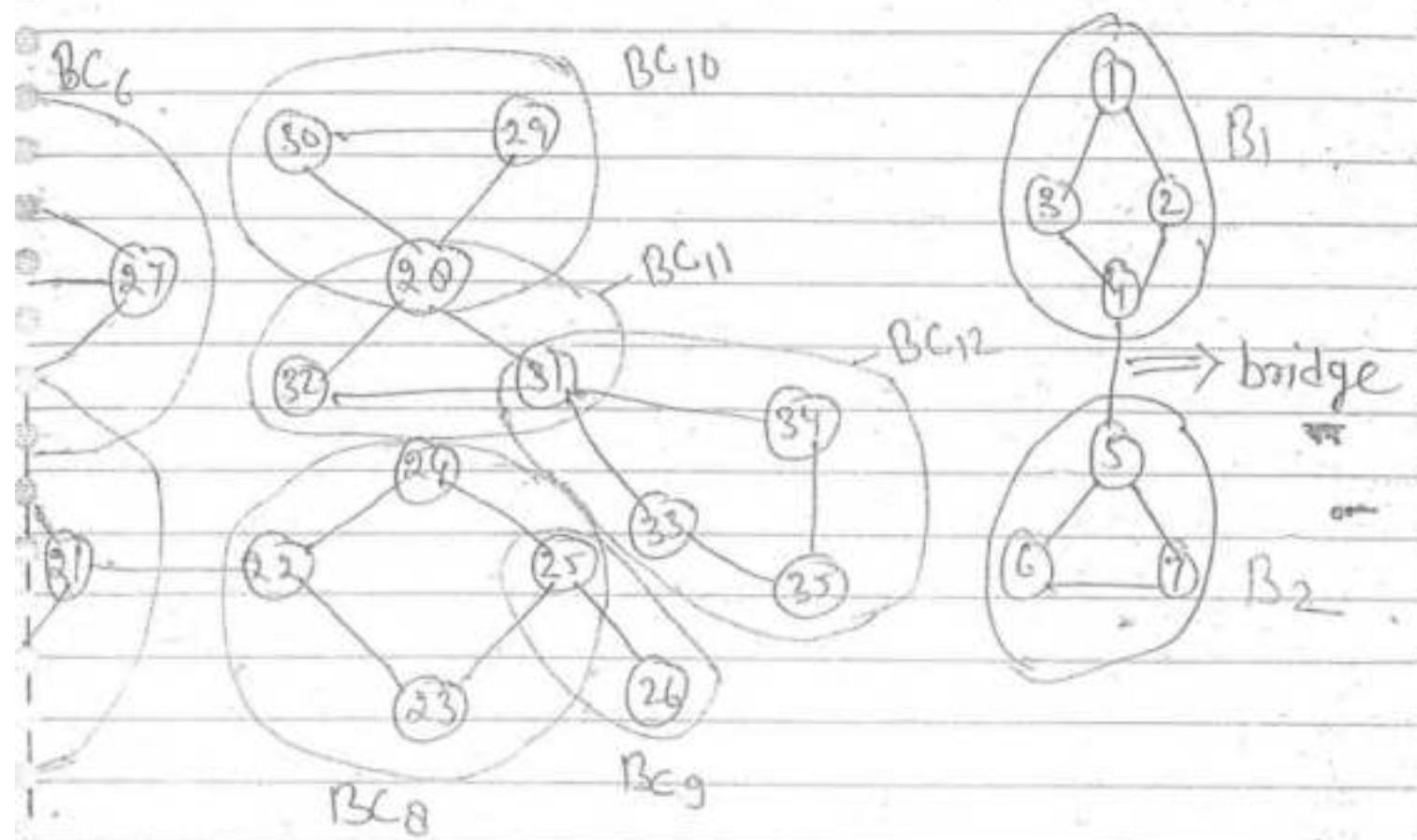
Note :-

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.

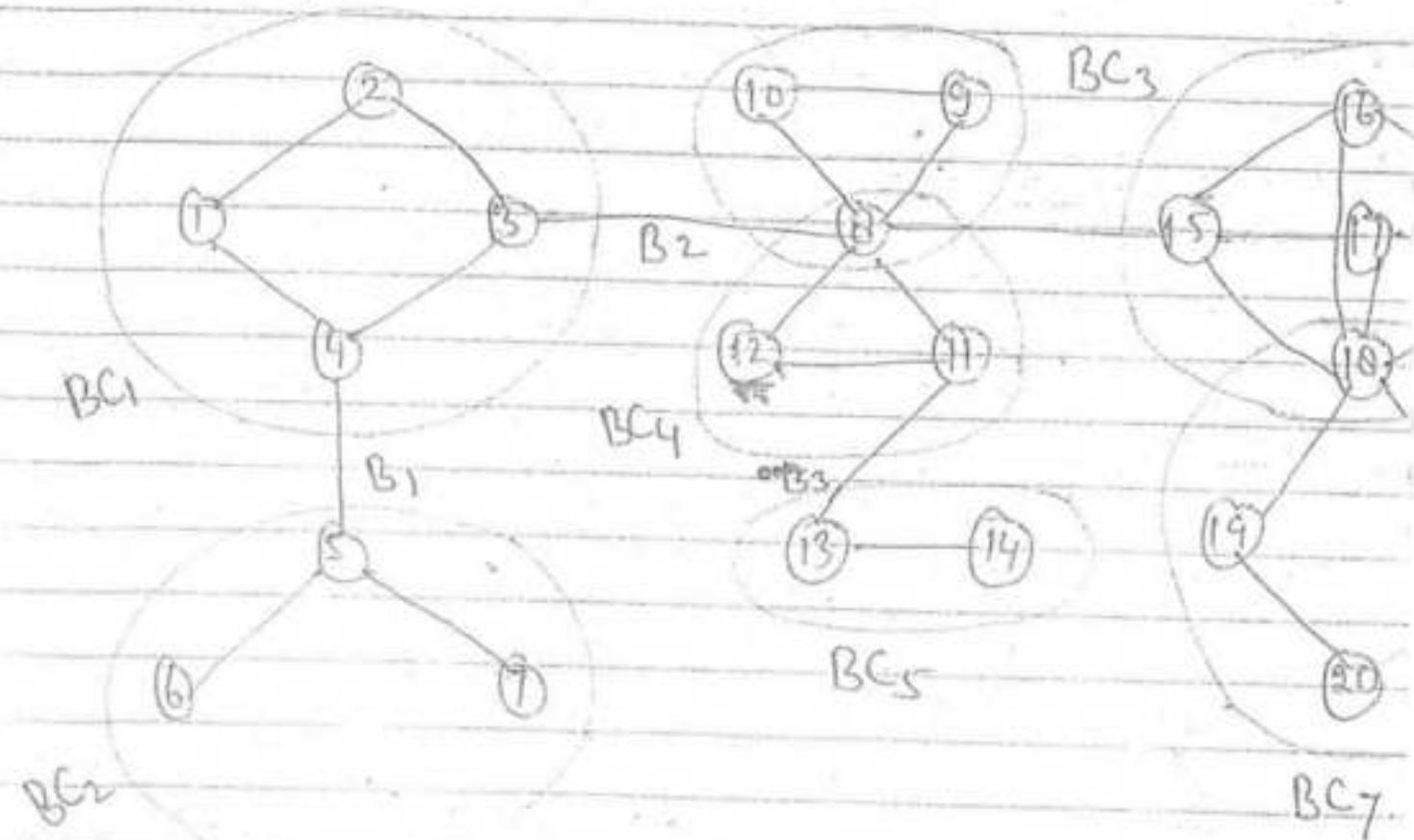


Using DFT algo we can find out bridge also.

Bridge = Any connected graph by deleting an edge if the graph is disconnected the edge is called bridge.

Using DFT algo we can find out bridge also.

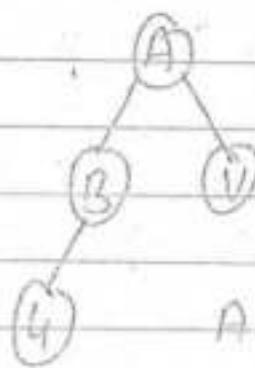
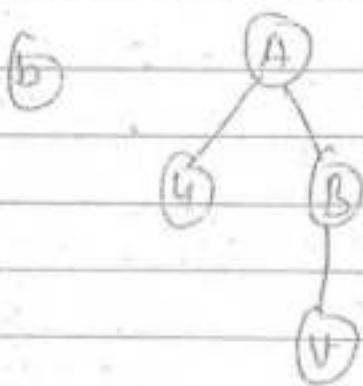
Find Out the no of articulation point of following graph.



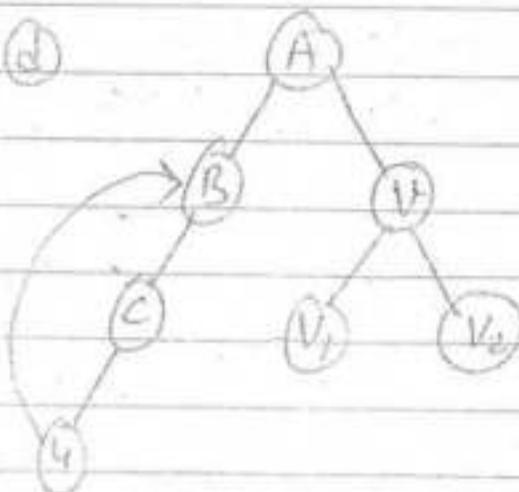
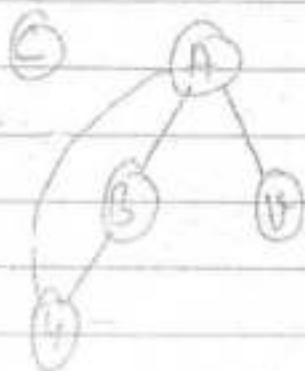
articulation point $\Rightarrow 14$
 bridge $\Rightarrow 8$

Note ① In a particular computer network the number of routers, which are behaving like a articulation point.

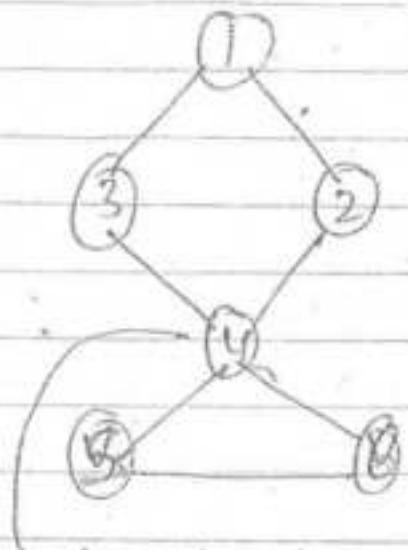
② Using DFT we can find all possible articulation point in given graph.



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.

(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.

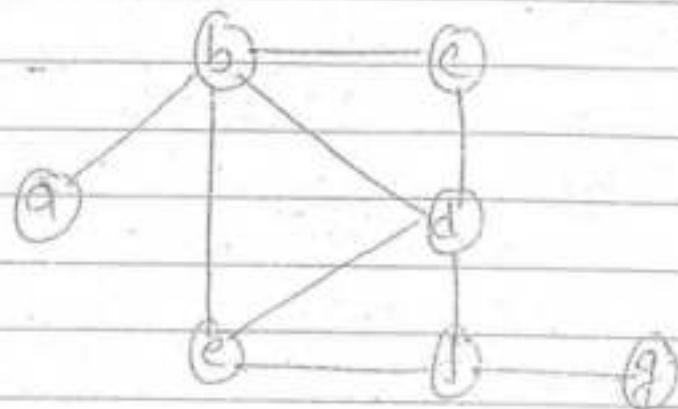
Let G_7 be undirected Graph.

Consider a DFT of g and let T be the resulting BFT 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?

- $\{u, v\}$ must be an edge in G
- if $\{u, v\}$ is not an edge in G_7 , then $\{u, v\}$ must have same parent
- if (u, v) is not an edge in g then v is leaf node in G .
- If $\{u, v\}$ is not an edge in G then u is leaf in T

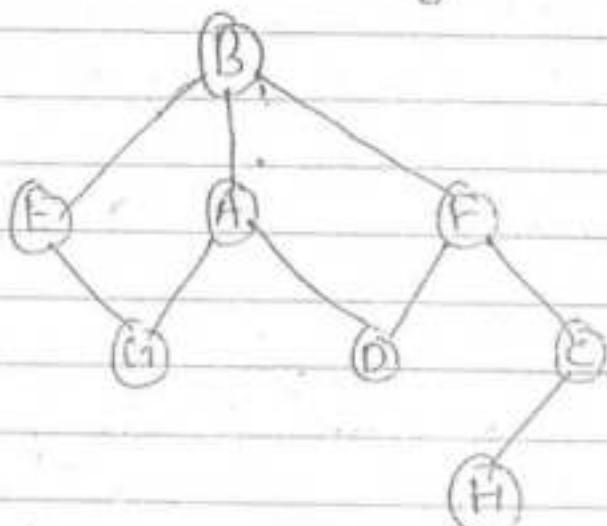
Consider the following graph.



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.

$v_1, v_5, v_7, v_0, v_6, v_t, v_3, v_4$

find DFT by considering always ascending order starting from v_3

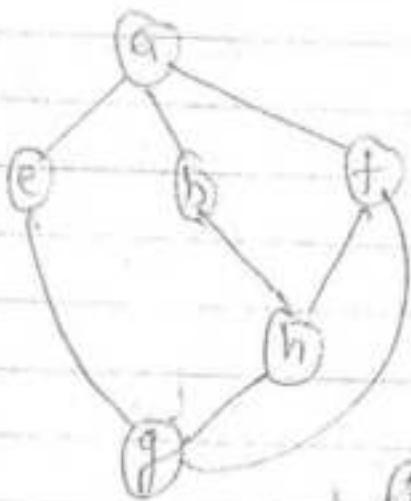
$v_3, v_1, v_2, v_6, v_8, v_7, v_4, v_5$

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

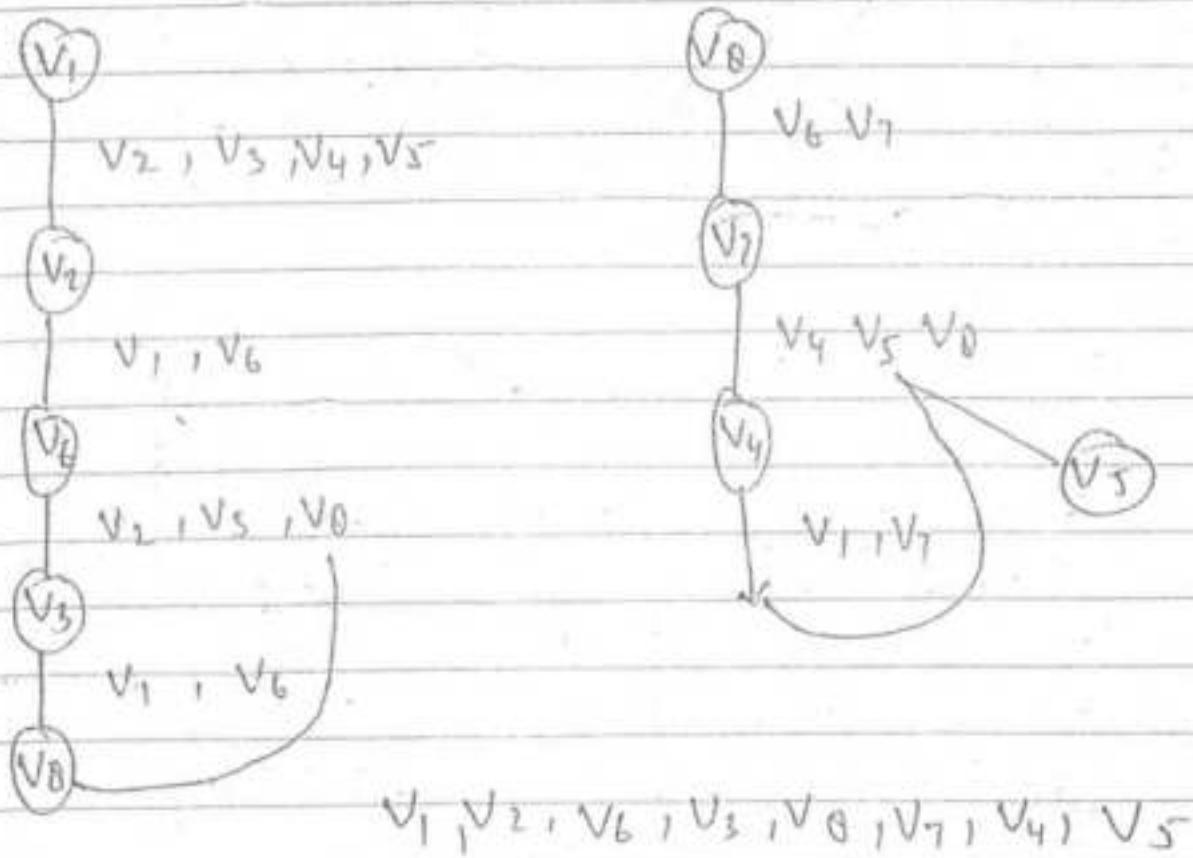
vertex \Rightarrow DFT

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_5)

①

② $w = v_1, v_6$

③

DFT(v_6)

①

② $w = v_3, v_8$

③

DFT(v_3)

①

② $w = v_1, v_6$

DFT(v_8)

①

② $w = v_6, v_7$

③

DFT(v_7)

①

② $w = v_4, v_5, v_0$

③

DFT(v_4)

①

② $w = v_1, v_6$

③

DFT(v_6)

①

② $w = v_1, v_7$

③

DFT

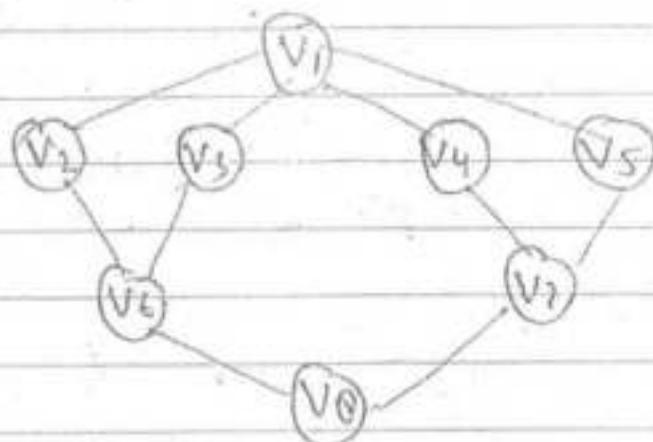
DFT(v)

① visited(v) = 1

② for all w adjacent to v

 ③ if (w is not visited)

 ④ DFT(w);



V₁ V₂ V₃ V₄ V₅ V₆ V₇ V₈ V₉ V₁₀

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

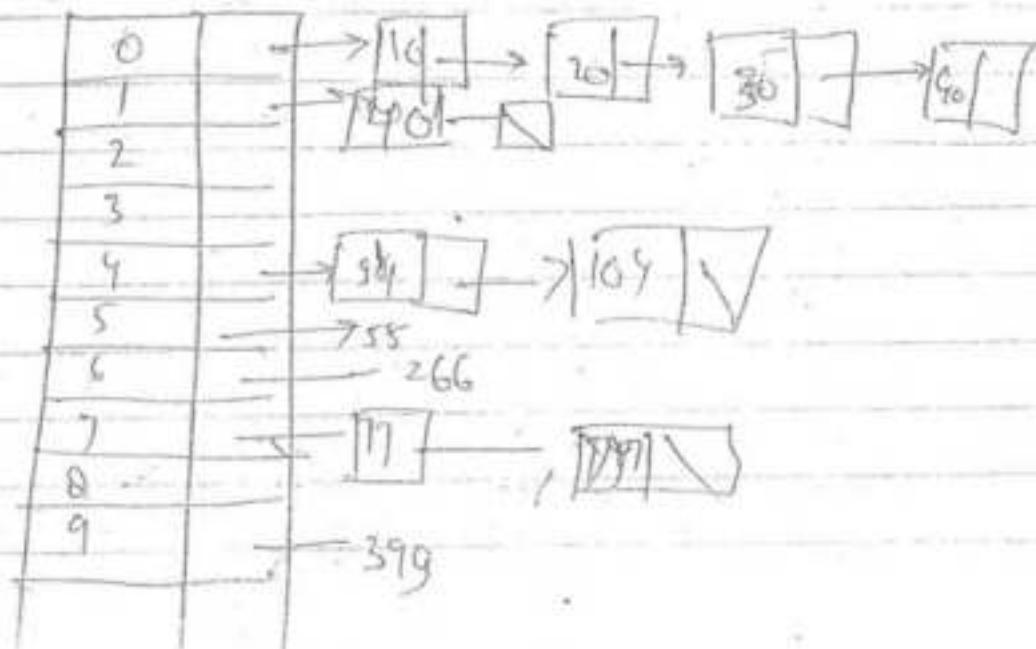
Double
Hashing

Collision - Resolution by Chaining

Key: 10 77 94 899 461 565 104 266 747

Hash function, Key modulo 10

Collision Resolution Technique = Chaining

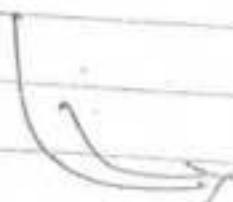


- D We are taking extra space extra to store
 - ④ When collision occurs, instead using a vacant space. Even table have empty space.
- D In the worst case it will be having linear searching.
- ③ Unlimited collisions will be handled by this method.
- D Does not required pre-knowledge of how many are coming into hash.
- If the elements are stored in hash table using Chaining, we can delete those elements 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 Probing

Keys: 25, 30, 77, 55, 62, 78, 99, 52, 101, 41

Hash funⁿ = Key modulo 10

Collision Resolution technique = Linear Probing.

$$h'(K) = K \bmod m$$

$$h(K, i) = (h'(K) + i) \bmod m$$

$$\forall i \in \{0, 1, 2, 3, \dots, m-1\}$$

0	30
1	10)
2	62
3	52
4	41
5	25
6	55
7	77
8	70
9	99

$$\checkmark h'(25) = 25 \bmod 10 \\ = 5$$

$$h(25,0) \doteq (h'(k) + 0) \bmod 10 \\ = 5$$

$$\checkmark h'(30) = 30 \bmod 10 \\ = 0$$

$$h(30,0) = (0+0) \bmod 10 \\ = 0$$

$$\checkmark h'(77) = 77 \bmod 10 \\ = 7$$

$$h(77,0) = (7+0) \bmod 10 \\ = 7$$

$$\checkmark h'(55) = 5$$

$$h(55,0) = (5+0) \bmod 10 \\ = 5$$

$$h(55,1) = (5+1) \bmod 10 \\ = 6$$

$$\checkmark h'(62) = 2$$

$$h(62,0) = 2$$

$$\checkmark h'(52) = 2$$

$$h(52,0) = (2+0) \bmod 10 \\ = 2$$

$$h(52,1) = (2+1) \bmod 10 \\ = 3$$

$$h'(41) = 1$$

$$h(41,0) = (1+0) \bmod 10 \\ = 1$$

$$h(41,1) = (1+1) \bmod 10 \\ = 2$$

$$h(41,2) = (1+2) = 3$$

$$h(41,3) = (1+3) \bmod 10 \\ = 4$$

Q. Let $h(K) = K \bmod 7$.

Keys 1, 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, 11, 2, 3, 23, 5 and 15 are

inserted into an initially empty, hash-table length
i.e., using linear probing.

$$\text{hash fun} = K \bmod 10$$

0	
1	10
2	12
3	13
4	2
5	3
6	23
7	
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
1	0
2	31
3	1
4	
5	7
6	23
7	15
8	11
9	4
10	3

$\text{Key} = 48$

$$u = 48 \times 48;$$

$$u = (48 \times 48) / 16 = 48 \times 3$$

$$u = (48 \times 3) + 48; = 107$$

$$u = 0$$

return (0)

$\text{Key} = 23$

$$u = 23 \times 23$$

$$u = 49$$

$$u = 49 + 23 = 72$$

$$u = 6$$

$\text{Key} = 1$

$$u = 2 \times 1 = 3$$

$$u = 36$$

$$u = 36 / 16 \equiv 2$$

$$u = 3 \% 11 = 3$$

$$u = 3$$

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	53
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 resulting in the hash table shown above.

- a) 10 , b) 20 ~~c) 30~~ d) 40

~~Soln~~

$\frac{46, 34, 42, 25}{\text{---}}$, $\frac{52, 33}{\text{---}}$

$$4! = 24$$

$\frac{34 \quad 42 \quad 25}{\text{---}}$, $\frac{52 \quad 46 \quad 33}{\text{---}}$

$\frac{11}{\text{---}}$

$$3! = 6$$

$$\text{total} = 24 + 6 = 30$$

~~Q10)~~

- ① In linear probing we have probing one slot after another..
- ② If an empty slot 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	38
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

$$\checkmark h'(30) = 30 \bmod 10$$

$$= 0$$

$$h(30, 0) = (h'(30) + C_1 \cdot 0 + C_2 \cdot 0) \bmod 10$$

$$= (0 + 0 + 0) \bmod 10$$

$$= 0$$

$$\checkmark \frac{h'(20)}{0} = 20 \bmod 10$$

$$h(20, 0) = (h'(20) + 0 + 0^2) \bmod 10$$

$$= 0$$

$$h(20, 1) = (h'(20) + 1 + 1 \cdot 1^2) \bmod 10$$

$$= 0 + 1 + 1$$

$$= 2$$

$$h'(30)$$

$$0 \equiv 30 \pmod{10}$$

$$h(30,0) = (h'(30) + 0 + 0^2) \pmod{10}$$

$$= 0$$

$$h(30,1) = (h'(30) + 1 + 1 + 1 \cdot 1^2) \pmod{10}$$

$$= 0 + 1 + 1$$

$$h(30,2) = (h'(30) + 1 + 2 + 1 \cdot 2^2) \pmod{10}$$

$$= (0 + 2 + 4) \pmod{10}$$

$$= 6$$

✓ $h'(40) = 40 \pmod{10}$

$$= 0$$

$$h(40,0) = (0 + 0 + 0^2) \pmod{10}$$

$$= 0$$

$$h(40,1) = (0 + 1 + 1^2) \pmod{10}$$

$$= 2$$

$$h(40,2) = (0 + 2 + 2^2) \pmod{10}$$

$$= 6$$

$$h(40, 3) = (0 + 3 + 3^2) \bmod 10$$

$$= \underline{\underline{42}} \quad \underline{\underline{2}}$$

$$h(40, 4) = (0 + 4 + 4^2) \bmod 10$$

$$= \underline{\underline{0}}$$

$$h(40, 5) = (0 + 5 + 5^2) \bmod 10$$

$$= \underline{\underline{0}}$$

$$h(40, 6) = (0 + 6 + 6^2) \bmod 10$$

$$= \underline{\underline{2}}$$

$$h(40, 9) = (0 + 9 + 9^2) \bmod 10$$

$$= \underline{\underline{0}}$$

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 same path. This problem is known as Secondary clustering bcoz of secondary clustering a large scaling little bit increases.

Keys

= 10, 22, 31, 4, 15, 18, 17, 88, 59

$$M = 11$$

$$c_1 = 1, c_2 = 3.$$

$$h'(k) = k \quad ; \quad h(k, i) = (h(k) + c_1 \cdot i + c_2 \cdot i^2) \bmod M$$

$$C.R.F = Q.P.$$

0	22
1	59
2	
3	88
4	4
5	18
6	17
7	10
8	15
9	31
10	10

$$\checkmark h(10, 0) = (10 + 0 + 0) \bmod 11 \\ = 10$$

$$\checkmark h(22, 0) = (22 + 0 + 0) \bmod 11 \\ = 0$$

$$\checkmark h(31, 0) = (31 + 0 + 0) \bmod 11 \\ = 9$$

$$\checkmark h(15, 0) = (15 + 0 + 0) \bmod 11 \\ = 4$$

$$h(15, 1) = (15 + 1 + 3 \cdot 1^2) \bmod 11 \\ = 9$$

$$\checkmark h(10, 0) = (15 + 0 + 0) \bmod 11 \\ = 7$$

$$\checkmark h(17,0) = (17+0+0) \bmod 11 \\ = 6$$

$$\checkmark h(88+0) = (88+0+0) \bmod 11 \\ = 0$$

$$h(88,1) = (88+1+3) \bmod 11 \\ = 4$$

$$h(88,2) = (88+2+3 \cdot 2^2) \\ = (102 \bmod 11) = 5$$

$$\checkmark h(59,0) = (59+0+0) \bmod 11 \\ = 4$$

$$h(59,1) = (59+1+3) \bmod 11 \\ = 63 \bmod 11 = 8$$

$$h(59,2) = (59+2+3 \cdot 2^2) \bmod 11 \\ = 7$$

$$h(59,3) = (59+3+3 \cdot 2^2) \bmod 11 \\ = 1$$

L.P > Q.P

more searching less searching

Double Hashing (Brook Hashing) Pedicnic

L.P



Primary Cloustring



Quadratic Probing



S.C



Double Hashing

$$h(LK, i) = (h_1(K) + i \cdot h_2(K)) \bmod m$$

$$\forall i \in \{0, 1, 2, 3, \dots, m-1\}$$

$$h_1(K) = K \bmod m$$

$$h_2(K) = 1 + (K \bmod (m-1))$$

~~Q8~~

$$M = 10$$

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	

$$h_1(10) \stackrel{def}{=} 10 \bmod 10$$

$$= 0$$

$$\checkmark h_1(10, 0) = (0 + 0 \cdot h_2(k)) \bmod 10$$

$$= 0$$

$$h_1(25) \stackrel{def}{=} 25 \bmod 10$$

$$= 5$$

$$h_1(25, 0) = 5 + 0 \cdot h_2(k) \bmod 10$$

$$= 5$$

$$h_1(14) \stackrel{def}{=} 14 \bmod 10$$

$$= 4$$

$$h_1(14, 0) = (4 + 0 \cdot h_2(k)) \bmod 10$$

$$= 4$$

$$h_1(16) \stackrel{def}{=} 16 \bmod 10$$

$$= 6$$

$$h_1(16, 0) = (16 + 0 \cdot h_2(k)) \bmod 10$$

$$= 6$$

$$h_1(20) = 20 \bmod 10 \\ = 0.$$

$$h_1(20,0) = (0 + 0 \cdot h_2(5)) \bmod 10 \\ = 0$$

$$h_2(20) = 1 + (20 \bmod 9) \\ = 1 + 2 \\ = 3$$

$$h(20,1) = (0 + 1 \cdot 3) \bmod 10 \\ = 3$$

$$h_1(30) = (0 + 0 \cdot h_2(30)) \bmod 10 \\ = 0$$

$$h_2(30) = 1 + 3 \bmod 9 \\ = 1 + 3 = 4$$

$$h(30,1) = (0 + 1 \cdot 4) \bmod 10 \\ = 4$$

$$h(30,2) = (0 + 2 \cdot 4) \bmod 10 \\ = 0$$

$$h_1(35) = 35 \bmod 10$$

$$= 5$$

$$h_2(35)_2 = 1 + 35 \bmod 10$$

$$= 1 + 8 = 9.$$

$$h(35,0) = (5+0 \cdot h_2(5)) \bmod 10$$

$$= 5$$

$$h(35,1) = (5+1 \cdot 9) \bmod 10$$

$$= 4$$

$$h(35,2) = (5+2 \cdot 9) \bmod 10$$

$$= 3$$

$$h(35,3) = (5+3 \cdot 9) \bmod 10$$

$$= 2$$

$$h_1(45) = 45 \bmod 10$$

$$= 5$$

$$h_2(45) = 1 + 45 \bmod 9$$

$$= 1$$

$$h(45,0) = (5+0 \cdot 1) \bmod 10$$

$$= 5$$

$$h(45,1) = (5+1 \cdot 1)$$

$$= 6$$

$$h(45,2) = (5+2 \cdot 1)$$

$$= 7$$

- * In linear probing whenever collision is occurred we will always add $i + 1$. 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.

27/07/01

eg

$$M=13, h_1(K) = K \bmod M, h_2(K) = 1 + K \bmod 11$$

$$29, 69, 72, 98, 10, 50$$

$$h(K, i) = (h_1(K) + i \cdot h_2(K))$$

0	
1	
2	
3	29
4	69
5	98
6	
7	72
8	
9	
10	10
11	50
12	

$$h_1(29) = 29 \bmod 13$$

$$h_1(29) = 3$$

$$h(29, 0) = (3 + 0 \cdot h_2(K)) \\ = 3$$

$$h(69, 0) = (4 + 0 \cdot h_2(K)) \\ = 4$$

$$h(72, 0) = (7 + 0 \cdot h_2(K)) \\ = 7$$

$$h_1(98) = 98 \bmod 13 \\ = 7$$

$$h_2(98) = 1 + 98 \bmod 11 \\ = 1 + 10 \\ = 11$$

$$h(98, 0) = (7 + 0 \cdot h_2(K)) \bmod 11 \\ = 7$$

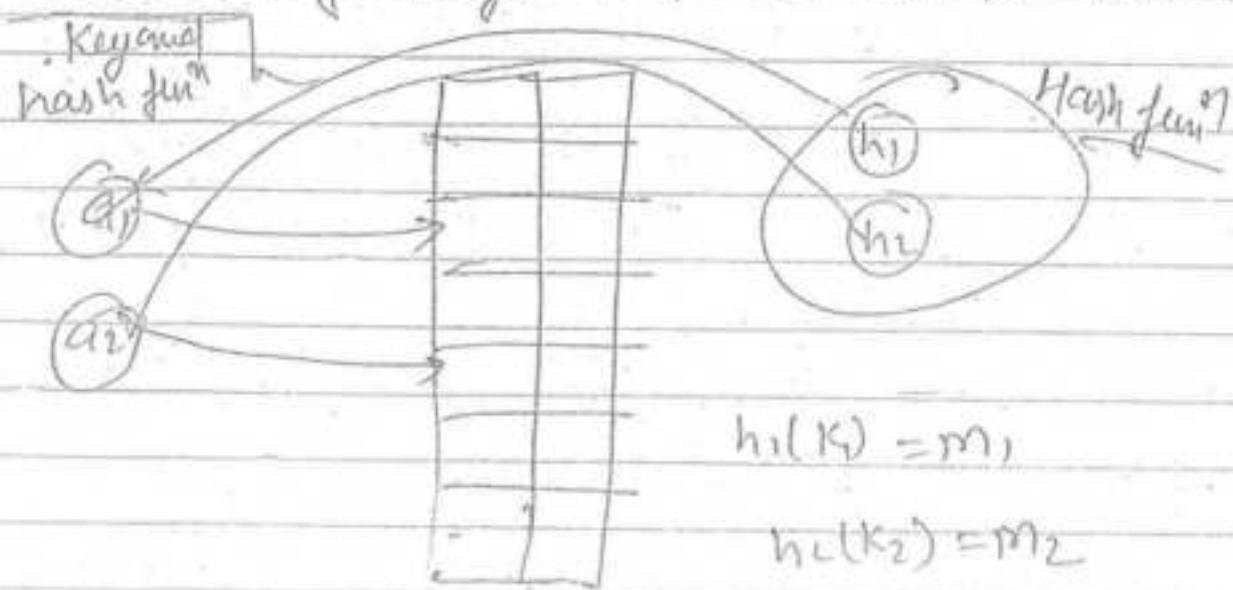
$$h(98, 1) = 7 + 1 \cdot 11$$

$$= 10 \bmod 11$$

$$= 5$$

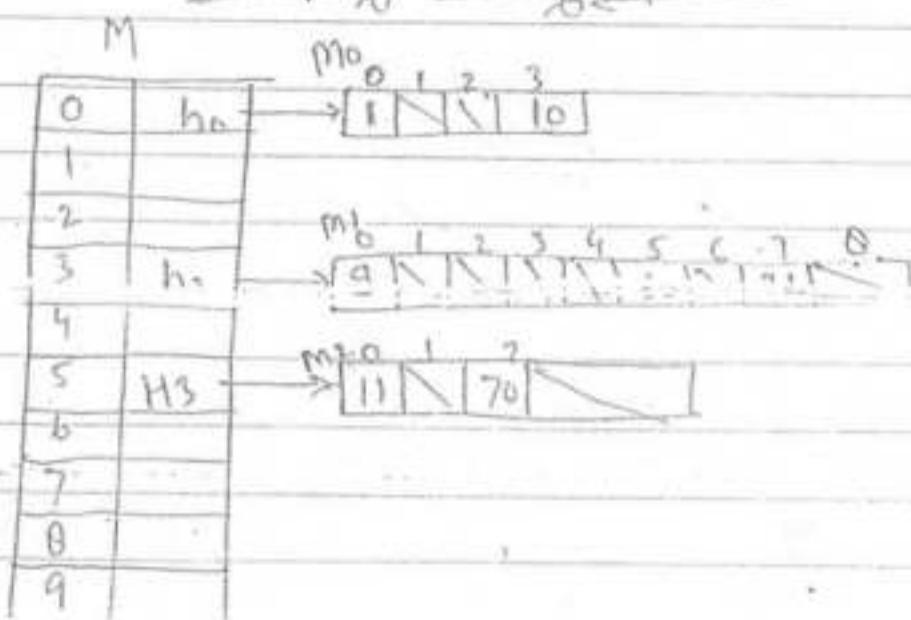
Uniqueness of Hashing

For any choice of hash function there exist bad set of keys that all hash to same slot.



The solⁿ to this problem pick hash funⁿ randomly, which is known as Universal Hashing -

Perfect Hashing



$$h(1) = 0$$

$$h_0(1) = 0$$

$$h(10) = 0$$

$$h_0(10) = 3$$

$$h(9) = 5$$

$$h_2(9) = 0$$

$$h(60) = 3$$

$$h_2(60) = 5$$

$$h(7L) = 3$$

$$h_2(7L) = 7$$

$$h(11) = 5$$

$$h_3(11) = 0$$

$$h(70) = 5$$

$$h_3(70) = 2$$

Note-

~~Instead~~ 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

Y.L
(accepted)
(rejected)

Y.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 $\Rightarrow O(n)$
- (2) Bubble Sort $\Rightarrow \Theta(n^2)$
- (3) Quick Sort $\Rightarrow O(n \log n)$
- (4) Euler Graph $\Rightarrow \Theta(V^2)$
- (5) Prim's algo $\Rightarrow (E+V) \log V$
- (6) Merge Sort $\Rightarrow O(n \log n)$
- (7) LCS $\Rightarrow O(n^2)$
- (8) All pair shortest Path $\Rightarrow O(n^3)$
- (9) Matrix chain multiplication $\Rightarrow O(n^3)$
- (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 ques.
        {
            find ith smallest element
        }
    } → O(n)
  
```

called P class and another class called

NP class

Set of NP-Clas

$x \rightarrow x$

(1) Travelling Sales Person.

(2) Hamiltonian

(3) 0/1 Knapsack

(4) Sum of Subsets

(5) Linear Search

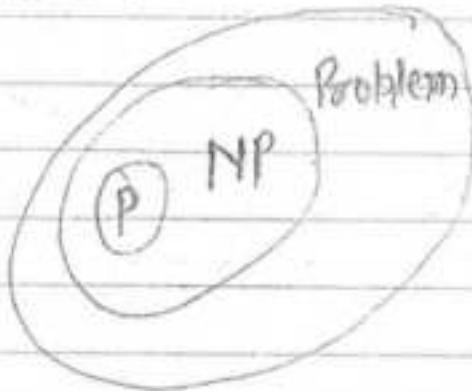
(6) Quick Sort

(7) Merge Sort

(8) Euler Graph.

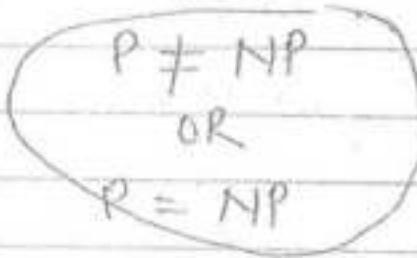
every P class problem is NP class problem.

$$P \subseteq NP$$

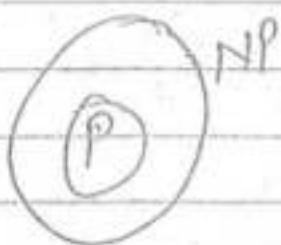


$$P \subseteq NP$$

OR



P & NP



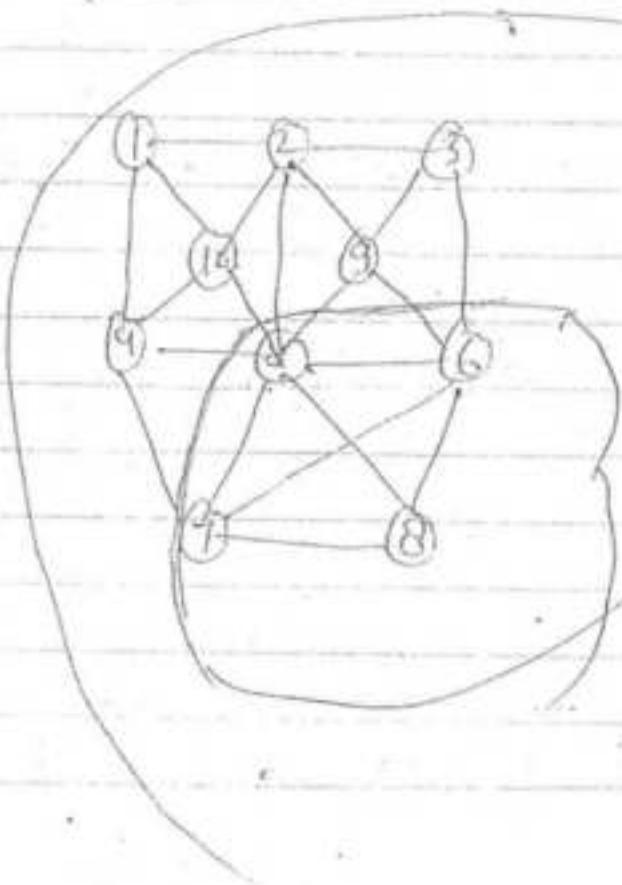
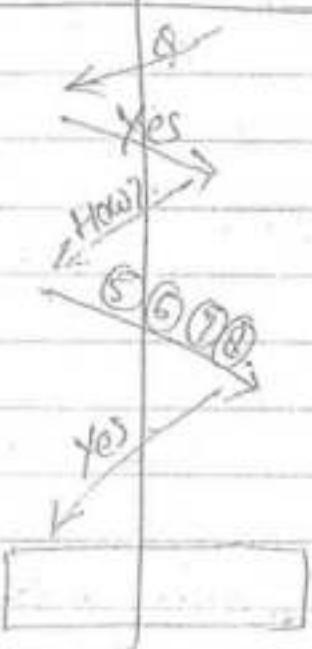
$\gamma - (1\text{-million dollar})$
↓
PCNP

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.



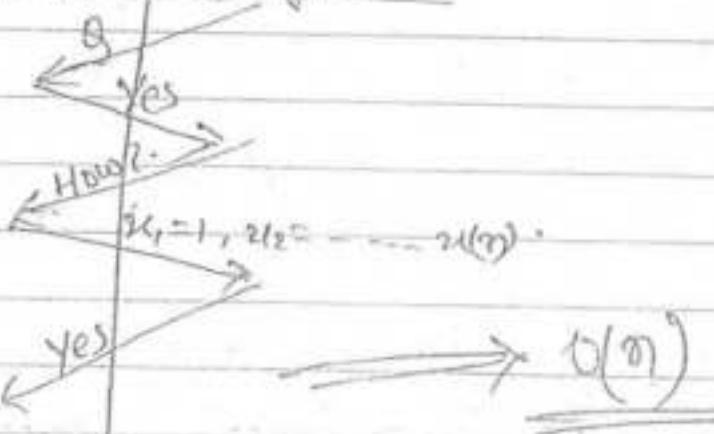
i/p : A boolean expression (CNF) and each term of CNF contain 3-variables.

$$\text{eg. } (u_1 \vee u_2 \vee u_3) \wedge (u_1 \vee u_2 \vee u_3) \wedge (u_1' \vee u_2' \vee u_3') \\ \wedge (u_1 \vee u_{100} \vee u_{50}) \wedge (u_{99}' \vee u_1' \vee u_{100}')$$

Q : Does the given boolean expression contain truth value for all variables u_1, \dots, u_n such that given expression becomes true ?.

~~Idn~~

Prover | Verifier



3-SAT is NP complete as 2-SAT is NP complete

~~AES~~Polynomial time reduction

$$A : bu + c$$

$$B : au^2 + bu + c$$

$$\downarrow \begin{array}{l} \text{Polynomial} \\ \text{exponential} \end{array}$$

$$A : bu^2 + bu + c$$



- (1) Graph
(2) Graph

16
A - NPC

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.

$$L' \leq_p L, \forall L' \in \text{NP} \quad \text{then}$$

L is NPH



(My Problem L is at least
Hardest as NP-Problem)

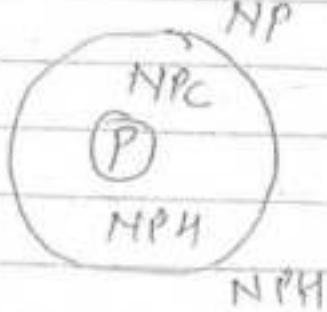


NP-Complete

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 \xrightarrow{NPC}
 A is NP-H

② If L is P-class Problem and.
 A is polynomially reduced to L ie $A \leq_p L$
 then $\xleftarrow{}$
 A is also p-class Problem

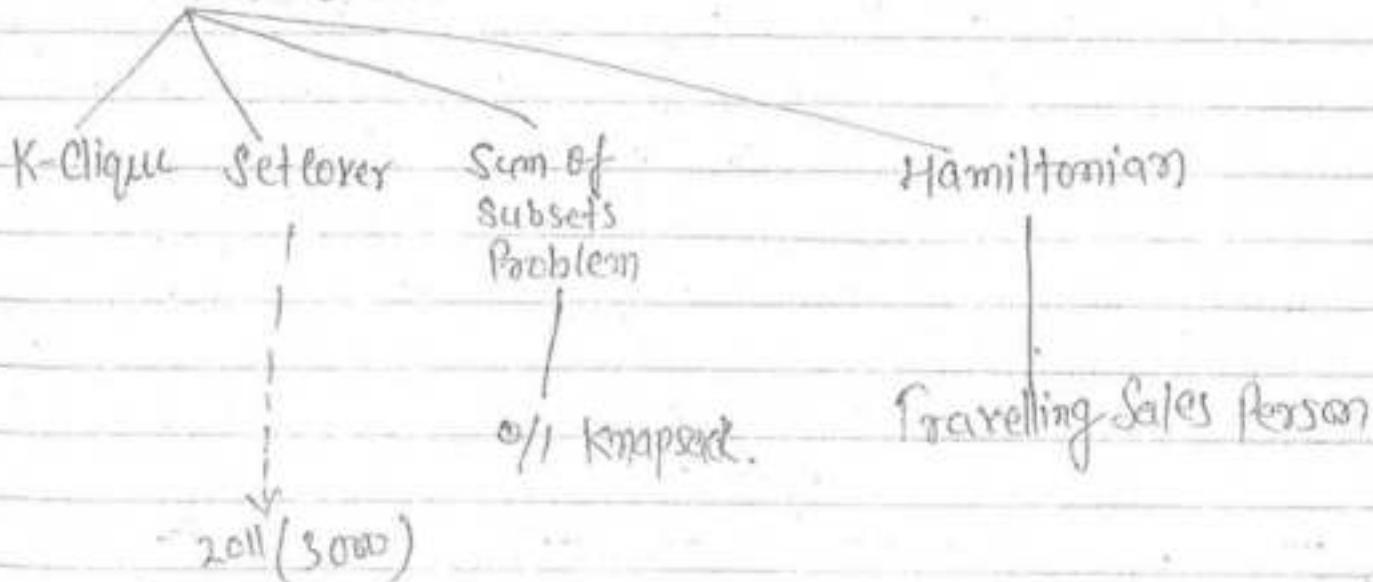
the of $L \in NP$

$A \leq_p L$ then A is also NP class.

(i) 3-SAT \Rightarrow NP complete.



Vertex cover



⇒ If any NP Complete Problem is polynomial time solvable then $P = NP$

⇒ 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) $\forall L \in NP, L \leq_p A$ (A -NP)
- c) If $C \in NP$ and $A \leq_p^{NPC} C$ then
c is NPC (Note 1)
- X d) If $B \in NP$ and $B \leq_p A$ then
B is NPC (Note 2)



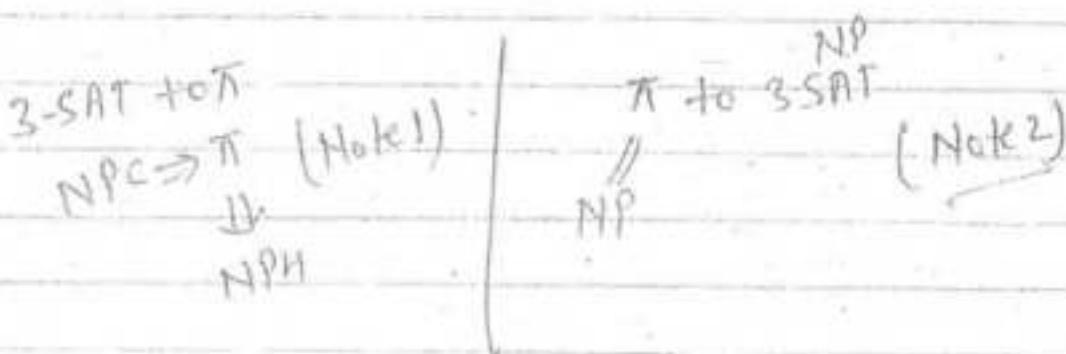
Ram and Shyam want to show that a problem π is NP-Complete.

Ram shows a polynomial time reduction from π to 3-SAT.

Shyam shows in ||| 3-SAT to π

Then which one of the following is true.

- (a) π - is NPC
- (b) π - is NPH
- (c) π - is NP
- (d) π - is P.

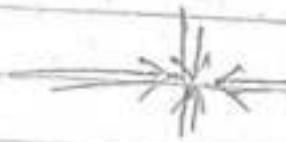
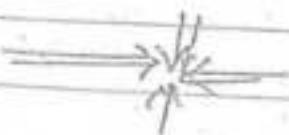


Let S be a NPC and Q and R be 2 other probles not known to be in NP.

If $S \leq_p^{NP} R$ & $Q \leq_p^{\text{NP}} S$ then which of the following is true? NP

- (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	n^2	$n \log n$	$n \log n$
M-S	$n \log n$	$n \log n$	$n \log n$
H-S	$n \log n$	$n \log n$	$n \log n$
Selection	n^2	n^2	$n \log n$
Inversion	n^2	n	n^2
Bubble Sort	n^2	n^2	n^2

Sorting

Comparison Based
 Quick sort, Merge sort
 Heap sort
 $\log n$
 Selection sort
 n^2
 Insertion sort
 Bubble sort

Non-Comparison Based
 Counting sort
 Radix sort
 Bucket 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 ($j=n$; $j \geq l$; $j--$) $\rightarrow O(n)$

{
 $b[c[a[j]]] = a[j]$;

$c[a[j]] = c[a[j]] - 1$;

}

$O(n+K)$

W

O(1)

A :

2	5	3	0	2	3	0	3
1	2	3	4	5	6	7	8

 $(n=8)$

①	C:	<table border="1"> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	0	0
0	0	0	0	0	0			
		0 1 2 3 4 5						

②	C:	<table border="1"> <tr> <td>2</td><td>0</td><td>2</td><td>X³</td><td>0</td><td>1</td></tr> </table>	2	0	2	X ³	0	1
2	0	2	X ³	0	1			
		0 1 2 3 4 5						

C:	<table border="1"> <tr> <td>2</td><td>0</td><td>2</td><td>3</td><td>0</td><td>1</td></tr> </table>	2	0	2	3	0	1	// how many times it's present in the given array.
2	0	2	3	0	1			
	0 1 2 3 4 5							

③	C:	<table border="1"> <tr> <td>2</td><td>0</td><td>4</td><td>7</td><td>7</td><td>8</td></tr> </table>	2	0	4	7	7	8
2	0	4	7	7	8			
		0 1 2 3 4 5						

<table border="1"> <tr> <td>2</td><td>2</td><td>4</td><td>7</td><td>7</td><td>8</td></tr> </table>	2	2	4	7	7	8	// how many elements are present such that all the elements are $\leq i$
2	2	4	7	7	8		
	0 1 2 3 4 5						

C:	<table border="1"> <tr> <td>2</td><td>2</td><td>4</td><td>7</td><td>7</td><td>8</td></tr> </table>	2	2	4	7	7	8	\Rightarrow	C:	<table border="1"> <tr> <td>1</td><td>2</td><td>2</td><td>4</td><td>7</td><td>7</td></tr> </table>	1	2	2	4	7	7
2	2	4	7	7	8											
1	2	2	4	7	7											
	0 1 2 3 4 5			0 1 2 3 4 5												

B ₅	<table border="1"> <tr> <td>0</td><td>0</td><td>2</td><td>2</td><td>3</td><td>3</td><td>3</td><td>5</td></tr> </table>	0	0	2	2	3	3	3	5
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:

4	0	1	5	0	1	2	3
1	2	3	4	5	6	7	8

(1) c: ^{0 1 2 3 4 5}

c	c	c	c	c	c
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)

e:	0	2		4	5	6	7	8
	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	2	3	4	5	6	7	8	9
	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 (10^n)

Pass 2 : 007 911 420 825 137 143 451 569 375 786
 2nd LSB (10^n)

Pass 3 : 007 137 143 375 420 451 569 786 825 911
 3rd LSB
 $\Rightarrow O(dn)$ $d_n = \text{no of digit}$ (10^n)
 $\Rightarrow O(n)$

$10^n \times 3$

\downarrow

30^n

\downarrow

$O(n)$

~~329 457 657 720 355~~

Pass 1 : ~~329 457 657 720 355~~ 436 457 657 329 839

Pass 2 : ~~720 355 436 839 355~~ 457 657 \rightarrow 720 329 839

Pass 3 : ~~329 355 436 457 657~~ 720 839 \rightarrow 720 329 436 457 657

Radix sort not in place
but stable \rightarrow $O(n)$

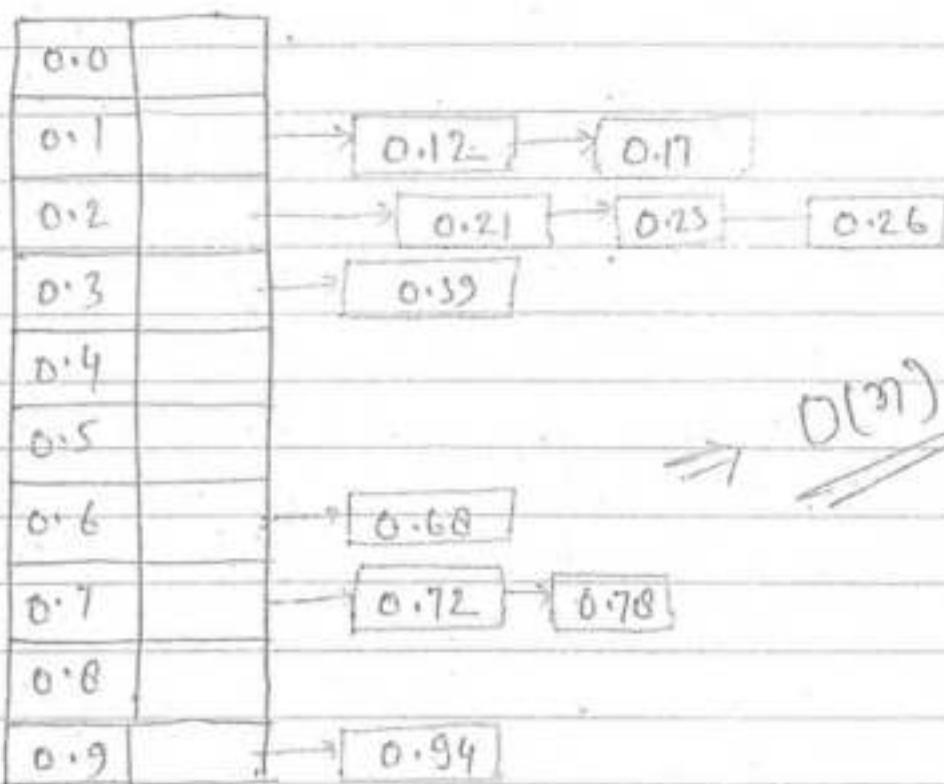
Bucket Sort (Assume uniform distribution)

Assume that all the elements are in the range $[0, 1]$; [including 0 Excluding 1]

i/p $\frac{70}{100}, \frac{17}{100}, \frac{39}{100}, \frac{26}{100}, \frac{72}{100}, \frac{94}{100}, \frac{21}{100}, \frac{12}{100}, \frac{23}{100}, \frac{68}{100}$

$\rightarrow 0.70, 0.17, 0.39, 0.26, 0.72, 0.94, 0.21, 0.12, 0.23, 0.68$

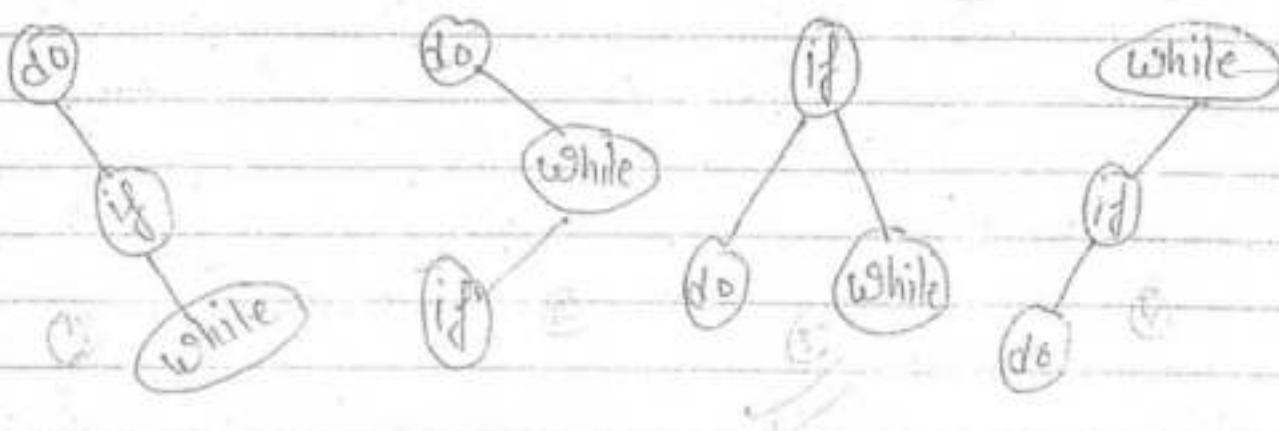
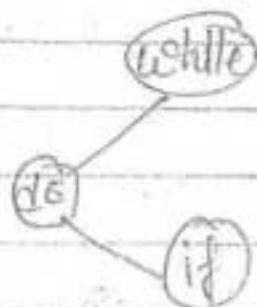
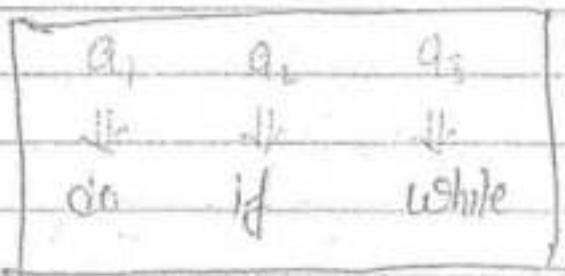
$M = 10$



$12 \boxed{17} 21 23 26 39 68 \boxed{72} 70 \boxed{94}$

Bucket sort also not in place

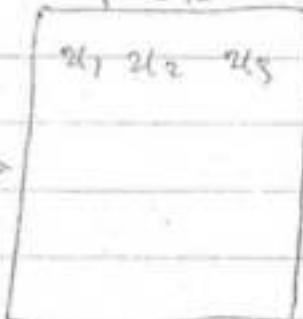
Optimal Cost Binary Search tree (OBST)



English



French



100

(1)

$$do = 20$$

$$20 \times 1 = 20$$

$$if = 50$$

$$50 \times 2 = 100$$

$$while = 30$$

$$30 \times 3 = 90$$

$$\underline{210}$$

$$(2) \quad 20 \times 1 = 20$$

$$50 \times 3 = 150$$

$$30 \times 2 = 60$$

250

$$(3) \quad 20 \times 2 = 40$$

$$50 \times 1 = 50$$

$$30 \times 2 = 60$$

150

$$(4) \quad 20 \times 3 = 60$$

$$50 \times 2 = 100$$

$$30 \times 1 = 30$$

190

$$(5) \quad 20 \times 2 = 40$$

$$50 \times 3 = 150$$

$$30 \times 1 = 30$$

220

For the given 3 ~~first~~ keywords do, if, while
with the frequency 20, 50, 30 respectively.
5 difference BST. In these 5 BST optimal
one is (3)

* Cost of a BST purely Based on.

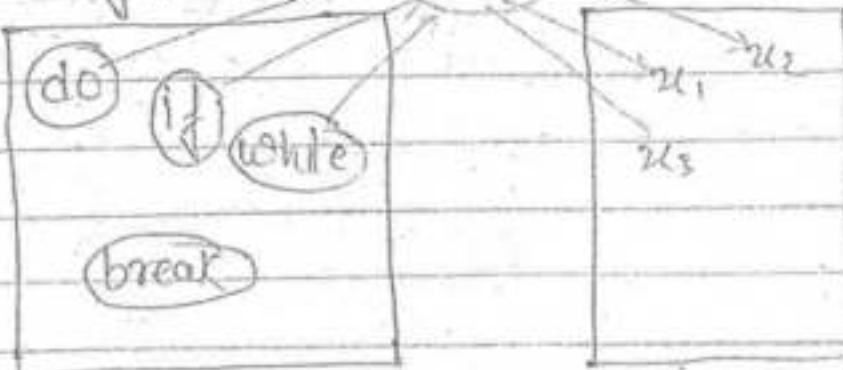
(i) no of leafels.

(ii) highest frequency Key words should be there
near to root.

English

BST

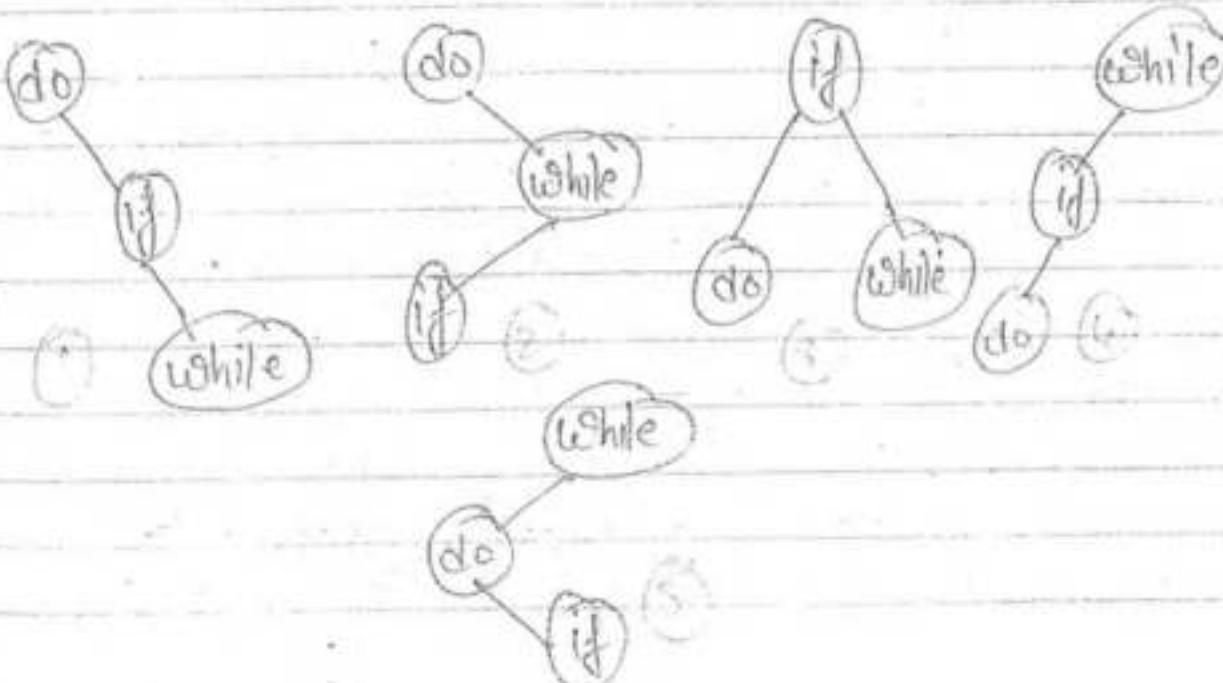
French



Cost of any BST going will be going to cost required are success full mode \leftarrow cost required for all unsuccessful mode.

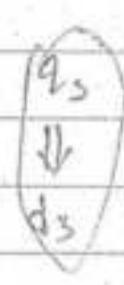
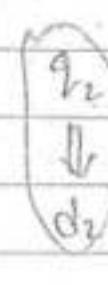
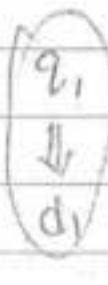
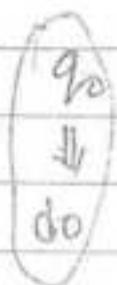
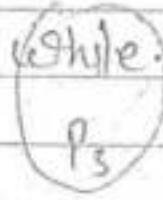
$$\text{Cost(BST)} = \text{Cost(Successfull mode)} + \text{Cost(Unsuccessful mode)}$$

~~* do , if , while~~



3

$$n=3$$



ans

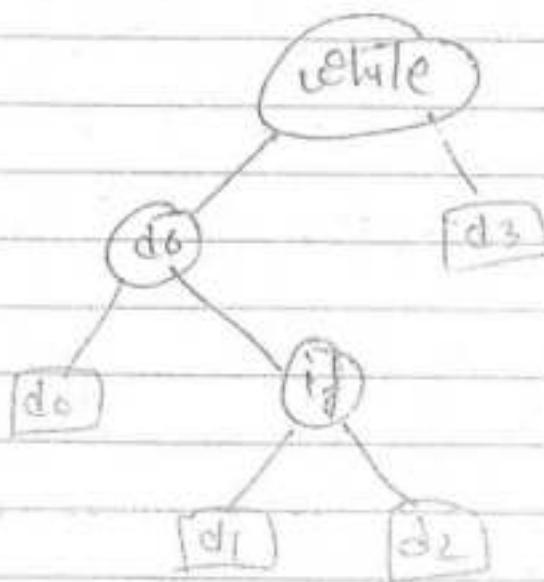
or

$$\text{Cost (BSR)} = \text{Cost}(\text{successful node}) + \text{Cost}(\text{unsuccessful node})$$

$$= \sum_{i=1}^n p_i \cdot l_i + \sum_{i=0}^m q_i \cdot l_i$$

$p_i \Rightarrow$ Probability of node.

$l_i \Rightarrow$ level of node.



$$\text{Cost (SSN)} = 2.0 \times 2 + 5.0 \times 3 + 3.0 \times 1$$

+

$$\text{Cost (USS)} = q_0 \times 3 + q_1 \times 4 + q_2 \times 4 + q_3 \times 2$$

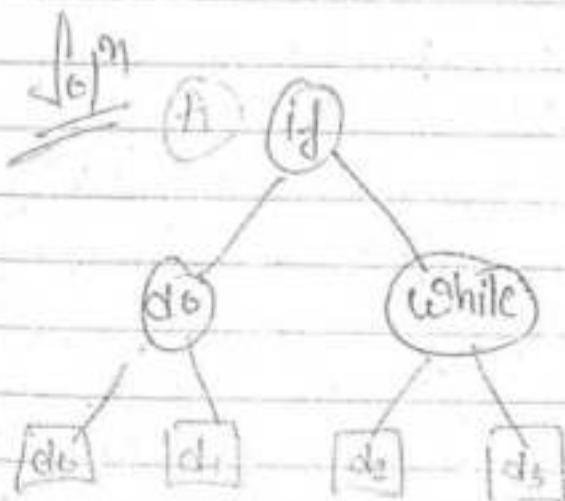
Σ

$$n = 3$$

$$a_1, a_2, a_3 = (\text{do}, \text{if}, \text{while})$$

$$(p_1 - p_3) \quad 0.5 \quad 0.1 \quad 0.05$$

$$(q_0 - q_5) \quad 0.15 \quad 0.1 \quad 0.05 \quad 0.05$$

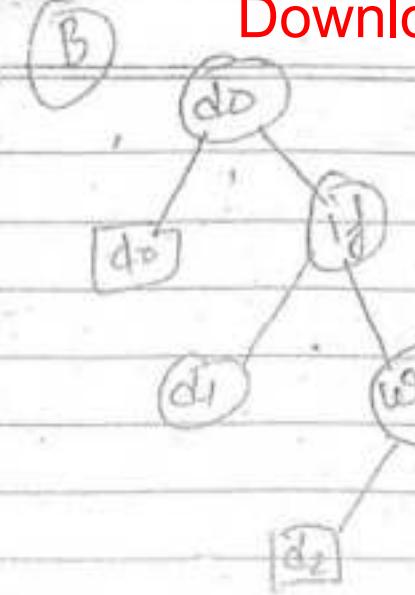


$$\begin{aligned} \text{Cost (SSN)} &= 0.1 \times 1 + 0.5 \times 2 + 0.05 \times \\ &= 1 + 1.0 + 0.1 = 2.10 \end{aligned}$$

$$\begin{aligned} \text{Cost (USS)} &= 0.15 \times 3 + 0.1 \times 3 + 0.05 \times 3 \\ &+ 0.05 \times 3 \\ &= .45 + .3 + .15 + .15 \\ &= 1.05 \end{aligned}$$

$$\text{Cost (BSI)} = 2.10 + 1.05$$

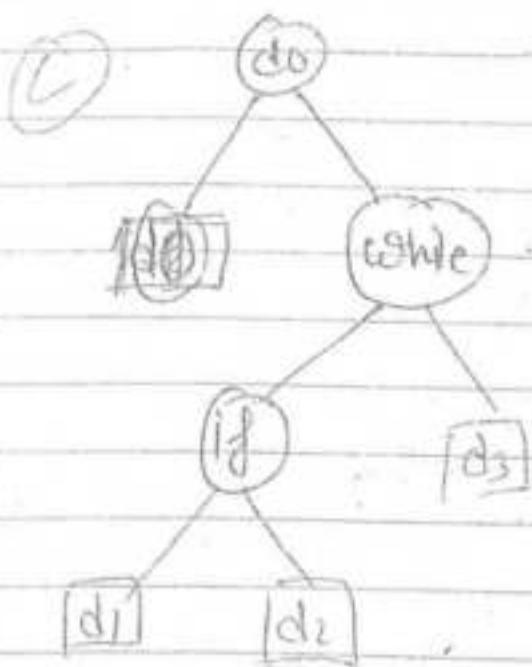
$$= \boxed{3.25}$$



$$\begin{aligned}
 \text{Cost(SSN)} &= 0.5 \times 1 + 0.1 \times 2 + 0.05 \times 3 \\
 &= 1.5 + 1.2 + 1.5 \\
 &= 4.20
 \end{aligned}$$

$$\begin{aligned}
 \text{Cost(USN)} &= 0.15 \times 2 + 0.1 \times 3 + 0.05 \times 4 \\
 &\quad + 0.05 \times 4 \\
 &= 1.30 + 1.3 + 1.20 + 1.20 \\
 &= 1.00
 \end{aligned}$$

~~Cost(BST) = 1.85~~

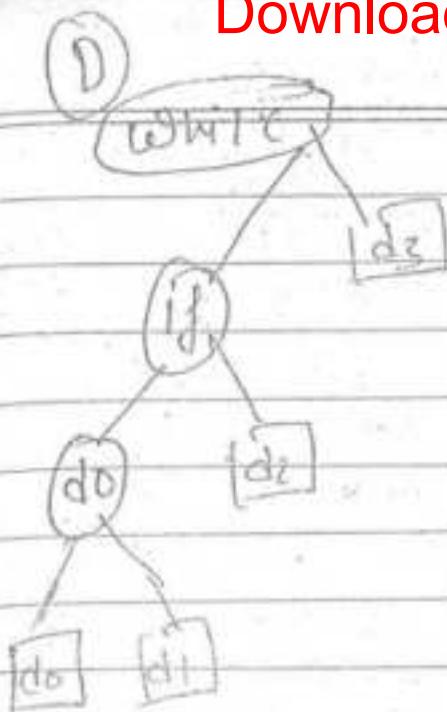


$$\begin{aligned}
 \text{Cost(SSN)} &= 0.5 \times 1 + 0.05 \times 2 + 1 \times 3 \\
 &= 1.5 + 1.0 + 1.3 \\
 &= 3.8
 \end{aligned}$$

$$\begin{aligned}
 \text{Cost(USN)} &= 0.15 \times 2 + 0.1 \times 4 + 0.05 \times 4 \\
 &\quad + 0.05 \times 3 \\
 &= 1.30 + 1.4 + 1.20 + 1.15 \\
 &= 1.05
 \end{aligned}$$

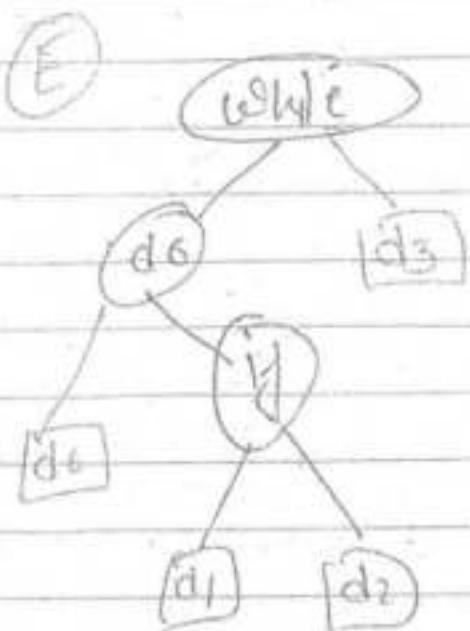
~~Cost(BST) = 1.9 + 1.05~~

~~= 1.95~~



Cost(BST) =

= 3.00



Cost(BST) =

= 2.50

In the above 5 BST ~~tree B is~~ tree B is optimal cost binary tree becoz higher probability made it self a 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.

\rightarrow

\leftarrow

8)

$$W(i, j) = \sum_{K=i}^j p_k + \sum_{K=i+1}^j q_k$$

do if while	
$p_1 - p_3 = 0.5$	$0.1 \quad 0.05$
$q_1 - q_3 = 0.15$	$0.1 \quad 0.05 \quad 0.05$
q_0	$q_1 \quad q_2 \quad q_3$

$e[i, j]$ = The min cost required for the optimal cost BST which contains i to j elements.

$$e[a_1, q_2, q_3] = \min \begin{cases} e[1, 0] + e[2, 3] + p_1 + W[1, 0] + W[2, 3] \\ e[1, 1] + e[3, 3] + p_2 + W[1, 1] + W[3, 3] \\ e[1, 2] + e[4, 5] + p_3 + W[1, 2] + W[4, 5] \end{cases}$$

Dots

$$e[i, j] = \min \left\{ \begin{array}{l} \left. \begin{array}{l} I \text{ as the root} \\ e[i, i-1] + e[i+1, j] + p_i + w(i, i-1) \end{array} \right\} \\ \left. \begin{array}{l} J \text{ as the root} \\ e[i, j-1] + e[j+1, j] + p_j + w(i, j-1) \end{array} \right\} \\ + w(i+1, j) \end{array} \right\}$$

OR

$$e[i, j] = \min \left\{ \begin{array}{l} \left. \begin{array}{l} e[i, r-1] + e[r+1, j] + p_r + w(i, r-1) \\ + w(r+1, j) \end{array} \right\} \\ \left. \begin{array}{l} r \leq r \leq j \end{array} \right\} \end{array} \right\}$$

$m=1$

a_1
↓

p_1

$q_0 \ q_1$

(q)

[do] [di]

Ans : AAD

Q1

$$n = 1$$

$$d_0 = d_1$$

$$P_1 = 0.5$$

$$q_0, q_1 = 0.05, 0.15$$



$$= \underline{0.9}$$

$$e[1,1] = \left\{ \begin{array}{l} e[1,0] + e[2,1] + \underline{P_1} + w[1,0] + w[2,1] \\ \hline 0.05 + 0.15 + 0.5 + 0.05 + 0.15 \end{array} \right.$$

$$= \underline{1.9}$$

D.A.A. · 235