Computer Notes

DESIGN ANALYSIS & ALGORITHAM

PART-I

(1)

16/05/11

Asymptotic Notation

1. O - Notation.
2. Ω - Notation.
3. Θ - Notation

1) Let $f(n)$ & $g(n)$ be two +ve functions.

Big O Notation (upper bound)

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

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

there exists 2 constants $c > 0$ & $n_0 \geq 0$

such that

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

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

Q. $f(n) = n$

$g(n) = n^2$

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

$n = O(n^2)$

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

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

$f(n) = n^2$

$g(n) = n^2$

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

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

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

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

$g(n)$ is greater the c times of $f(n)$

Ω - Notation (lower bound)

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

there exist 2 constant $c > 0$ & $n_0 \geq 0$

such that

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

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

Θ - Notation (L.B = U.B)

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

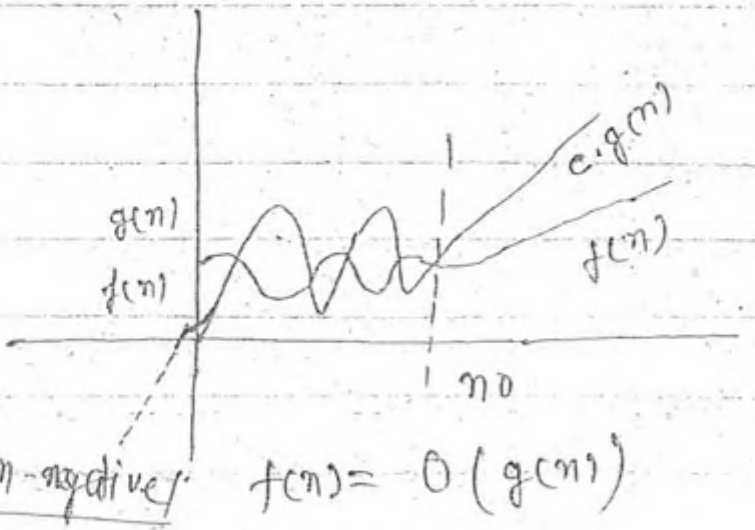
(1) $f(n) = O(g(n))$

&&

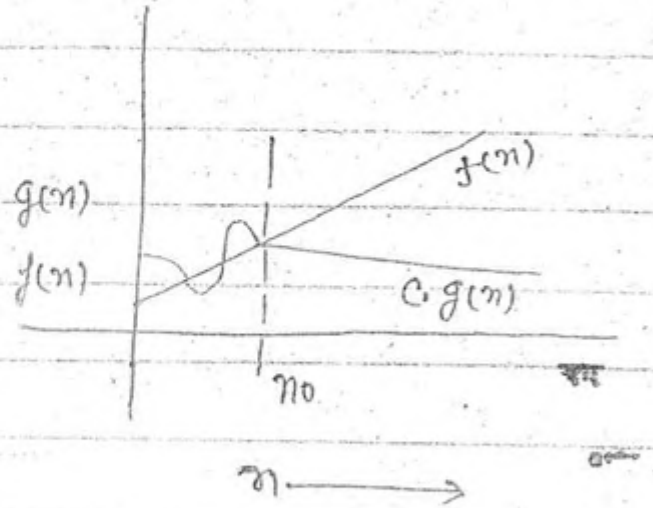
(2) $f(n) = \Omega(g(n))$

$$c_2\, g(n) < f(n) \leq c_1\, g(n) \quad \forall n, n \geq n_0$$

Big - O - Notation

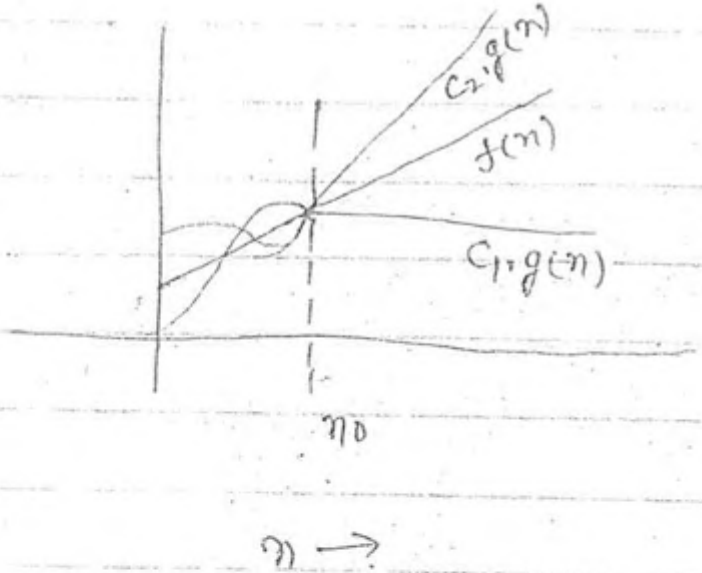


Omega - Notation



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

Theta - Notation



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

$A = n^2$

$B = n^3$

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

$A = n^2$

$B = n^2$

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

$A = n^2$

$B = n$

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

#

P

⇓

A

⇓

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

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

Best $\Rightarrow n^2$

worst $\Rightarrow n^2$ $\Rightarrow \Theta(n^2)$

Types of Time Complexity

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

Consider the following two function

$f(n) = n$.

$g(n) = n^2$

Which one of the following are true ?:

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

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

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

1) We can't compare

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

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

$\Downarrow$

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

$\Downarrow$

1

#

$f(n) = n^2$

$g(n) = 2^n$

then which one of the following is true?

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

Sol^n

n	n^2	2^n
1	1	2
2	4	4
3	9	8
4	16	16
5	25	32
6	36	64
7	49	128
8	64	256

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

$\Downarrow$

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

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

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

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

Ans

$n^2 \log n$ $\quad$ $n (\log n)^{10}$

n $\quad$ $(\log n)^9$

$\log n$ $\quad > \quad$ $9 \log(\log n)$

#

= $f(n) = 2^n$

$g(n) = n^n$

which one of the following is true ?.

a)

b)

c)

d)

Sol^n

2^n $\quad$ n^n

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

n $\quad < \quad$ $n \log n$

n	2^n	n^n
1	2	1
2	4	4
3	8	27
4	16	256
5	25	3125

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

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

$2^n \quad 1 \quad n^n$

Which one of the following is true or false?

✓ a) $100\, n \log n = O\left(\frac{n \log n}{100}\right)$

✗ b) $\sqrt{\log n} = O(\log(\log n))$

✓ c) $0 < x < y$ then

n^x is $O(n^y)$

✓ d) $2^n \neq O(n^k) \quad k > 0$.

If in any problem constant will not be consider

1st (a) $100 \cdot n \log n = O\left(\frac{n \log n}{100}\right)$

$n \log n \leq c \cdot \frac{n \log n}{10000}$

$\Downarrow$

1000000

Identify True/False for the following stmt.

a) $(n+k)^m = O(n^m)$ where k is constant.

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

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

if both for n are same dont apply log

Which one of the following stmt is False. ?

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

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

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

d) none

formulas

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

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

3) $a^{\log_b n} = n^{\log_b a}$ ✓ (n log n)

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

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

6) $\log_b a = 1 / \log_a b$

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

7. n^n

Sol)

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

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

c) n is $O(n^2)$ ✓
n is $\Omega(n^2)$ ✗

Which one of the following options provides the increasing order of asymptotic complexities of fun^n f_1, f_2, f_3, & f_4.

(4) $f_1(n) = 2^n$ 9

(2) $f_2(n) = n^{3/2}$ 9

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

(3) $f_4(n)$ $n^{\log_2 n}$ 5

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) $f_2\ f_3\ f_4\ f_1$ (5)

Sol^n

2^n $n^{\log n}$	n^n 2^n
n $(\log n)^2$	n^n 2^n

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

O, Ω, Θ

$n^2 = O(n^2)$ — Tight upper bound

$n^2 = O(n^3)$ — not T.U.B.

O

| $n^2 = O(n^2)$ | |
|---|---|
| $n^2 \neq n^2$ | TUB |
| $n^2 < n^3$ | NTUB |

O

$n^2 < n^3$

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

| | |
|---|---|
| $n^2 > n$ | NTUB |
| $n^2 = n^2$ | T.L.B. |

ω

$n^2 > n$ — NTLB

O Ω Θ o ω

$\leq$ $\geq$ $=$ $<$ $>$

$\Downarrow$ $\Downarrow$ $\Downarrow$ $\Downarrow$

U.B (T, NT) | L.B (T, NT) | U.B (NT) | L.B (NT)

$f(n) = n^1$

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

$f(n) = \sqrt{n}$

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

} not comparable.

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

$\sqrt{n} \leq n^{\sin\theta}$ $(\sin\theta = 1)$

$O(1)$

$\log(\log n)$

$\log n$

$\log n$

$\sqrt{n}$

n

n^2

n^3

n^k

2^n

Properties of Asymptotic Notation.

1) Reflexive ⇒

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

&

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

⇓

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

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

$R = \{a$

2. Symmetric

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

then

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

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

then

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

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

then:

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

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

$R = \{(1,1), (2,2), (3,3)\}$

$M = M^T$

Transitive

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

&

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

then

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

$2 < 3$ & $3 < 4$

$2 < 4$

$5 > 4 > 3$

Similarly Ω, Θ also satisfies transitivity.

Properties

1. $f(n) = O(f(n/2))$ eg $n = O(n/2)$

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

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

then.

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

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

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

7

If $f(n) = O(g(n))$ where n and n^9

&

$g(n) = O(h(n))$ where n^4 and n^5

then

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

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

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

$f(n) = o(g(n))$ && $g(n) \neq o(f(n))$

$g(n) = o(h(n))$ && $h(n) = o(g(n))$

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

2)

$a \leq b$
$b \geq a$ } $a = b$.

$f(n) = o(g(n))$ && $(g(n) \neq o(f(n))$

$\Downarrow$

$f(n) < g(n)$

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

If

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

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

P.

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

Q.1)

b)

$$T_1(n) = O(f(n))^n$$

$$T_2(n) = O(f(n))^{n^2}$$

2)

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

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

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

$$= O(n^2)$$

ii)

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

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

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

(iii) $T(n) = n!$

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

$= \underbrace{n \times n \times n \cdots}_{n}$

$= O(n^n)$

(iv) $T(n) = \sum_{i=1}^{n} x^i$ | $\frac{a(r^n-1)}{r-1}$

$= x^1 + x^2 + x^3 + \cdots x^n$

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

$= O(x^n)$

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

$T(n) = 2 + 0$

$= 2$

$= O(1)$

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

$5 + \frac{1}{n^2} = O(1)$

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

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

$n^2 + n = O(n^2)$
$\Downarrow$
$n.n + n = O(n^2)$

(vii) $n^3 + 16^{10} n^2 = O(n^3)$

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

Q Consider the following two functions. The fun^n are give like as

$$g_1(n) = n^3 \quad 0 \le n \le 10,000$$
$$= n^2 \quad n > 10,000$$

$$g_2(n) = n \quad 0 \le n \le 100$$
$$= n^3 \quad n > 100$$

In $g_1(n)$ and $g_2(n)$ what is the relation?

$\therefore g_1(n) = g_2(n)$

Sol^n

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

$$\Downarrow$$

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

In big O notation only n starting is avalable.

$g_1(n) = n^3 \quad 0 \leq n \leq 10,000$

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

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

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

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

$$n^2 \leq c \cdot n^3, \quad n \geq 10,001$$

$$\boxed{g_1(n) = O(g_2(n))}$$

Friday
17/06/11

1 Explain why running time algo A is atleast $O(n^2)$ is meaningless.

Best case n^2 [$\Omega(n^2)$]
worst n n^2 [$O(n^2)$]
⇓ [$\Theta(n^2)$]

Recursion

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

eg

fact(4) = 4 x fact(3) — Recursive steps
⇓
3x fact(2)
⇓
2 x fact(1) — Termination condition.
~~1 x fact(0)~~
⇓
1

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.

(*)

eg

Mul (2,3) = 2 + Mul(2,2) ← Recursive steps.

⇓

2 + Mul (2,1)

⇓

2 + Mul (2,0) ← Termination condition.

⇓

0

eg

Pow(2,3) = 2 * Pow(2,2)

⇓

2 * Pow(2,1)

⇓

2 * Pow(2,0)

⇓

1

Recurrence Relⁿ

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

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

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

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

Sol

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

eg: ✓ GCD(5, 50) = 5

✓ GCD(5, 7) = 1

✓ GCD(0, 5) = 5

✓ GCD(5, 0) = 5

✓ GCD(0, 0) = 1

GCD(5, 50)

⇓

5) 50 (10
50

0

GCD(5, 0) ⇒ 5

GCD(5, 7)

⇓

5) 7 (1
5

2

GCD(5, 2)

⇓

5) 2 (0
0

2

GCD(5, 7)

⇓

7) 5 (0
5

5

GCD(7, 5)

⇓

5) 7 (1
5

2

GCD(5, 2)

⇓

2) 5 (2
4

1

GCD(2, 1)

⇓

1) 2 (2
2

0

GCD(1, 0) = 1

GCD (5, 100)

⇓

100) 5 (0
0
5

GCD (100, 5)

⇓

5) 100 (20
100
(0)

GCD (5, 0)

⇓

5

GCD (100, 200)

⇓

200) 100 (0
0
100

GCD (200, 100)

⇓

100) 200 (2
200
(0)

GCD (100, 0)

⇓

100

GCD (a, b)

⇓

b) a (c
0
0
a

#

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} fibinocci number. ?

n	0	1	2	3	4	5	6	7	8	9	10
fib(n)	0	1	1	2	3	5	8	13	21	34	55

Recurrence Rel^n

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

```
fib ( 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 .?

```
GCD (int m, int n)
{
  int f,
  if (m==0 && n==0)
      return (1)
  if ( m==0 )
     return ( n )
  if ( n==0 )
     retur (m)
  else
  {
     f = GCD( n , m%n)
        return( f );
  }
}
```

```
Main ()
{
int x, n=3
x = fact (n);
Printf ("%d", x);
}
```

x: 1000 | n: 3 (2000)

3

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

n: 3 (3000)

fact (n)

(1) f [ ]
(2)
(3) f = 3 * fact (n) n: 2 (4000)
(4) return (f)

(1) f [ ]
(2)
(3) f = 2 * fact (n) n: 1 (5000)
(4) return (f)

(1) f [1]
(2) n==1 ⇒ return (1)
(3)
(4)

More space is Drawback of the Recursive Progra

Simple C Program of factorial n.

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

Solving Recurrence Relation

1. Substitution Method.
2. Recursive Tree Method.
3. Master Method.

1 Substitution Method →

Substitution: the given fun again and again untill the given fun is remove.

eg

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

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

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

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

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

$n-1$

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

$+ n-1 + n$

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

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

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

$= O(n^2)$

Solve the following recurrsion Relation

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

$= 1$ if $n = 1$

Soln

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

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

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

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

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

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

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

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

$n-1$

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

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

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

$= n!$

$= O(n^n)$

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

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

Sol^n

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

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

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

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

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

$\vdots$ K times

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

$= T(1) + Kc$

$= 1 + c * \log_2 n$

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

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

Soln

$$T(n) = 2T(n/2) + C$$
$$= 2[2T(n/2^2) + C] + C$$
$$= 2^2 T(n/2^2) + 2^1 C + 2^0 C$$
$$= \Downarrow$$
$$2^2[2T(\tfrac{n}{2^3}) + C] + 2^1 C + 2^0 C$$
$$= 2^3 T(n/2^3) + 2^2 C + 2^1 C + 2^0 C$$
$$\downarrow K$$
$$T(n) = 2^K T(n/2^K) + C[2^0 + 2^1 + 2^2 + \cdots 2^{K-1}]$$
$$= n * T(1) + C[2^0 + 2^1 + 2^2 + \cdots 2^{K-1}]$$
$$= n + C\left[\frac{1(2^K - 1)}{(2-1)}\right] = n + C[n-1]$$
$$= n(1+C) - C$$
$$= O(n)$$

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

$n = 2^K$

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

Sol^n

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

$$\downarrow K$$

$$T(n) = 2^K T(n/2^K) + Kn$$
$$= nT(1) + n * \log_2 n.$$
$$= n + n\log_2 n$$
$$= O(n \log n)$$

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

Soln

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

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

$$T(n) = 8\,T(n/2) + n^2 \quad \text{if } n > 2$$
$$= c \quad \text{if } n \leq 2$$

Soln

$$T(n) = 8\,T(n/2) + n^2$$

$$= 8\left[T(n/2^2) + (n/2)^2\right] + n^2$$

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

$$=$$

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

$T(n) = 8T(n/2) + n^2$ if $n > 2$

$= c$ if $n \leq 2$

$n = 2^K$

$K = \log_2 n$

$T(n) = 8T(n/2) + n^2$

$= 8[8T(n/2^2) + (n/2)^2] + n^2$

$= 8^2 T(n/2^2) + 2^1 n^2 + 2^0 n^2$

$= 8^2[8T(n/2^3) + (n/2^2)^2] + 2n^2 + 2^0 n^2$

$= 8^3 T[n/2^3) + 2^2 n^2 + 2^1 n^2 + 2^0 n^2$

$\vdots$

$K-1$

$\downarrow$

$T(n) = 8^{K-1} T(n/2^{K-1}) + n^2[2^0 + 2^1 + 2^2 + \cdots 2^{K-2}]$

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

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

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

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

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

$T(n) = 7T(n/2) + n^2$ if $n > 2$

$= c$ if $n \leq 2$

Soln

$T(n) = 7T(n/2) + n^2$

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

$= 7T\left(\frac{n}{2^2}\right) + 7(n/4)^2 + (n^2)^1$

$= 7T\left(\frac{n}{2^2}\right) + \left(\frac{7}{4}\right)^1 n^2 + \left(\frac{n}{4}\right)^0 n^2$

$= 7T\left(\frac{n}{n^3}\right) + \left(\frac{7}{4}\right)^2 n^2 + \left(\frac{7}{4}\right)^1 n^2 + \left(\frac{7}{4}\right)^0 n^2$

$\downarrow$ K-1

$T(n) = 7^{K-1} T\left(\frac{n}{2^{K-1}}\right) + n^2\left[\left(\frac{n}{4}\right)^0 + \left(\frac{7}{4}\right)^1 + \cdots \left(\frac{7}{4}\right)^{K-2}\right]$

$= \frac{7^K}{7} T(2) + n^2\left[\frac{1\left(\left(\frac{7}{4}\right)^{K-1} - 1\right)}{\frac{7}{4} - 1}\right]$

$= n^{\log_2 7} + n^2\left[\left(\frac{7}{4}\right)^K - 1\right]$

$= n^{\log_2 7} + n^2 \times \left(\frac{7}{4}\right)^K - n^2$

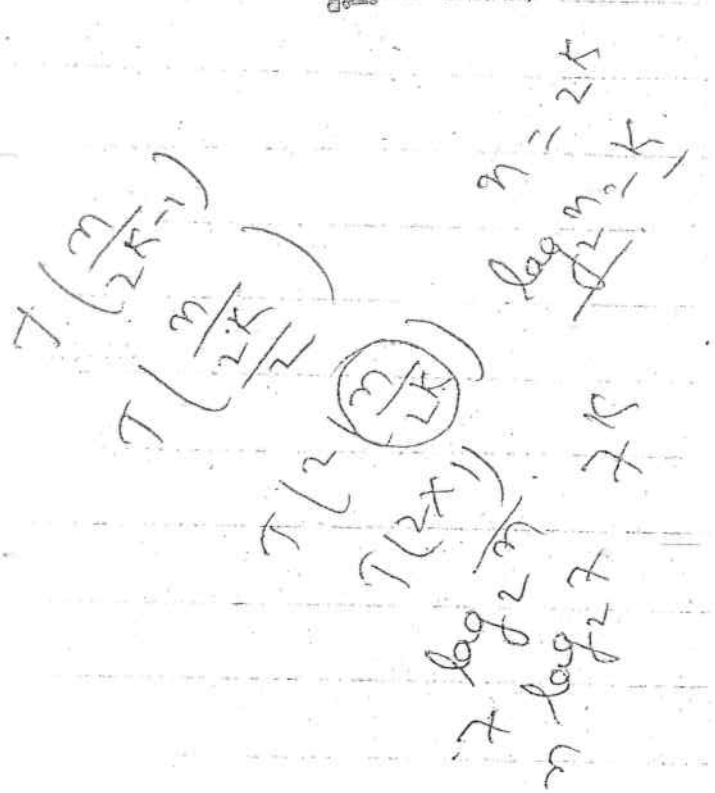
$$= n^{\log_2 7} + n^2 * \frac{n^{\log 7}}{n^2} - n^2$$

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

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

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

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



Page 11

*

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

Sol^n

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

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

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

$$\Downarrow$$

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

$$\downarrow \quad n-1 \text{ times}.$$

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

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

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

$$1 + \log n!$$

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

$$= 1 + n \log n.$$

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

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

$= 1$ if $n = 1$

Soln

$T(n) = T(n-1) + \frac{1}{n}$ ⟶ (1)

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

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

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

↓ n-1 time.

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

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

$\log n$

* $T(n) = 2T(n/2) + n\log n$ ... if $n \geq 1$

$= 1$ ... if $n = 1$

$n = 2^k$

$\log_2 n = k$

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

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

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

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

k time

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

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

$= n + n\left[(\log_2 n - 0) + (\log_2 n - 1) + (\log\frac{n}{2} - 2) \cdots \right.$

$\left. \cdots + (\log_2 n - (k-1))\right]$

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

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

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

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

$$T(n) = 5T(n/5) + \frac{n}{\log_5 n} \quad \text{if } n > 1$$

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

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

Sol:

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

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

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

$$= 5^3\left\{T\left(\frac{n}{5^3}\right) + n\log_5 \frac{n}{5^2} + \frac{n}{\log_5 \frac{n}{5}} + \frac{n}{\log_5 n}\right.$$

↓ K times.

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

$$= nT(1) + n\left[\frac{1}{\frac{(\log_5 n - 0)}{K}} + \frac{1}{\frac{(\log_5 (n-1))}{K}} + \frac{1}{\log_5 (n-2)} + \cdots\right.$$

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

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

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

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

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

$T(n) = T(n-2) + 2\log n$ if $n > 0$

$= 1$ if $n = 0$

$n - 2K = 0$

$n = 2K$

$K = n/2$

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

Sol^n

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

$= T(n-6) + 2^3 \log(n-4) + 2\log(n-2) + 2\log(n-0)$

↓ K times.

$T(n) = T(n-2\times K) + 2\log(n(2K-2)) + 2\log(n-(2K-4)) + \cdots + 2\log(n-0)$

$= T(0) + 2\log 2 + 2\log 4 + 2\log 6 + \cdots + 2\log 2K$

$= 1 + 2[\log 2 + \log 4 + \log 6 + \cdots \log 2K]$

$= [\log(2\times 1) + \log(2\times 2) + \log(2\times 3) + \cdots \log(2\times K)$

$= (\log_2^2 + \log_2^1) + (\log_2^2 + \log_2^2) + (\log_2^2 + \log_3^3) + \cdots (\log_2^2 + \log\frac{K}{2})$

$= K \log_2^2 + [\log_2 1 + \log 2 + \log 3 \longrightarrow \log K]$

$= K + \log K!$

$= K + K \log K$

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

$= O(\log n/2)$

11)

Q $T(n) = T(n-2) + n^2$ if $n > 0$

$= 1$ if $n = 0$

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

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

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

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

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

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

$$= T(n-2*K) + (n-(2K-2))^2 + (n-(2K-4))^2 + \cdots$$

$$= T(0) + 2^2 + 4^2 + 6^2 + \cdots + (n-0)^2 \quad (2K)^2$$

$$= 1 + [4 + 16 + 36 + 64 + \cdots (2K)^2]$$

$$= 1 + 2^2[1^2 + 2^2 + 3^2 + 4^2 + \cdots K^2]$$

$$= 1 + 2^2\left[\frac{K(K+1)(2K+1)}{6}\right]$$

$$= \frac{K^3}{6} = K^3$$

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

$$= O(n^3)$$

$n = 2K$

$K = n/2$

*

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

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

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

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

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

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

$=$

$\vdots$

$n-1$ times.

$\vdots$

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

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

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

$$S = 2^0(n-0) + 2^1(n-1) + 2^2(n-2) + \cdots$$

$$\cdots 2^{n-3}(3) + 2^{n-2}(2)$$

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

$$= \cdots$$

$$+ 2^{n-3}(3) + 2^{n-2}(2)$$

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

$$\cdots - 2^n$$

$$-S = -n\left[2^1 + 2^2 + 2^3 + \cdots 2^{n-2}\right] + 2^n$$

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

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

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

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

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

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

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

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

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

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

$$S = \frac{1}{2^1} \cdot 1 + \frac{1}{2^2} \cdot 2 + \frac{1}{2^3} \cdot 3 + \cdots \frac{1}{2^K} \cdot K \quad \text{(S)}$$

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

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

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

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

$=$

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

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

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

$$= O(n)$$

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

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

Sol: $T(n) = n^{1/2}\, T(n^{1/2}) + n.$

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

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

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

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

$$\downarrow K$$

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

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

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

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

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

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

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

$$\frac{1}{2^K} \log_2 n = 1$$

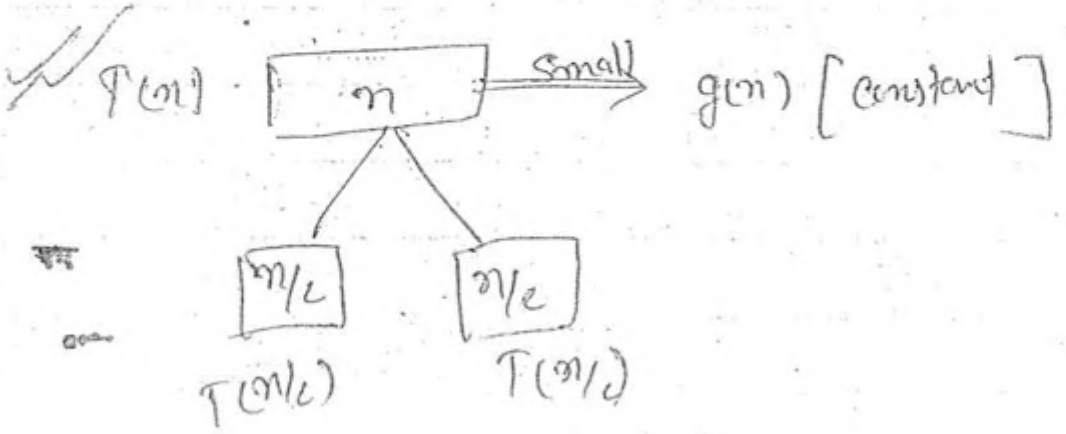
$$\log n = 2^K$$

$$\log(\log n) = K + \log_2 2$$

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

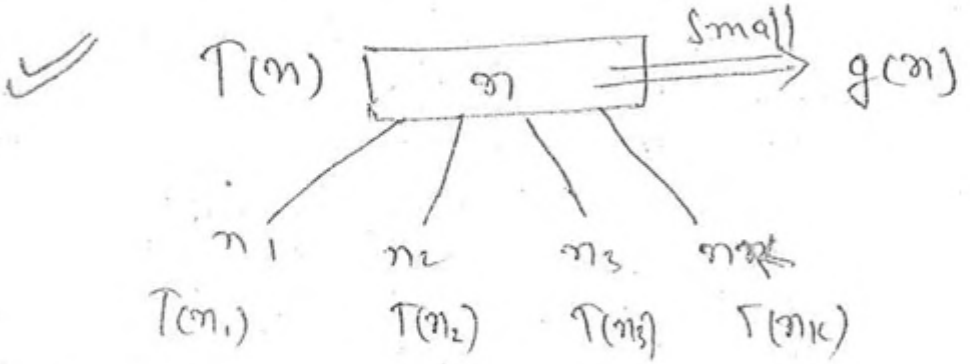
Divide and Conquer (DAC)

(1) Divide.
(ii) Conquer.
(iii) Combind



P $$T(n) = \begin{cases} g(n) & \text{if } n \text{ is small} \\ T(n/2) + T(n/2) + f(n) & \\ 2T(n/2) + \underbrace{f(n)}_{D\&C} & \text{if } n \text{ is big} \end{cases}$$

$f(n)$ is time required Divide / Combined.



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

Divide ⟹ Divide the given problem into small sub problem.

Conquer ⟹ Conquer the subproblem to get solution recursively.
If the subproblem is small then return solution recursively.

Combined ⟹ Combind the subproblem solution to get original problem soln.

Control abstraction of DAC

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

else

{

m = Divide (p, q)

Return Comb ind (DAC (P, m) , DAC (m+1, q))

}

}

Applications of DAC

1. Power of an element.
2. Finding max^m & 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 > 1 && integer n > 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);
}
```

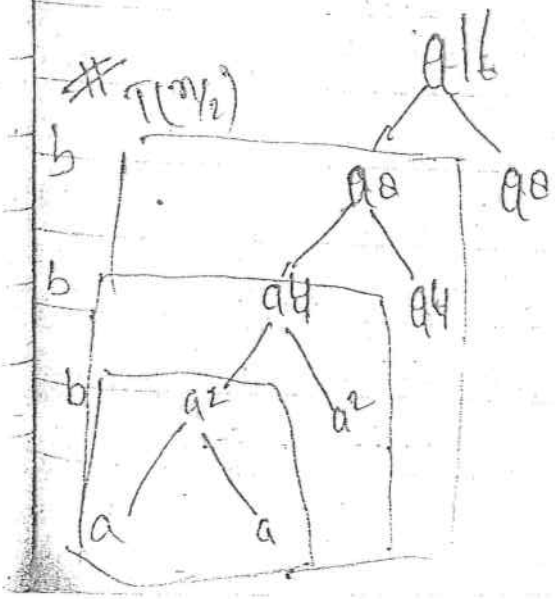
Time complexity = $\Theta(n)$

#

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

$$\Downarrow$$

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



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)
    }
}
```

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

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

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

$\Rightarrow O(\log n)$

$KC \Downarrow C \cdot \log n \Downarrow O(\log n)$

Q. Finding maximum and minimum in the given array of n element

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]
}
```

n-1

Comparisons

Best Case : 1(n-1)

Worst case : 2(n-1)

Time Complexity
↓
O(n)

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

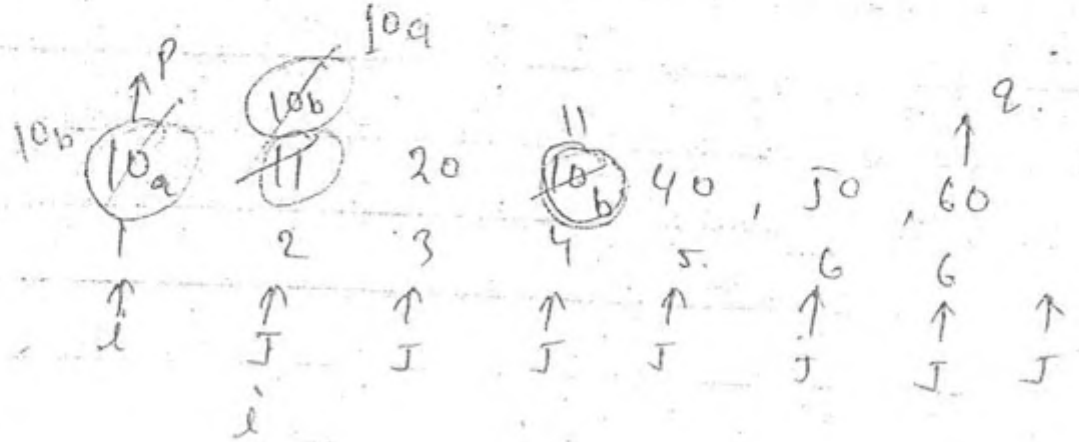
$= \frac{n-1}{2} + n - 1 \Rightarrow \frac{3}{2}(n-1)$

$= 1.5\,(n-1)$

1/ Apply partion algo on following Algo.

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

Sol:



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

Stable sorting Tech

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

Stable sorting Technique.

⇓

Q.S is not stable sorting.

Selection procedure

i/p : An array of n elements and K

o/p : Find K^{th} smallest element

eg

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

o/p = 10 K=4 | 11 K=5 | 31 K=9

A. (1) 1, 2, 3, (10), 11, 20, 21, 30, 31 ⇒ O(nlogn)
1 2 3 4 5 6 7 8 9

(2) Go to location a[K] ⇒ O(1)

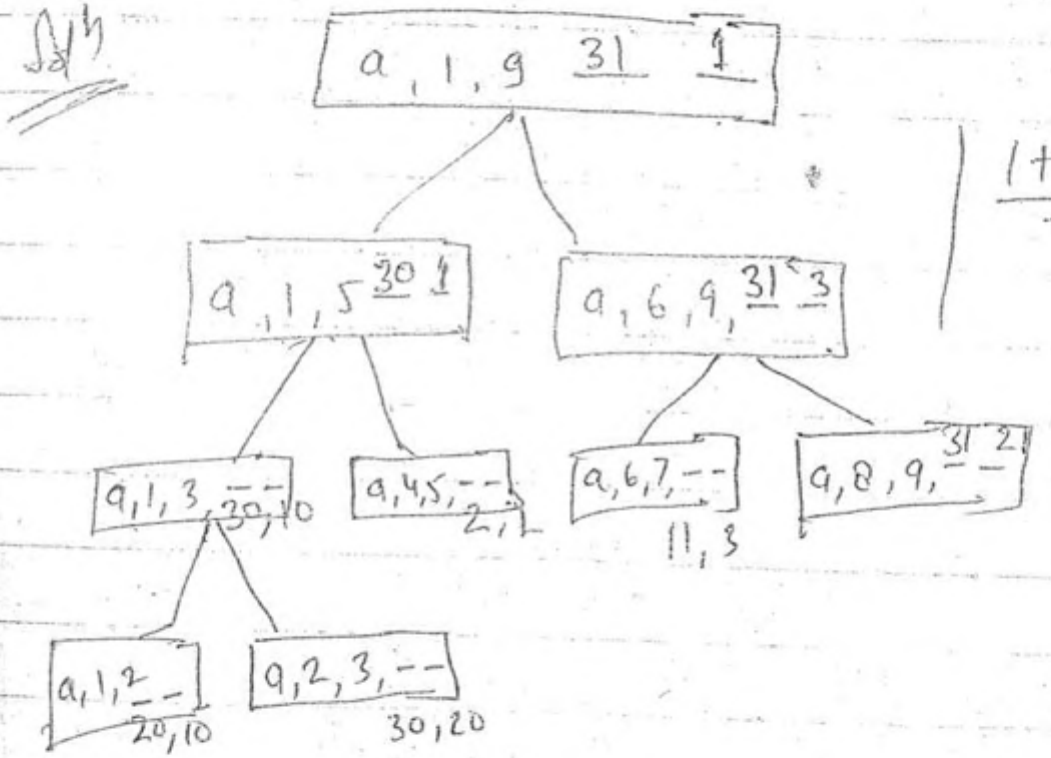
O(nlogn)

Using DAC Max & Min.

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

Max : 51
min : 1

Soln



DAC maxmin (a, i, J, max, min)

{

int mid,

if (i == J)

{ max = min = a[i];

i.e 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];

}

Small

else

{

mid = (i+J)/2 ⇒ c (constant)

DAC maxmin (a, i, mid, max1, min1) ⇒ T(n/2)

DAC maxmin (a, mid+1, J, max2, min2) ⇒ T(n/2)

```
if (max1 < max2)
        max = max2
    else
        max = max1

if (min1 > min2)
        min = min2
    else
        min = min1
return (max, min)
}
```

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

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

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

$$\Downarrow$$

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

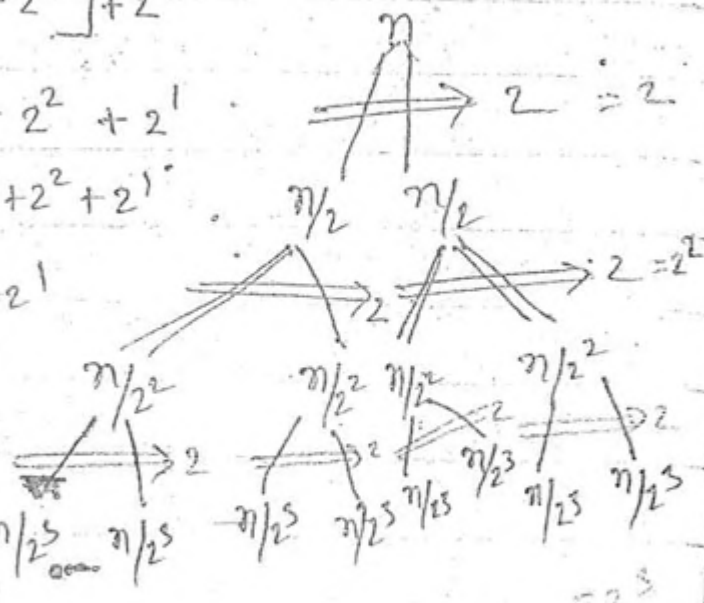
$T(n) = 2T(n/2) + 2$

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

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

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

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



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

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

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

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

$= 1.5n - 2$

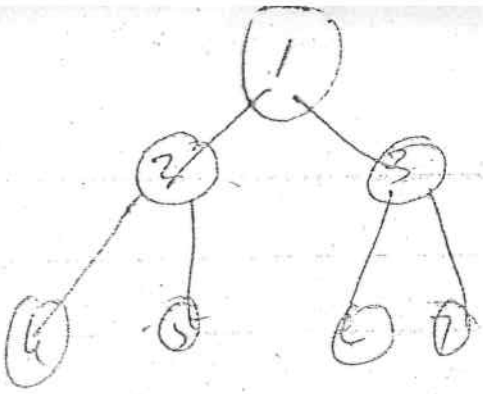
$= O(n)$ ⇓ It is best, worst & Avg case.

Note - Using DAC and without DAC Taking same time. So we how to consider space move, ~~t=5(n-1)~~

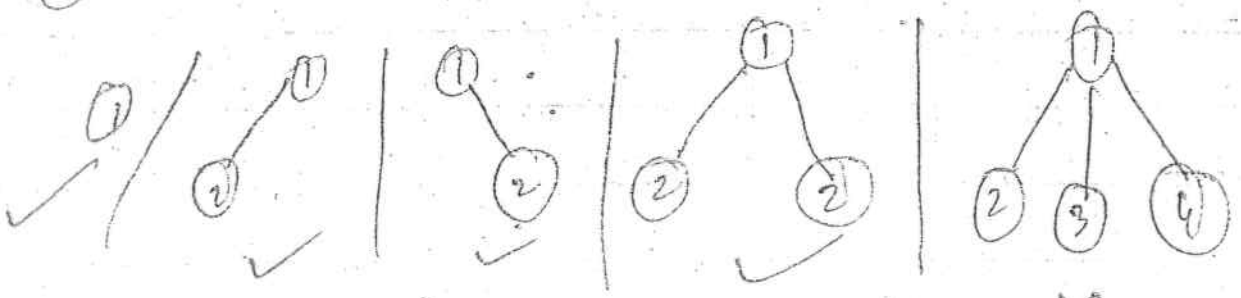
Now using DAC recursive program take more space campare to non-recursive Program.

Binary Search

Graph	Tree (DAG) Directed acyclic graph
1- no-root	(i) root
2- graph may be directed / indirected	(ii) Directed (Top → Bottom)
3- may contain cycle / non cycle	(iii) no cycle.
4- may be connected / disconnected	(iv) Connected.



Binary
⇓
atmost 2



X

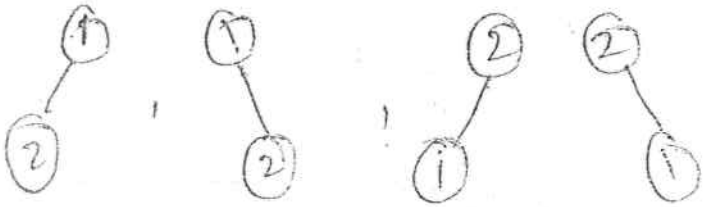
A Tree is a set of binary tree iff at every node maxm 2 children is present.

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

Sol^n $n=1$

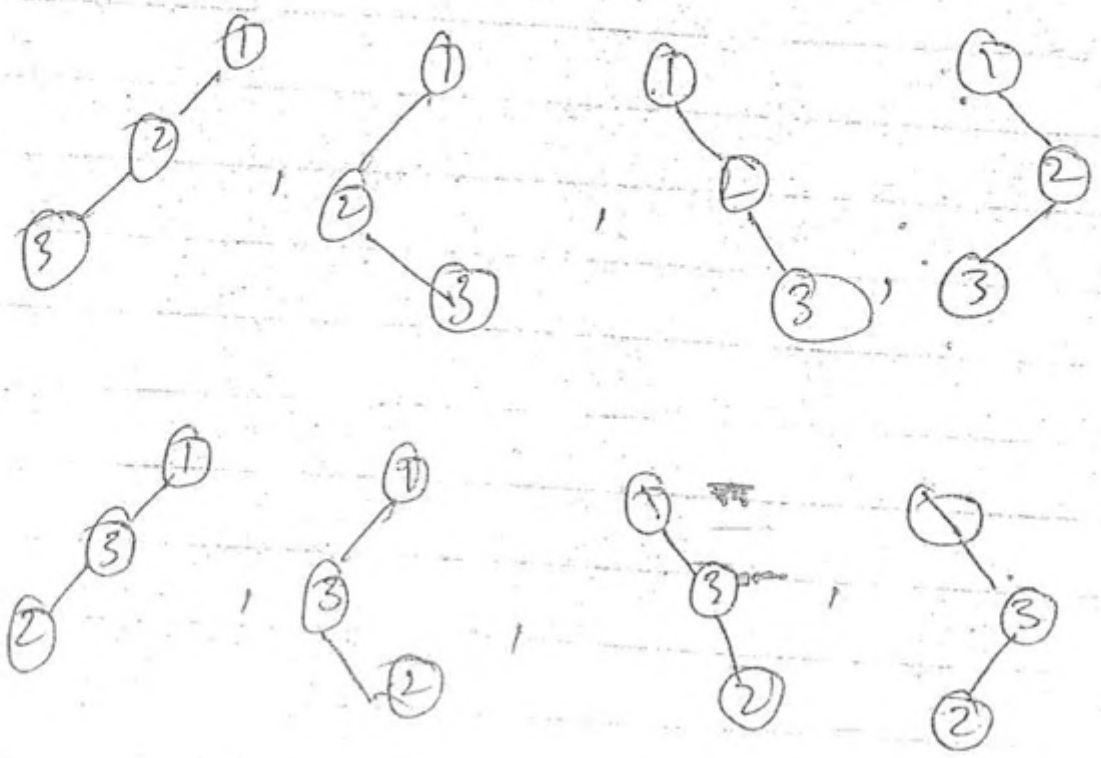
(1) ⟹ (1)

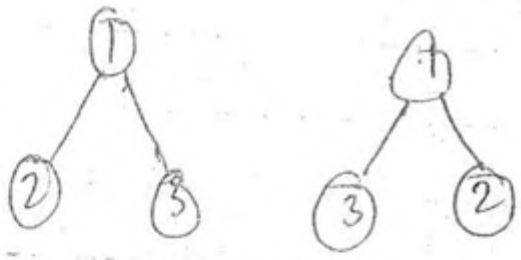
$n=2$ (1, 2)



⟹ 4

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





1 — 10
2 — 10
3 — 30

$\Rightarrow$ 30

the no of binary trees with n nodes

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

By formula

if $n=3$

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

$$= 30$$

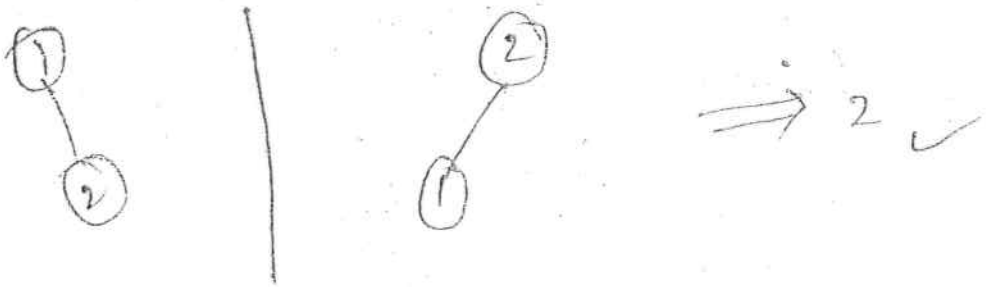
Binary Search Tree

(Repetation not allowed)

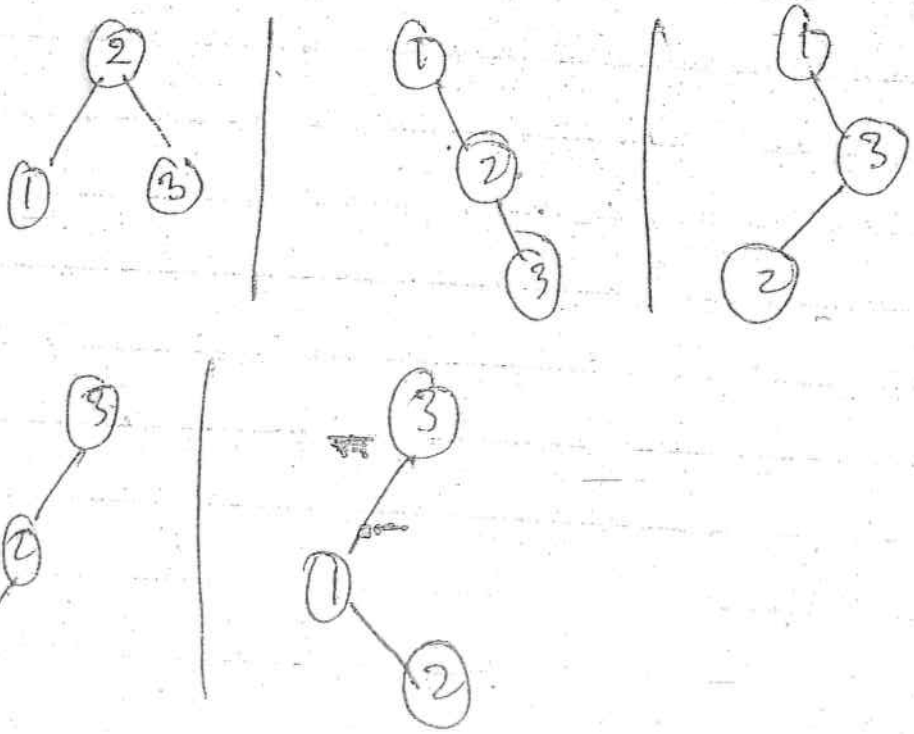
(1) at every node Root is greater than all the elements present Left Sub tree (LST)

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

eg $n=2$ (1,2)



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



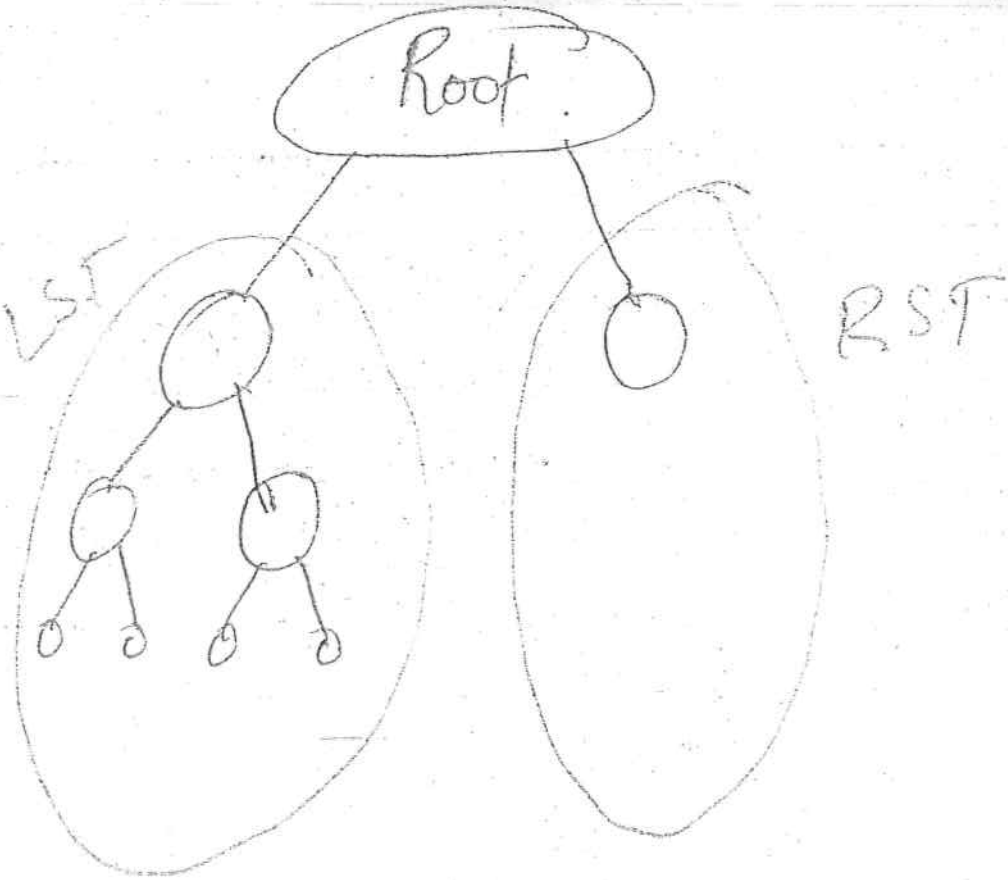
Binary Search Tree $= \frac{(2n)Cn}{n+1}$

$n = 1 \Rightarrow 1$

$n = 2 \Rightarrow 2$

$n = 3 \Rightarrow 5$

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



Searching -

Without Divide and Conquer

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

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

Soln

Ex

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

o/p : 7 | x=11 ✓

9 | x=31 ✓

1 | x=10 ✓

-1 | x=90 ✓

Without DAC

Linear Search

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

Best case - Ω(1)
Worst " → O(n)
Avg. case → (n+1)/2 = O(n)

Successfull search

Unsuccessful Search

Best case — $\Omega(n)$
Worst " — $O(n)$
Avg. " — $O(n)$

$\Downarrow$

$\theta(n)$

Binary Search (using DAC)

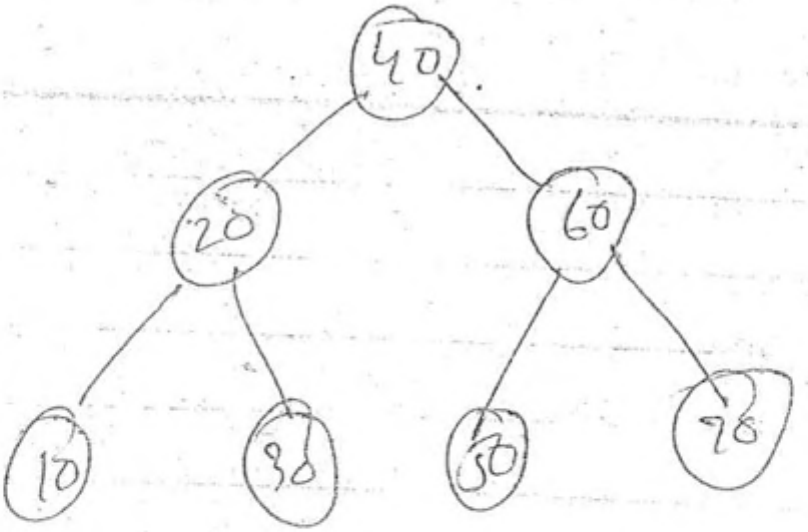
i/p = A sorted array of n-elements, element x.

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

eg

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

Sol^n.



$\log_2 n = 12.83$

$= 3$ Comparision.

```
Binary Search ( a , i , j , x )
{
  int mid ;
  if ( i == j )
  {
    if ( a[i] == x )
      return (i)
    else
    { return (-1)
  }
}
```

Small

```
else
{
    mid = (i+j)/2          ← C1
    if (a[mid] == x)
        return (mid)
    else
    {
        if (a[mid] > x)
            Binarysearch(a, i, mid-1, x)
        else
            Binarysearch(a, mid+1, j, x)
    }
}
```

Reccurence Relⁿ

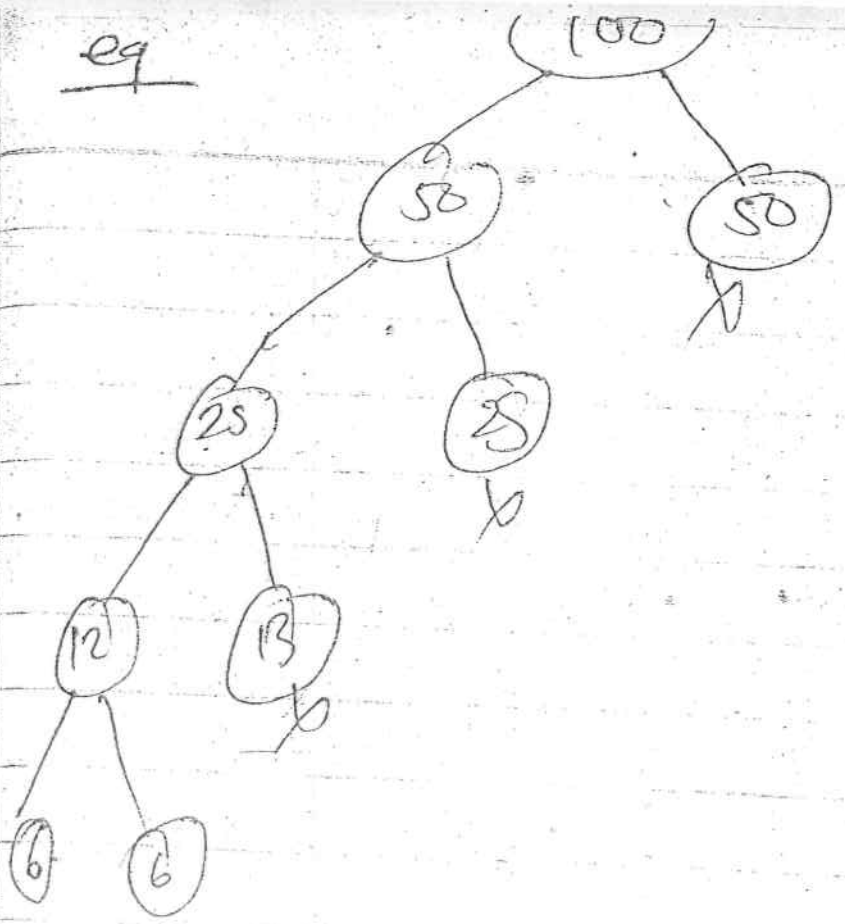
Let $T(n)$ be the time complexity to do Binary Search with DAC

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

$\Downarrow$

$O(\log n)$

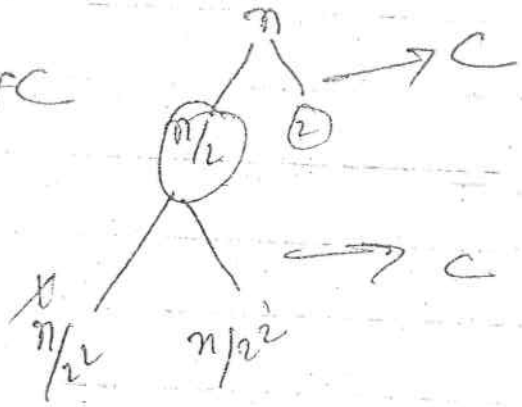
eg



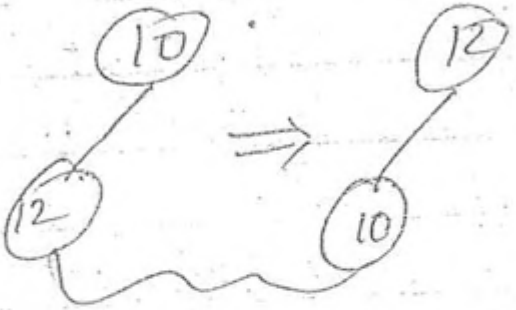
$T(n) = T(n/2) + c$

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

$\vdots$

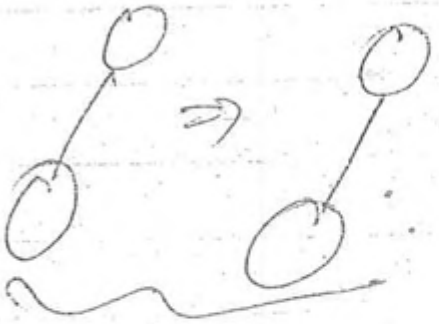


Unlabled Binay tree



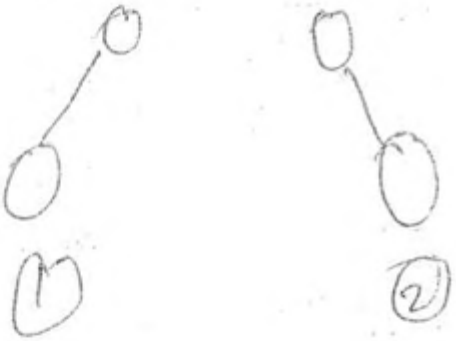
2 - different labled tree

Labled B-T



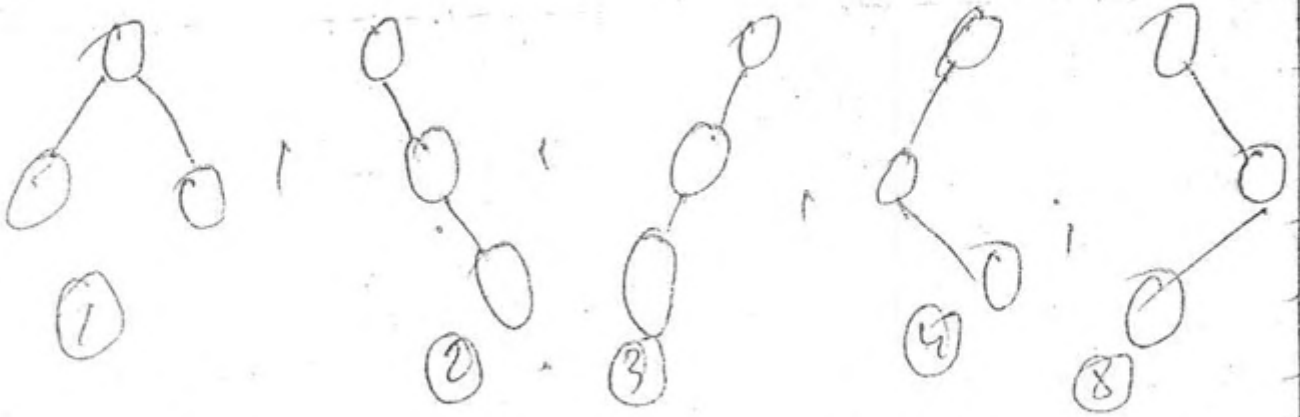
Both are same.

Q// $n = 2$ how many different unlabled B-T.



2//

$n = 3$



= 5//

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

a) 0

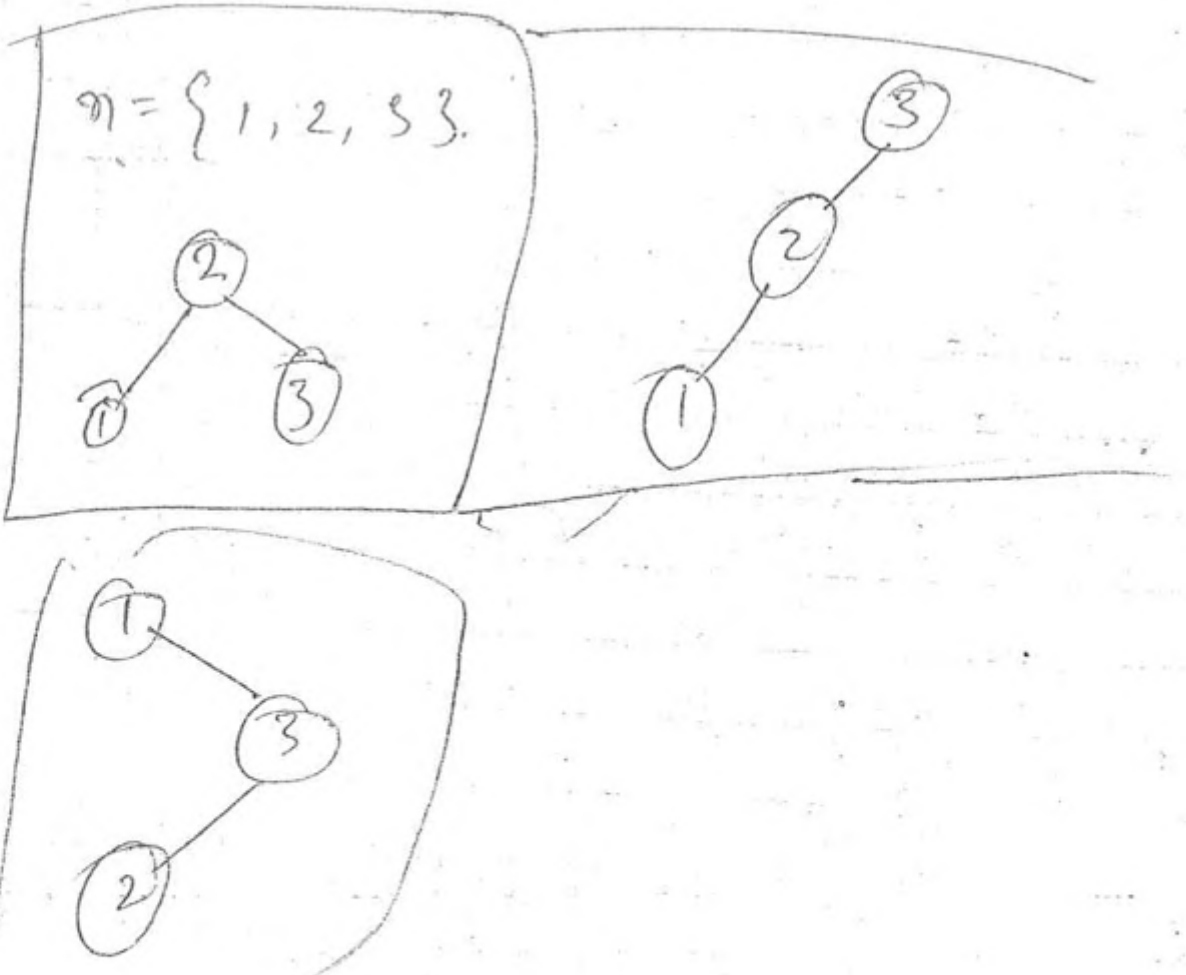
b) 1

c) $n!$

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

Sol/n

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



9

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

a) 61, 52, 14, 19, 40, 43

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

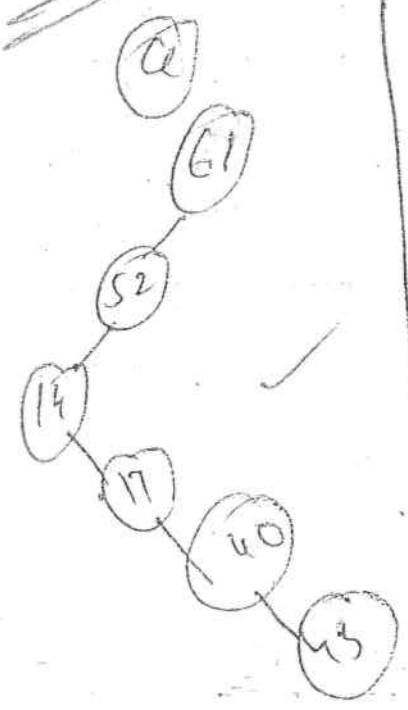
c) 10, 65, 31, 48, 37, 43,

d) 81, 61, 52, 14, 41, 43

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

Sol^{n}

b



(Linear ~~Binary~~ Search)

$\Omega(1)$ — Best case

$O(n)$ — Worst case

$\Theta(n)$ — Avg case.

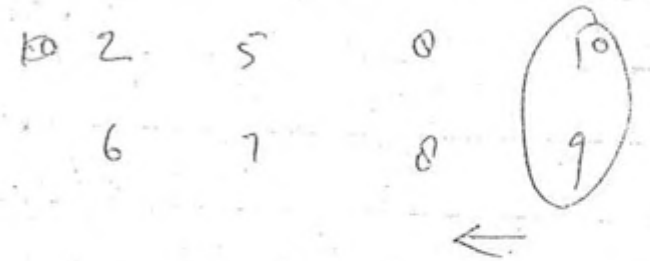
Binary Search means always goes left or right.

If Postion is greter than elment we go right side.

and if " " smaller " " then we go left side.

eg-1 -

2.



ele < Postion ⇒ right

ele > Postion ⇒ left.

$\Omega(0)$ — Best Case

$O(\log n)$ — Worst "

$\Theta(\log n)$ — Avg Case.

#

I/P : An array n-elements.

O/P : Find repeated elements.

eg

i/p A : 10, 20, 30, 1, 2, 3, 10, 20, 30

1 1 1 1 1 1 1 1 1

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

$\Rightarrow O(n^2)$

(A2)

(i) Short the given array by taking

$\Rightarrow O(n \log n)$

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

(ii) Scan the array to check

$a[i] = a[i+1]$ for all i

$O(n)$

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

$n = 1000$

$n^2 = 10,00,1000$

$n = 1000$

$n \log n = 1000 \times 3$

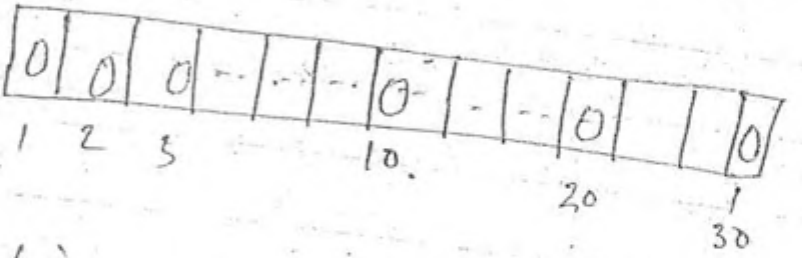
$= 3000$

(A3)

(1)

```
for ( i=1 , i<=n , i++ )
    flag [ a[i] ] = 0        ⟹ O(n)
```

flag:



(2)

```
for (i=1 , i<=n , i++ )
{ if (flag [a[i]] == 0 )
```

flag [a[i]] = 1
else
Print f (a[i])
}

$\Rightarrow O(n)$

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

#

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

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

eg

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

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

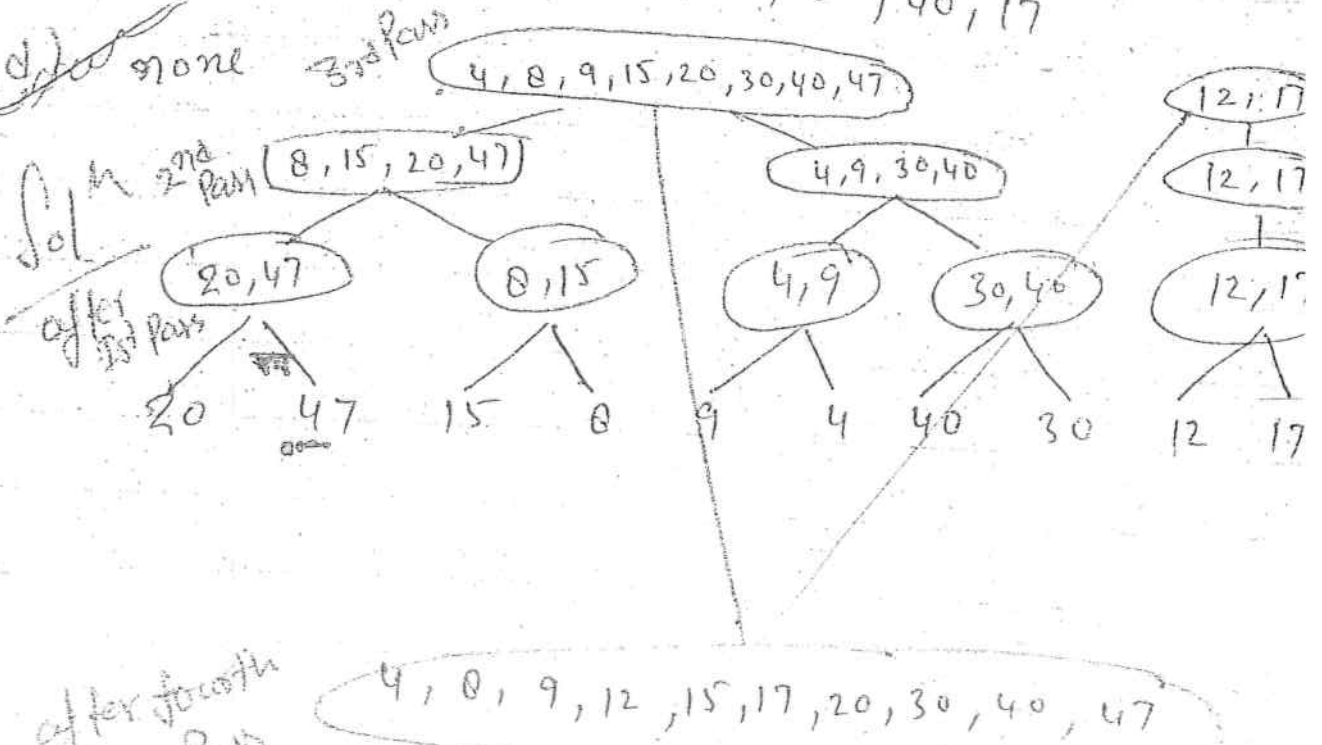
$\Rightarrow O(mn)$

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

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

d) none

Sol^n



after fourth pass: 4, 0, 9, 12, 15, 17, 20, 30, 40, 47

order = $\log_2 16 = 4$

Quick Sort (Tony Hoare)

(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. ⇒ Partation Algo
(2) Conquer the subproblem recursively to get solutions of subproblems.

→ Partial Divide and Conquer approach.

```
Partation (a, P, q)
{
    x = a[P];
    i = P;
for (J = P+1 ; J ≤ n , J++)
{
  if (a[J] ≤ x)
  {  i = i+1
     swap (a[i], a[J])
  }
}
```

A2.

i) Short the nut array in ascending order.
$\Rightarrow O(n \log n)$

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

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

\# 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 the elements present on RH.S of it.

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

In this leader = 7, 100

(A1) Apply linear search to check every element present on R.H.S. of it smaller or not

$\Rightarrow O(n^2)$

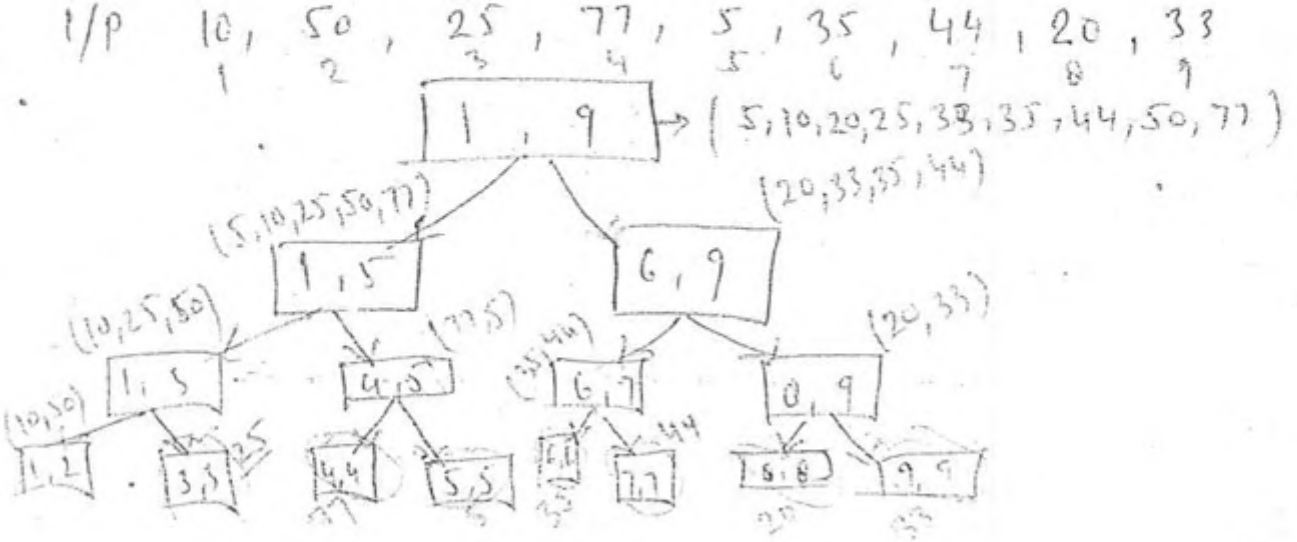
(A2) We cant apply binary search because order of the elements changing.

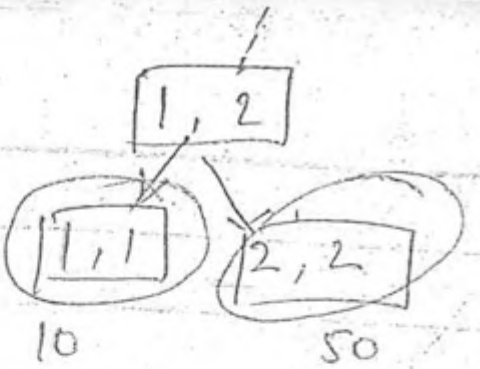
eg: 100, 70, 50, 40, 30, 10

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

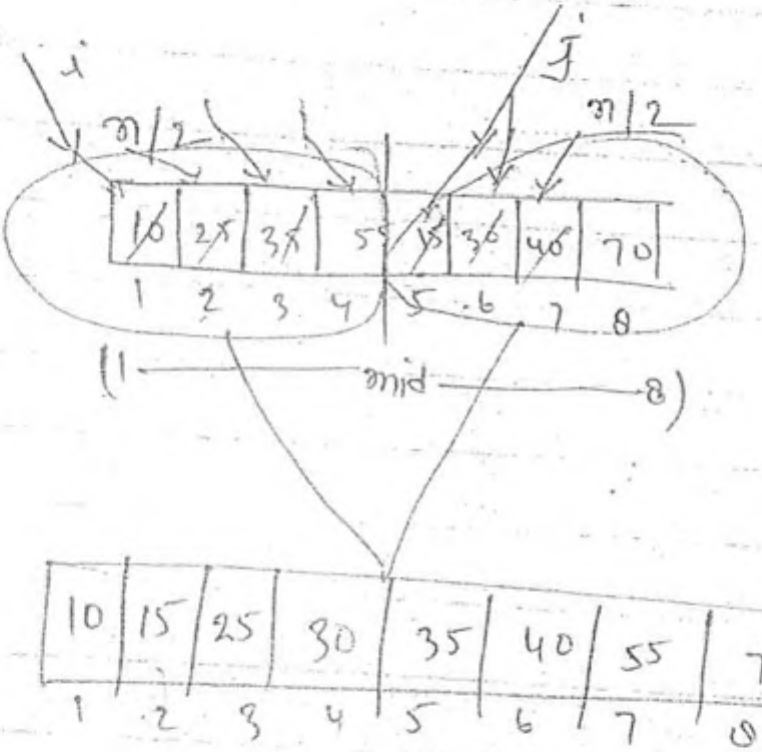
MergeSort

Divide and Conquer. Merge Sort.





Algo - eg



10	15	25	30	35	40	55	70
1	2	3	4	5	6	7	8

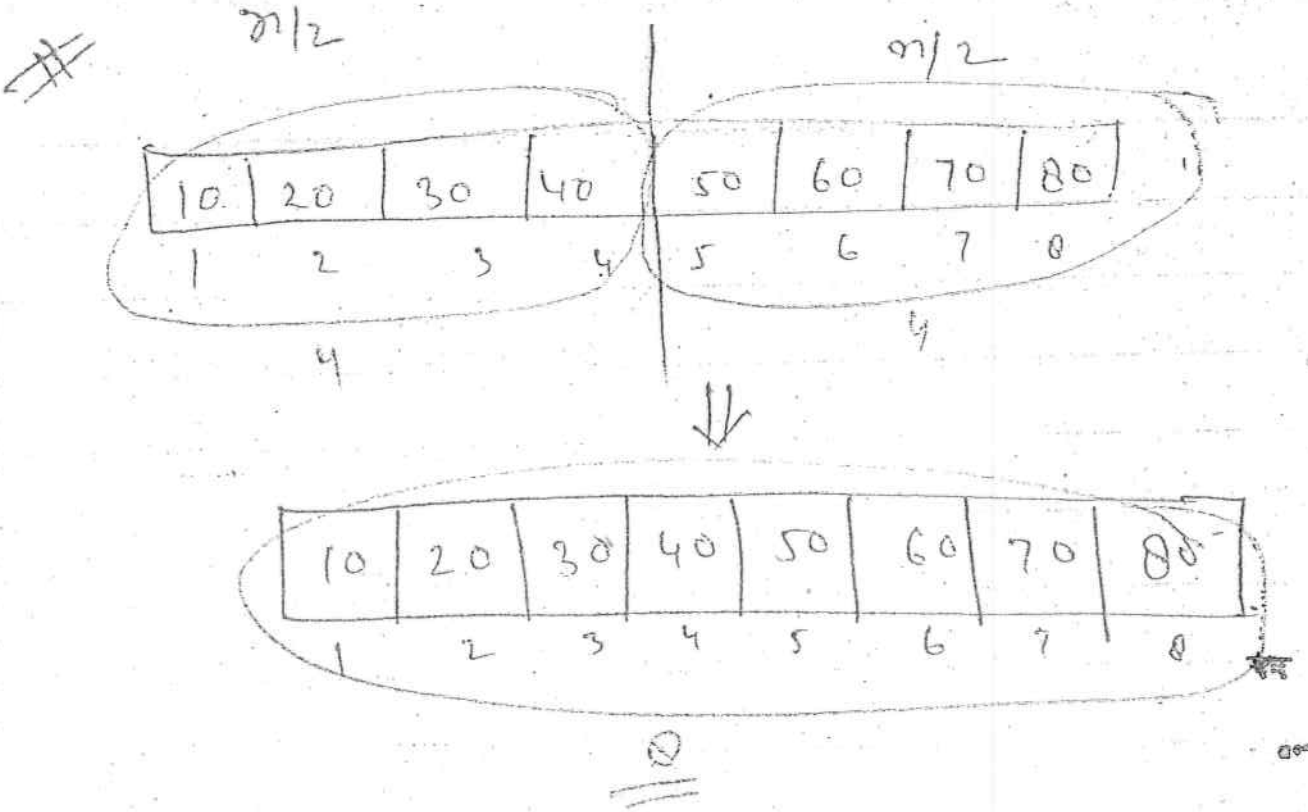
(10, 15) ⇒ 10
(25, 15) 15
(25, 30) ⇒ 25
(35, 30) ⇒ 30
(35, 40) ⇒ 35
(55, 40) ⇒ 40
(55, 70) ⇒ 55

70 ⇒ 70

total Comparison = $\frac{4+4-1}{} = 7$

$= n/2 + n/2 - 1$

$= n - 1$



(10, 50) ⇒ 10

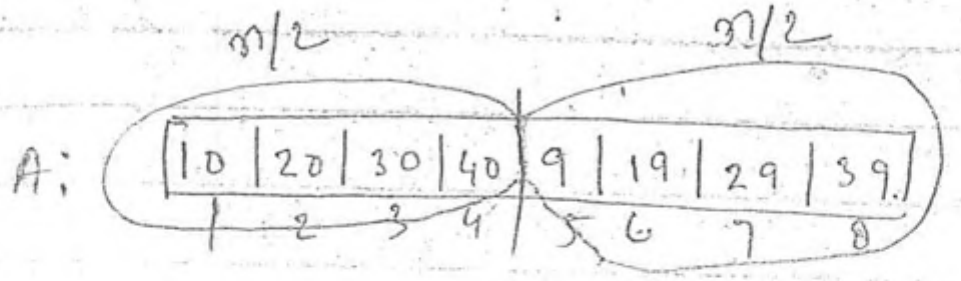
(20, 50) ⇒ 20

(30, 50) ⇒ 30

(40, 50) ⇒ 40

Best Case ⇒ $n/2 \Rightarrow O(n)$

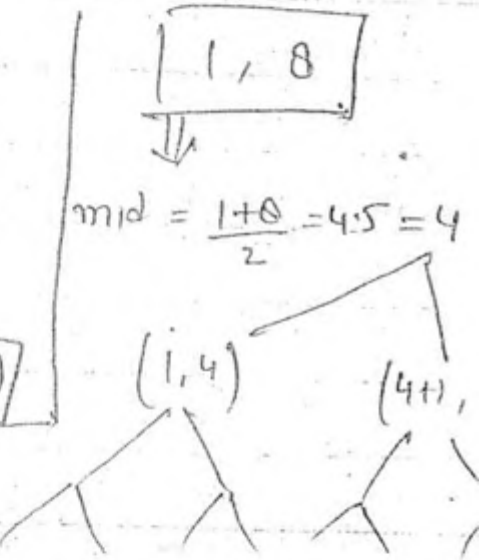
Worst Case ⇒ $n/2 + n/2 - 1 \Rightarrow O(n)$



Using Divide and Conquer

```
MergeSort ( a, i, j )
{
  int mid;
  if ( i==j) return (a(i))
  else
  {
    mid = (i+j)/2 ⇒ C
    Mergesort( a, i, mid ) ⇒ T(n/2)
    Mergesort(a, mid+1, j) ⇒ T(n/2)
    Merge ( a, i, mid, j)
    return (a)
  }
}
```

Small



#

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

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

$$\Downarrow$$

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

$$= 2\left[2T(n/2) + n/2\right] + n$$

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

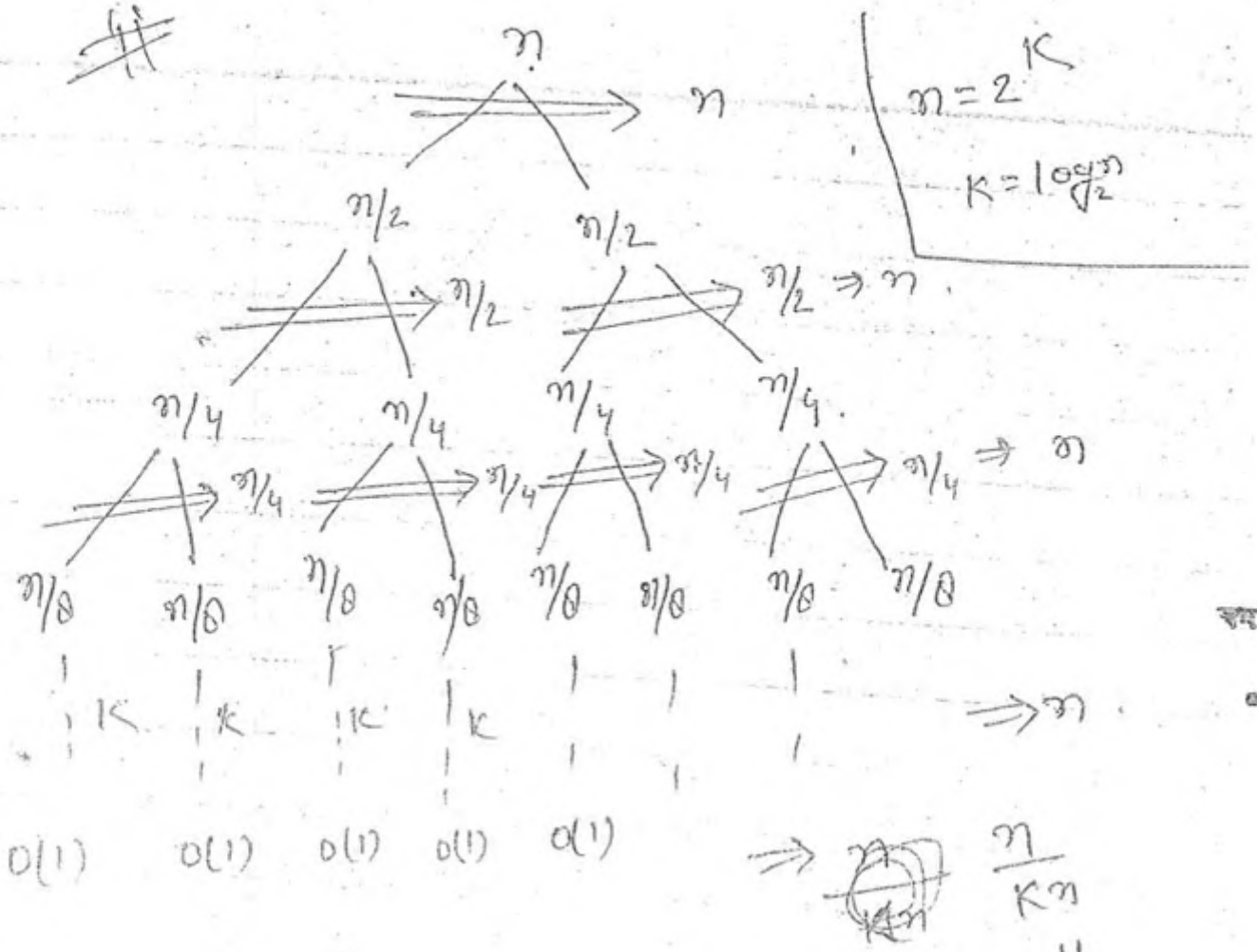
$$= 2^3 T(n/2^2) + 3n$$

$$\vdots$$

$$\vdots\ K$$

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

$$\theta(n \log n)$$



$n = 2^K$

$K = \log_2 n$

Time Complexity of Mergesort.

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

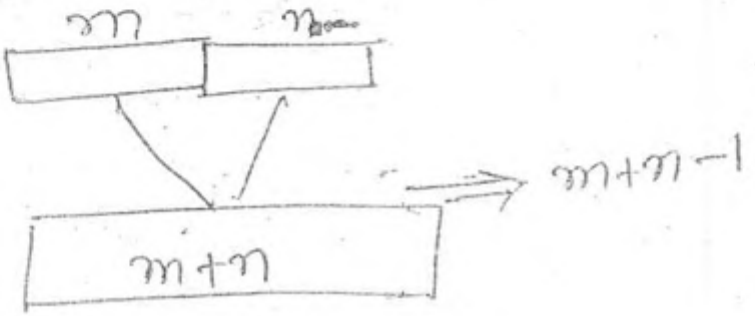
Worst case = $O(n \log n)$

i.e. Best Case = Worst Case.

(1) Merge Sort is preferable for larger size array no for smaller size. Smaller size can be preferable by insertion sort

(2) Merge sort is not inplace sorting technique because in merge sort process it is taken an extra array of size m.

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



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

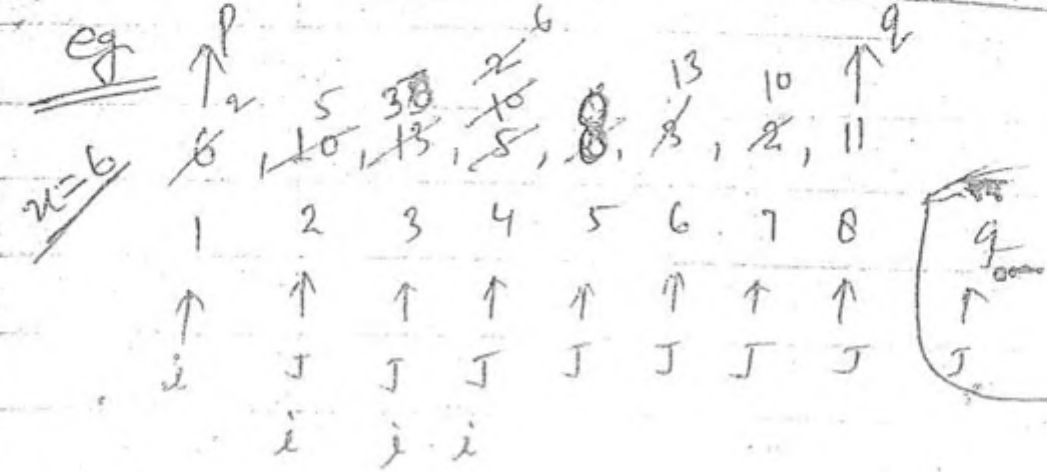
20, 47, 15, 8, 9, 4, 40, 30, 12, 17

then what will be the output after 2nd Pass of the algo.

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

```
    swap(a[i], a[p])
    return(i);
}
```

eg

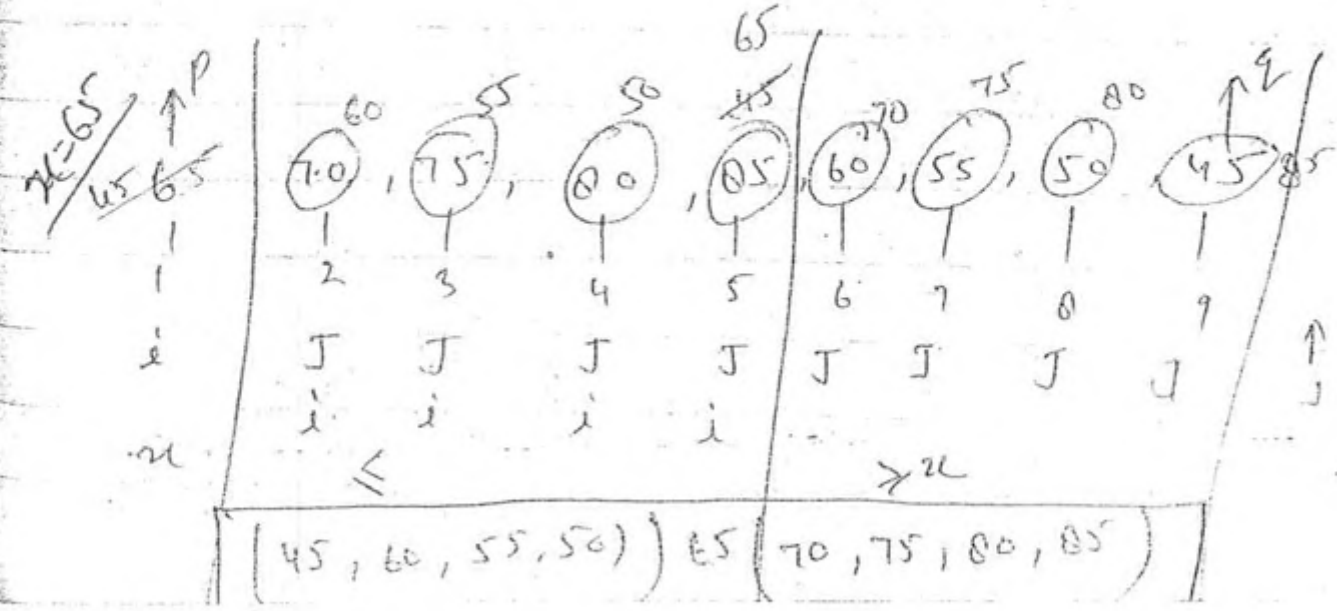


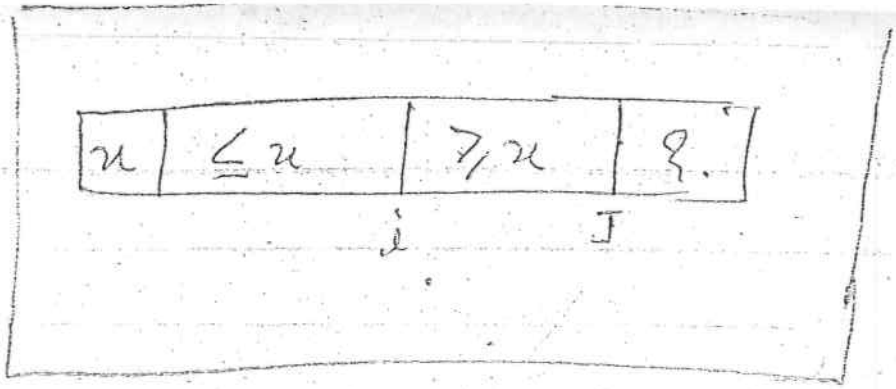
obser

get

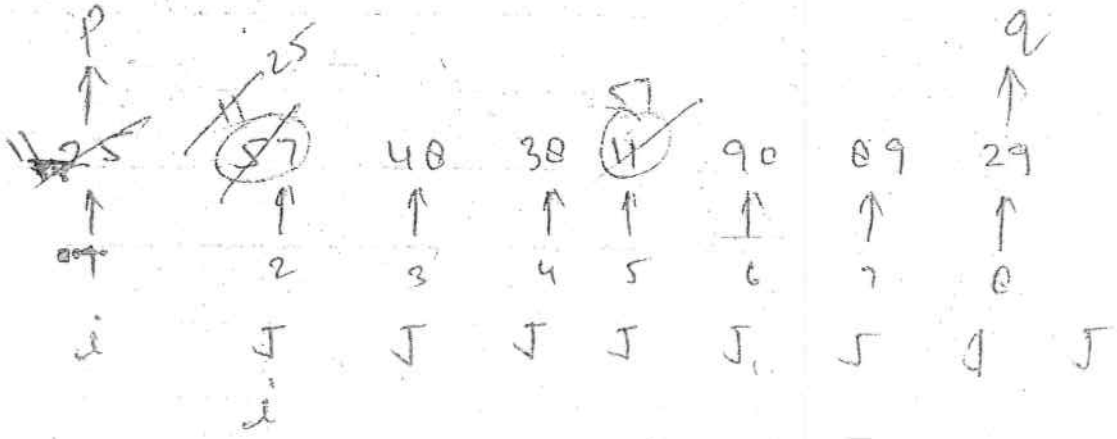
(2 5 3) 6 (8 13 10 11)

eg Apply partion algo to the following inpu





P



(11) 25 (40, 30, 57, 90, 09, 29)

P(1)

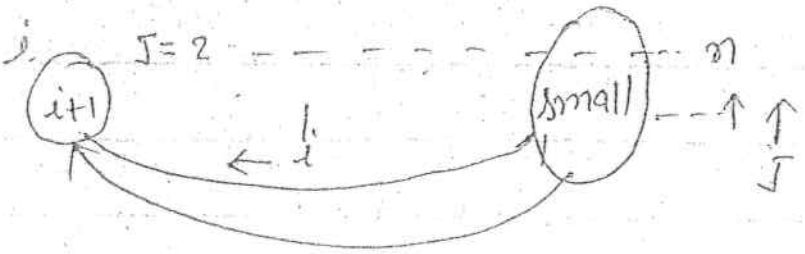
10 20 30 40 50 60 70 80 90 100

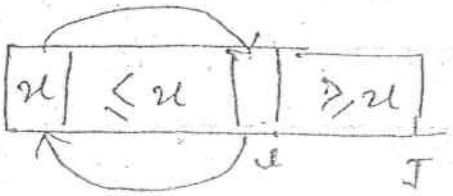
P(2) -

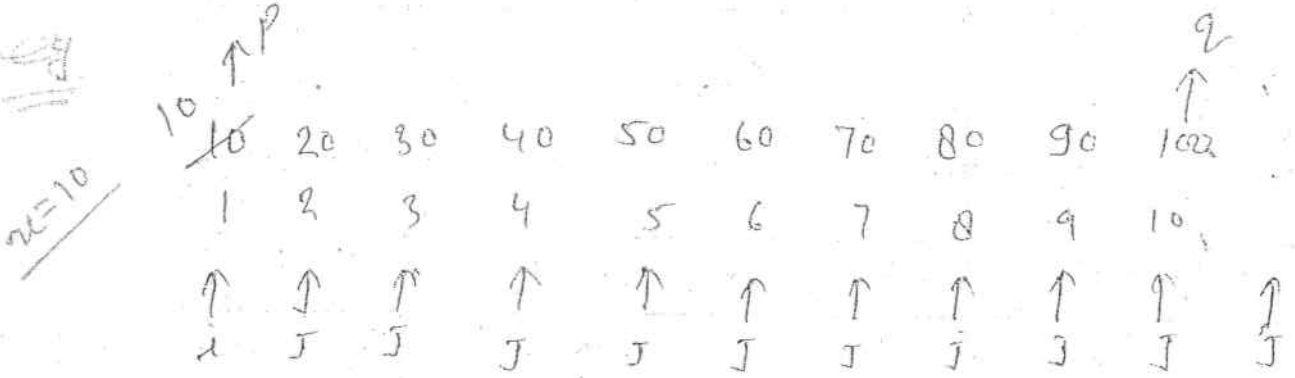
100 90 80 70 60 50 40 30 20 10

100

Partation Algo

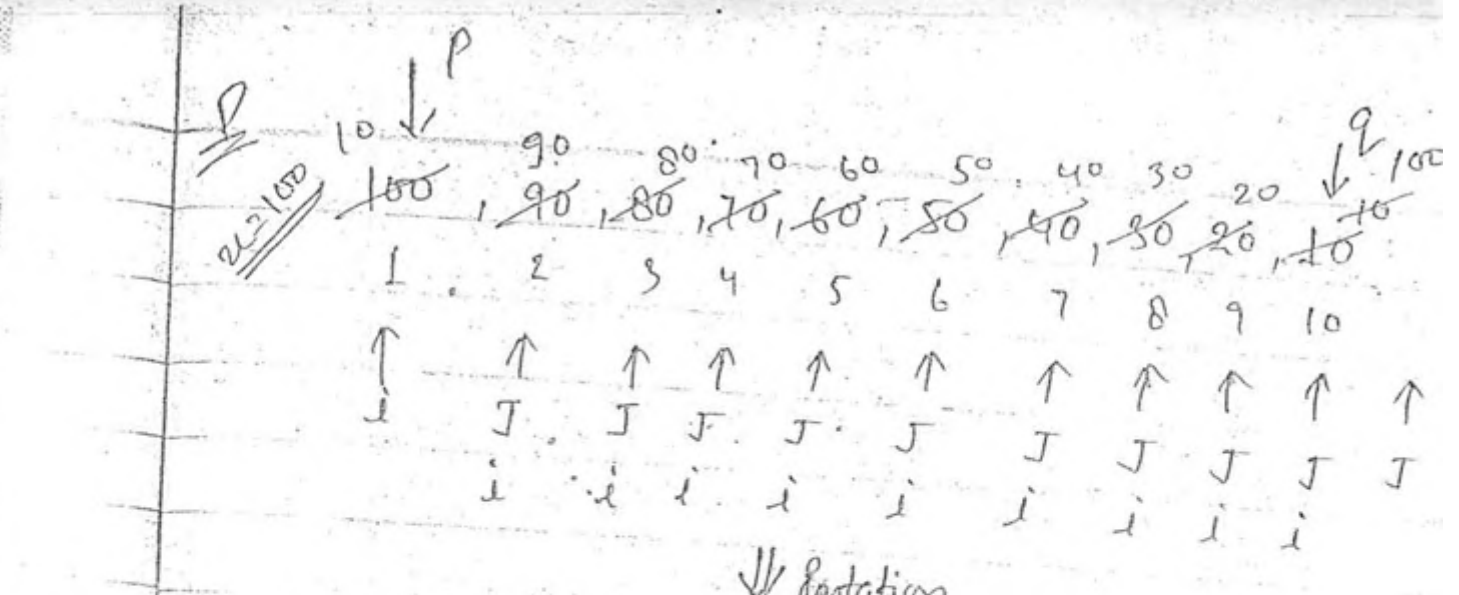






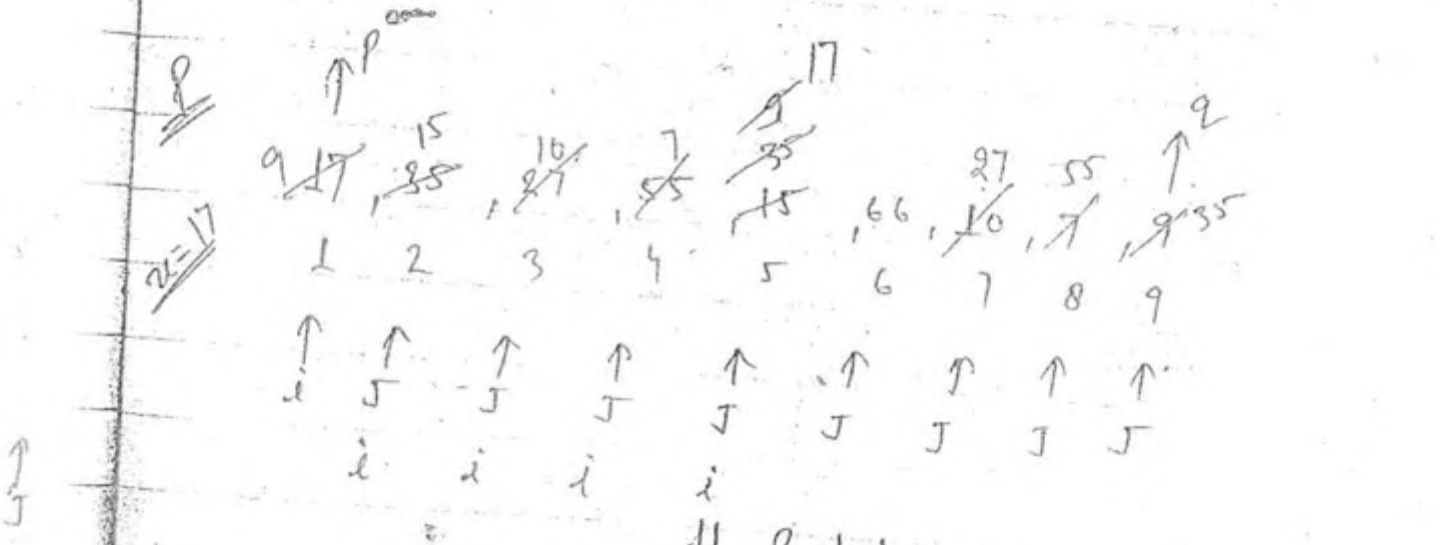
⇓ Partation

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



Partation

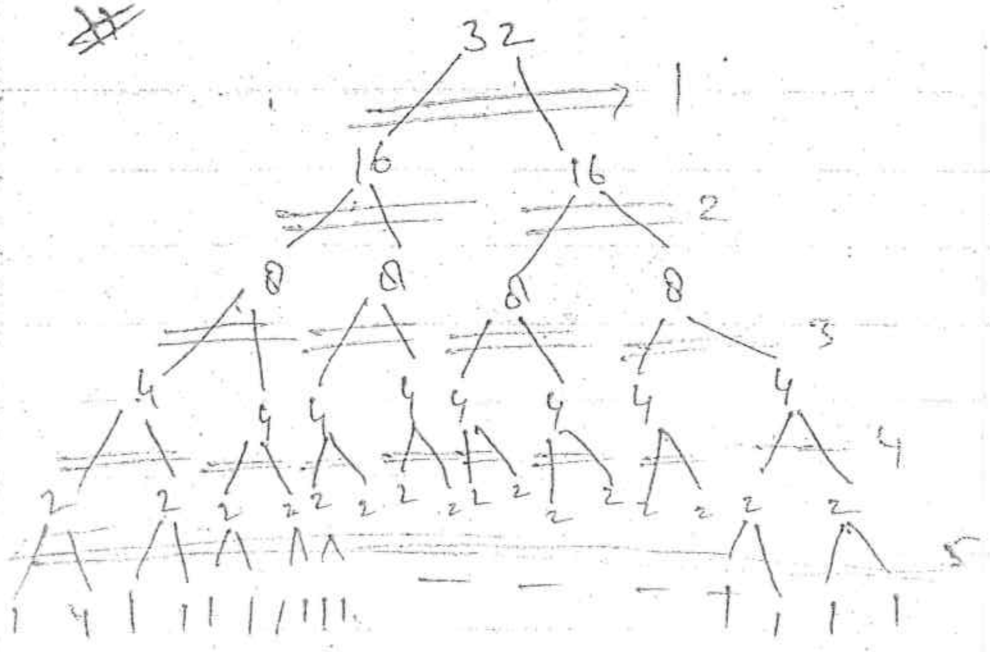
(10, 90, 80, 70, 60, 50, 40, 30, 20) 100 ( )



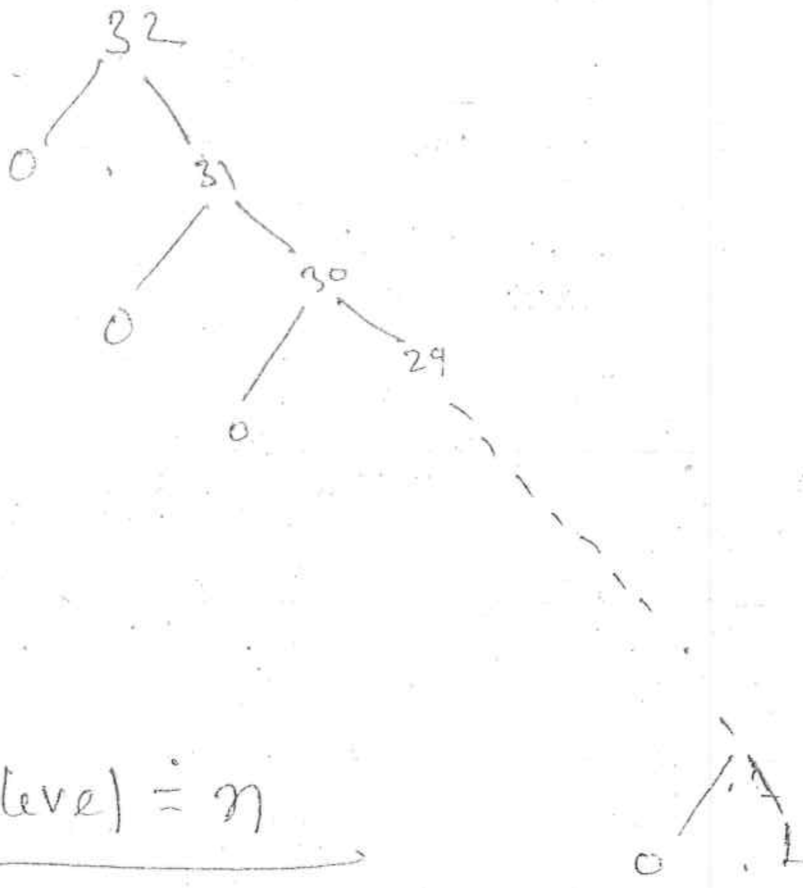
Partation.

(9 15 10 7) 17 (66, 27, 55, 55)

No of level is $= \log_2 n$



level 5



no of level = n

Quicksort

```
Quicksort (a, p, q)
{
If (p == q) return (a[p])
else
{
m = Partation (a, p, q);
Quicksort (a, p, m-1);
Quicksort (a, m+1, q);
return (a)
}
}
```

Small

Big

n

n/2

n/2

Best case T.C of Quicksort Algo.

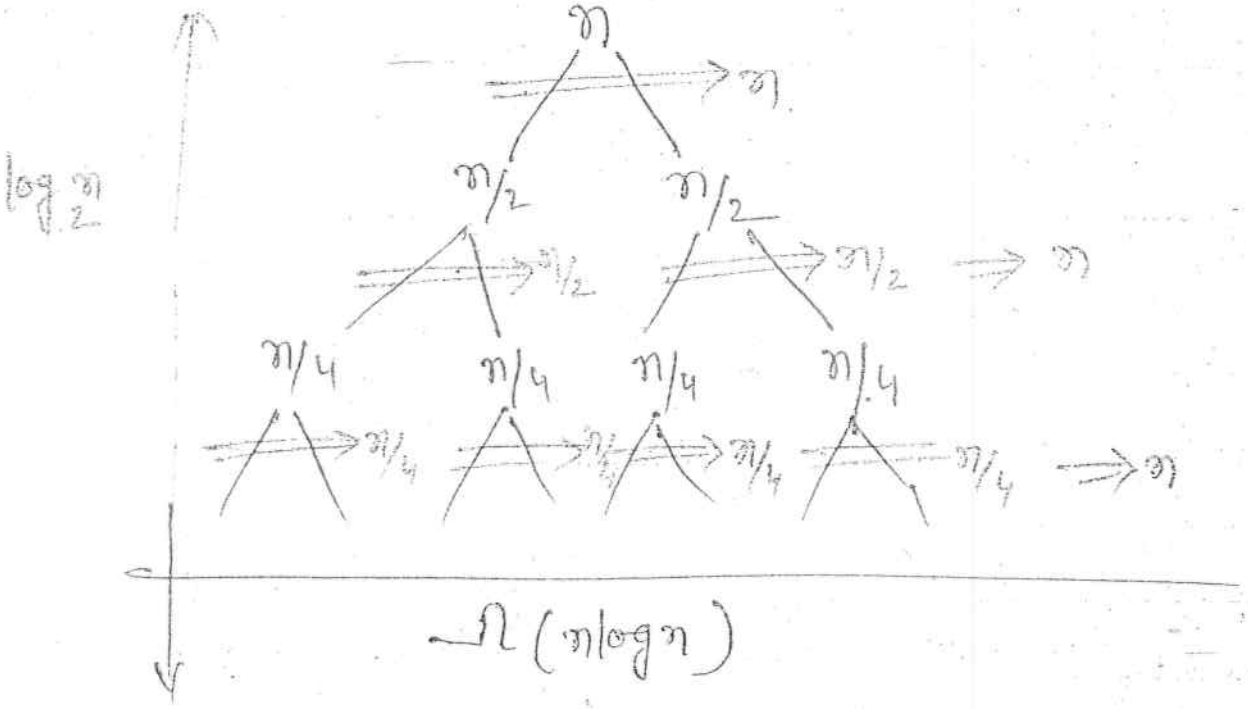
n elements.

Partation ⇒ n.

If we are lucky.

Recurrence Relation

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



eg: 10, 20, 30, 40, 50

⇓ Partation ⇒ 5

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

⇓ Partation ⇒ 4

( ) 20 (30, 40, 50)

⇓ Partation ⇒ 3

( ) 30 (40, 50)

⇓ Partation ⇒ 2

( ) 40 (50)

⇓ Partation ⇒ 1

( ) 50 ( )

50 40 30 20 10

⇓ Partation ⇒ 5

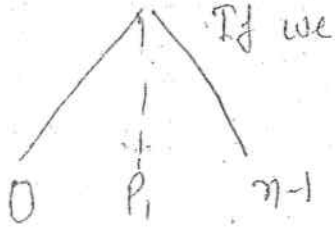
(40, 30, 20, 10) 50 ( )

Worst Case

n - elements.

⇓ Partation $\Rightarrow n$

If we are not Lucky.



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

Average Case

LULULULU - - - - - - $\Rightarrow$ Lucky.

Lucky Unlucky.

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

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

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

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

$$\Downarrow$$

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

$$= 2L(n/2) + O(n)$$

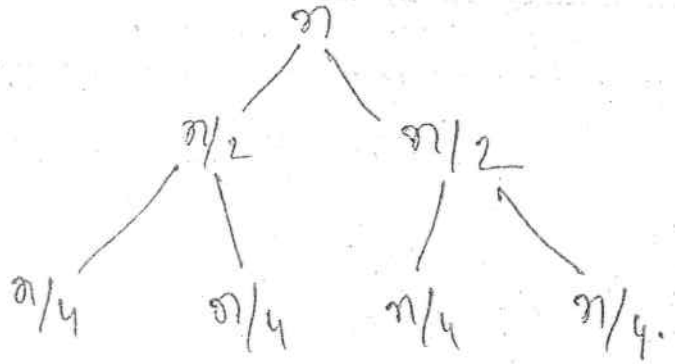
$$\Downarrow$$

$$O(n \log n)$$

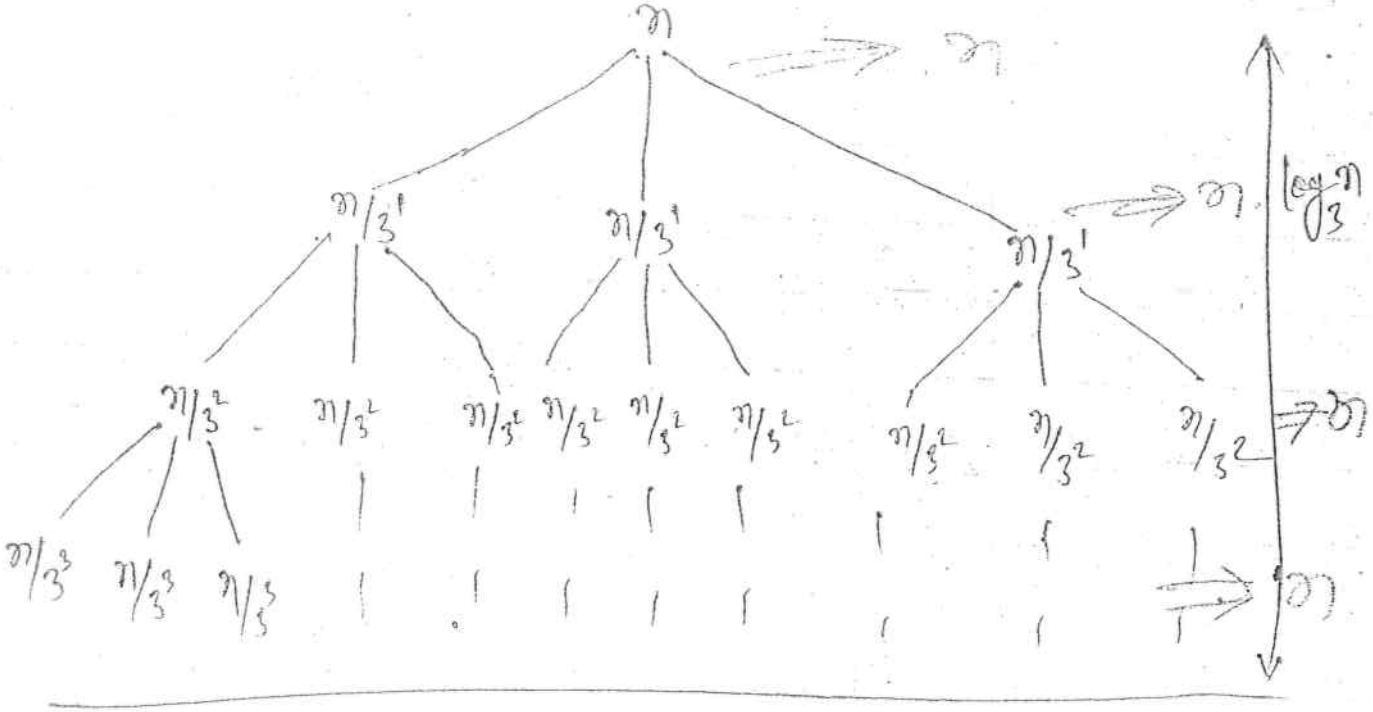
Average case → $O(n \log n)$

Recursive Tree Method

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

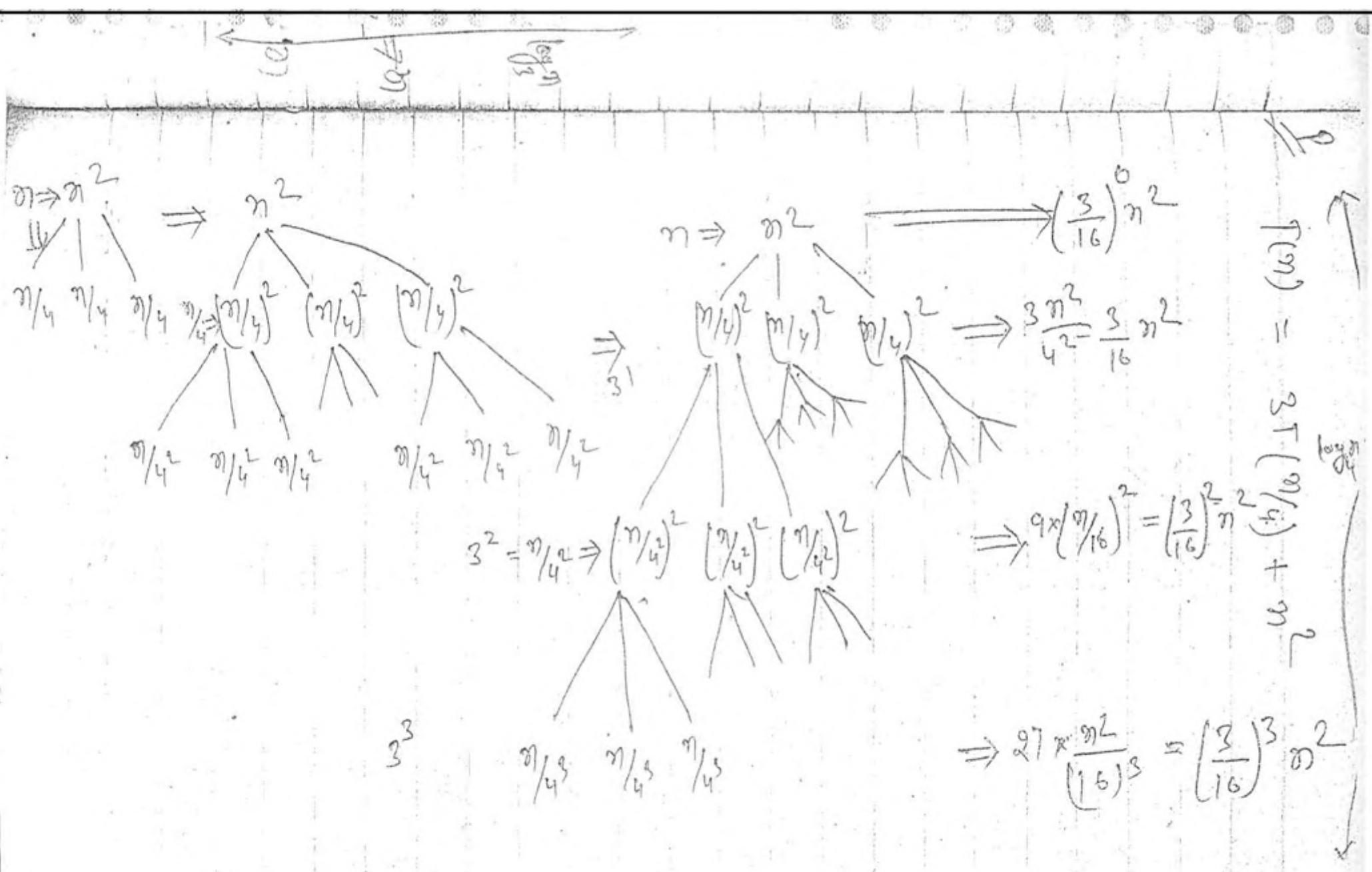


Ex $T(n) = 3T(n/3) + n$



$n \log n$

$\Rightarrow O(n \log n)$



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

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

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

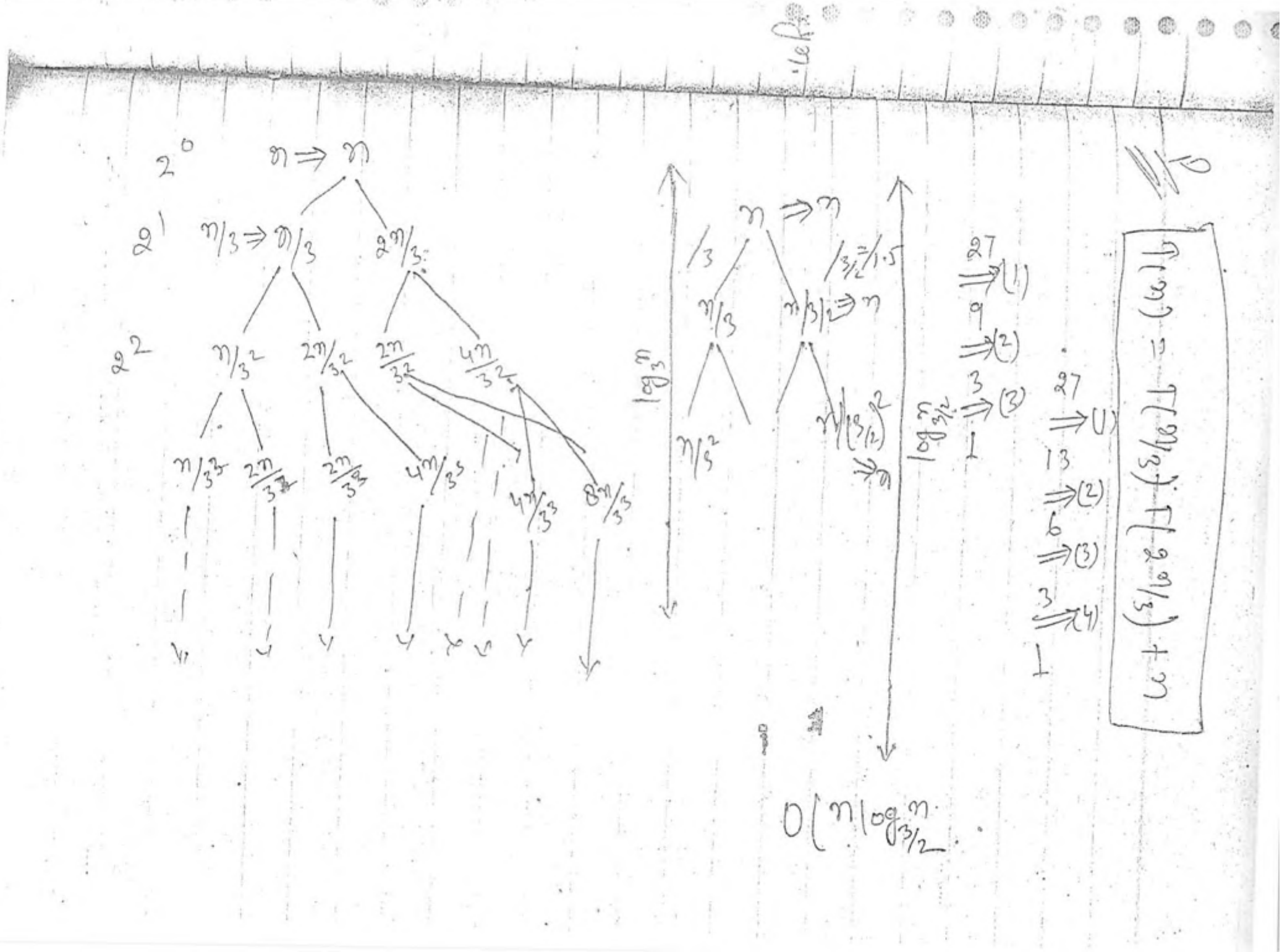
$$= \left(\frac{3}{16}\right)^2 n^2 + \left(\frac{3}{16}\right)^1 n^2 + \left(\frac{3}{16}\right)^0 n^2 \quad \Rightarrow n\log_4 n.$$

$n + n + \cdots + n$

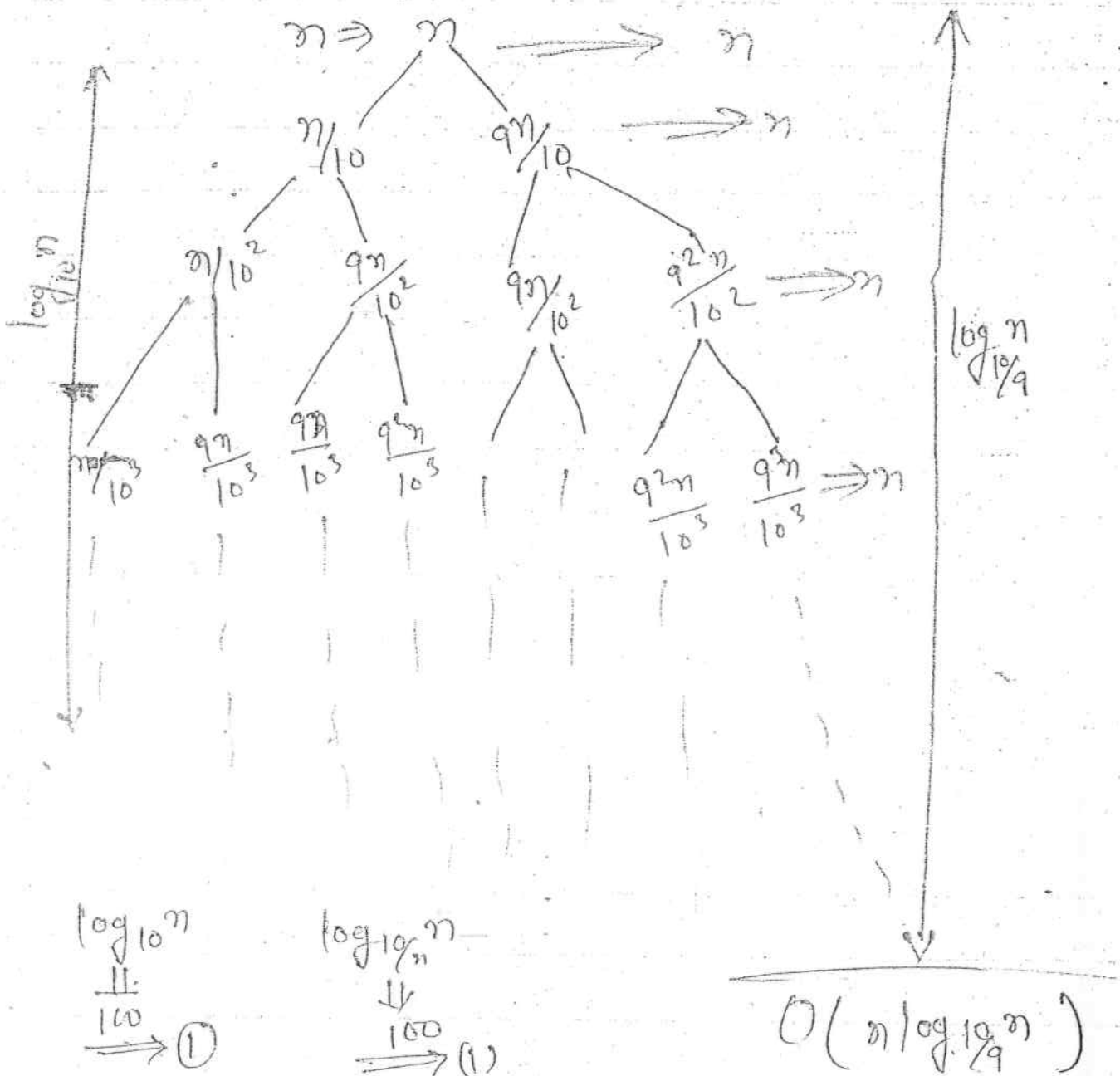
$$\Rightarrow n^2\left[\left(\frac{3}{16}\right)^0 + \left(\frac{3}{16}\right) + \cdots + \left(\frac{3}{16}\right)^{\log_4 n}\right]$$

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

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



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



$\log_{10} n$

$100 \Rightarrow (1)$

$10 \Rightarrow (2)$

1

$\log_{10/9} n$

$100 \Rightarrow (1)$

$50 \Rightarrow (2)$

$25 \Rightarrow (3)$

$12 \Rightarrow (4)$

$(6) \Rightarrow (5)$

$3 \Rightarrow (6)$

1

```
Qsort( a, P, q).
{
  if (P==q)
  else
  {
  m = Partation (a, P, q)
    q.s (a, P, m-1)
    q.s (a, m+1, q)
  }
}
```

Best case ⇒ (Choosing Pivot)

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

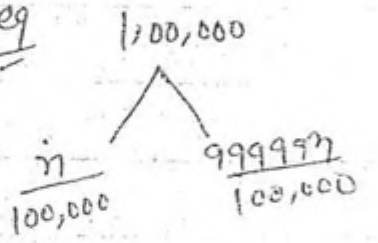
Worst case ⇒ $= 2T(n/2) + n.$

$$T(n) = O(1) + O(n) + T(0) + T(n-1)$$

$$\Downarrow$$

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

eq



Average case ⇒

$$T(n) = O(1) + O(n) + T(n/5) + T(4n/5)$$

$$= T(n/5) + T(4n/5) + O(n) \Rightarrow O(n \log_{5/4} n)$$

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

P

After apply few Passes of Quick Sort on the given array we got following output.

1, 10, 5, 8, 25, 44, 55, 30, 70

then

how many pivot elements are there in the above output.

Ans There are three Pivot element

1, 25, 70

P In Quick sort of n-element the $(n/4)^{th}$ 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.

Sol^n

$$T(n) = O(n) + O(n) + T(n/4) + T\left(\frac{3n}{4}\right)$$

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

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

1

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^{n}

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

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

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

2

Randomized Q. Sort:

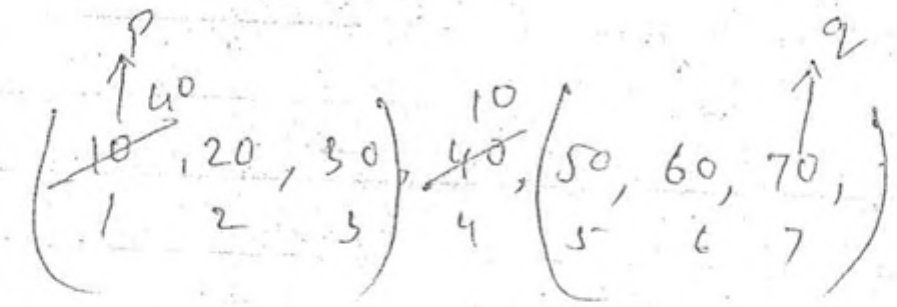
(1) It is independent of input order.

(2) ~~Further~~ For the sorted input array it will $O(n \log n)$ time worst case.

(3) No specific input behaves as worst case.

Drawback of R Q S

eg



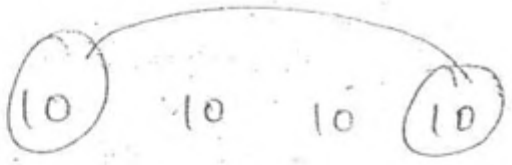
4 = random generater (1, 2, 3, 4, 5, 6, 7)

swap (a[p], 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

eg



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

( ) 10 (10, 10, 10)

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

A2 (1) for (i=1 , i≤K ; i++)
find i^{th} smallest element and delete
$\Rightarrow (K-1)n$

(2) find smallest element $\Rightarrow n$

$O(Kn)$

Algo

```
Selectionprocedure (a, i, j, K)
{
if (i==j)                                   → b
{
if(K==i)
return (a[i])
else
return (-1);
}
else
{
m= partation(a,i,j)                         → n
if (m==K) return (a[m])                     → 2W_c
else{
if (m<K) selection Procedure(a, m+1, i, K)
else selection procedure(a, i, m-1, K)
}
}
```

n(log n)

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

Matrix Multiplication

Without DAE

Matrix Addition -

$A_{n \times n}$

$B_{n \times n}$

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

C contains n^2 elements

for every elements required one ~~element~~ addition.

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

$\Rightarrow O(n^2)$

```
for (i = 1 to n)
    for (j = 1 to n)
C(i)(j) = A[i][j] + B[i][j]
```

Matrix Multiplication

eg

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2\times3}, \quad B = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}_{3\times3}$$

$$C_{2\times3} = A_{2\times3} * B_{3\times3}$$

$$C = \begin{bmatrix} - & - & - \\ - & - & - \end{bmatrix}_{2\times3}$$

If

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

n

Matrix Multiplication

⇒ $A_{n\times n}$

$B_{n\times n}$

n)

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

TB[4][3]

C - Contains n^2 elements.

For every elements n-multiplication

So

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

$\Rightarrow O(n^3)$

(1)
```
for ( i = 1 to n)
for ( j = 1 to n)
    c[i] [j] = 0
```
$\Rightarrow n^2$

(2)
```
for ( i = 1 to n )
for ( j = 1 to n)
for ( k = 1 to n)
```
$\Rightarrow n^3$

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

$O(n^3)$

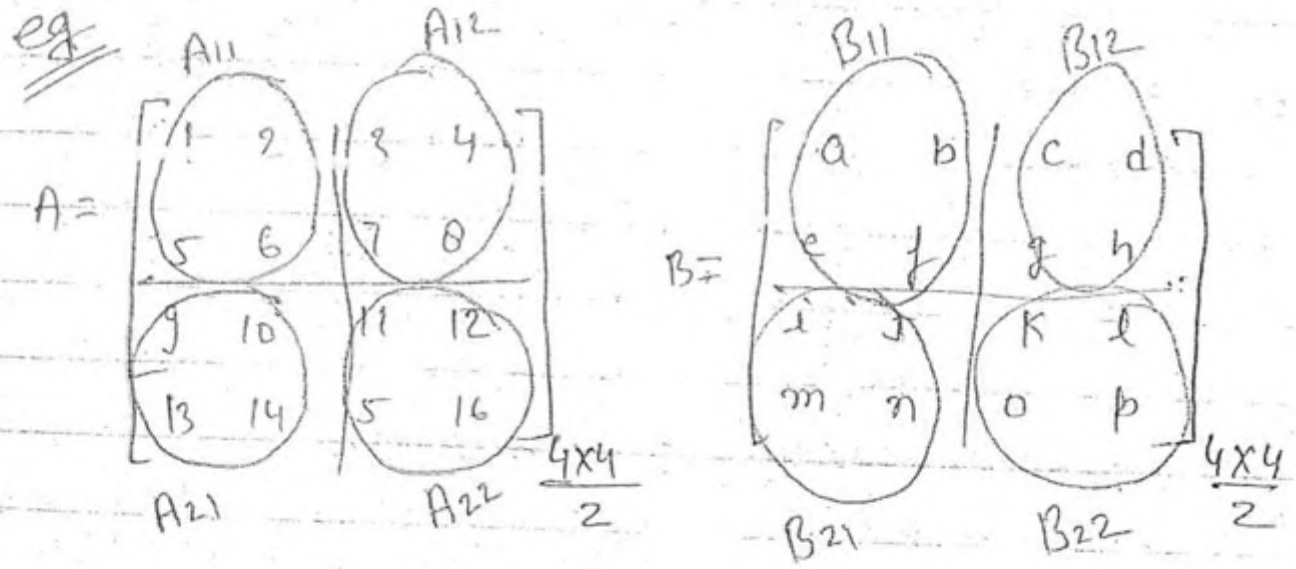
Using DAC

(1) Given two matrixs sizes are $\leq 2 \times 2$ then problem is small stop.

(2) Assume given two matrixs are powers of 2

(3) Assume given matrices are squre matrices.

eg



$C = A \times B$

$4\times4 \quad 4\times4 \quad 4\times4$

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

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

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

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

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

$$T(4) = 8\,T(2)$$

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

$$T(50) = 8\,T(25)$$

$$T(n) = 8\,T(n/2)$$

$$T(n) = \begin{cases} c & \text{if } n \le 2 \times 2 \\ 8T(n/2) + 4(n/2)^2 \\ \Downarrow \\ 8T(n/2) + n^2 \end{cases}$$

$$T(4) = \begin{cases} c \text{ if } n \le 2 \times 2 \\ 8T(4/2) + 4 \text{ additions of size } 4/2 * 4/2 \end{cases}$$

$$T(n) = \begin{cases} c \text{ if } n \le 2 \times 2 \\ 8T(n/2) + 4(n/2)^2 \text{ if } n \times 2 \end{cases}$$

$$\Downarrow$$

$$8T(n/2) + n^2$$

$$\Downarrow$$

$$O(n^3)$$

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

So without DAC is better one compare to DAC.

Strassen's matrix multiplication

The decresed number of multiplications from 8 to 7

The increses number of additions from 4 to 16 then the reccurence relⁿ.

सूत्र

$$T(n) = \begin{cases} c & \text{if } n \le 2 \times 2 \\ 7T(n/2) + cn^2 & \\ \Downarrow & \\ O(n^{\log_2 7}) & \\ \Downarrow & \\ O(n^{2.81}) \Rightarrow O(n^{2.32}) & \end{cases}$$

Master Theorem

This method will be applicalole only those reccurence relⁿ in the following form.

$$T(n) = aT(n/b) + f(n)$$
$$= 2T(n/2) + n$$

$$a \geq 1, \quad b > 1$$

Case 1:-

$f(n) = O(n^{\log_b a - \epsilon})$ where $\epsilon > 0$ (greater)

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

Case 2:- $f(n) = \theta(n^{\log_b a})$

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

Case 3:- $f(n) = \Omega(n^{\log_b a + \epsilon})$ $\epsilon > 0$ (greater)

and

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

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

eg

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

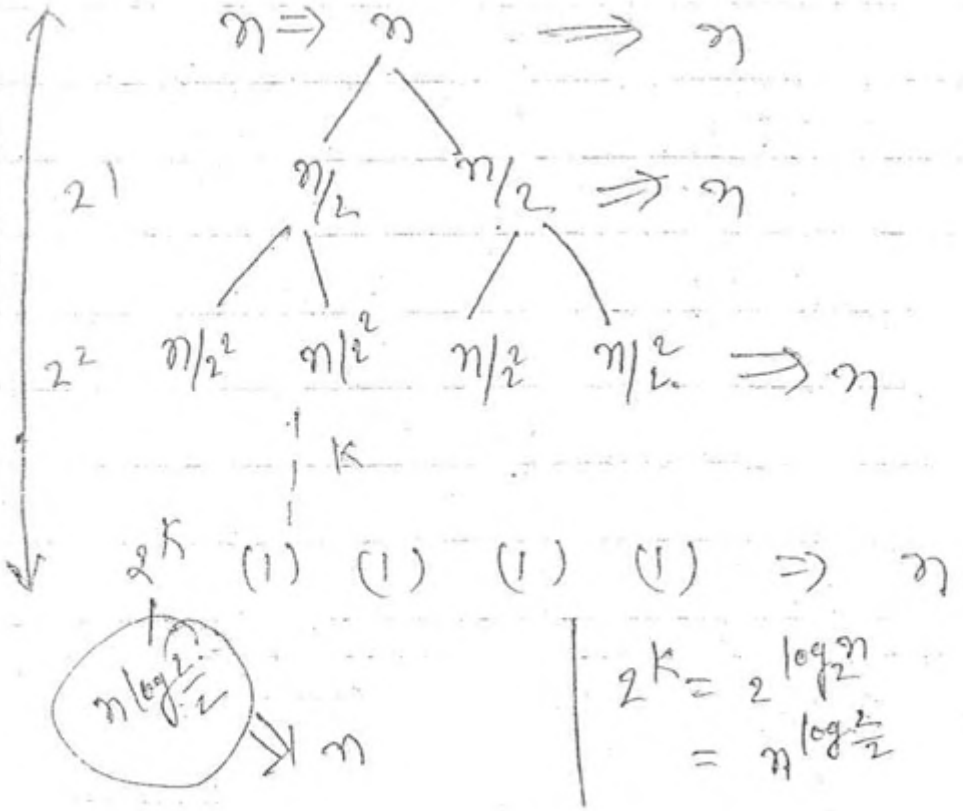
$a = 2$, $b = 2$, $f(n) = n$.

$f(n) = n$.

$n^{\log_b a} = n^{\log 2/2} \Rightarrow n$

Case 2:

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



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

$= n^{\log 2/2}$

$T(n) = 8T(n/2) + n^2$

$a = 8, \quad b = 2$

$f(n) = n^2$

$n^{\log_b^a} \Rightarrow n^3$

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

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

$a = 2, \quad b = 2$

$f(n) = 1$

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

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

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

$a = 4, \quad b = 2$

$f(n) = n^3$

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

Case 3:-

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

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

$a = 2 \quad , b = 2$

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

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

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

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

$a = 1 \quad , b = 2$

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

$f(n) \Rightarrow n$

Case 3:-

$T(n) = \theta(n)$

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

$a = 2$, $b = 1$

it is not in own required format.

Conversion

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

(1) Assume $n = 2^K$

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

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

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

$a = 2$, $b = 2$

$f(K) = K$

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

$K \cdot \log K$

$$O(\log n \cdot \log(\log n))$$

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

(i) n^{2^K}

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

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

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

$a = 1$

$b = 2$

$f(K) = 1$

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

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

$= \theta(\log(\log n))$

** Find time complexity of the following C-Program.

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

{ b = √n ⇒ c
c = A(b) ⇒ T(√n)
return (c) ⇒ c
}.

a) O(n) (b) O(log n), c) O(log(log(n))) d) none

Sol^n

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

$$T(n) = \begin{cases} k & \text{if } n \le 1 \\ T(\sqrt{n}) + c & \text{if } n > 1 \end{cases}$$

$$\Downarrow$$

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

$$\Downarrow$$

$$\log k$$

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

Find time complexity for following C-Program.

```
DAA (n)
{
  if (n<=1) return ;
  else
  {
    return (DAA (n/2) + DAA(n/2) + n) ;
  }
}
```

a = DAA (n/2)
b = DAP (n/2
c = a+b+n.
return (c)
}.

Sol/n

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

$\Downarrow$

$O(n)$

$$T(n) = 2T(n/2) + C$$
$$= 2[2T(n/2^2) + C] + C$$
$$= 2^2 T(n/2^2) + 2^1 + 2^0 C$$
$$= 2^3 T(n/2^3) + 2^2 C + 2^1 C + 2^0 C$$
$$\vdots$$
$$\downarrow k$$

(88) $= 2^K + (n/2^K) + [2^0 + 2^1 + 2^2 + \cdots + 2^{K-1}]$

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

$= n + n$

$\Rightarrow O(n)$

#

```
A(n)
{
if (n<=1) return;
else
return[n * A(√n))
}
```

```
{
a = √n
b = A(a)
c = n * b
return(c);
}
```

Soln

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

$\Downarrow$

$O(\log(\log_2 n))$

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

$\Downarrow$

$O(n^2)$

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

```
Sum = 0
for ( i = 1 to n ) {
    Sum = Sum + i
```

$\Downarrow O(n)$

---

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

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

$\Downarrow n \quad \Downarrow n \quad \Downarrow n$

$\underbrace{\qquad\qquad\qquad\qquad}_{n}$

*
```
fact (n)
{
if (n<=1) return 1
else
return (n * fact (n-1))
}
```

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

Cube root

```
for ( i = 1 to 1000)
{
if ( i^3 ≤ 1000)
    else
    return (i-1)
}
```

⇒ (10)

$m^3 < n$

Conclusion

1) Binary Search ⇒ $T(n/2) + c \Rightarrow O(\log n)$

2) Power of an element ⇒ $T(n/2) + c \Rightarrow O(\log n)$.

3) Find Max, Min ⇒ $2T(n/2) + 2 \Rightarrow O(n)$

4) Merge sort ⇒ $2T(n/2) + n \Rightarrow O(n \log n)$

5) Quick sort ⇒ $2T(n/2) + n \Rightarrow \Omega(n \log n)$

$T(n-1) + n \Rightarrow O(n^2)$

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

7) Matrix multiplication $\Rightarrow$
- $8T(n/2)+n^2 \Rightarrow O(n^3)$
- $7T(n/2)+n^2 \Rightarrow O(n^{2.8})$ (Strassen)

Insertion Sort

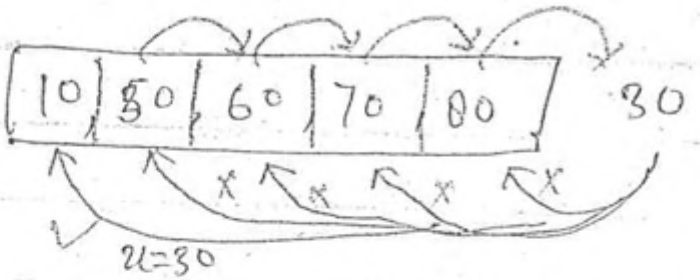
Eg 10, 50, 60, 70, 80, 30, 40, 90, 100

(1) 10 50

(2) 10 50 60

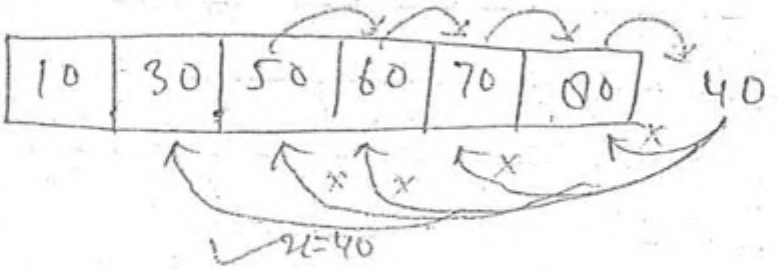
(3) 10 50 60 70

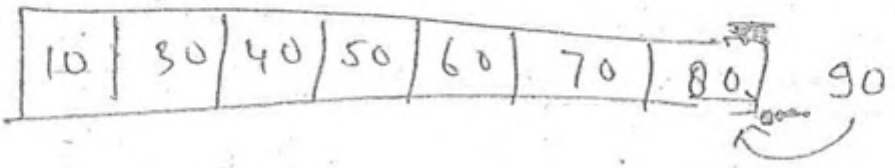
(4) 10 50 60 70 80

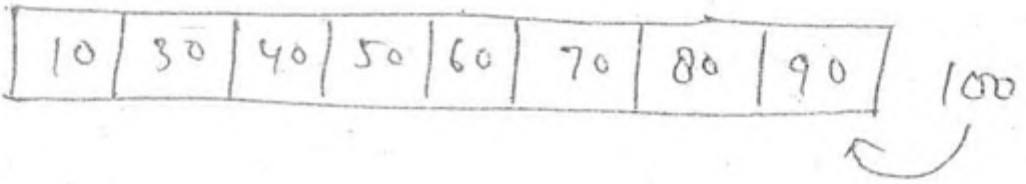
(5) 

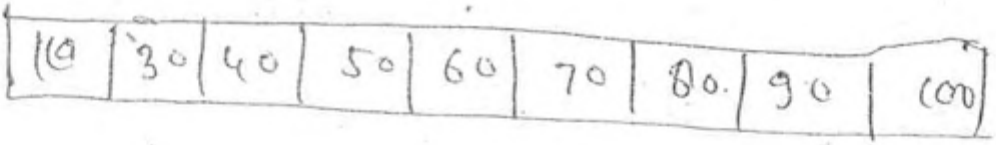
3)

2.0)



(6) 

(7) 

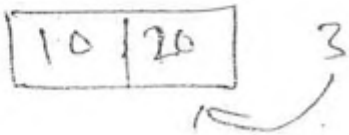
(8) 

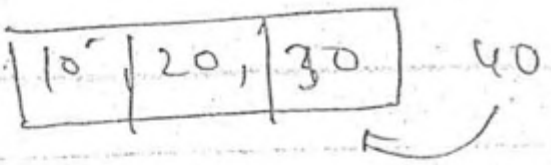
Best case

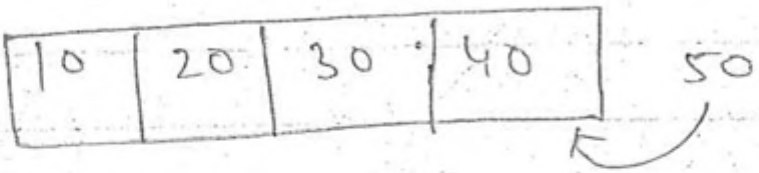
Increasing order

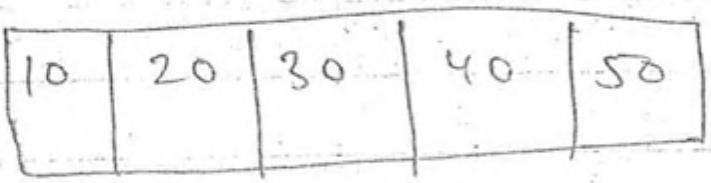
10, 20, 30, 40, 50

(1) 

(2) 

3) 

4) 

5) 

Best case of T.C of Insertion sort is $\Omega(n)$ in $(n-1)$ time (bcm)

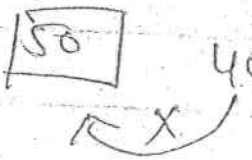
Note For smaller size of input Insertion sort is Better.

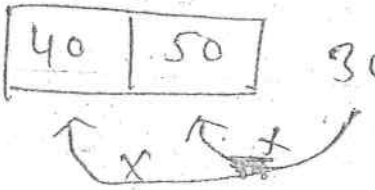
For greater " " " Merger

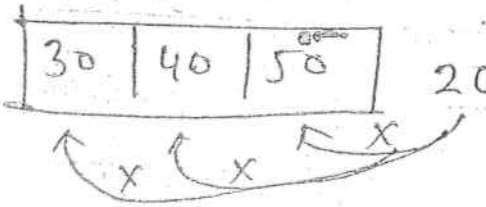
" " Better.

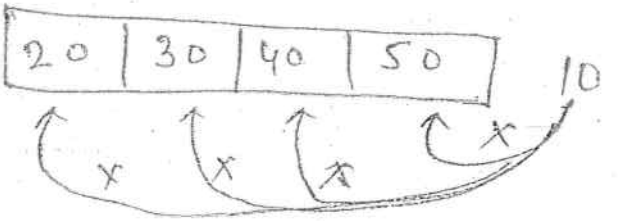
worst case

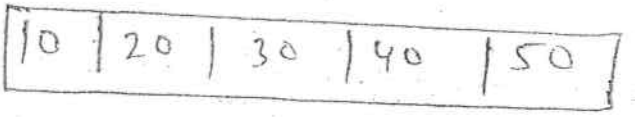
50, 40, 30, 20, 10

1) 

2) 

3) 

4) 



Comparision	Swapt
$1 \Rightarrow n$	$1 \Rightarrow n$
$2 \Rightarrow n$	$2 \Rightarrow n$
$3 \Rightarrow n$	$3 \Rightarrow n$
$4 \Rightarrow n$	$4 \Rightarrow n$
1	1
!	!
$n-1$	$n-1$
$\downarrow$	$\downarrow$
n	n

total Comparision $O(n)^2$

" swapt $O(n)^2$

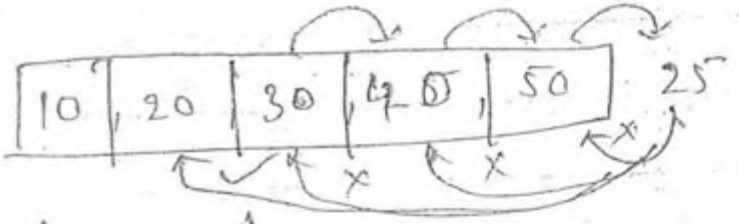
$\Rightarrow n^2 + n^2$

$O(n^2)$

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

worst case T.C

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



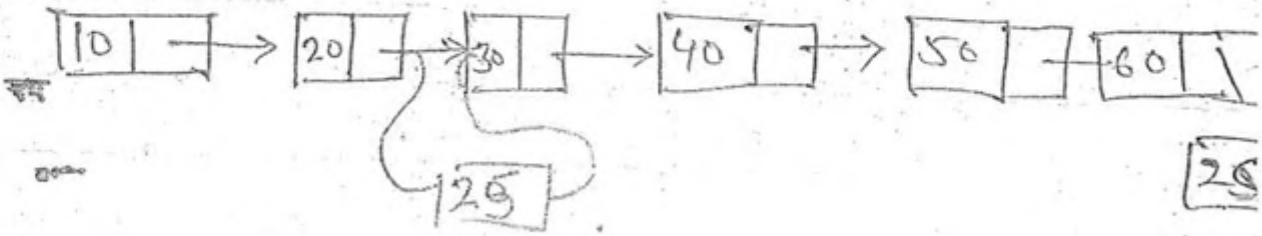
Sol^n

Comparision	Swapt
$1 \Rightarrow \log n$	$1 \Rightarrow n$
$2 \Rightarrow \log n$	$2 \Rightarrow n$
$3 \Rightarrow \log n$	$3 \Rightarrow n$
⋮	⋮
$n-1 = \log n$	$n-1 \Rightarrow n$

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

total swapt = $O(n^2)$ $\Rightarrow$ $n \log n + n^2$

$O(n^2)$



Comparsion	swapt
$1 \Rightarrow n$	0
$2 \Rightarrow n$	0
$3 \Rightarrow n$	0
!	0
!	0
$n-1 \Rightarrow n$	
n^2	0

$n^2 + 0$

$O(n^2)$

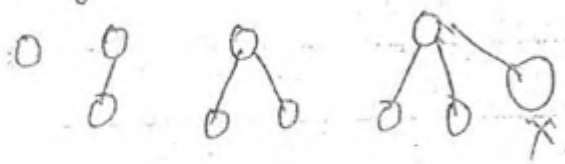
Avrge case also leads to $O(n^2)$

Tuesday

Imp

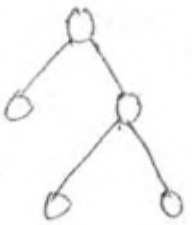
Heap sort

1) Binary tree

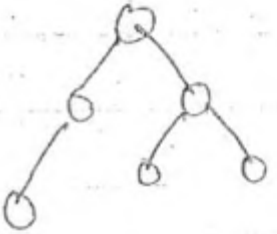


2) Strict Binary tree (Full binary tree) in the given binary tree at every node exactly zero or two children.

eg



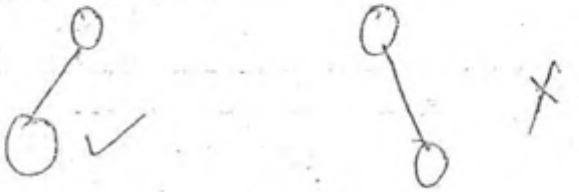
BT ✓
SBT ✓



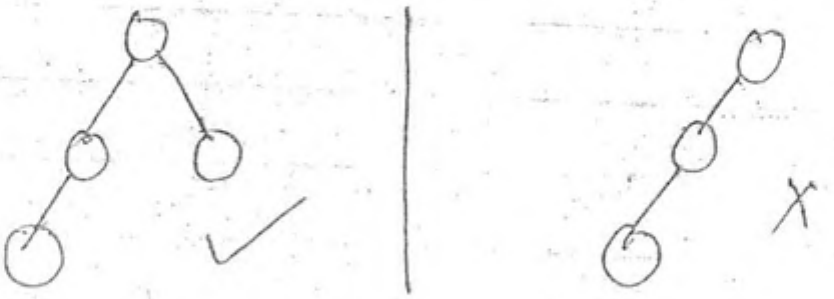
BT ✓
SBT ✗

(3) Almost Complete Binary tree :-

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

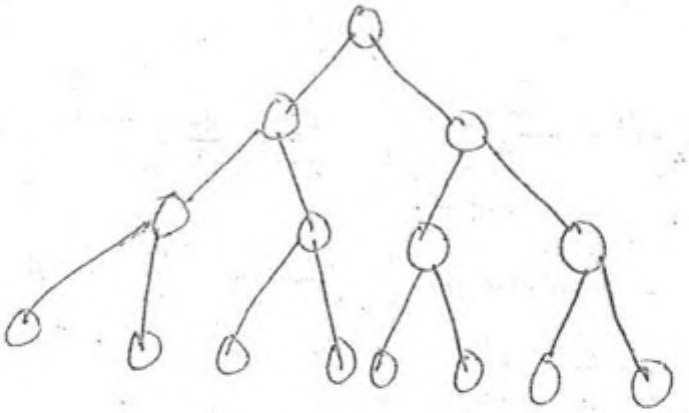


2) at every node after completing present level completly then only go to next level

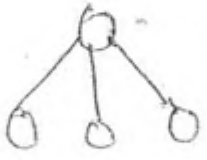


4) Complete binary tree :-

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

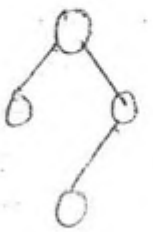


BT ✓
ACBT



BT ✗

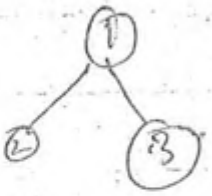
BT ✓
ACBT ✓
CBT ✗



BT ✓
ACBT ✗
CBT ✗

Q. Construct almost complete binary tree for no^{s} 1, 2, 3.

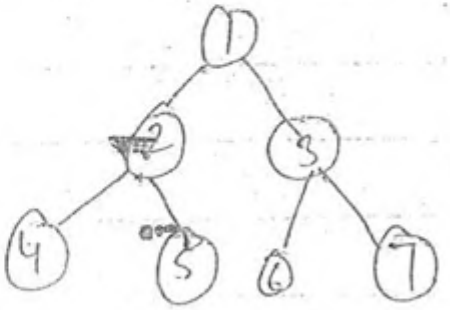
$n = 3$ $2^2 - 1$



$\log_2 3 \Rightarrow 2$

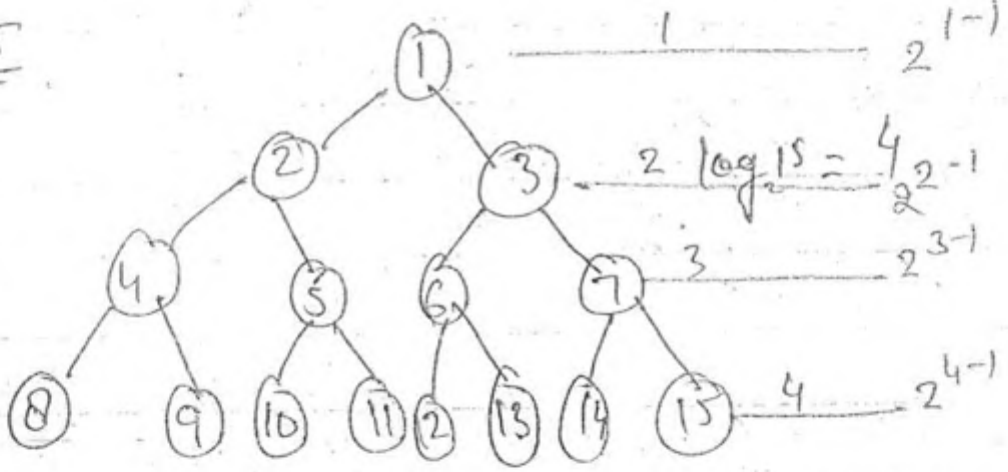
$n = 7$

$2^3 - 1$



$\log_2 7 \Rightarrow 3$

$n = 15$



all are BT, CBT, ACBT

// The max^{m} 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) = $\boxed{2^i - 1}$

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

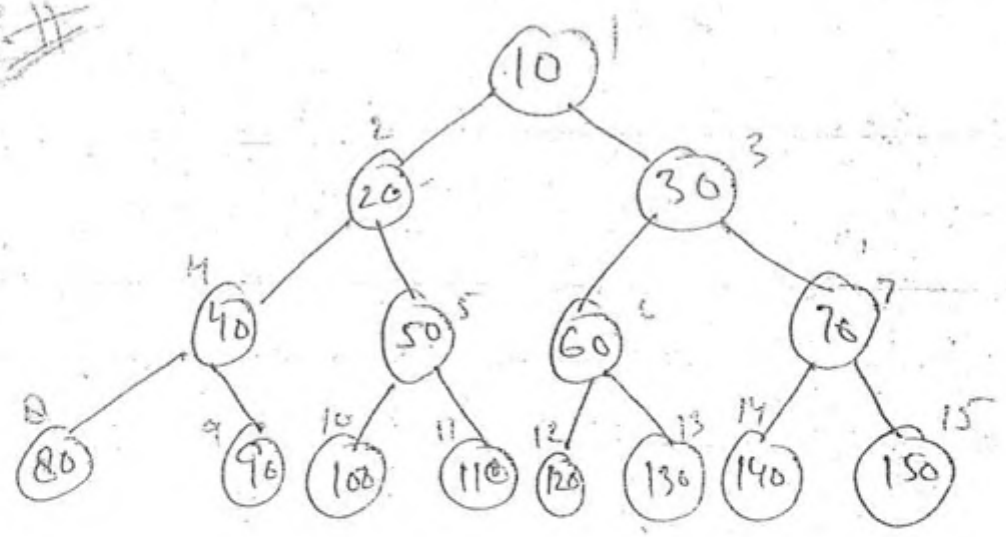
$$n = 2^i - 1$$

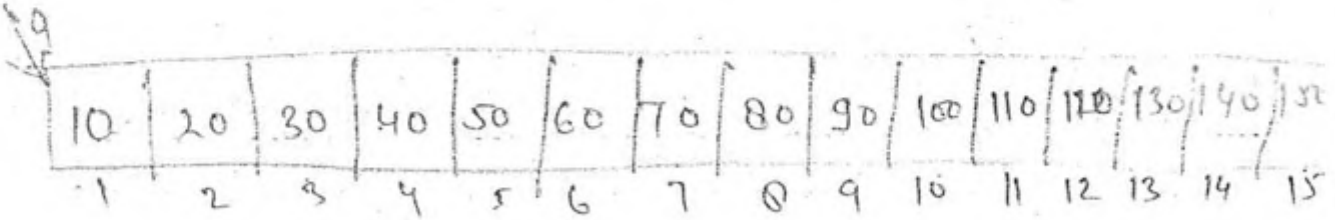
$$2^i = n + 1$$

$$\boxed{i = \log_2(n+1)}$$

where i is level

Eg:-

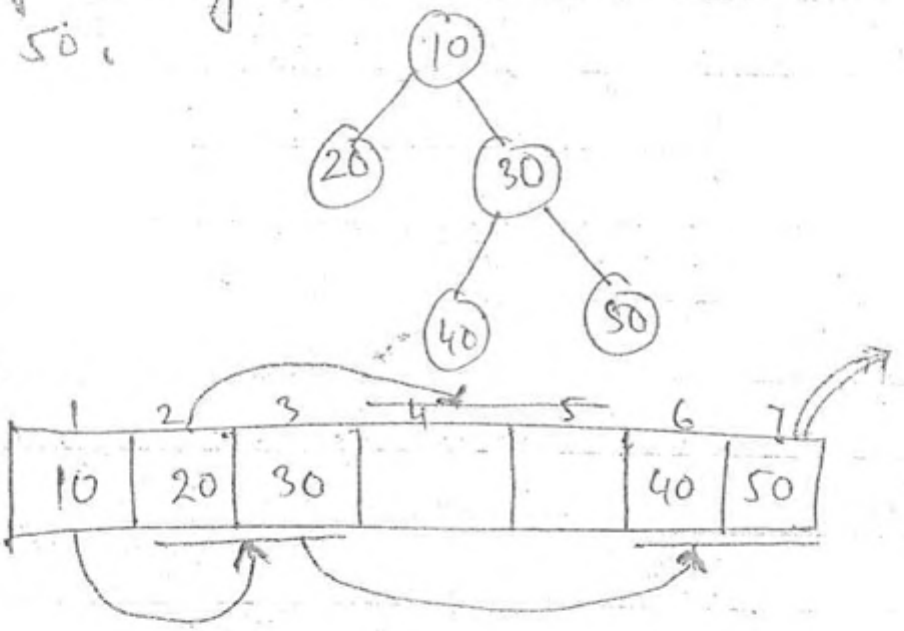




the node present at ith replace

left (i) = 2 * i
right (i) = (2*i)+1
Parent (i) = $\left[\frac{i}{2}\right]$

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



Soln

7

Heap Sort

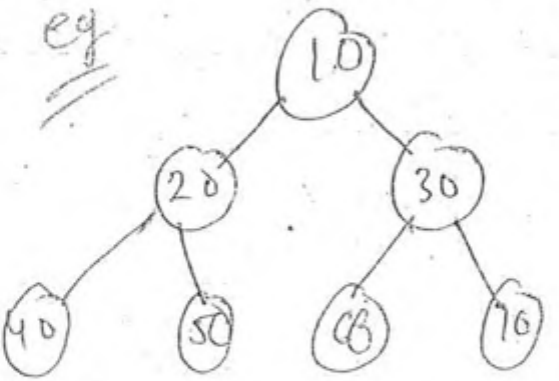
Heap tree → An almost complete binary tree is said to be Heap tree iff.

It should satisfy heap property.

two types of heap property.

Min heap	Max heap
In the almost complete binary tree at every node present is min^m comparing their children is called min heap tree	In the almost complete binary tree at every node present is maximum comparing their childrenm is called Max heap tree.

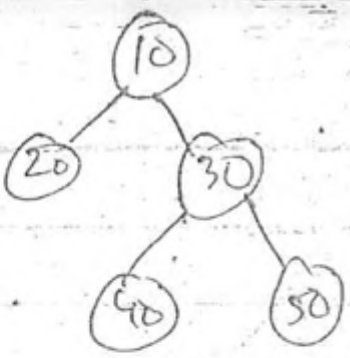
eg



ACBT ✓

Min ✓

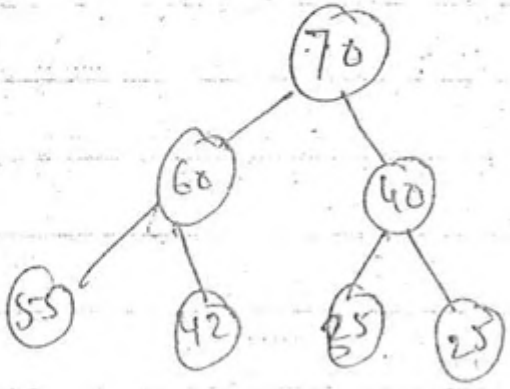
eg



ACBT X

Min X

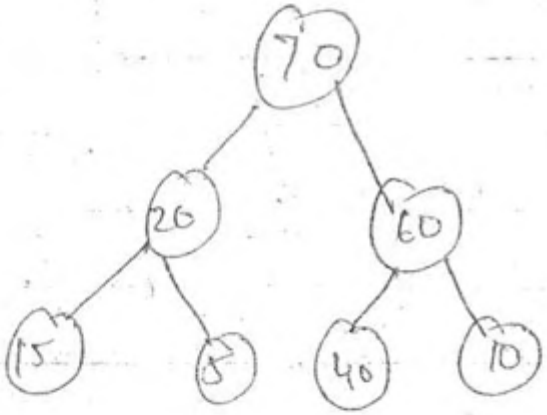
eg



ACBT ✓

Max ✓

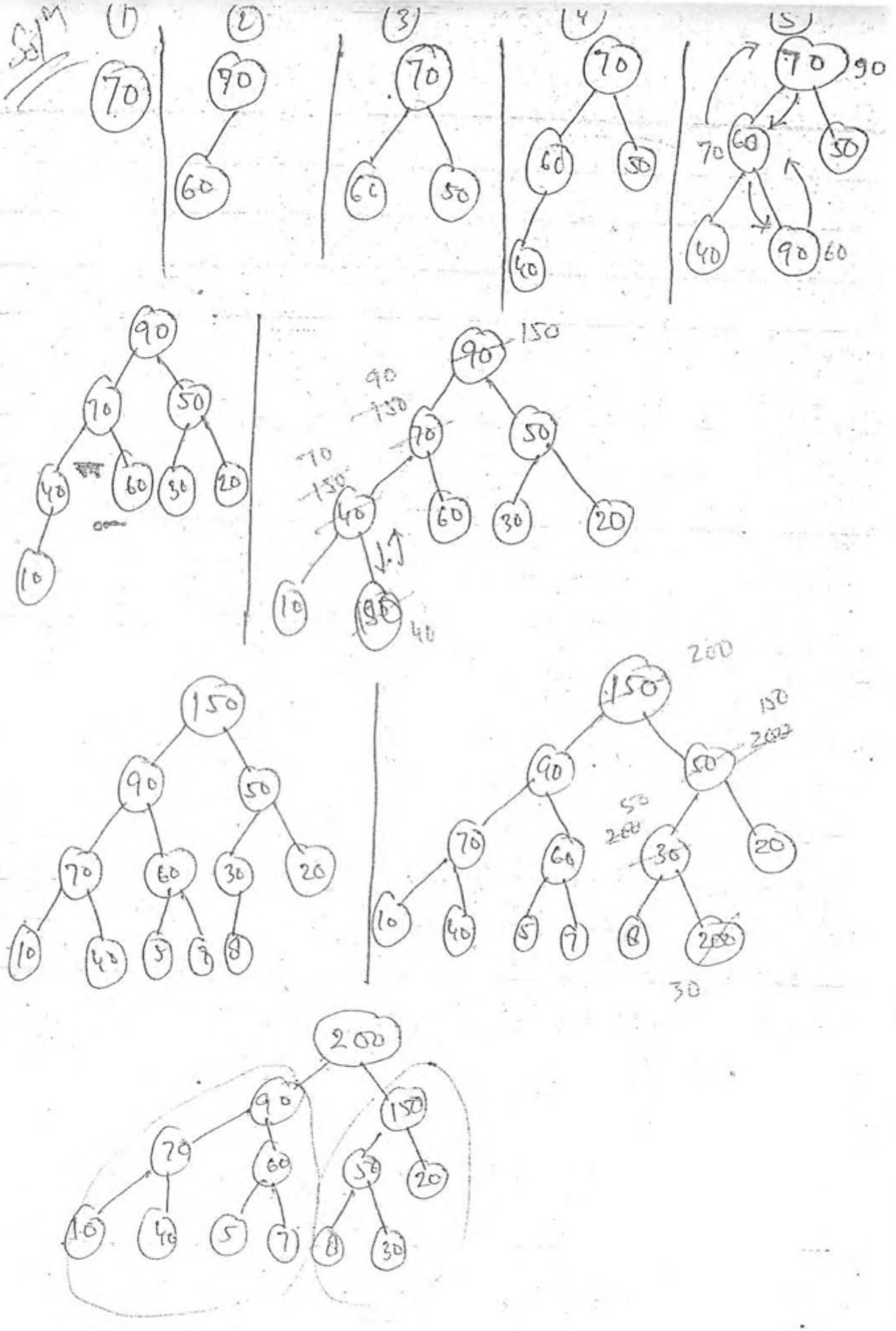
eg



ACBT ✓

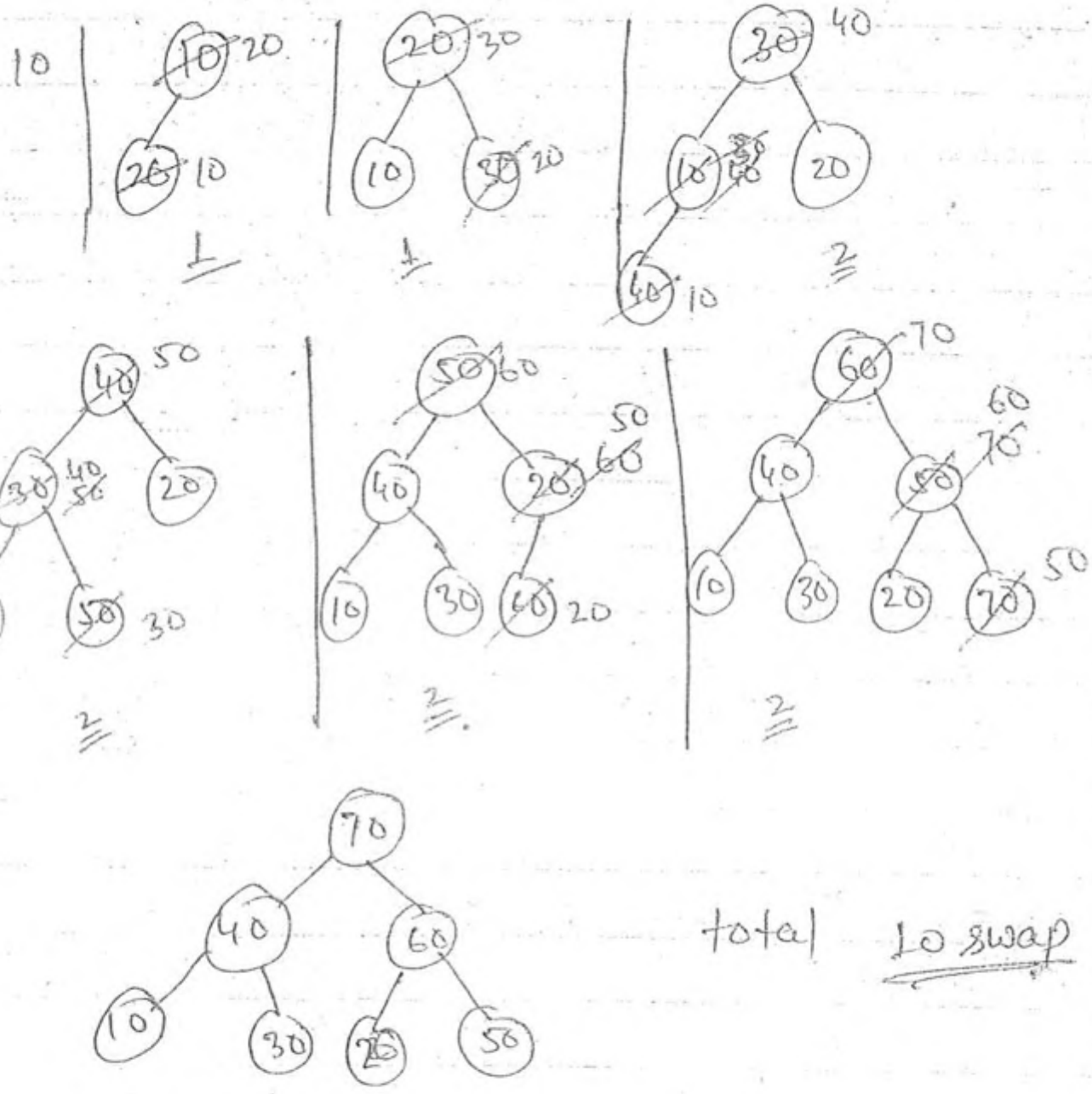
max ✓

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

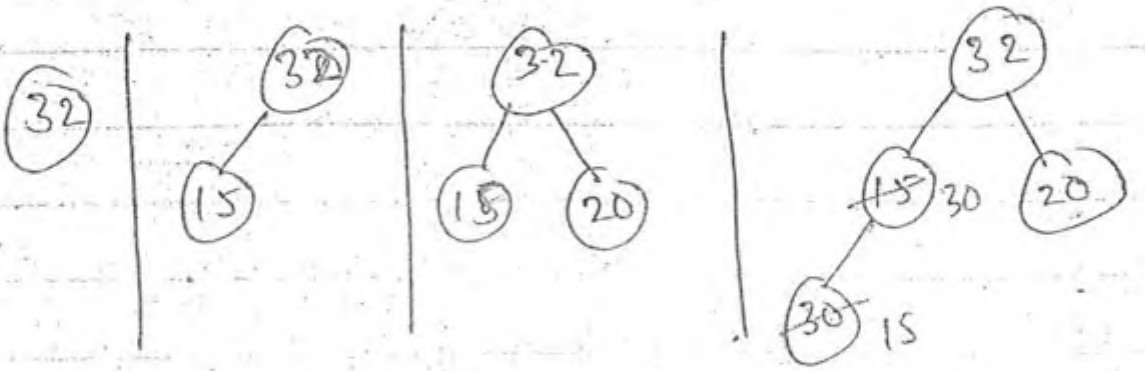


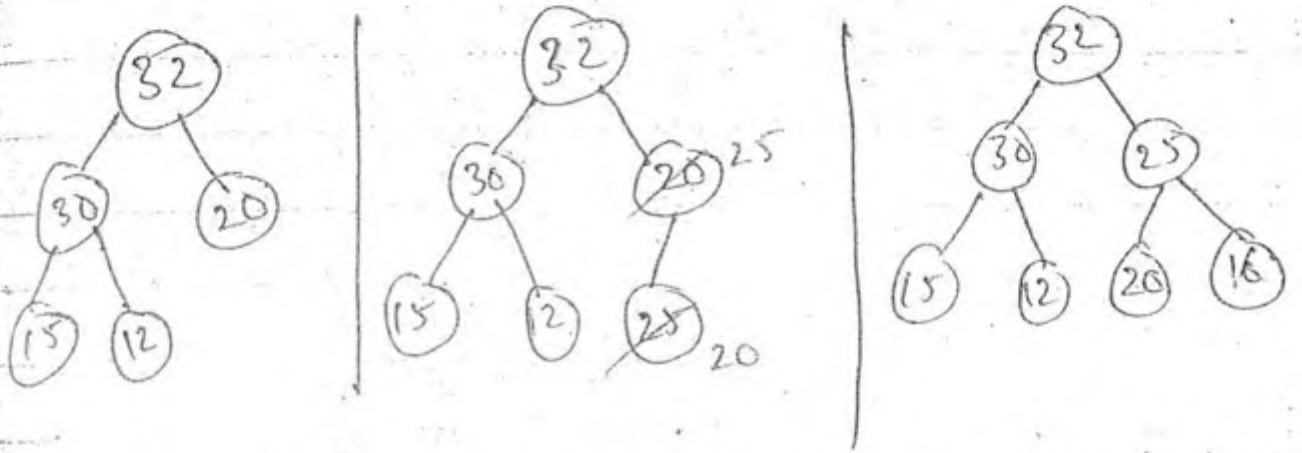
#

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



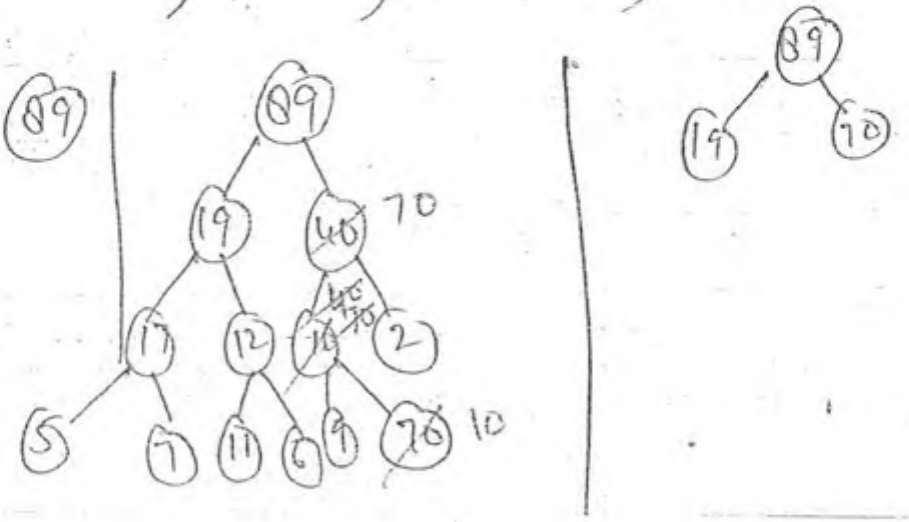
* The elements: 32, 15, 20, 30, 12, 25, 16 are inserted one after another into a max heap. The resultant max heap tree is.



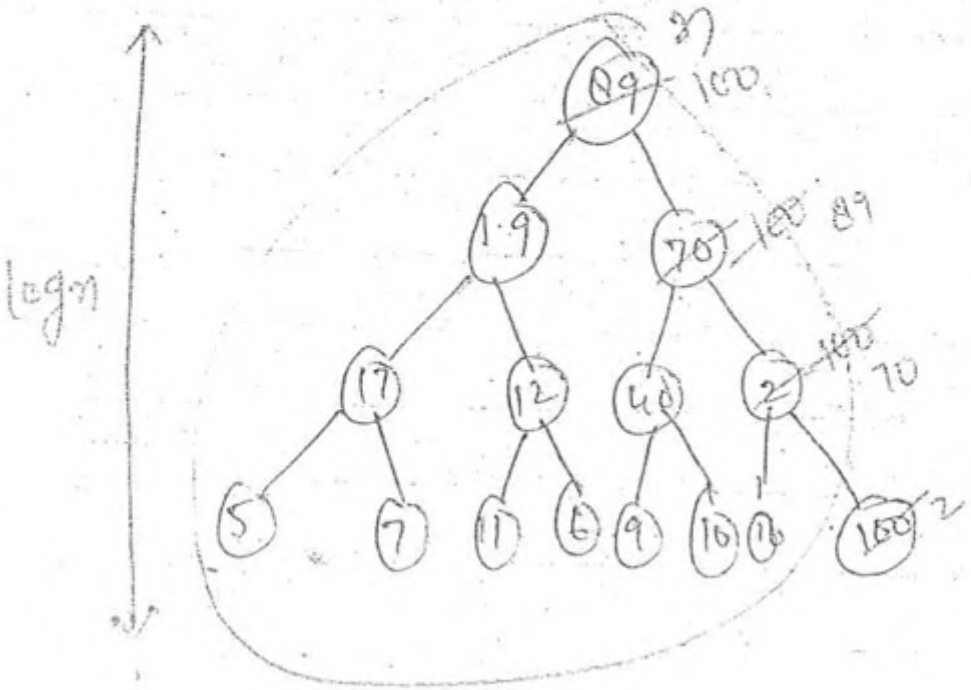


The min^m no of interchanges needed to convert the array. 89, 19, 40, 17, 12, 10, 2, 5, 7, 11, 6, 9, 70 into heap. with max^m element at the root is

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



Q How much time it will take to insert an element into max^m or min heap which already contains n elements.



Max heap ~~$\Omega(1)$~~

Best case $\Rightarrow \Omega(1)$

eg. Let we have to insert 1

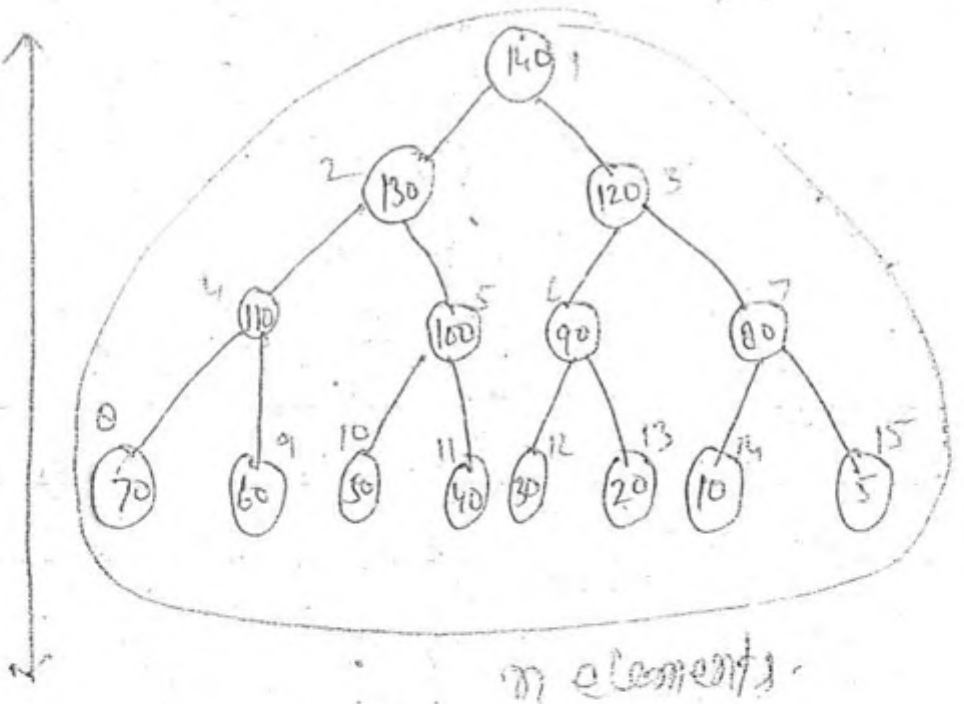
Worst case $\Rightarrow O(\log n)$

Let we have to insert 100 then all parent will be affected. Hence total no of. of swapping = no of level in tree.

Note

In order to insert an element into min heap or max heap will take order of $O(\log n)$ times $\Rightarrow$ $O(\log n)$ worst case

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



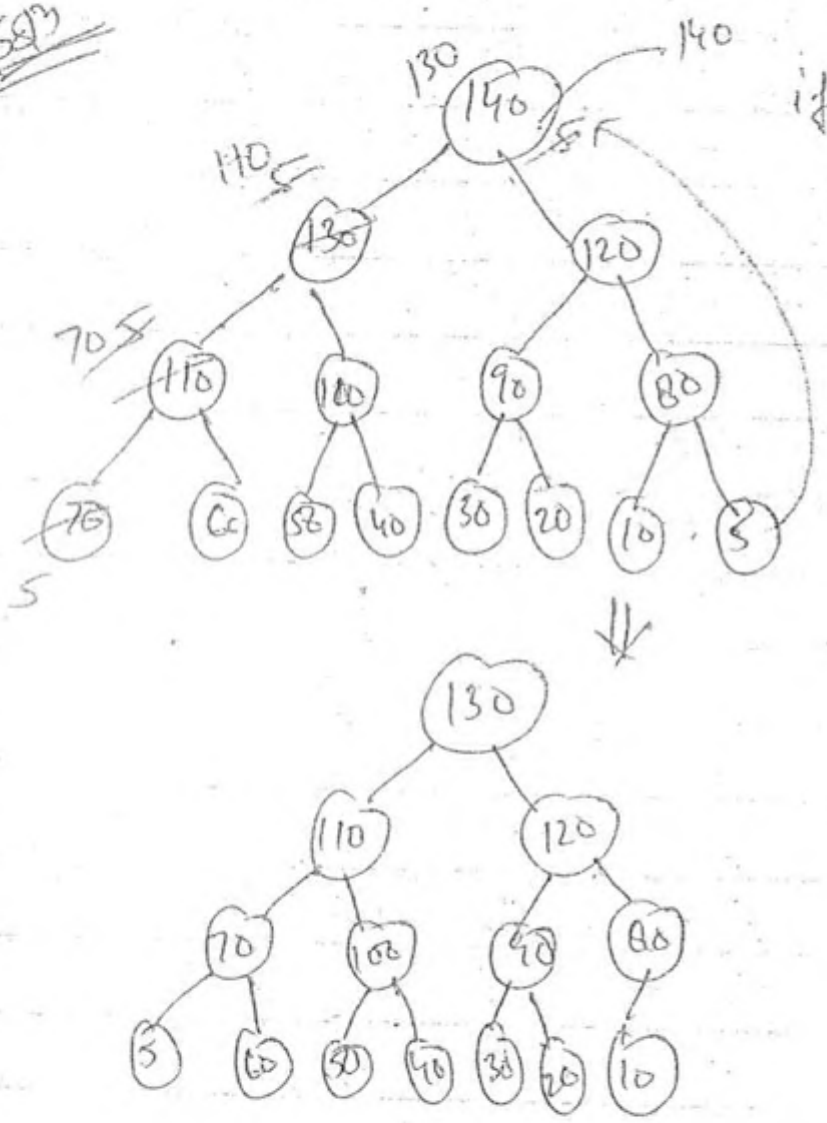
a.

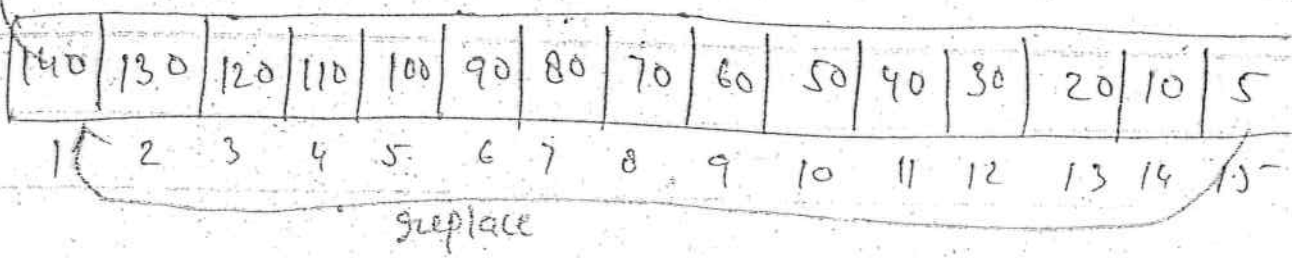
140	130	120	110	100	90	80	70	60	50	40	30	20	10	5
1	2	3	4	5	6	7	8	9	10	11	12	13	14	

Jan 19

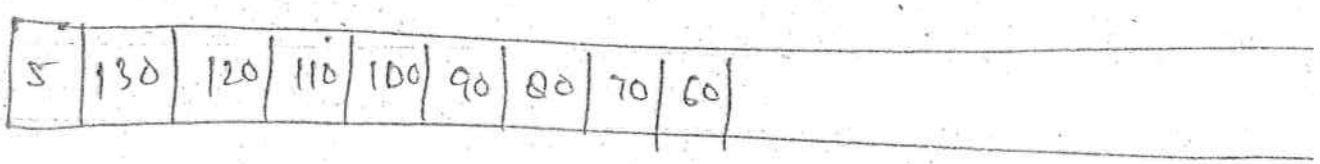
Note ⇒ In How to delete an element from min heap or max heap which contains already n elements requires order of $\log n$ times.

Soln





To delete 140 replace it with 5

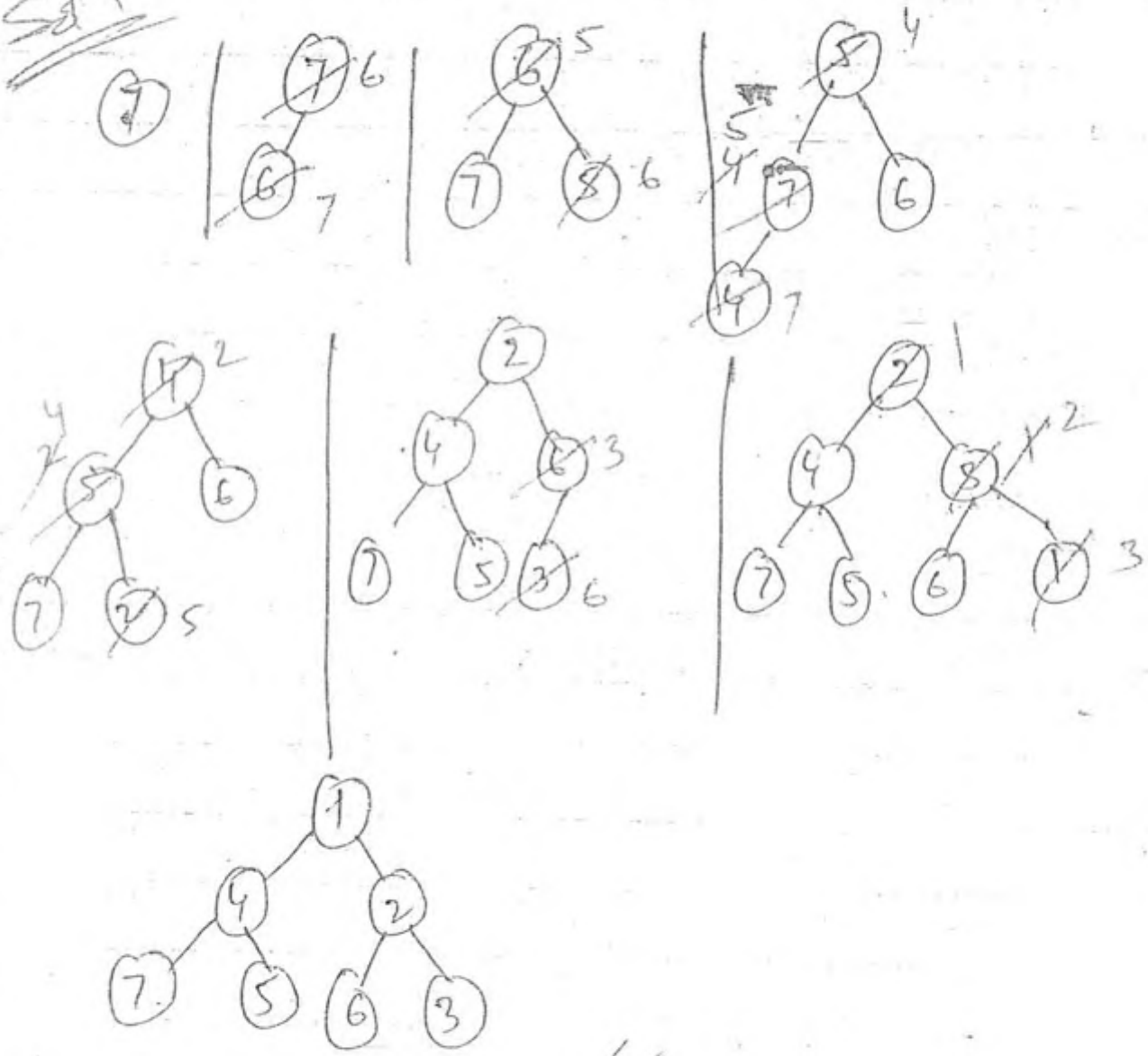


Q. Draw the mean heap tree that results from insertion of the following elements one after another into empty min heap.

7, 6, 5, 4, 2, 3, 1.

Show the resulted min heap tree after deleting root from the min heap.

Soln



P1 Consider the following max heap implemented using array.

P2 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

P2 What are the contents of the array after deleting on the correct answer to the about problem.

a) 14, 13, 120, 10, 8

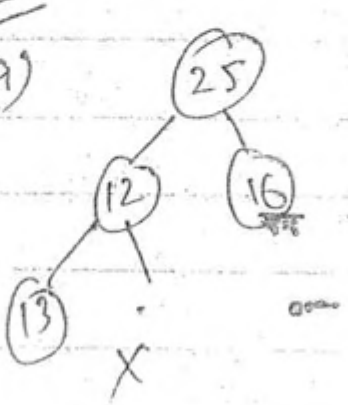
b) 14, 12, 13, 8, 10

c) 14, 13, 8, 12, 10

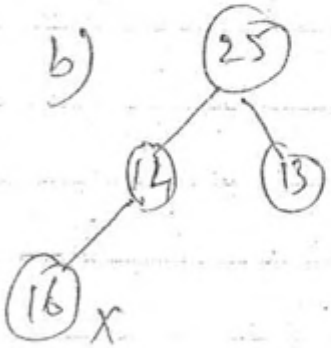
d) 14, 13, 12, 8, 10

Sol^{n} P.

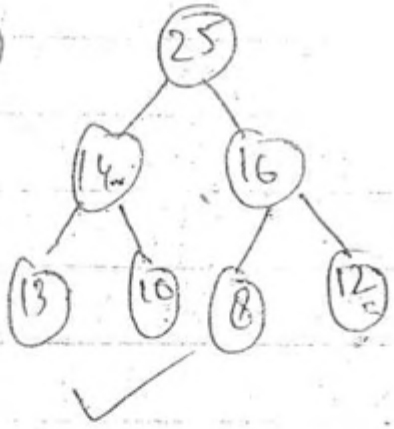
(a)



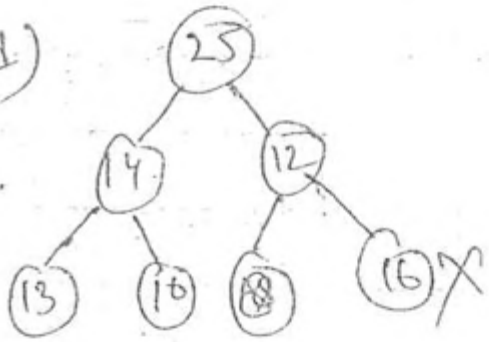
b)



c)



d)



Sol^{n} P.

Q1 A 3-Array max-heap is like a binary max heap but insert of 2-children nodes. have 3-children.

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

a) 1, 3, 5, 6, 8, 9

b) 9, 6, 3, 1, 8, 5

c) 9, 3, 6, 8, 5, 1

d) 9, 5, 6, 8, 3, 1

Q2 Suppose the elements 7, 2, 10 & 4 are inserted in the order into valid 3-array max heap found in the about problem which is true.

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

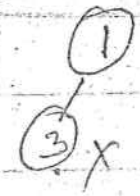
b) 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

c) 10, 9, 4, 5, 7, 6, 8, 2, 1, 3.

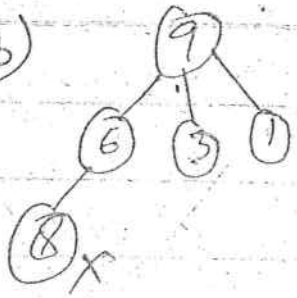
d) none.

Sol^n

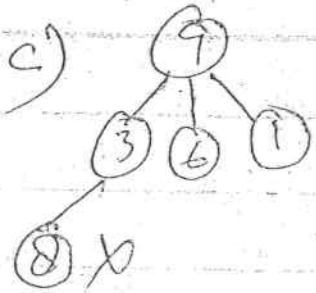
a)



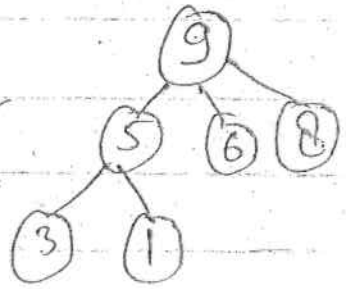
b)



c)



d)



Heap Sort (Assending order)

(1) create max heap tree (heap tree) by inserting one after another (n) — $\log n$

$\Rightarrow O(n \log n)$

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

$O(n \log n)$

$\Rightarrow \Theta(\log n)$

Min Heap

(1) Finding min^n $\Rightarrow O(1)$

(2) Deleting min^n $\Rightarrow O(\log n)$

(3) eg: Getting 100^{th} smallest element

$\Rightarrow (100 \log n) \Rightarrow O(\log n)$

(4) Getting max^m element

$\Rightarrow O(n)$

Max heap

1) Finding Max^m $\Rightarrow O(1)$

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

3) Getting 100th max^m element

$100 \log n \Rightarrow O(\log n)$

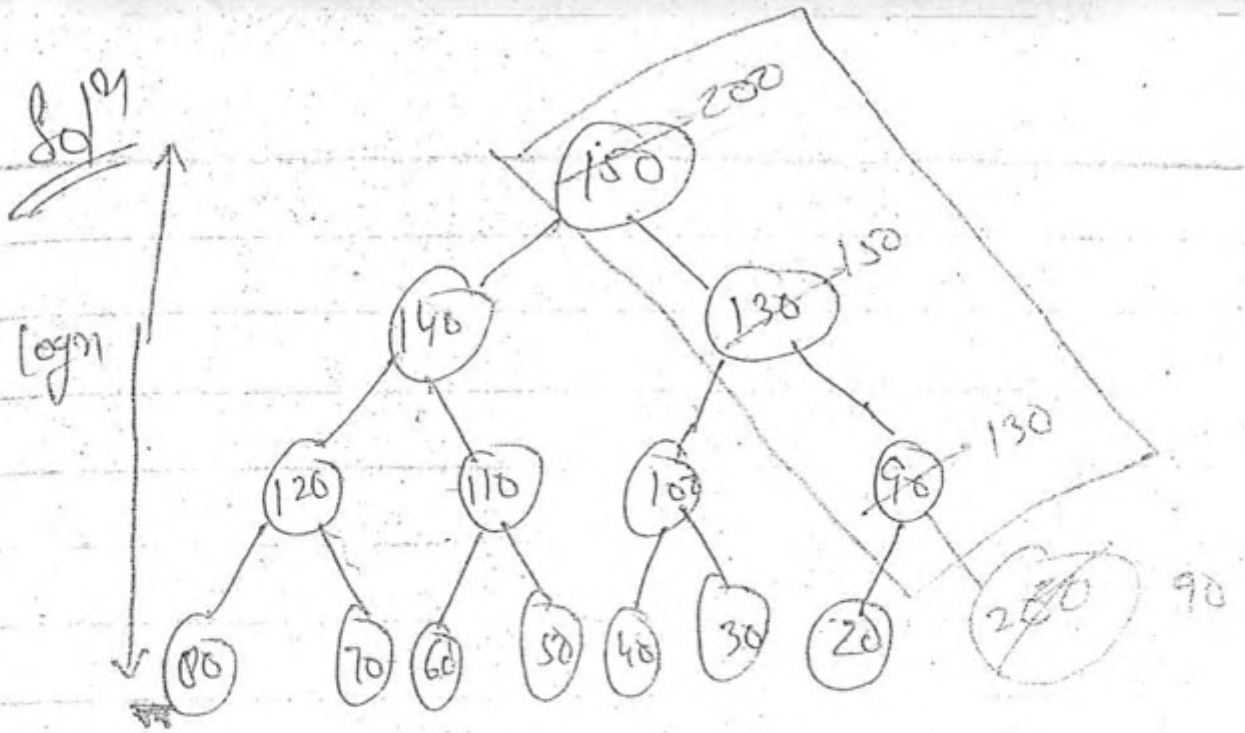
4) Getting min^m element $\Rightarrow O(n)$

P Consider the process of inseting n elements into maxheap where max heap is representing using array.

Suppose we peform a binary search on the path from new leaf to root to find the position for the newley inserted element, The T.C is.

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

c) $O(n)$ d) $O(n \log n)$



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

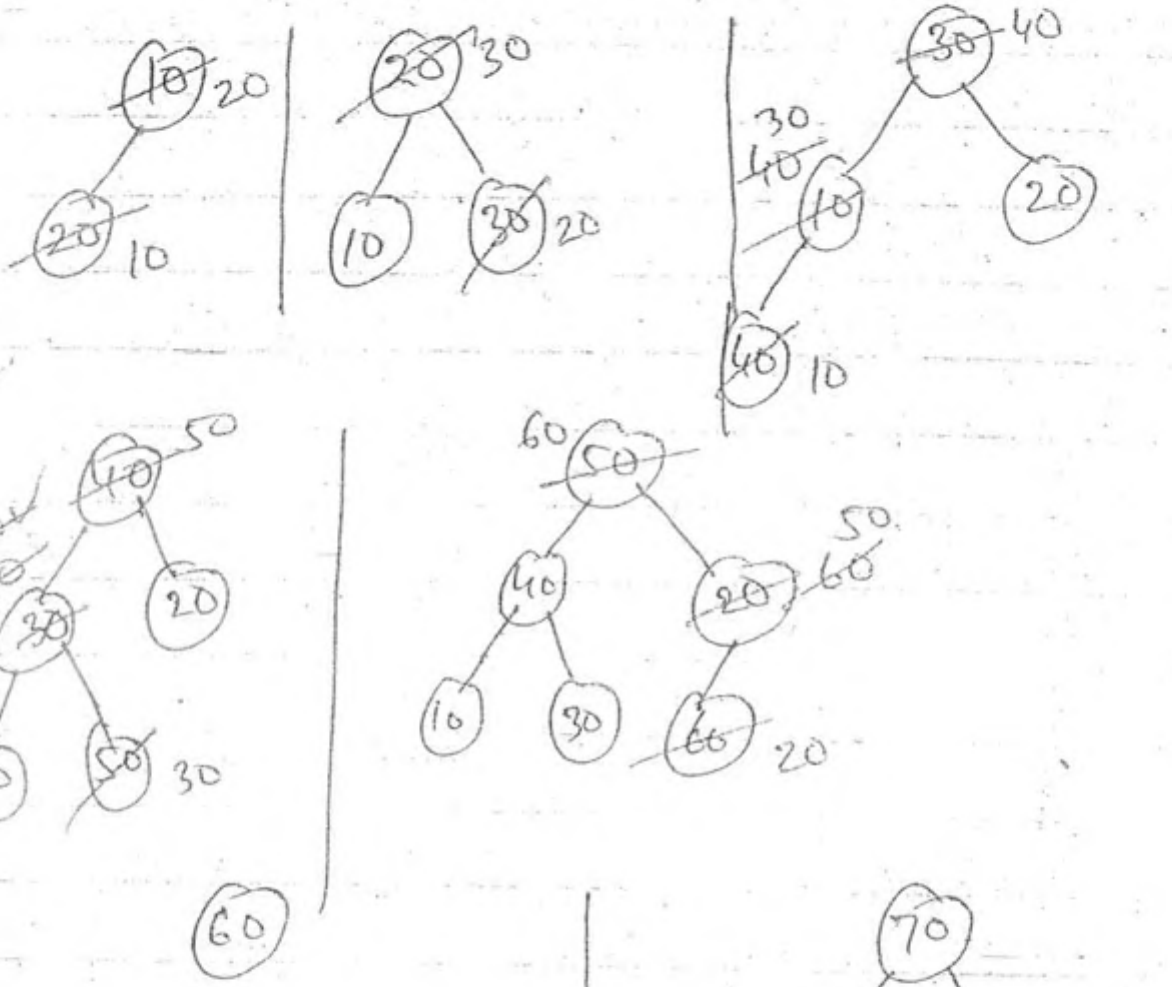
$\Rightarrow O(\log n)$

Binary search is not Possible.

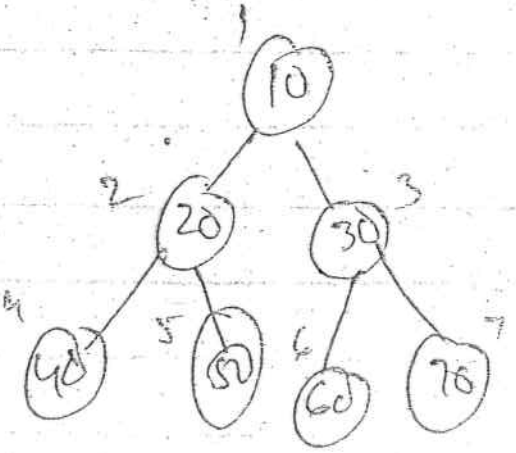
Build Heap Method

Note:- Using Build heap method we can create Heap tree which contain n-elements with $O(n)$ times.

P/ Create maxheap tree for the following elements using build heap method.
10, 20, 30, 40, 50, 60, 70

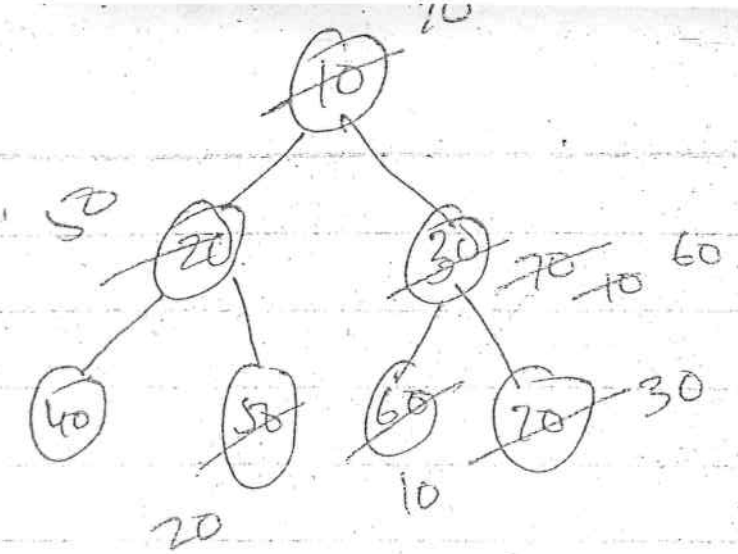


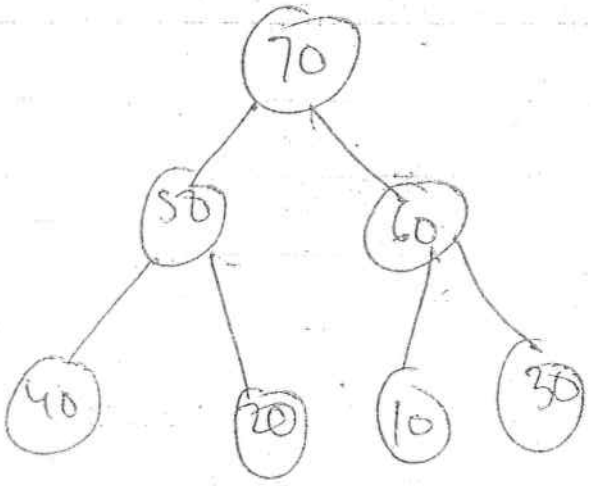
Build heap metho




```
Buildheap(a, n)
{
for(i [n/2] down to 1)
max_heapify.(a, i)
```

```
Maxheapify(a, i)
{
  l = left(i)
  r = right(i)
  ~~max=i;~~
  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)
}
```

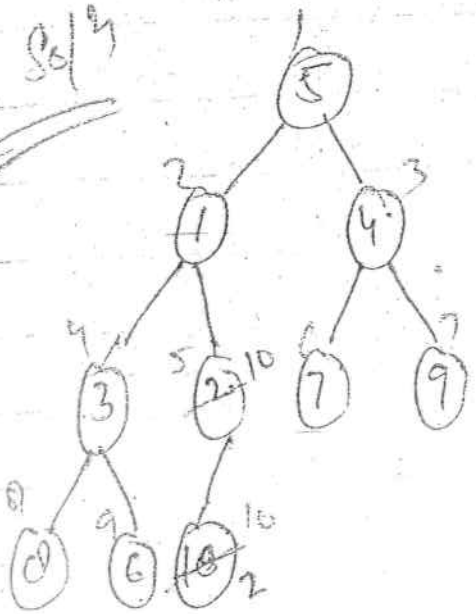




Q/ Construct max heap tree for the following elements using Build heap method.

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

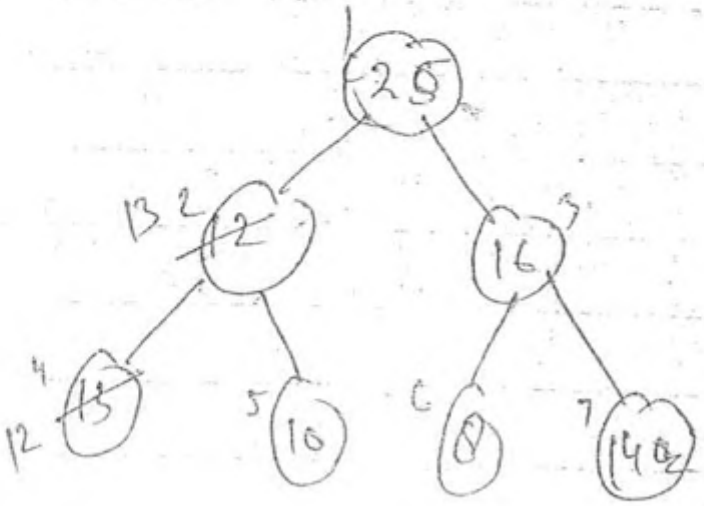
Sol/



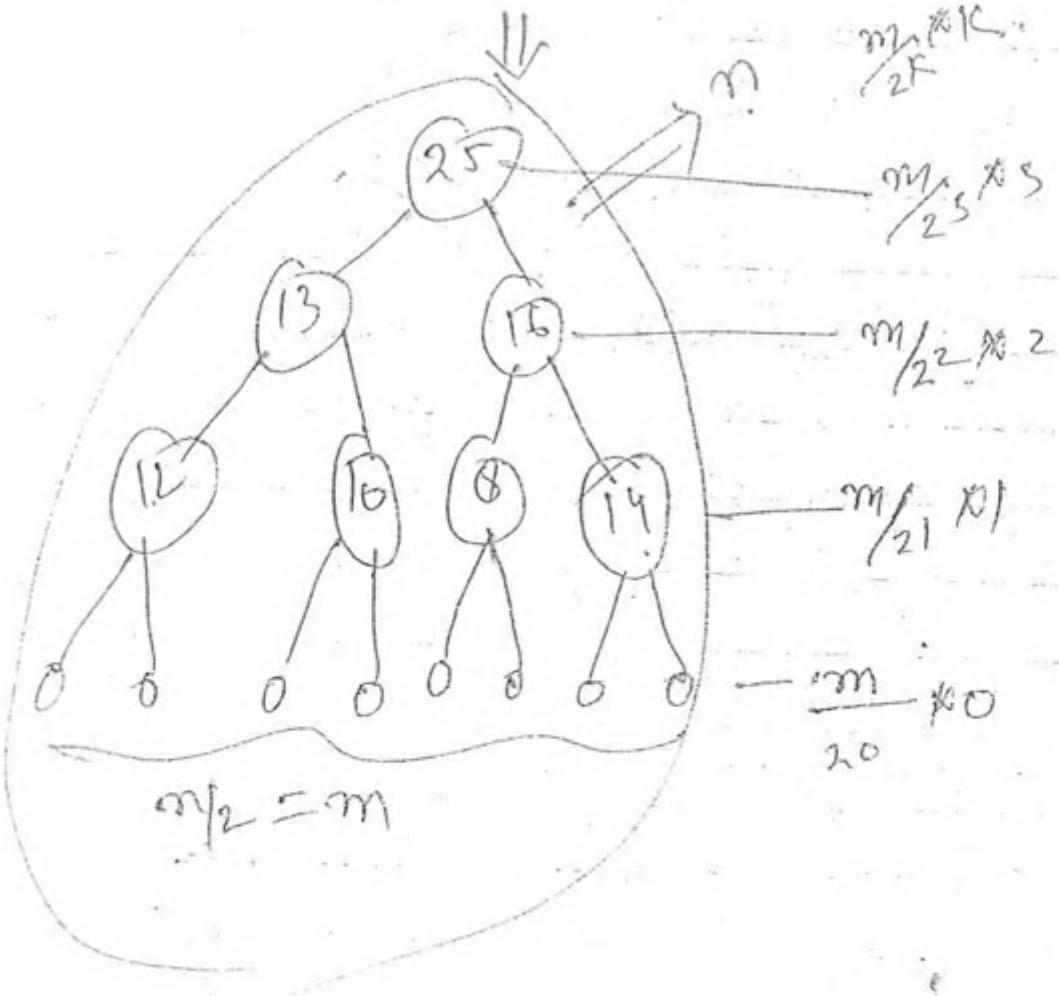
$\rightarrow i = n/2 = \frac{10}{2} = 5$

P Construct a maxheap tree for the following using Build heap method.

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



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



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

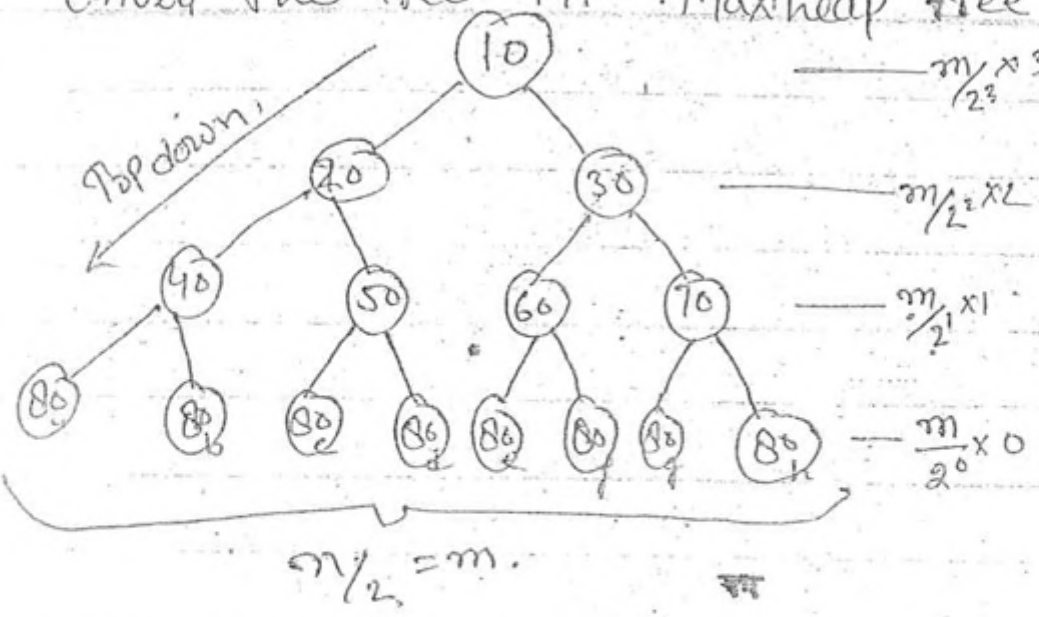
$$= n\left[\frac{0}{2^0} + \frac{1}{2^1} + \frac{2}{2^2} + \frac{3}{3^3} + \frac{4}{4^4} + \cdots \frac{K}{2^K}\right]$$

$$= n\,[\text{Constant}]$$

$$= n/2 * \text{Constant}$$

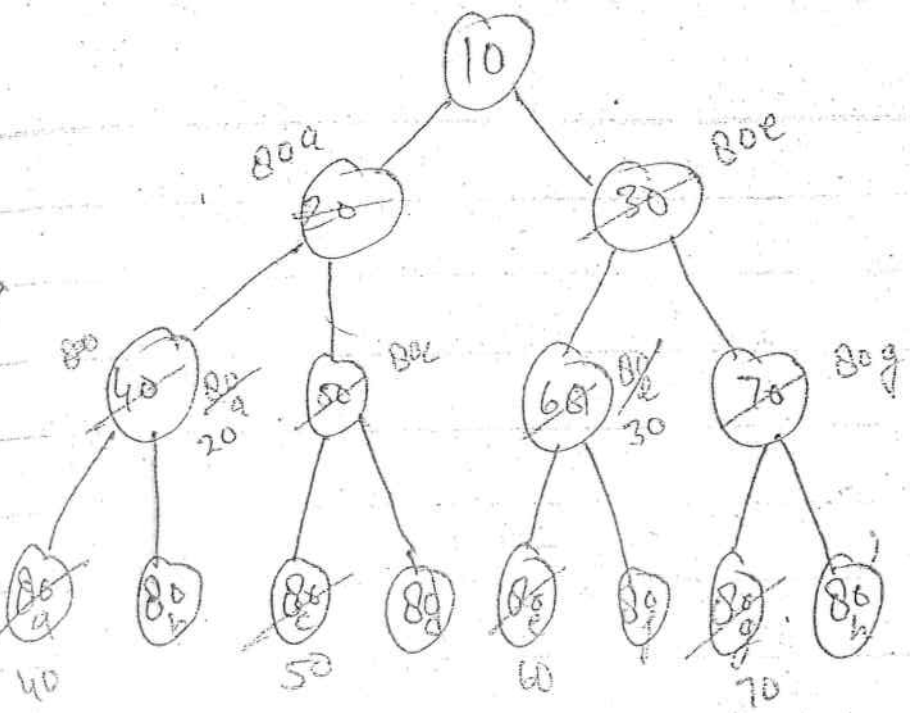
$$= O(n)$$

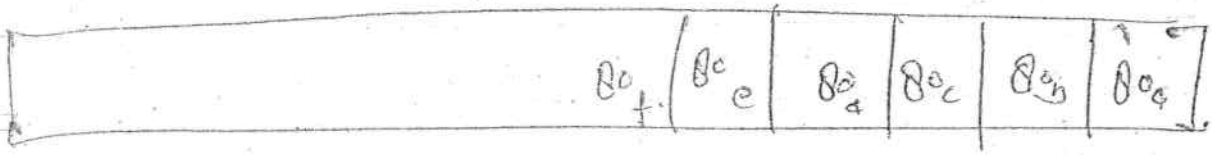
P// Convert the tree into Maxheap tree –



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

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



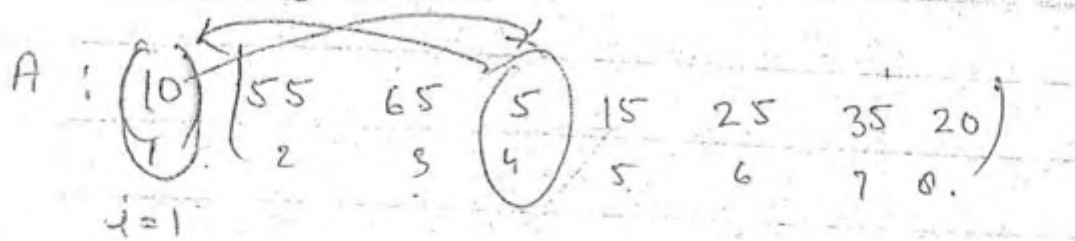


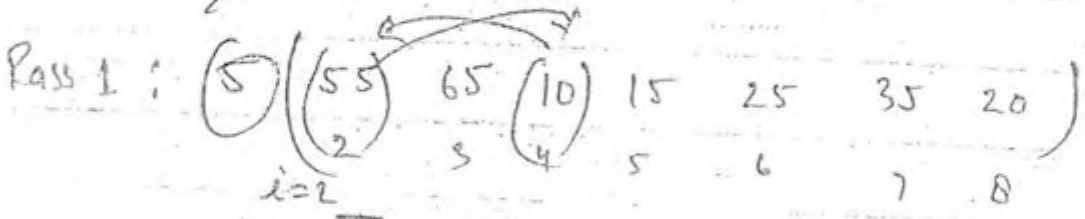
1) Heap sort not ~~stable~~ stable sorting technique.

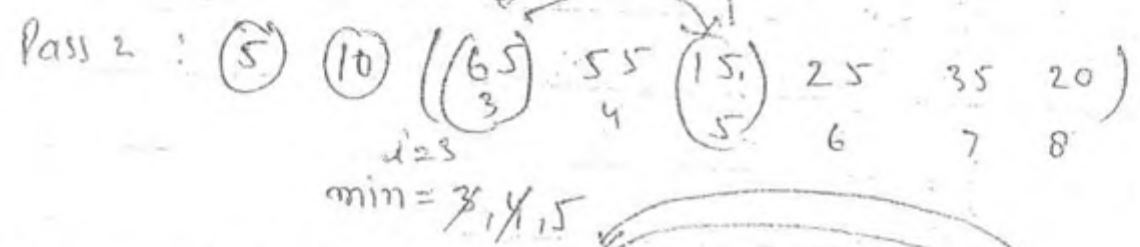
2) In-place

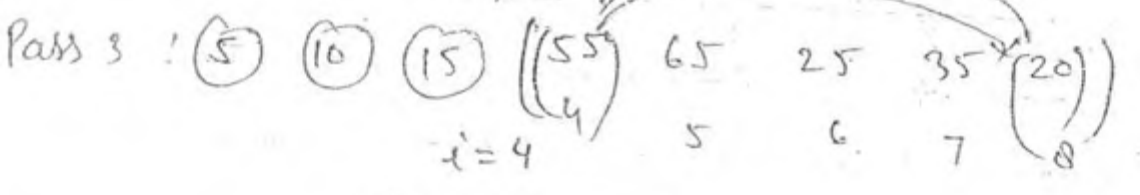
3) ~~Construct~~.

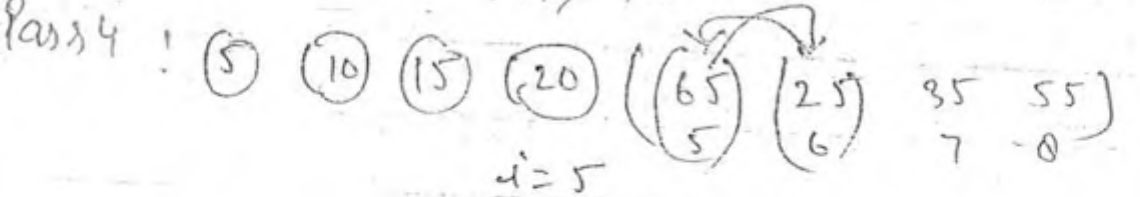
Selection Sort

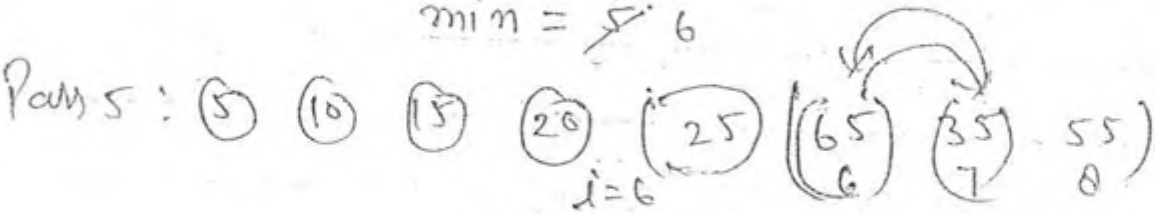


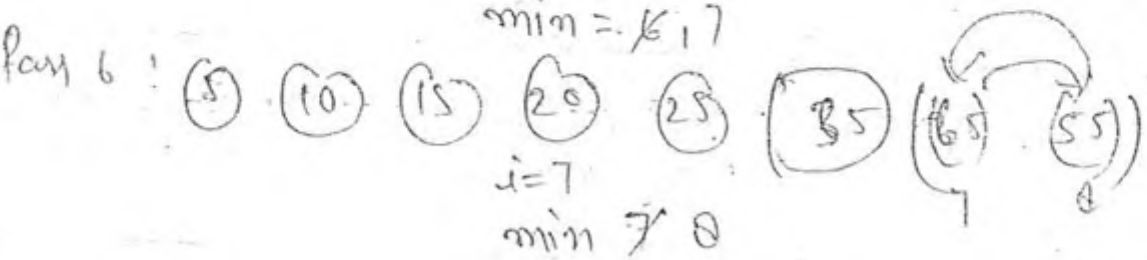












Pass 7: (5) (10) (15) (20) (25) (38) (55) (65)

Selection sort

(1) the min^m 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 cast and worstcase)

Bubble Sort

```
Bubblesort (a)
{   flag = 0
for ( i=1 ; i ≤ n-1 ; i++)
{   if ( flag == 0)
    for ( j = 1 ; j ≤ n-1 ; j++)      flag = 1
    {
        if ( a[j] > a [j+1]
        {swap ( a[j], a[j+1])
        flag = 0
        }
    }
}
```

modification tekkam

eg

Pass 1 — 10, 15, 25, 40 | (50)

Pass 2 — 10, 15, 25 | (40)

Pass 3 — 10, 15 | (25)

Pass 4 — 10 | (15)

Q)

40 30 50 20 50
50 40 30 20

30 40 20 40 | 50
30 40 20 | 50

20 30 | 40

20 | 30

Q)

329 457 657 039 436 720 355
039 436 720 355

(1) 329 457 657 436 720 355 | 039

(2) 329 457 657 436 355 | 720

(3) 329 436 457 355 | 657

(4) 329 436 355 | 457

(5) 329 355 | 436

(6) 329 | 355

Stable
Inplace:

min = O

max = $O(n^2)$

eg

10, 20, 30, 40 | 50

10 20 30 | 40 50

10 20 | 30 40 50

10 | 20 30 40 50

10 20 30 40 50

if no of flag

$\Downarrow$

$O(n^2)$

$\Omega(n^2)$

$\Downarrow$

$\Theta(n^2)$

with flag

$\Downarrow$

$O(n^2)$

$\Omega(n)$

(5) Optimal Solution.

(3) Solution space ⇒ All possible sol^n with n input
Feasible Solution ⇒ It is one of the sol^n in sol^n space which will satisfied our condition.
Optimal sol^n ⇒ It is one of the feasible sol^n which optimes our target.

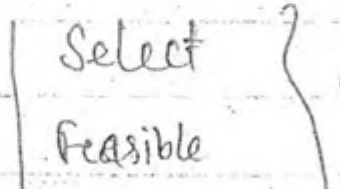
minimizing cost
maximizing profit.

Control Abstraction of Greedy

```
Greedy (a, n)
{
   Solution = ϕ
for (i=1 ; i≤n ; i++)
   {
      x = Select (a, n
      if (Feasible (x))
         add (x, Solution)
```

```
    }
    return (Solution)
}
```

Flow of Control



It is the flow of control
But each & every step is interval

It can choose any student
check any condition —
so it is called as control abstraction of greedy.

Applications of Greedy

(1) Greedy Knapsack
(2) Job sequencing with Deadlines
(3) Minimum Cost spanning tree
(4) Huffman Coding.
(5) Optimal merge pattern
(6) Single source shortest Path
    (i) Dijkstra's algo
    (ii) Bellman- Ford algo

Greedy knapsack

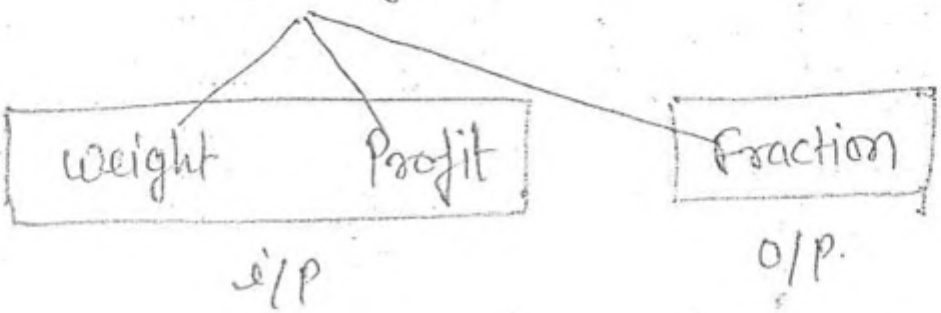
(OR)

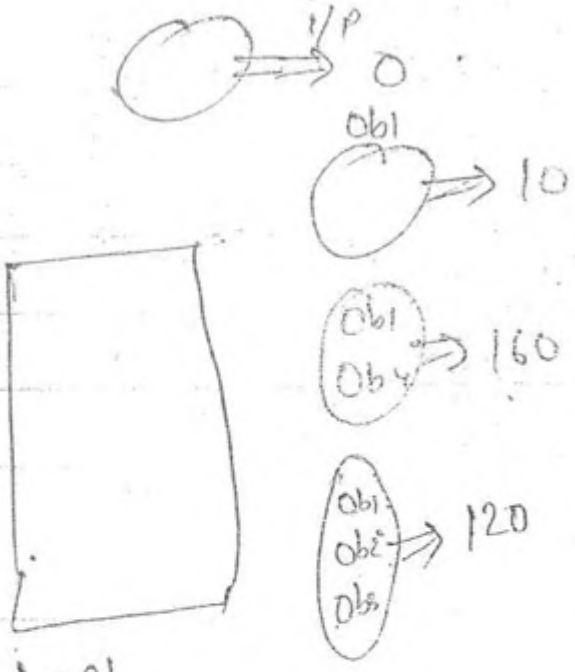
Real knapsack

(OR)

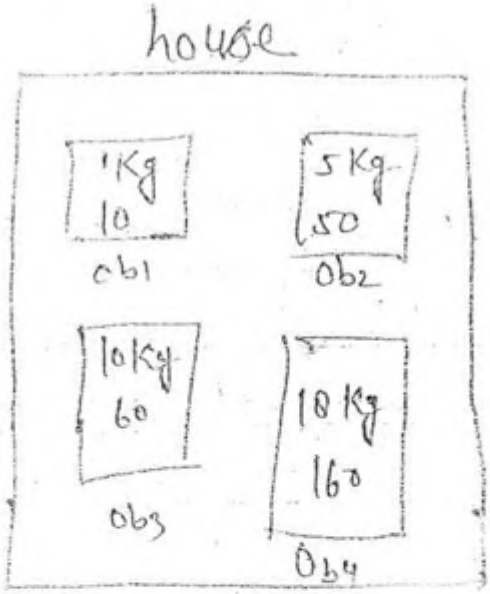
Fractional knapsack

n - object





C = 20 kg



4 – object $C = 20$

(n)

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 $\sum_{i=1}^{n} w_i > C$

Feasible $\sum_{i=1}^{n} w_i * x_i \leq C$

Optimal $\sum_{i=1}^{n} P_i x_i \quad max$

P

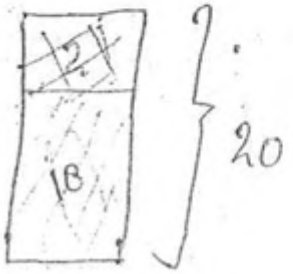
$n = 3 \quad c = 20$

object	Ob_1	Ob_2	Ob_3
profit	25	24	15
weight	18	15	10

Sol^n

Greedy about profit

$(\underline{1} \quad \underline{2/15} \quad \underline{0})$



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

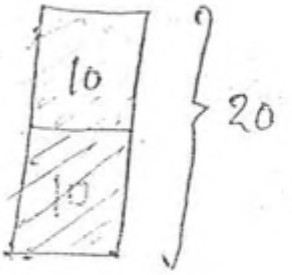
$= 25 + 32$

$= 28.2$

future

Greedy about weight

$(\underline{0} \quad \underline{10/15} \quad \underline{1})$



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

$= 0 + 16 + 15$

$= 31$

Present

Greedy about both (P & W)

ob_1 : $10 \Longrightarrow 25$

$1 \Longrightarrow \frac{25}{10} = 1.3$

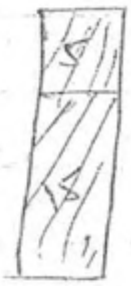
ob_2 : $15 \Longrightarrow 24$

$1 \Longrightarrow \frac{24}{15} = 1.6$

ob_3 : $10 \Longrightarrow 15$

$1 \Longrightarrow \frac{15}{10} = 1.5$

$(0 \quad 1 \quad 5/10)$



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

$= 24 + 7.5$

$= 31.5$

Note:- In Greed Knapsack who will also get optimal soln by considering both ~~profit~~ 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

ob1 : $5 \Rightarrow 5$

$1 = 5/5 = 1$ (3)

ob2 : $4 \Rightarrow 2$

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

ob3 : $6 \Rightarrow 2$

$1 \Rightarrow \frac{2}{6} = = .33$ (5)

ob4 : $2 \Rightarrow 4$

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

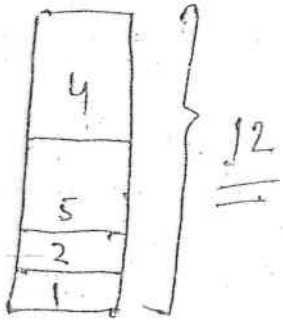
ob5 : $1 \Rightarrow 5$ ✓

$1 = 5/1 = 5$ (1)

(1 1 0 1 1)

$= 1\times5 + 1\times2 + 0\times2 + 1\times5 + 1\times4$

$= 16$



Q. $n = 7$ $C = 15$.

Object	Ob_1	Ob_2	Ob_3	Ob_4	Ob_5	Ob_6	Ob_7
Profit	10	5	15	7	6	18	3
Weight	2	3	5	7	1	4	1

then find out Optimal Soln.

Soln

1 → $\frac{10}{2} = 5$ — (2)

2 → $5/3 = 1.6$ — (6)

3 → $15/5 = 3$ — (4)

4 → $7/7 = 1$ — (7)

5 → $7/1 = 6$ — (1)

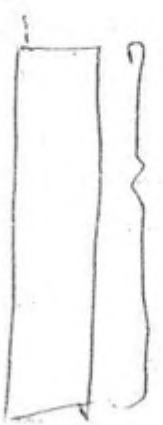
6 → $18/4 = 4.5$ — (5)

7 → $3/1 = 3$ — (3)

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

$= 1 \times 10 + \frac{2}{3} \times 5 + 1 \times 15 + 0 \times 7 + 1 \times 6 + 1 \times 18 + 1 \times 3$

$= 55.33$



15 14. 12 10 3 2

Algo

(1) for (i=1 ; i≤n , i++) ⟹ O(n)

a[i] = Pi/wi

(2) Arrang Array in A in decresing order of Pi/wi ratio. ⟹ O(n log n)

(3)
```
P=0
for (i=1 , i≤n , i++)
{
    if (C ≠ 0)
    {
        P = P + Pi
        C = C - wi
    }
}
return (P)
```

take one by one object untill capacity of Knapsack becomes zero.

⟹ O(n)

O(n log n)

(3)
```
for (i=1 ; i≤n , i++)
{
    if ( C>0)
    {
        if ( C > wi)
        {
            P=P+Pi , C=C-wi
        }
        P=P+Pi , C=0
    }
}
```

$\left(\frac{0}{1} - - \rightarrow\right)$

$\frac{1}{2}$

$1/10$ → Solution space for greedy Knapsack is infinite

$1/10$

14/7/11

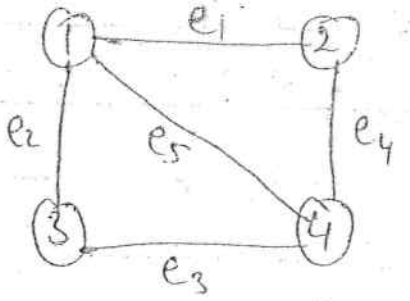
Minimum Cost Spanning Tree

$G(V, E)$

V = Set of vertices.

E = Set of edges.

eg



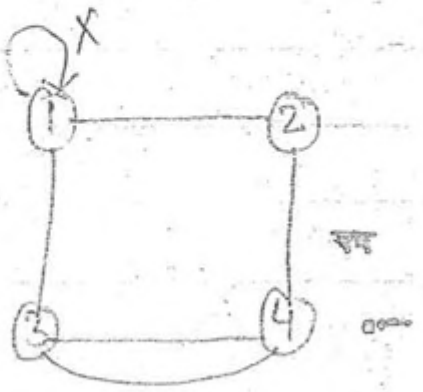
$V = \{1, 2, 3, 4\}$.

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

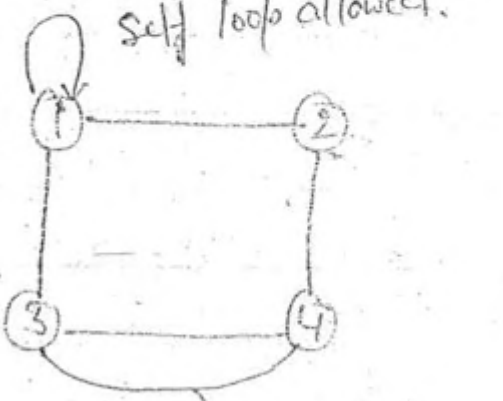
Types of Graph

Simple graph

Multi Graph.



no-self loop
parallel edge.



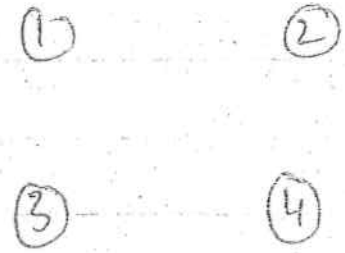
Simple Graph

Types of Graph

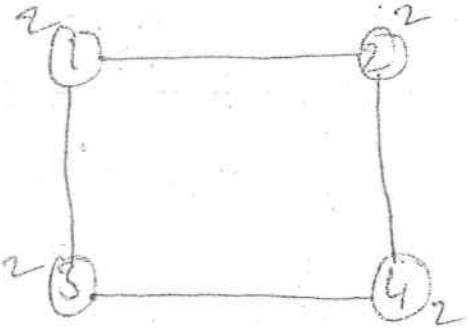
(i) Null Graph.
(ii) Regular "
(iii) Complete "
(iv) ⋮

↕ edges.

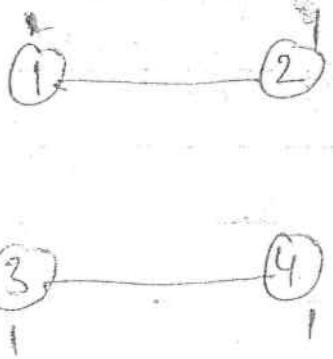
Null Graph



Regular Graph

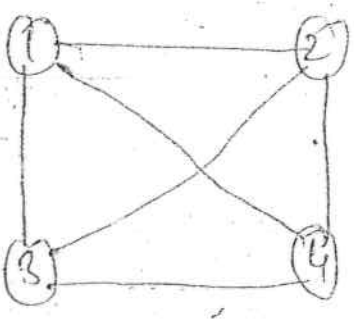


2-Regular graph

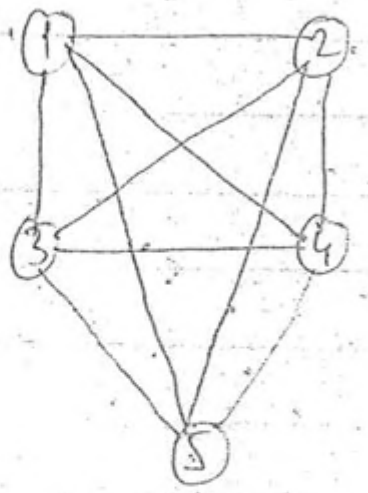


1 regular graph.

Complete Graph



K_4 Complete graph



K_5 Graph

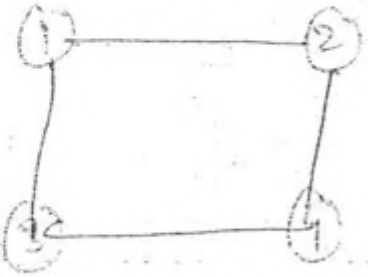
* Max^{m} no of ~~Total edge~~ edges in n vertex simple graph is equal to no of edges in complete graph.

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

* The no of edges in a n vertex K regular

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

*

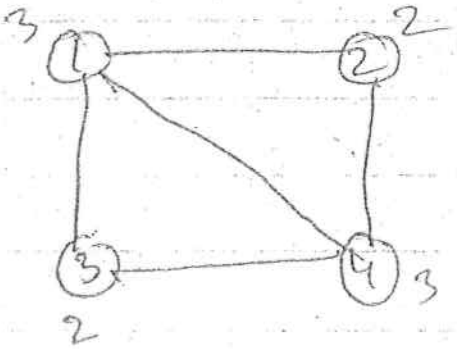


K_4 graph

$(n-1)$

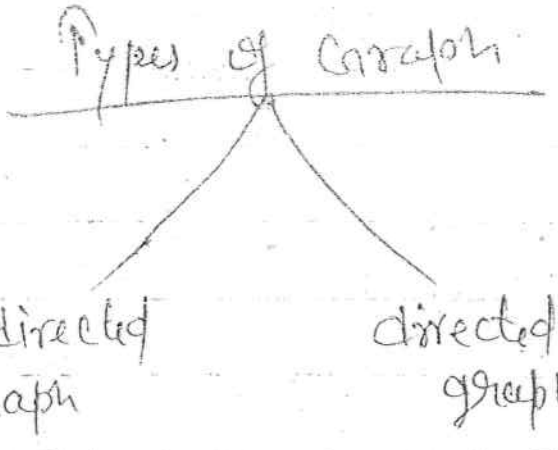
every complete graph is a regular graph.

Sum of degree theorem (undirected)



$3 + 2 + 2 + 3 = 2 * |E|$

$$\sum_{i=1}^{n} \text{degree}(V_i) = 2 * |E|$$



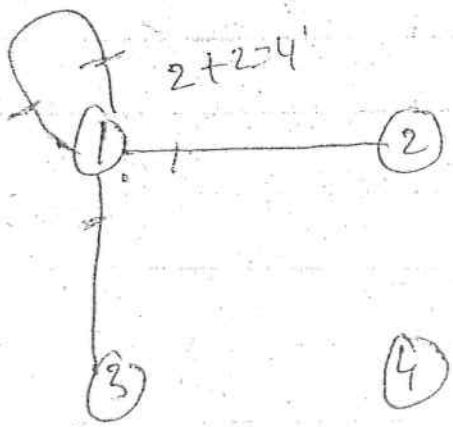
(1,2) ✓

(2,1) ✓

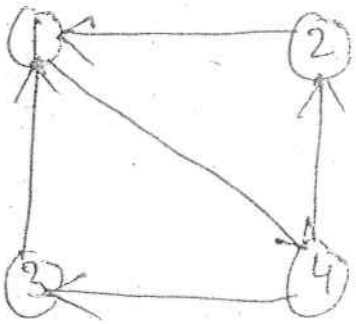
(1,2) ✓

(2,1) ✗

Self loop is count by degree 2



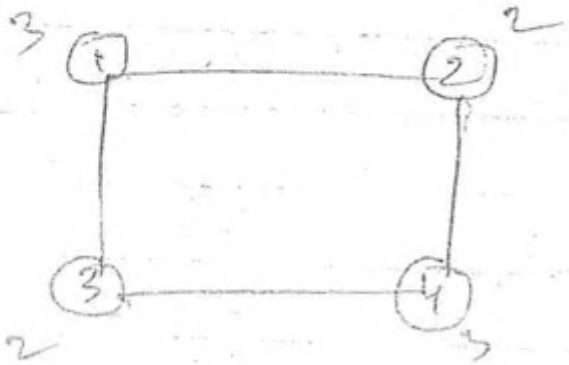
Directed graph (Sum of degree Theorem)



In degree (+)	Outdegree (-)
$1^+ = 2$	$1^- = 1$
$2^+ = 1$	$2^- = 1$
$3^+ = 1$	$3^- = 1$
$4^+ = 1$	$4^- = 2$
5	5

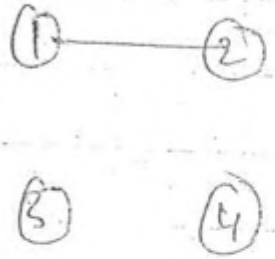
$$\sum_{i=1}^{n} (v_i)^+ = \sum_{i=1}^{n} (v_i)^- = |E|$$

Hand Shaking Lama



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

0 vertices with odd degree



2 vertices with odd degree.

4- vertices with odd degree.

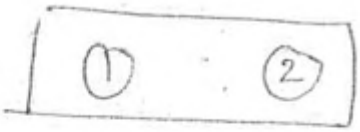
How many different simple Graphs are possible with n -vertices ?.

Sol^n

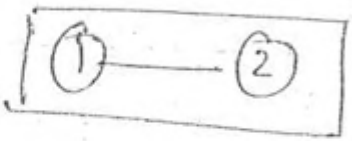
$n=1$

$\Rightarrow 1$

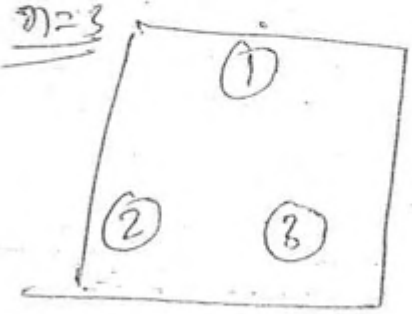
$n=2$

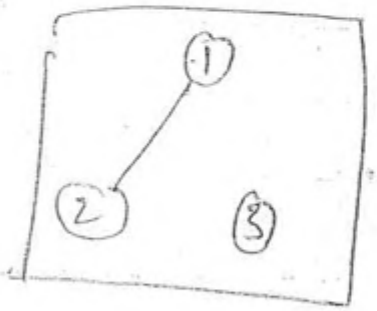


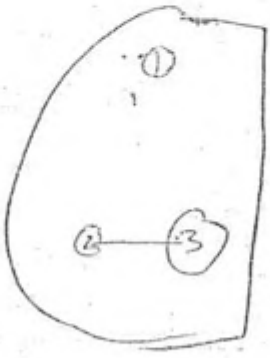
$\Rightarrow 2$

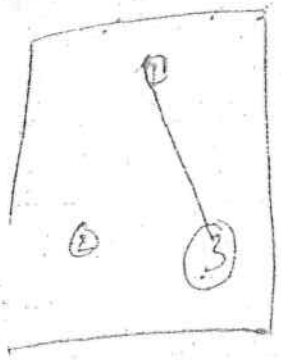


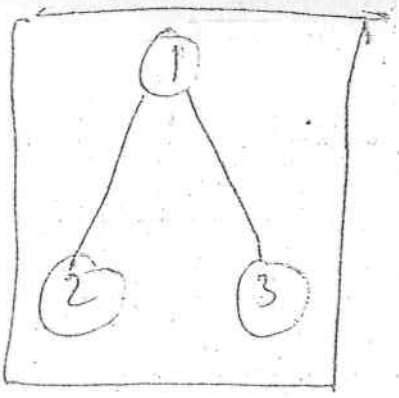
$n=3$

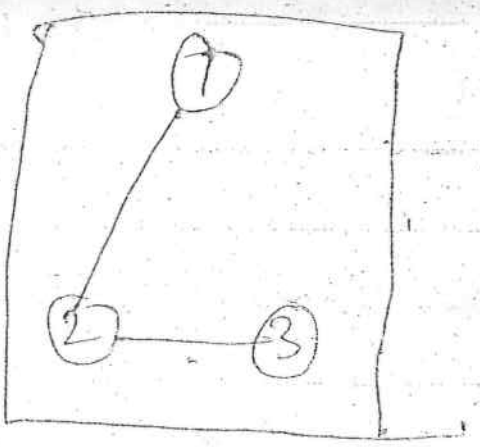


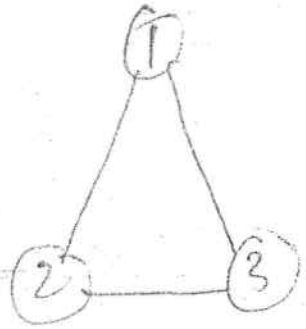












$\Rightarrow$ B

total simple graph if node is ?

a) n b) 2^n

c) 2^{n-1} d) $2^{\frac{n(n-1)}{2}}$

The no of simple graph possible with n vertices is equal $\frac{n(n-1)}{2}$

where $\frac{n(n-1)}{2}$ is max^n no of edge possible with n vertices.

DESIGN ANALYSIS & ALGORITHAM

PART-II

(1)

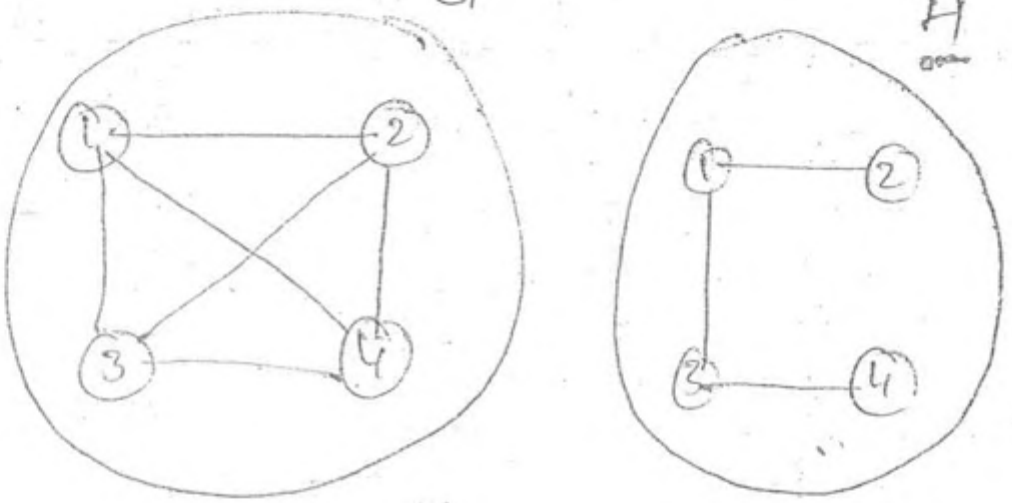
Spanning Tree

A connected subgraph H of a Graph G is said to be spanning tree iff.

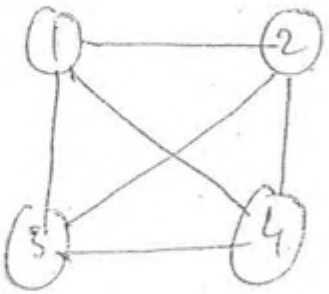
i) H should contain all the vertices of G.

(2) " " " (n-1) edges if G contain n-vertices.

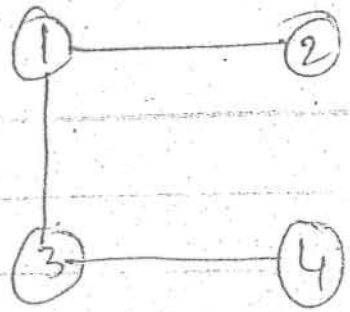
eg



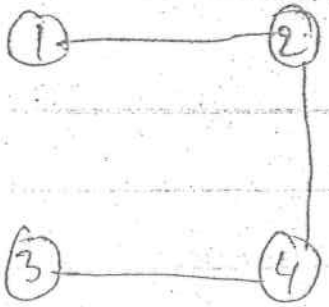
Q. Find all possible different spanning tree for the following graph.



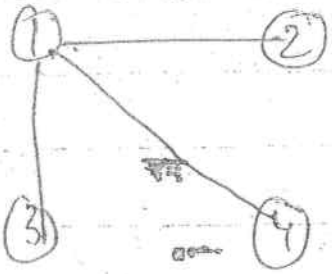
(5)



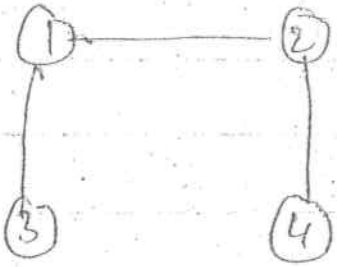
(i)



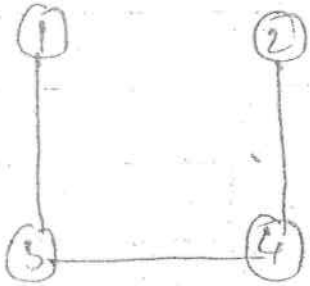
(ii)



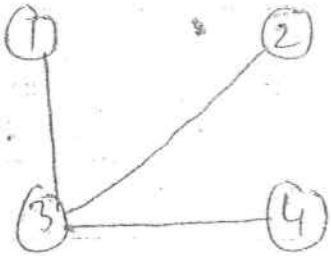
(iii)



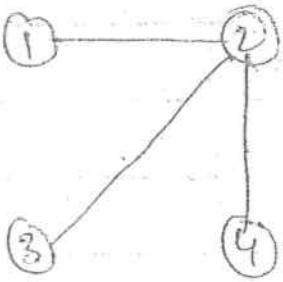
(iv)



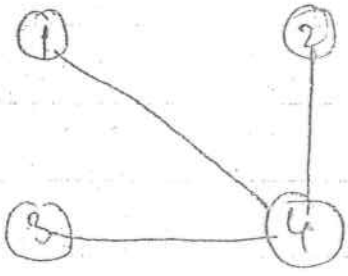
(v)



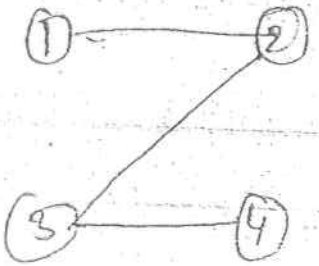
(vi)



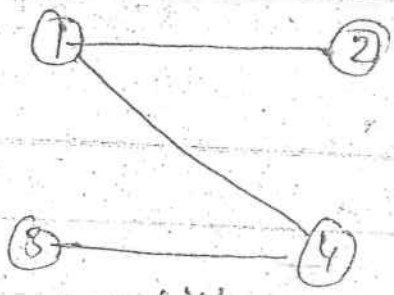
(vii)



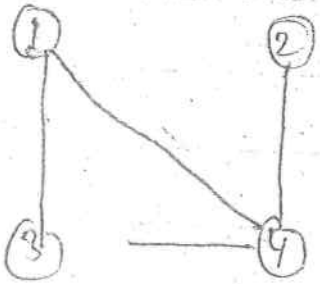
(viii)



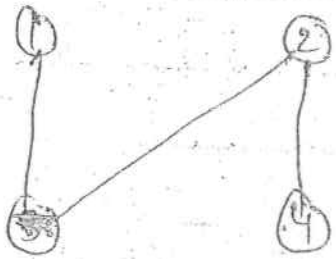
(IX)



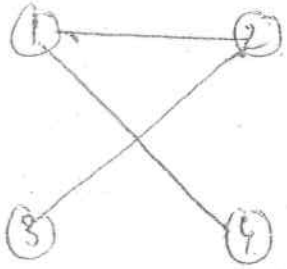
(X)



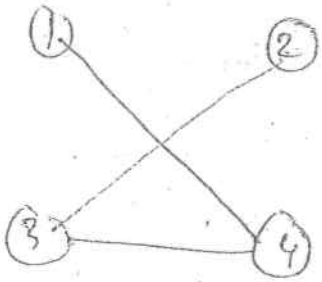
(XI)



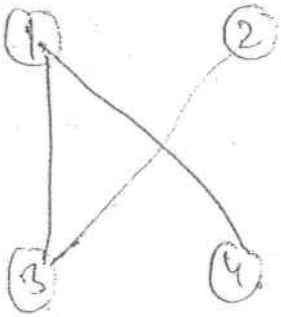
(XII)



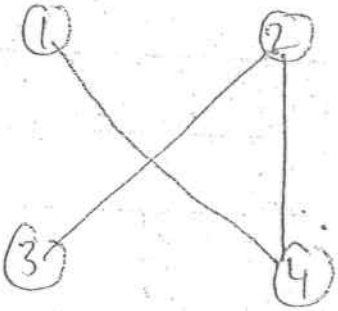
(XIII)



(XIV)



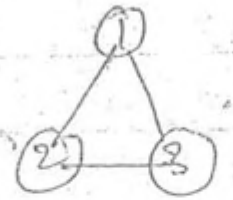
(XV)

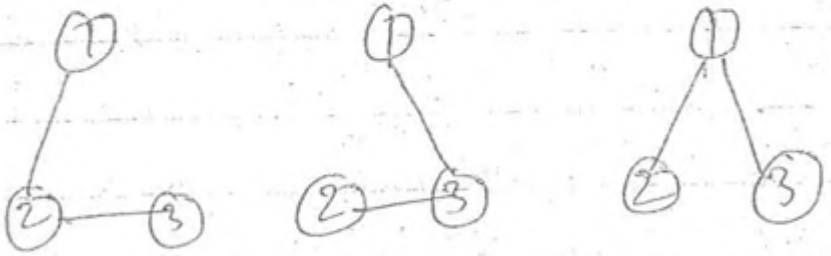


(XVI)

Find all

K_3





$ST(K_3) = 3 \quad (3^{3-2})$

$ST(K_4) = 16 \quad (4^{4-2})$

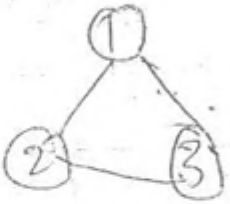
$ST(K_5) = 125 \quad (5^{5-2})$

$ST(K_n) = n^{n-2}$

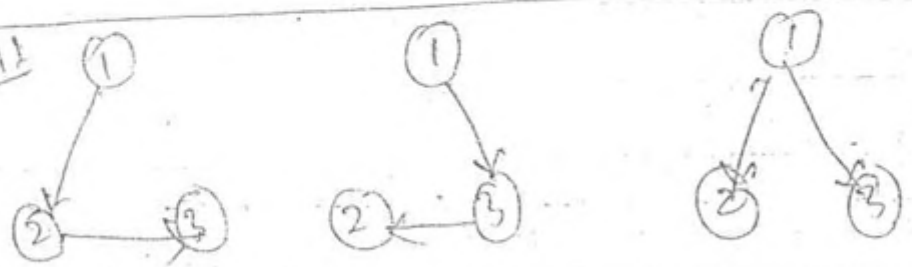
The no of possible spanning trees in complete graph with n vertices.

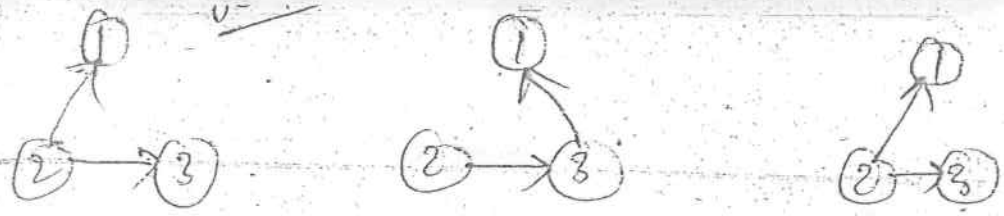
$= n^{n-2}$

$K_3 =$

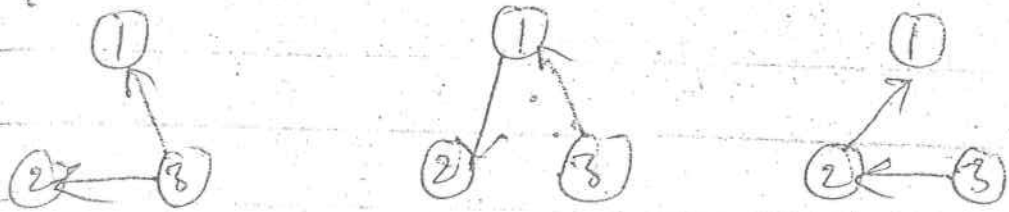


root:





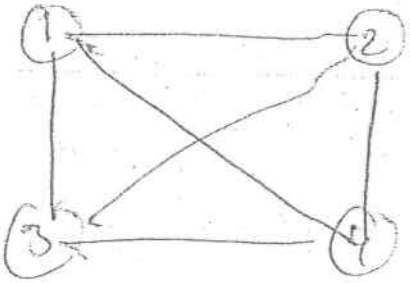
root 3



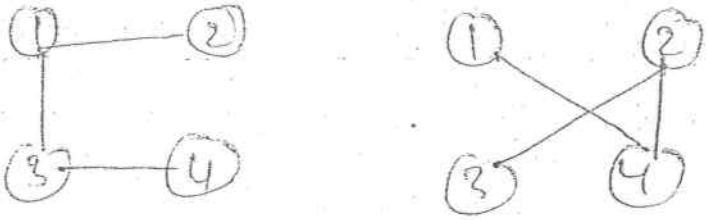
Note → The no of root a spanning tree in a with n vertices comple graph.

$$= n * (n^{n-2})$$

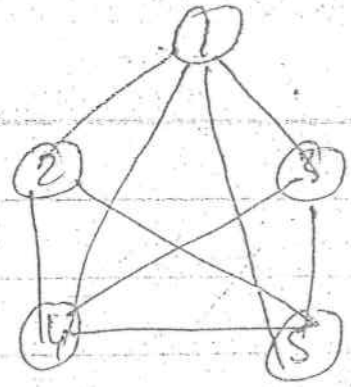
K4

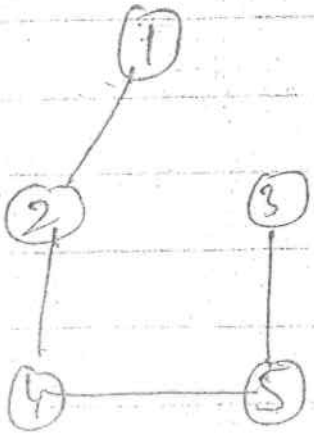


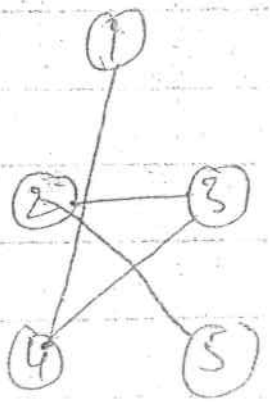
edge disjoint ST



K_5







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

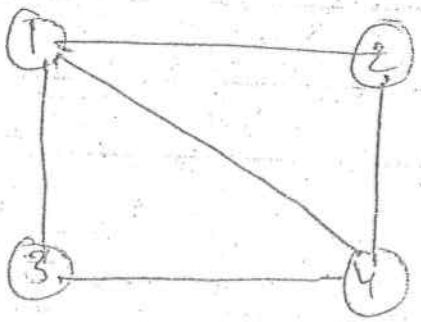
$K_4 = 2 \Rightarrow \left\lfloor \frac{4}{2} \right\rfloor \Rightarrow 2$

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

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

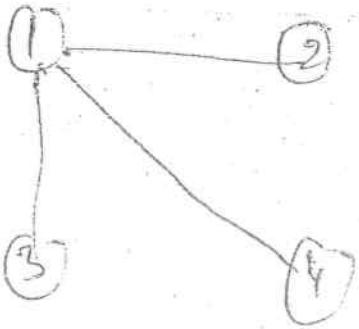
$EDST(K_n) = \left\lfloor \frac{n}{2} \right\rfloor$

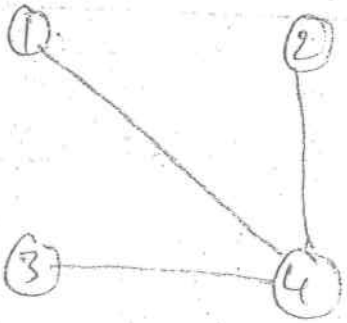
1. How many spanning tree are there for the following graph.



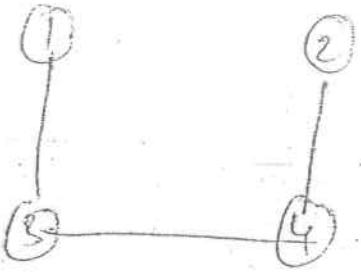
a) 16
b) 10.
c) 8 ✓
d) 12

Sol:

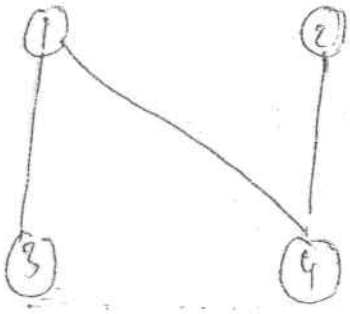


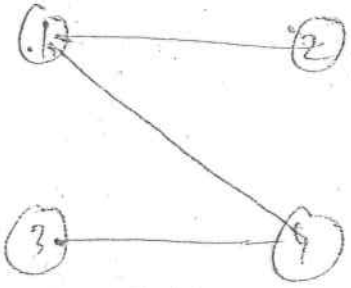


2



4





2

Kirchaff Theorem

1) Find adjacency matrix M for the given graph.

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & \phi \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

⇒ Dense graph (more edge)

⇒ $O(V^2)$

Adjacency Matrix

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

Those Graph which has less edges such graph is calles sparce graph (less edg)

(2) (i) replace all non digonal 1's by 1's.

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

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

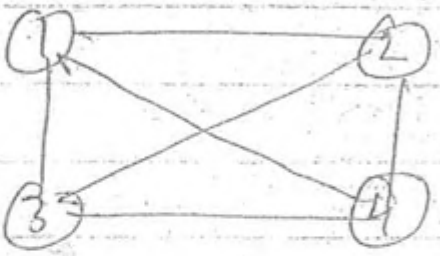
(3) Find Co-factor of any element which will give no of spanning tree.

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

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

$$= 10 - 2 = 8$$

2) Find no of spanning tree for the following graph using Kirchoff Theorem



Sol^n

(1) M_2

	1	2	3	4
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0

(2) $M =$

3	-1	-1	-1
-1	3	-1	-1
-1	-1	3	-1
-1	-1	-1	3

(3) find cofact of M_{11}

$$= (-1)^{1+1} \begin{vmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{vmatrix} = 16$$

for a given graph more than 1 MST is may be possible but min^m cost be same..

If the given graph contain Distinct edge weight are then it give max^m one max^m SCT.

Q Let G be an undirected connected Graph with distinct edge weight.
Let e max be the max^m weight and emin be min^m weight then which one of the following false.

(a) Every min^m CST must contain. Emin.

(b) If e max is in MST, then its removed must disconnect G.

(c) No min^m CST contain emax

(d) G 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

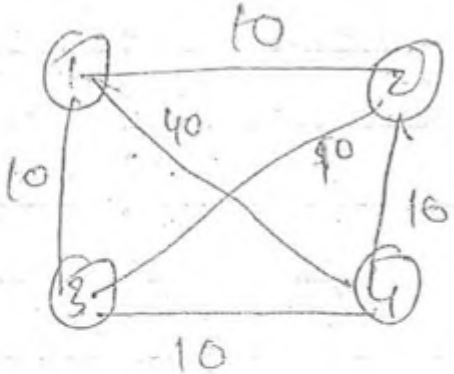
a) e may be there in MST

b) If e is not in MST than all edges have same weight w in that cycle.

c) If every ⊗ min^m EST should contain an edge with w.

x d) Every MSET should contain edge e.

80/9



w = 10

Q An undirected graph G has n-nodes. its adjcent matrix is given by an $n \times n$ square matrix, whose elements are defined as follows.

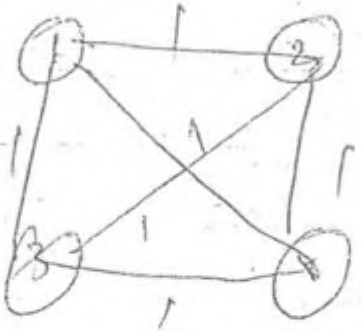
i) all diagonal elements are 0's.
ii) all non diagonal elements are 1's. } means complet graph.

True ?.

a) G has no MST
b) G has unique MST, each of cost $n-1$
c) G has multiple MST with different cost
d) G has multiple MST each of cost $n-1$

Sol^n

K4



16 ⇒ MST
each of cost = 3

$K_n \Rightarrow n^{n-2}$ MST
each of cost $n-1$

= Let T and T' be 2-spanning Tree of a connected graph G.

Suppose that an edge e is in T but not in T' and all other remaing edge are same Then which one of the following is true about T & T' after performing following 2-operations.

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

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

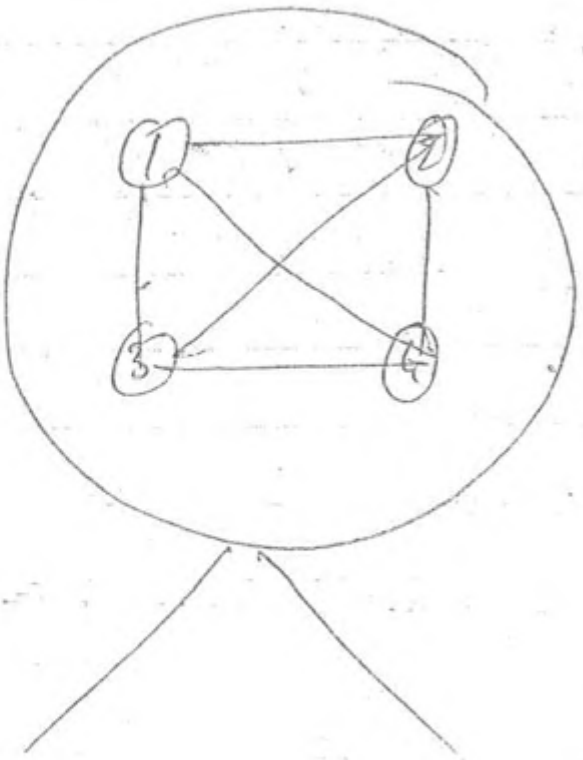
(iii) both T and T' are still ST

b) only T is ST

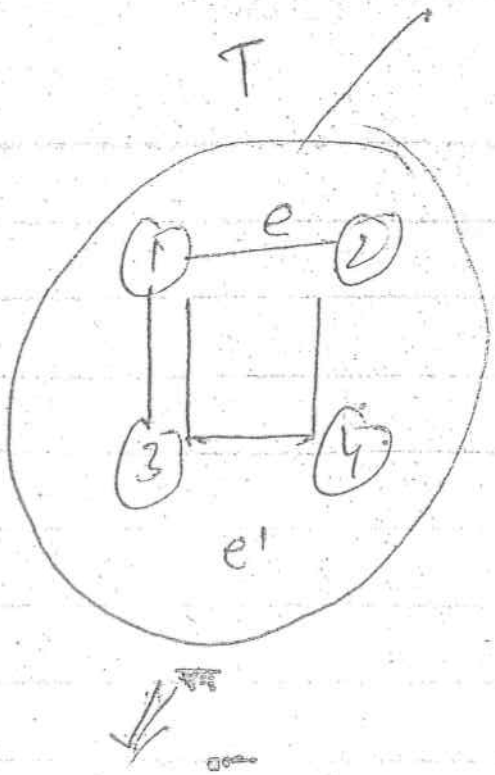
c) only T' is ST

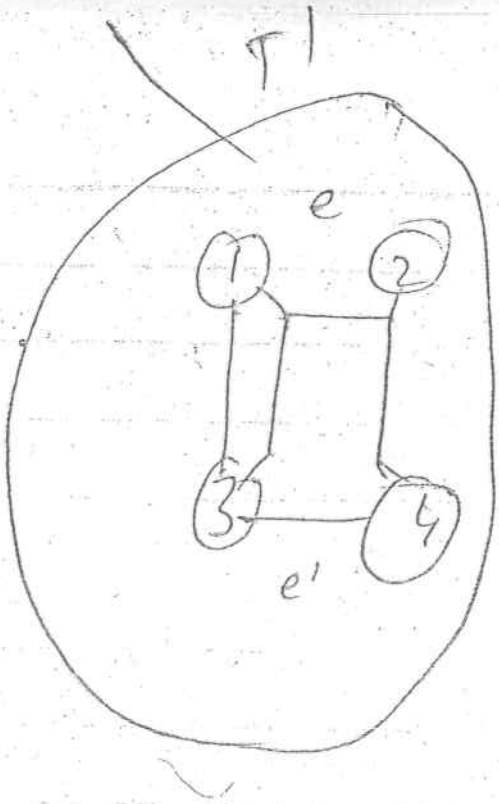
d) both are not.

Soln

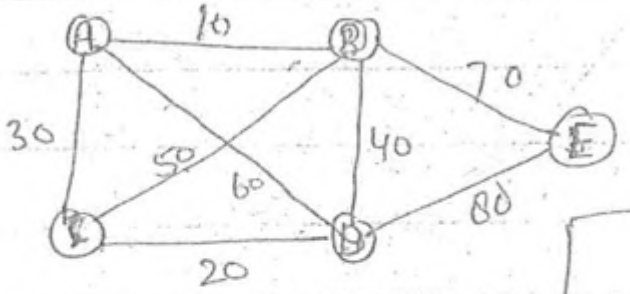


G



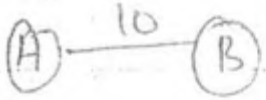


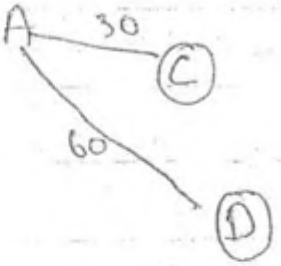
Prim's Algo



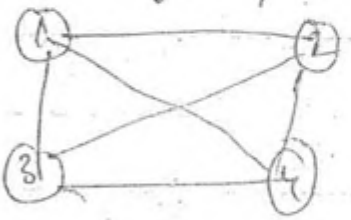
(i) Take any vertex & find adjecent of that vertex

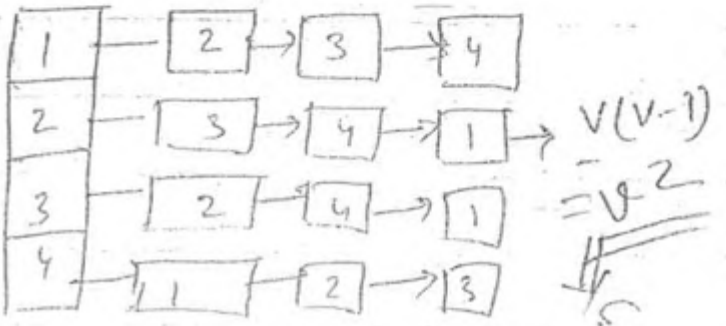
and take minm of all these

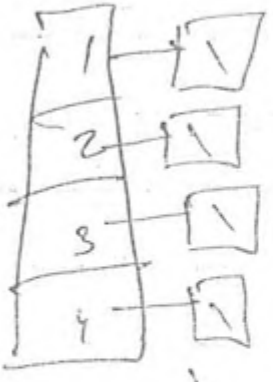




Adjcency list representation



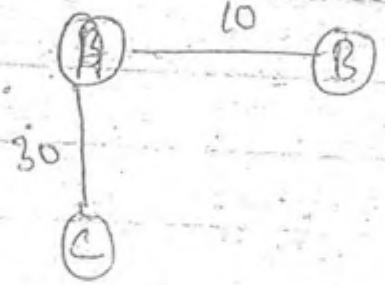


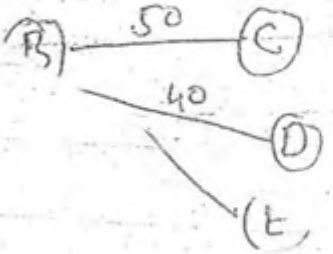


$\theta(V+E)$

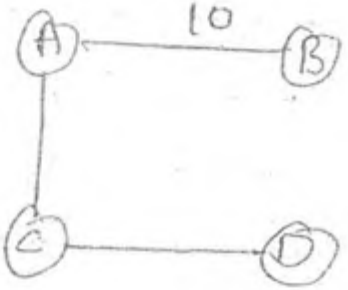
(4)

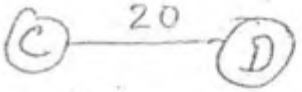
of the new vertex find adjacent and min^m
: cost vertices & previous



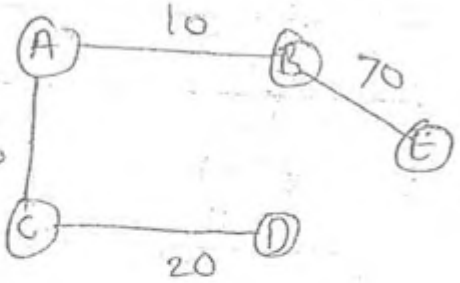


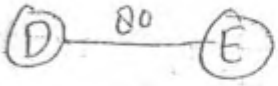
(3) for the new vertex find adjacent and take min^m of three adjacent and previous all adjacent.





(4) for the — — — — —





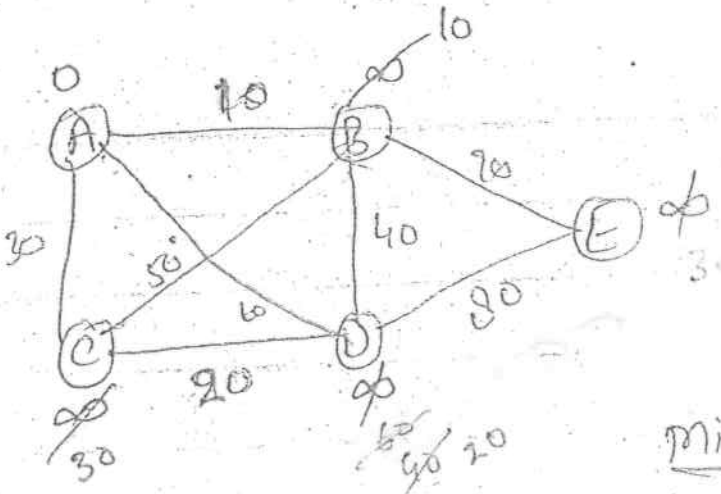
BD ×.

BC ×

AD ×

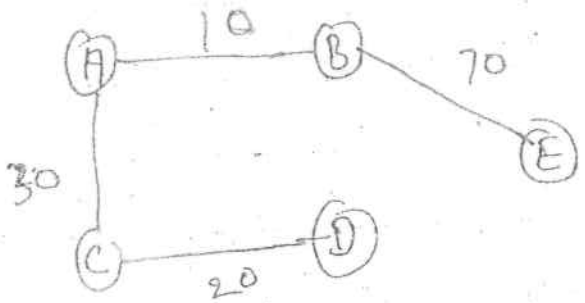
Note Prim's & Kruskal algo Both generate same cost
Can generate diff. MST.
But in B/w algo Prims doesn't have any forest and Kruskal have forest

#

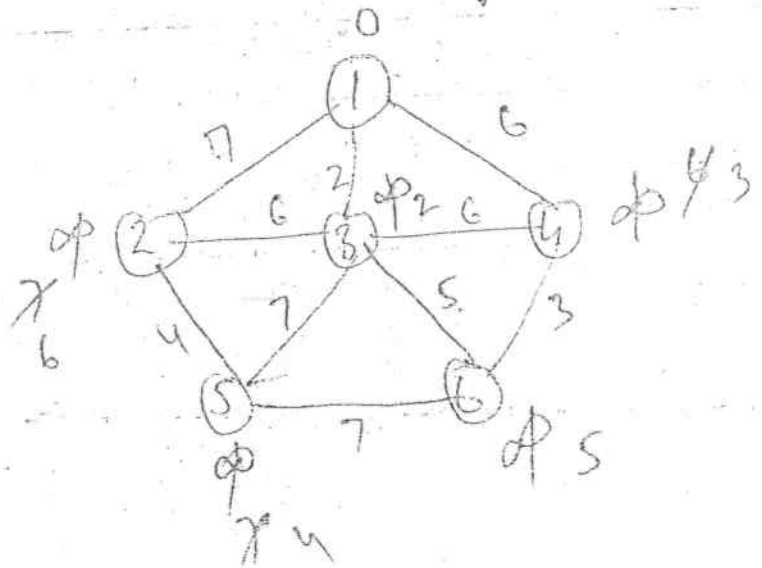


min heap tree

A	B	C	D	E
∞	∞	∞	∞	∞
0	∅	∅	∅	∞
	10	30	60	∅
		30	40	70
			20	70
				70

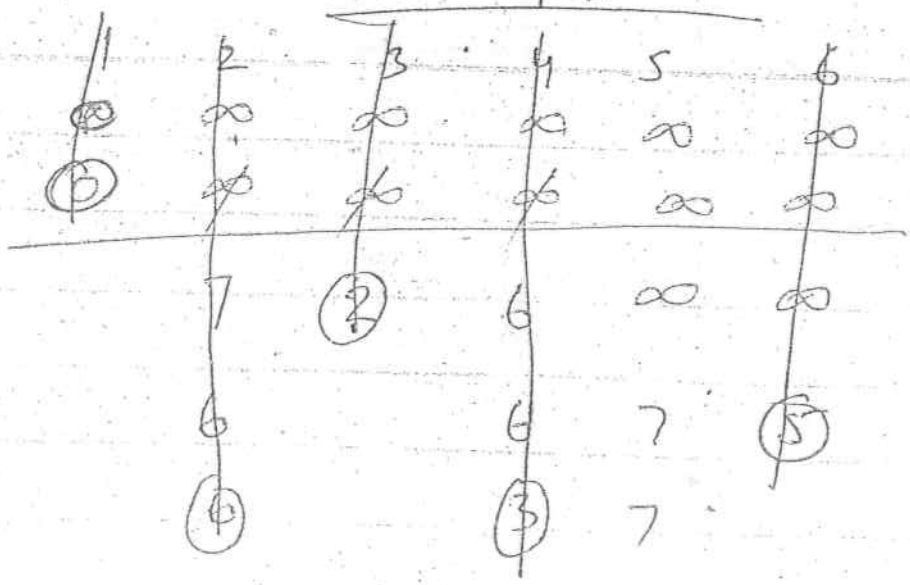


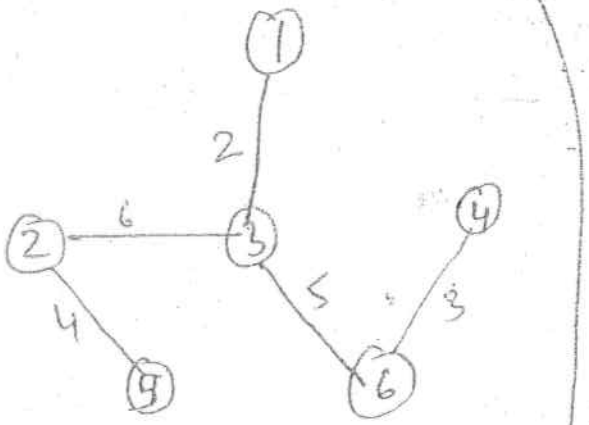
Consider the following graph.



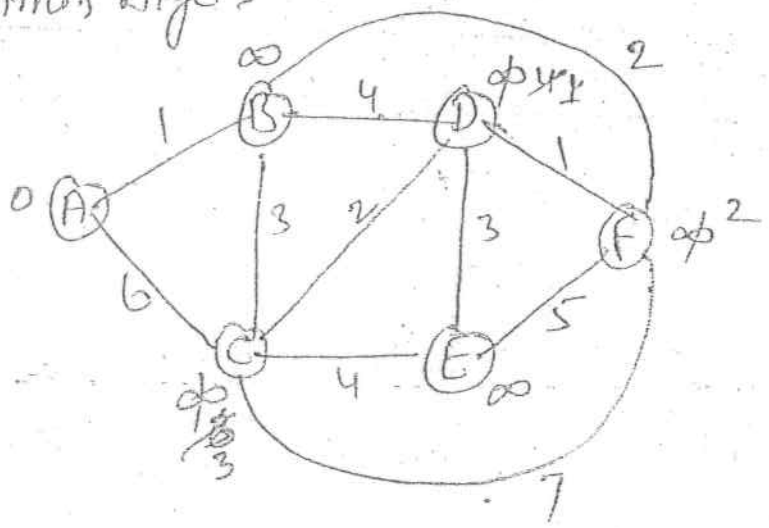
Sol^n

Min heap tree



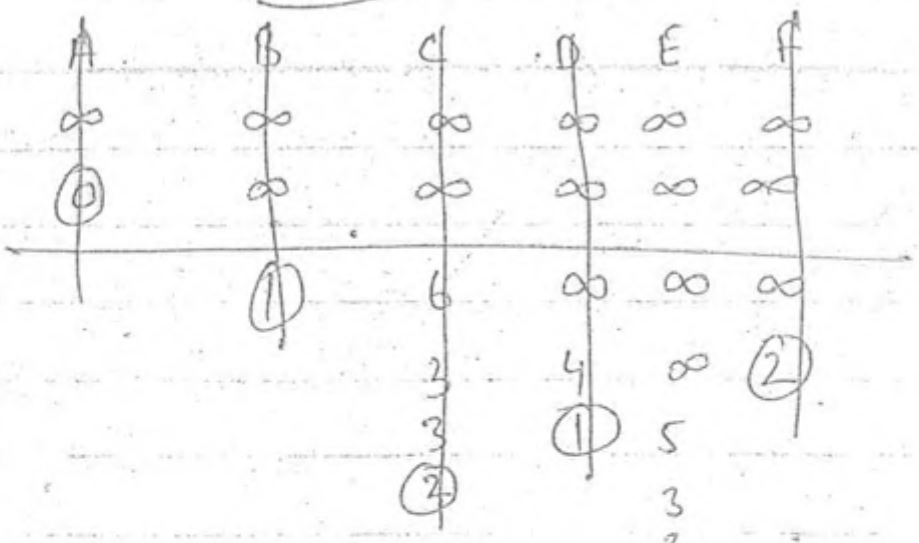


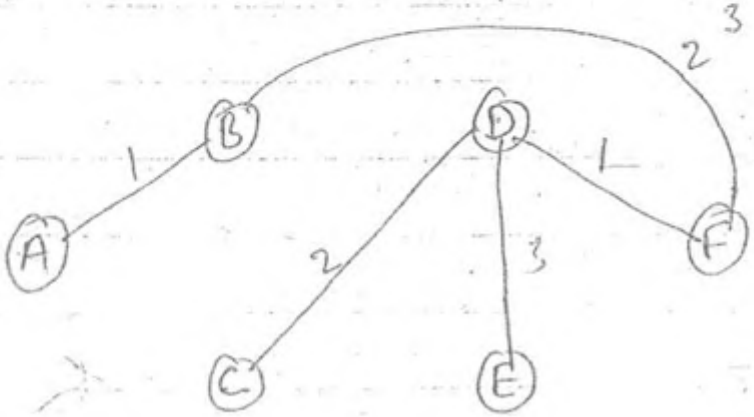
P Construct min CST for the following Graph using Prim's Algo.



Min heap tree

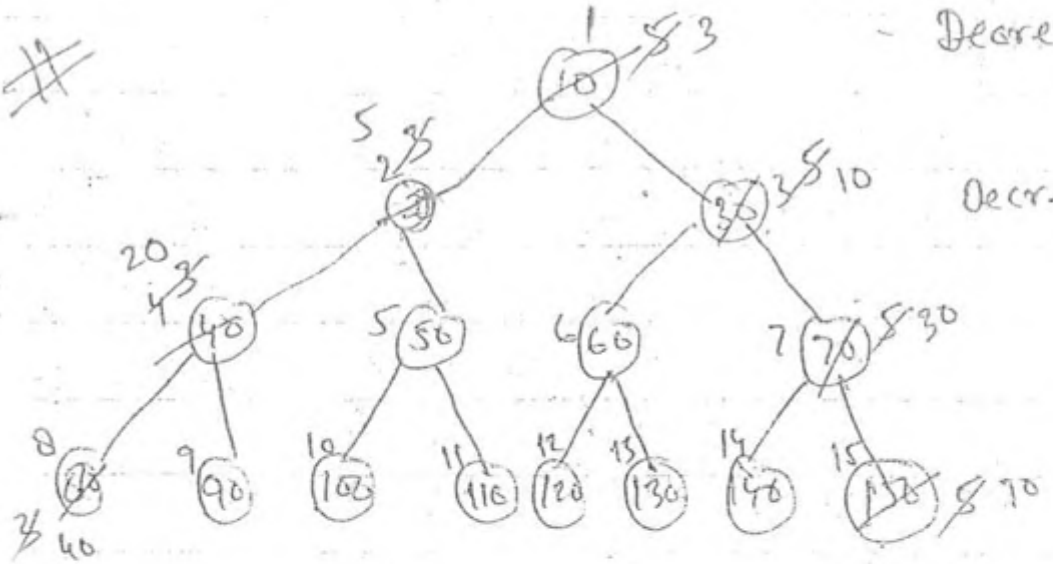
Sol^n





Initialization

#



- DecreaseKey(0, i, 5)
  ↓
  15

DecreaseKey(0, 1, 3)
↓
8

⇒ It is also operation of heap. To decrease the key in the min heap tree it will take O(log n) time.

⇒ 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 x ∈ V                    ⟹ O(V)
            x.min = ∞.
    (2) s.min = 0                                ⟹ O(1)
    (3) Create minheaptree (Q) for all
        vertices.                                ⇒ O(V)
    (4) while (Q ≠ ϕ)                            ⟹ O(log V)
        {
            u = extractmin (Q)
            for all v adjcent u
                if ( d(u,v) < v.min )            V-1
                    v.min = d(u,v)               → decreases key of O(log V)
        }
```

Initialization

$= O(V) + O(1) + O(V) + V(\log V + (V-1)\log V)$

$= V \log V + V^2 \log V$

$= V \log V + E \log V$

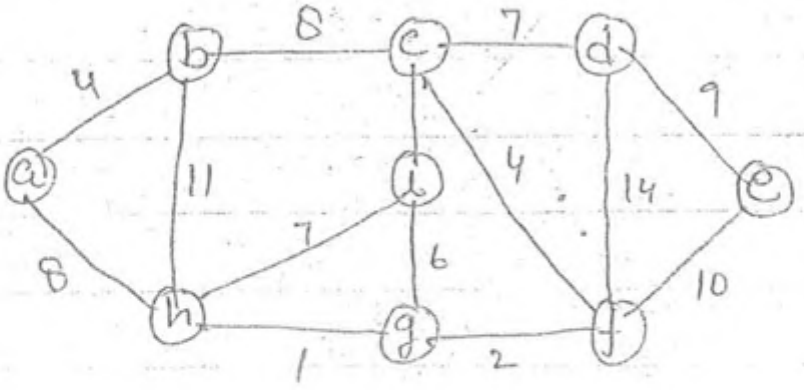
$= \boxed{O(V+E) \log V}$

using minheap.

$= \boxed{O(E + V \log V)}$

using Fibinocci minheap

Consider the following Graph.



Which one of the following sequence of edges of MCST is not true when the algo is started.

a) (a,b) (b,c) (c,i) (c,f) (f,g) (g,h) (c,d) (d,e)

b) (a,b) (a,h) (h,g) (g,f) (f,c) (c,i) (c,d) (d,e)

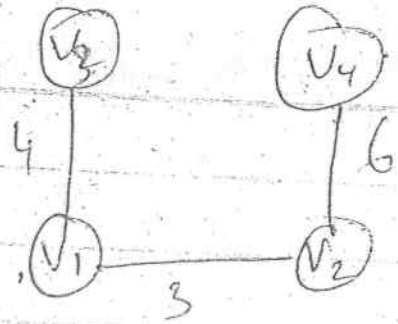
c) (a,b) (b,c) (f,c) (c,i) (f,g) (g,h) (c,d) (d,e)

X d) (a,b) (b,c) (h,g) (f,g) (c,i) (c,f) (c,d) (d,e)

disconnected Graph.
So. Prims no Possible

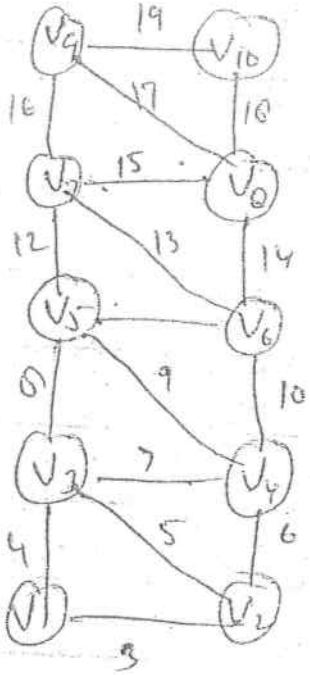
(1)

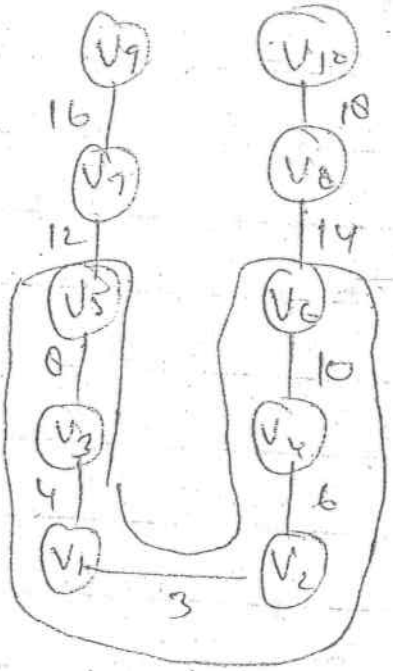
Soln



13

(2)





P Consider a complete undirected graph G(V,E) with vertex set {0,1,2,3,4}. Entry Wij in the matrix W below is the weight of the edge (i,j)

$$W = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 4 \\ 1 & 0 & 12 & 4 & 9 \\ 0 & 12 & 0 & 7 & 3 \\ 1 & 4 & 7 & 0 & 2 \\ 4 & 9 & 3 & 2 & 0 \end{bmatrix} \end{array}$$

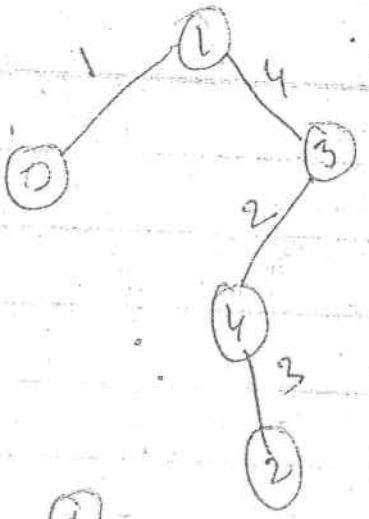
① What is the Minm possible weight of a SF T for the above graph, such that vertex 0 if leafnodes in T

a) 7, b) 0 c) 9 d) 10

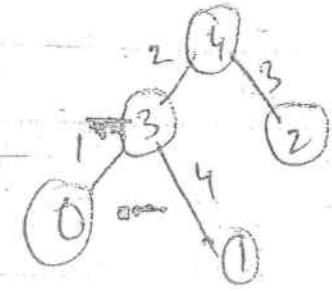
② What is the minm possible weight of a path P from vertex 1 to vertex 2 for the above graph G such that P contain atmost 3 edge.

a) 7 b) 8 c) 9 d) 10

Soln P(1)

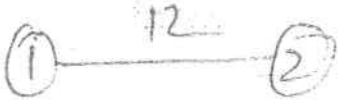


10

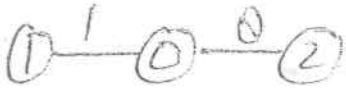


10

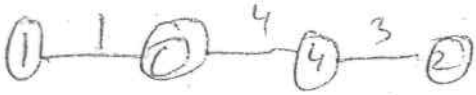
P(2)



x

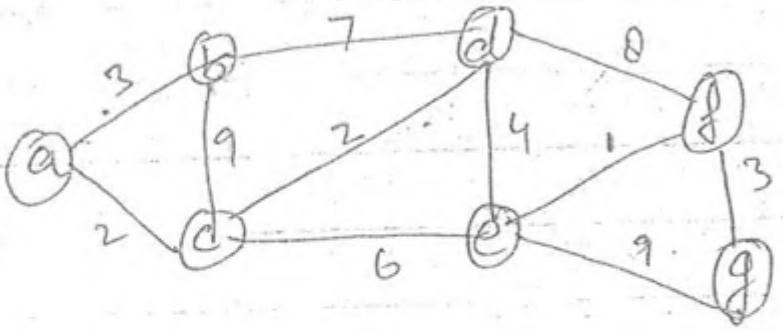


9



8

= Consider the following graph.



If Prims & Kruskal algo is used for generating ~~to generate~~ or mST of above graph, then which one of the following is true.

a) Prims is 25% decrease comparing with Kruskal.

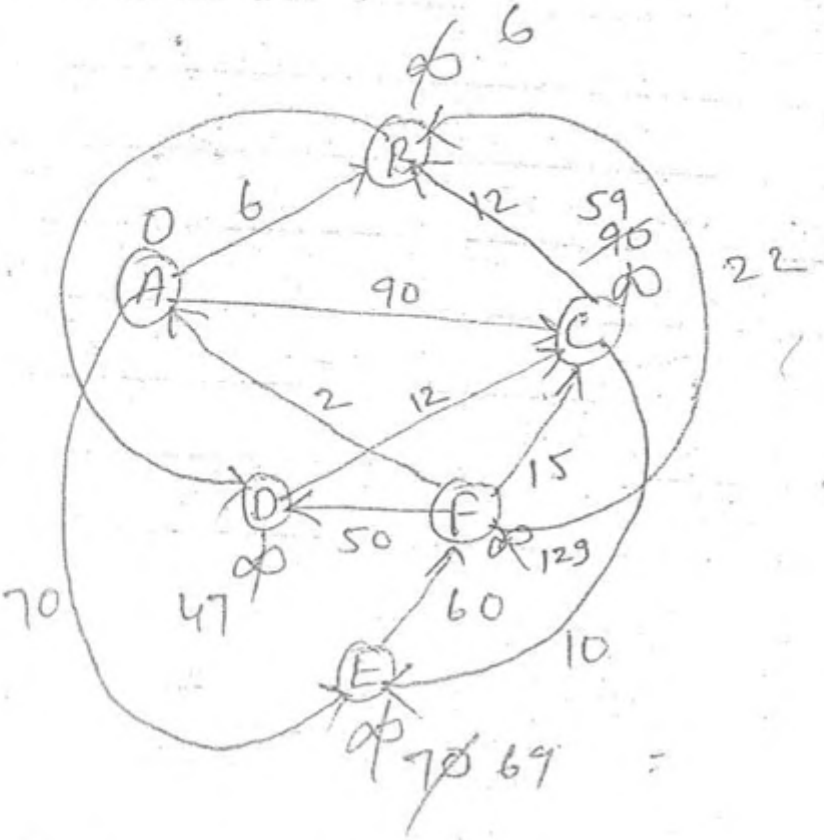
b) Kruskal is 10% increase comparing with prims.

c) Kruskal is 7.3% decrease with Prims ~~algo~~ ~~10%~~

d) Prims decrese 10% increse with kruskal.

Ans Both Prims and Kruskal will give same answer

Let G be a directed weighted graph shown ~~in~~ 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

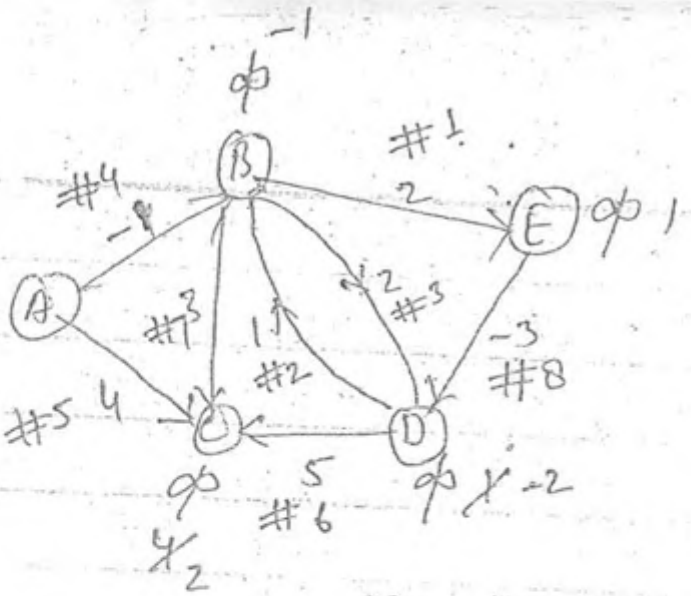
9

=

Not in min heap	In Min heap A	B	C	D	E	F
	(0)	∞	∞	∞	∞	∞
A		(6)	90	∞	70	∞
A B			90	(47)	70	∞
A, B, D			(59)		70	∞
A, B, D, C					(69)	
A, B, D, C, E						129
A, B, D, C, E, F						(129)

```
while ( Q ≠ ϕ )                                   → log V
{
    u = extract min(Q)
    for all adjacent
    V { vertices v to u
    (V-1) {  decreasekey (v)                       → log V
    }
}
```

eg



apply |V|-1 = 5-1 = 4 time (i = 4 time).

Soln

A	B	C	D	E	
0	∞	∞	∞	∞	
0	-1	∞	∞	∞	
0	-1	4	∞	∞	i=1
0	-4	2	∞	∞	
0	-1	2	∞	1	
0	-1	2	1	1	i=2
0	-1	2	-2	1	
0	-1	2	-1	1	i=3
0	-1	2	-1	1	i=4

Note - If given graph contain -ve weight cycle, Bellman ford algo report saying that -ve weight cycle ~~not there~~ exist)

Otherwise it will find out shortest path (when -ve weight cycle not form source to every vertex)

$O(VE)$

$O(VV^2)$ = dense

$O(V.V)$ = sparse.

eg

Job Sequencing with Deadline

(1) Single process is available. (1 unit of time for each job)
(2) Interleaving is not allowed.
(3) Arrived time all of the job are same.
(4) Every job requires 1 - unit of time.

eg

$n = 4$.

Jobs

Profit Deadlines.

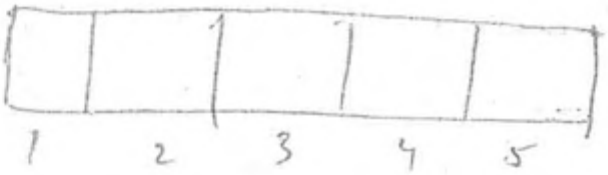
Jobs	J_1	J_2	J_3	J_4.
Profit	P_1	P_2	P_3	P_4.
Deadline.	d_1	d_2	d_3	d_4.

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 $\{J_1\}, \{J_2\}, \{J_3\}, \{J_4\}, \{J_5\}, \{J_6\}$.

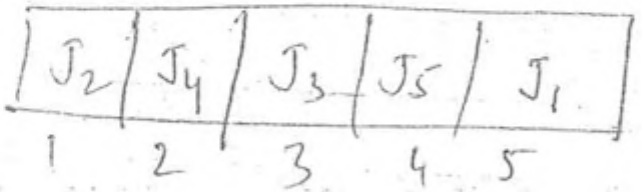


Soln ⇓ Decreasing order of profit

J_4 J_1 J_3 J_2 J_5 J_6

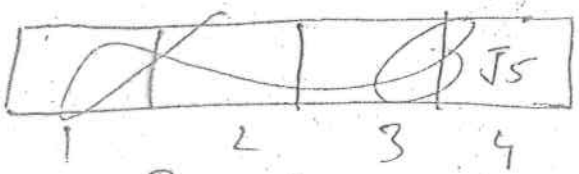
30 24 22 10 12 10

1 5 3 3 4 2



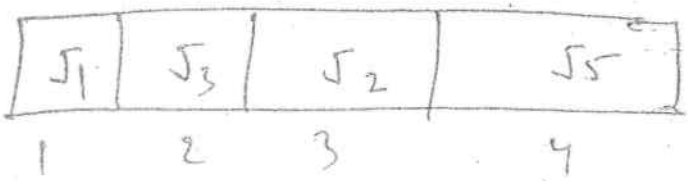
= 10 + 30 + . = 106

Job	J_1	J_2	J_3	J_4	J_5
Profit	10	20	15	5	80
Deadline	3	3	3	4	4



Decreing order of ~~deadline~~ Profit

J_5	J_2	J_3	J_1	J_4
80	20	15	10	5
4	3	3	3	4



$= 10 + 15 + 20 + 80 = 125$

2. $n = 9$

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
Profit	15	20	30 (1)	18 (4)	18 (5)	10	23 (3)	16	25 (2)
Deadline	7	2	5	3	4	5	2	7	3

We are given 9 Jobs. Execution of each Job requires 1 unit of time.

(1) Which one of the following is true.

a) all Jobs are completed.

b) J_1 and J_6 are left.

c) J_5 & J_0 " left only.

d) J_4 & J_6 "

(2) max^m Profit

a) 147 b) 165

c) 167 d) 160.

Soln

Decreasing order of Profit

J_3	J_9	J_7	J_2	J_4	J_5	J_8	J_1	J_6
1								
X 30	X 25	X 23	X 20	10	X 10	~~16~~	15	10
1	1	1	1				X	
5	3	2	2	3 ✓	4	7	7	5 ✓

array ⇒

J_2	J_7	J_9	J_5	J_3	J_1	J_8
1	2	3	4	5	6	7

⇒ 147

Algo

(1) Sort all the Jobs in decreasing order of profits. ⇒ O(n log n)

(2) Find max^m deadlines in the given array of n jobs. O(n)

n-deadline and take the same size array and start from RHS of the array.

3. For every slot I apply linear search to find a job which contain deadline $\geq d$

$O(n^2)$

$O(n^2)$

Huffman Coding

(1) Binary tree
(2) Encoding Technic
(3) Data compression mechanism.

eg

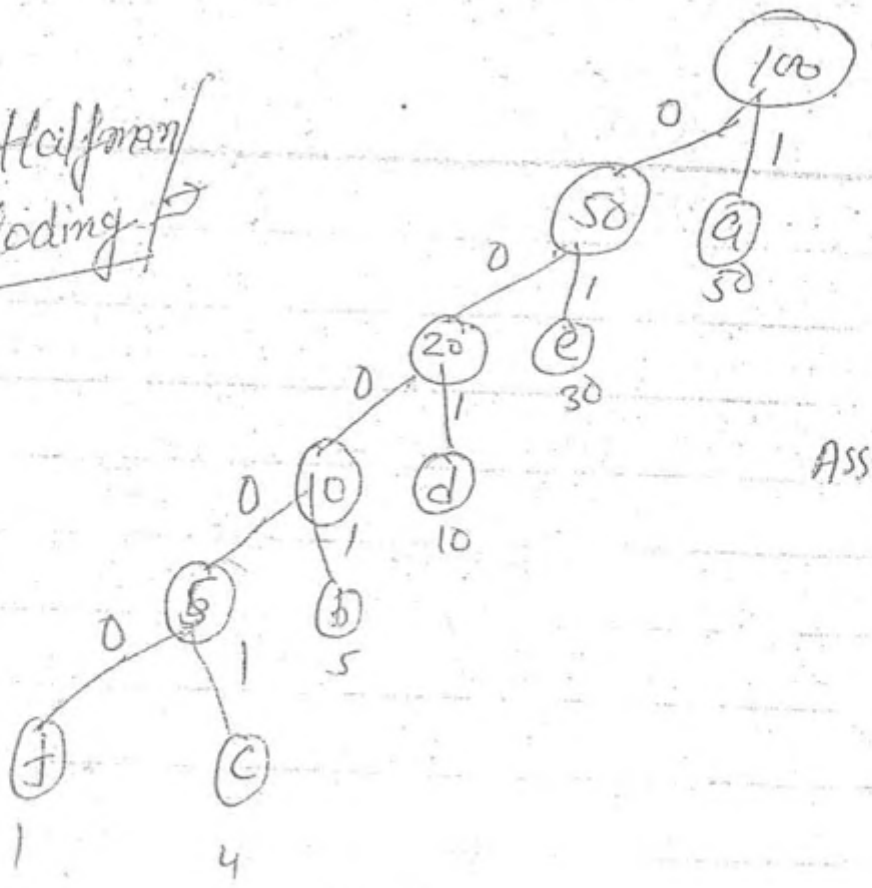
100-Character

	A (Asca)	3-bit (not ASCII)
a — 50	a - 50 x 7	a - 000 = 50 x 3
b — 5	b - 5 x 7	b - 001 = 5 x 3
c — 4	c - 4 x 7	c - 010 = 4 x 3
d — 10	d - 10 x 7	d - 011 = 10 x 3
e — 30	e - 30 x 7	e - 100 = 30 x 3
f — 1	f - 1 x 7	f - 101 = 1 x 3
	700 bytes.	300 bits

M → 0/1 Destination → Compress

Computer

2. Huffman coding



Assume left = 0

Right = 1

a = 1

b = 0 0 0 1

c = 0 0 0 0 1

d = 0 0 1

e = 0 1

f = 0 0 0 0 0

after 2nd compression

a - 1 * 50 = 50

b = 5 x 4 = 20

c = 4 x 5 = 20

d = 10 x 3 = 30

e = 30 x 2 = 60

f = 1 x 5 = 5

185

This is called Huffman coding

185 bit to perform for 100 char

100 char = 185 bits.

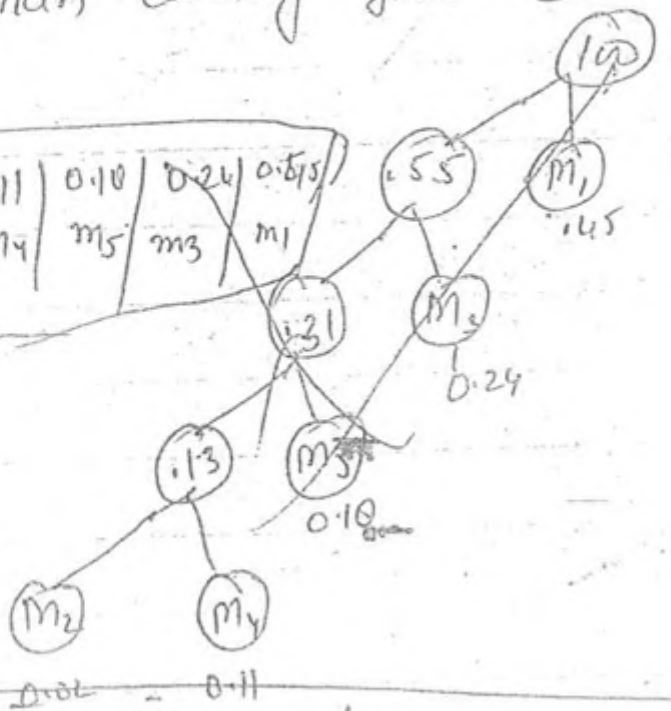
1 = $\frac{185}{100}$ = 1.85 bit/msg

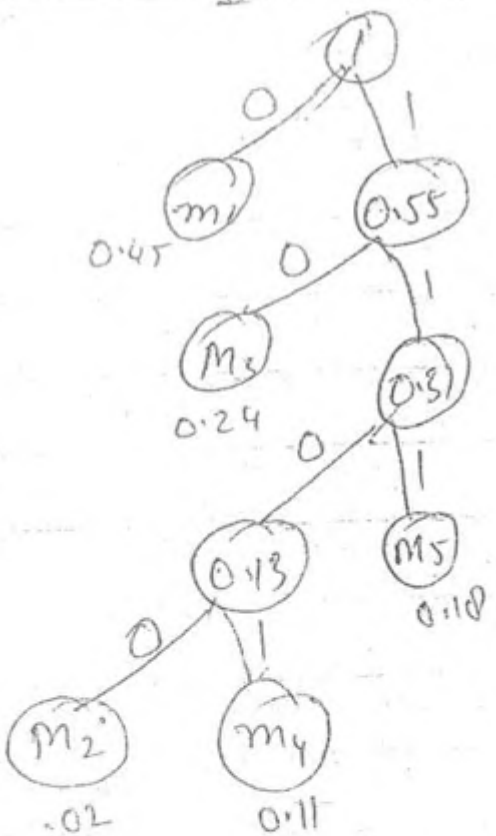
Message $M = (m_1, m_2, m_3, m_4, m_5)$

0.45 0.02 0.24 0.11 0.18

Using Huffman coding find Encoded form of Messages.

Id/n	0.02	0.11	0.18	0.24	0.45
	m_2	m_4	m_5	m_3	m_1





$m_1 = 0$

$m_2 = 1100$

$m_3 = 10$

$m_4 = 1101$

$m_5 = 111$

Total no of bit to transfer

$= 0.45 \times 1 + 0.02 \times 4 + 0.24 \times 2 + 0.11 \times 4 + 0.18 \times 3$

$= 1.99$ bits.

Avg no of bit to transfer

$= \frac{1.99}{1} = 1.99$ bit/msg

2. Suppose the letters → a, b, c, d and e and their Probability → $\frac{1}{2}$ $\frac{1}{2^2}$ $\frac{1}{2^3}$ $\frac{1}{2^4}$ and $\frac{1}{25}$ respec

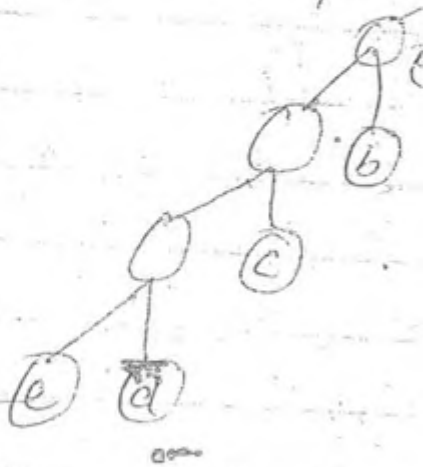
(1) Which one of the following is Huffman coding for the letters a, b, c, d & e respectiv

a) 11, 10, 011, 010, 001

b) 0, 10, 110, 1110, 1111

c) 1, 10, 01, 001, 0001

d) 110, 100, 010, 000, 001



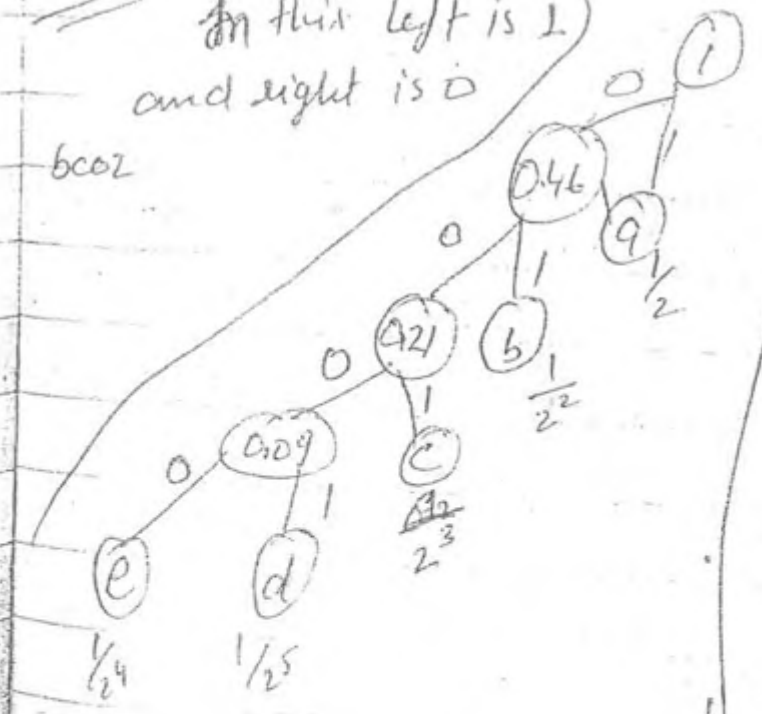
(2) What is the avg no of bits are required per message.

a) 3 b) 3.75 c) 2.10 d) 1.78.

Sol^n

(In this left is 1) and right is 0

bcoz



a = 0
b = 10
c = 110
d = 1110
e = 1111

Avg no of bits required

$$\sum_{i=1}^{n} P_i * b_i$$

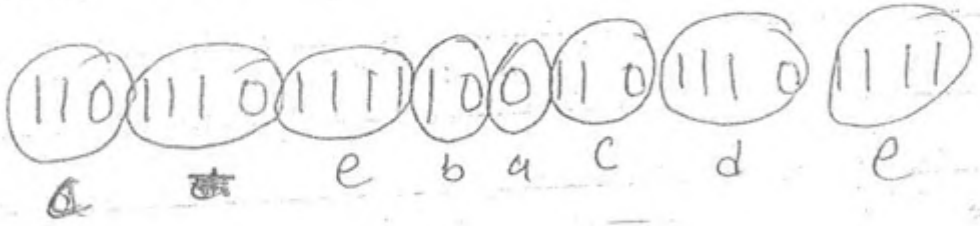
P_i = Probability of msg M_i

b_i = # of bits to represent msg M_i

$= \frac{1}{2} \times 1 + \frac{1}{2^2} \times 2 + \frac{1}{2^3} \times 3 + \frac{1}{2^4} \times 4 + \frac{1}{2^5} \times 5$

$= 1.78$ bits/msg.

3. If the encoded msg is 1101111011111001101110111 what will be the equivalent Decoded msg.

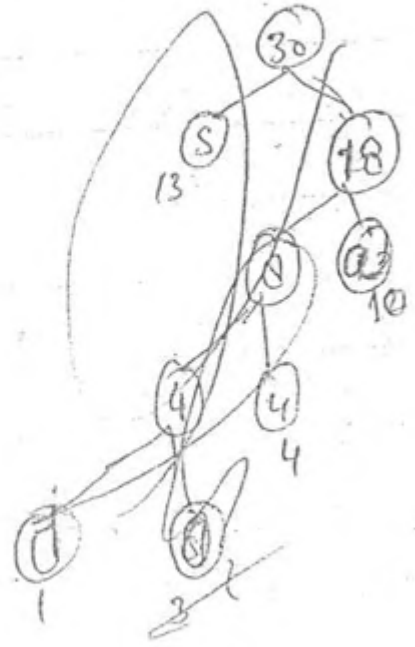


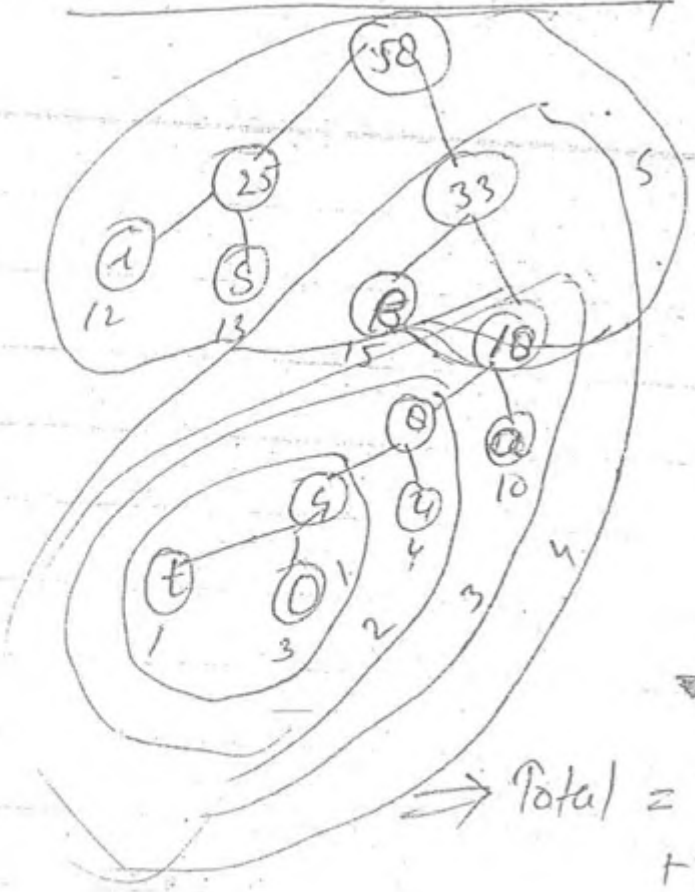
4. A file contain char a, e, i, o, u, s and d.
and frequcy 10 15 12 3 4 13 1

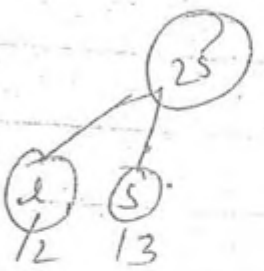
If we use Huffman coding for Data coding what will be the avg code length?

a) 146/58 b) 144/58 c) 140/58 d) none.

Soln







$\Rightarrow$ Total = $10\times3 + 15\times2 + 12\times2 + 3\times5 + 13\times2 + 1\times5$

$= 146$

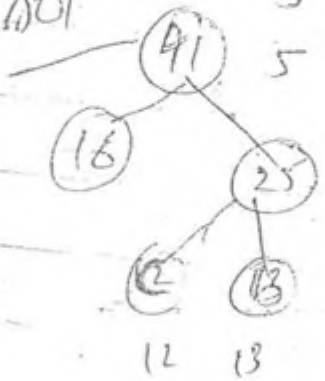
$Avg = 146/58$

P Consider the frequencies for the chars. a, b, c, d, e and f

a	b	c	d	e	f
45	13	12	16	9	5

If we use Huffman code for Data ~~error~~ compression. What will be the avg code length.

Sol^n



f	e	c	b	a	d.
5	9	12	13	15	16

(a) $\frac{220}{100}$ (b) $\frac{222}{100}$ (c) $\frac{224}{100}$ ✓

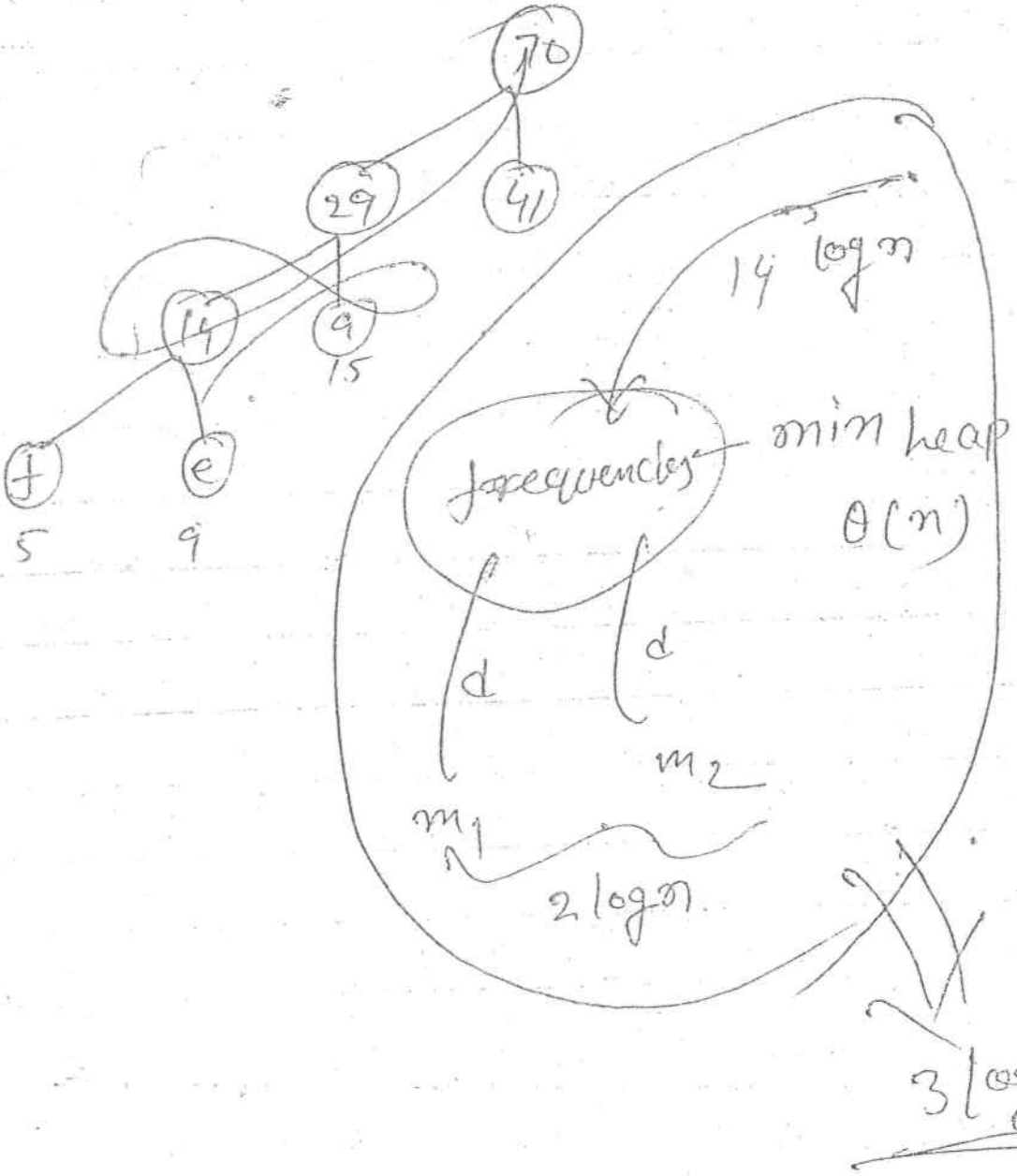
(d) none

Service no 7017

Name DLH-LKC

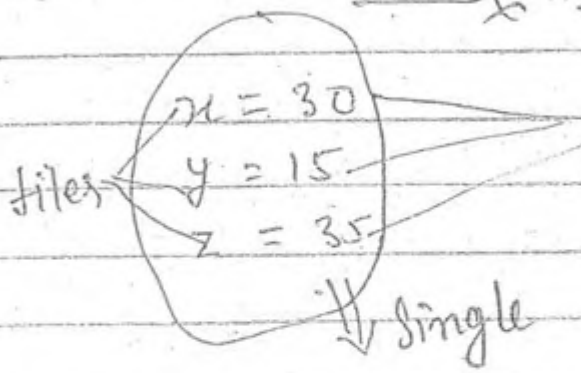
Type - Sleeper Coach

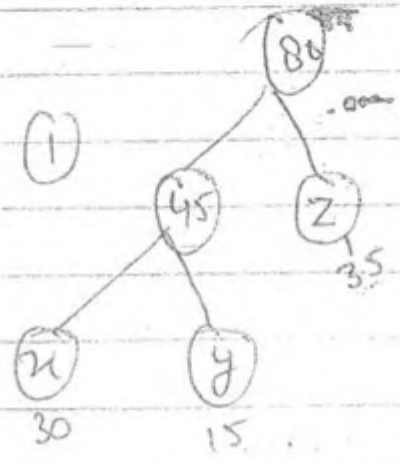
7:30 - 8:15

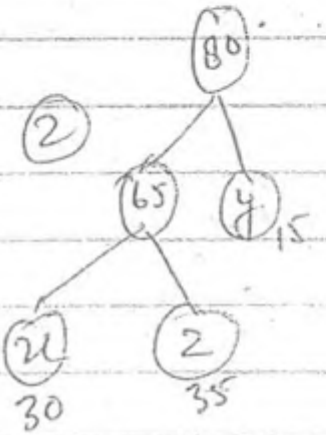


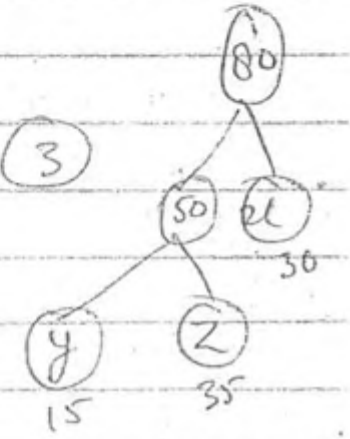
Tuesday
19/07/011

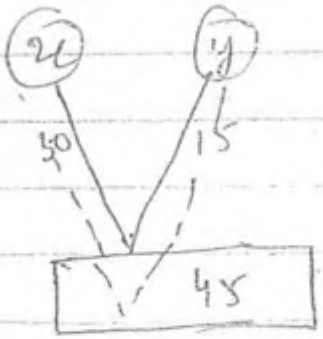
Optimal Merge Pattern









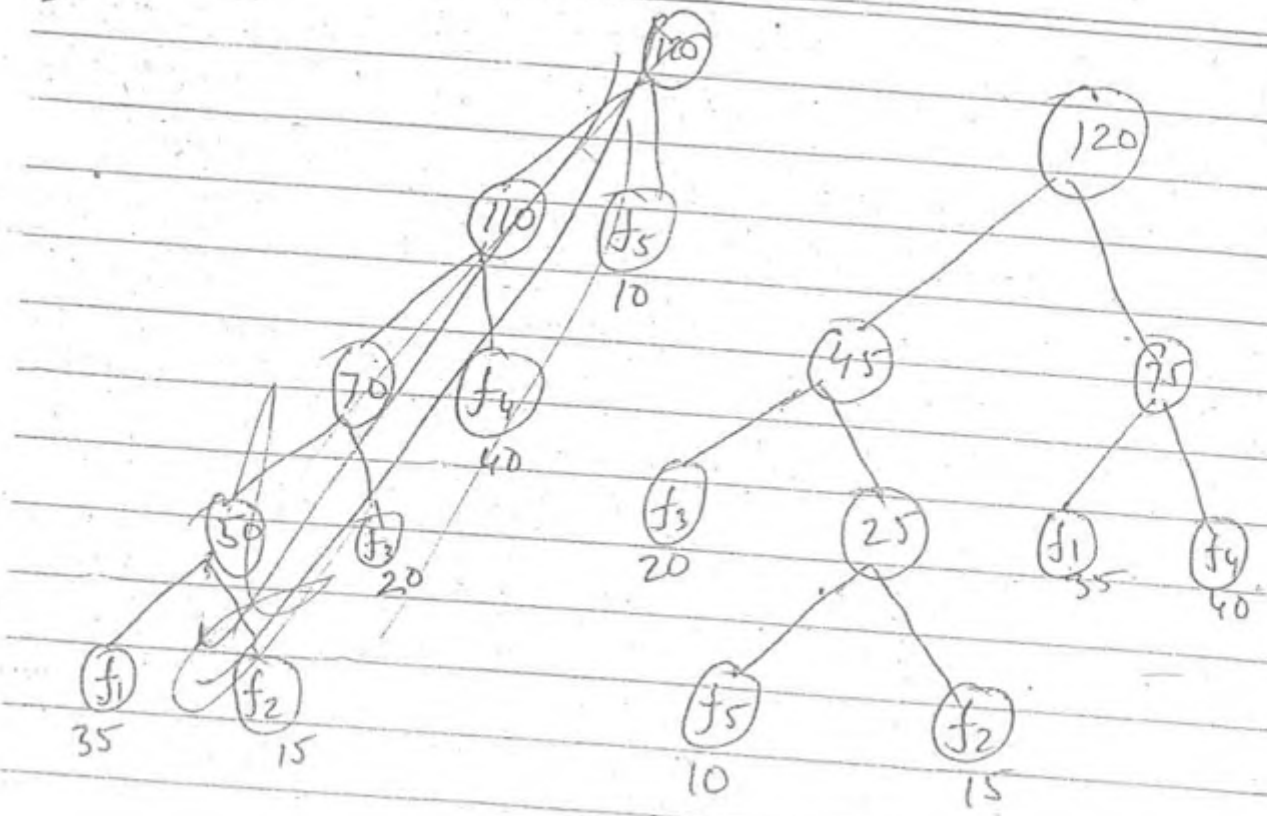


no of record movement

Q. The mimm no of record movement to merge 5-files

$f_1 = 35$, $f_2 = 15$, $f_3 = 20$, $f_4 = 40$, $f_5 = 10$

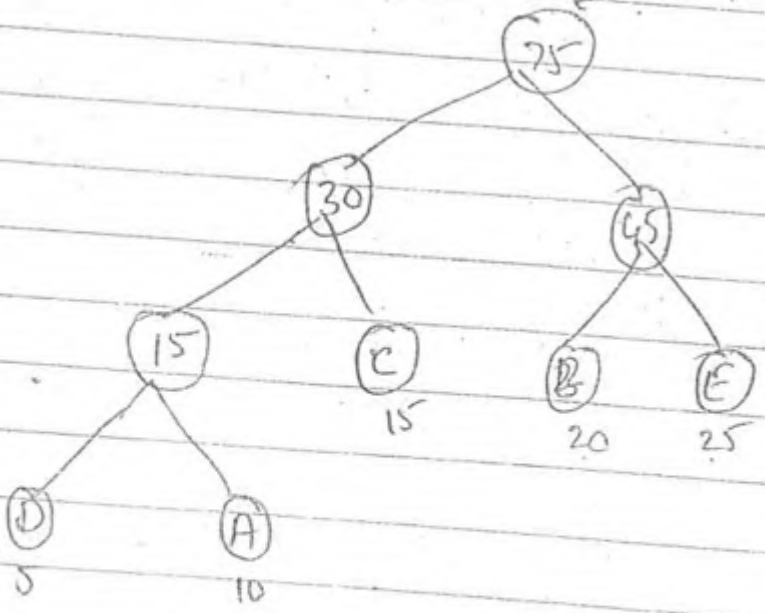
Soln



$25 + 45 + 75 + 120 = 265$

= Min no of Record mgt required to merge files.

$A = 10$, $B = 20$, $C = 15$, $D = 5$, $E = 25$



$= 15 + 30 + 45 + 75$

$= 165$

$4 \times 3 \log n$

$(n-1) 3 \log n$

$\Rightarrow O(n \log n)$

Conclusion

(1) Knapsack ⇒ $O(n \log n)$

(2) MST
- Kruskal ⇒ $e \log v$
- Prims ⇒ $(V+E) \log v$ (B. Min heap)
  or
  $E + V \log v$ (F. min heap)

(3) Job sequencing with Deadline ⇒ $O(n^2)$

(4) Huffman Coding ⇒ $O(n \log n)$

(5) Optimal merge Pattern ⇒ $O(n \log n)$

(6) Single Source Shortest Path
- (i) Dijkstra's ⇒ $(V+E) \log V$ OR $E + V \log V$
- (ii) Bellman-ford ⇒ $O(VE)$

Dynamic Programming (Time is over)

	1st	2nd	3rd	4th
1-	85 %	Fail	85 %	85 %
2-	80 %	80 %	80 %	80 % → 80 (topper)
3	84 %	85 %	Fail	86 %

Dynamic programming will give always optimal solution Greed may not be 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

Subsequence ⇒ A Subsequence of a given sequence is just the given sequence in which 0 (or) more elements left out.

eg $x = (A, B, C, B, D, A, B)$
$\quad\quad 1\ 2\ 3\ 4\ 5\ 6\ 7$

$S_1 = (A, A, B)$ ⇒ Subsequence.
$\quad\quad 1\ 6\ 7$

$S_2 = (A, B, B)$ ⇒ Subsequence
$\quad\quad 1\ 4\ 7$

$S_3 = (C, D, B)$ ⇒ Subsequence.
$\quad\quad 3\ 5\ 7$

$S_4 = (A, A, B, C)$ ✗
$\quad\quad 1\ 6\ 7\ 3$

$S_5 = (A\ B\ C\ B\ D)$ ⇒ Subsequence
$\quad\quad 1\ 2\ 3\ 4\ 5$

Common Subsequence: Given two sequence x and y we say that a sequence z is a common subsequence of x and y if z is a subsequence of both x and y.

eg $x = (A, B, B, A, B, D)$

$y = (A, A, B, B, D, D)$

$CS_1 = (A, A, D)$ — common subsequence.

$CS_2 = (A, A, B, D)$ — " "

$CS_3 = (A, B, B, D)$ — " "

$CS_4 = (\quad)$ ✗

In the above problem CS_2 and CS_3 are the longest common subsequence for the given sequence x and y.

eg $x = (A, B, C, B, D, A, B)$

$y = (B, D, C, A, B, A)$

$CS_1 = (B, C, A, B)$ — longest common subsequence

$CS_2 = \{A\}$

$CS_3 = \{A, B\}$

$CS_4 = \{B, C, A\}$

$CS_5 = \{\phi\}$

~~$x = (A, B, C, B, D, A, B)$~~

~~$y = (B$~~

$x = (a_1, a_2, a_3, a_4 \ldots\ldots a_m)$

$y = (b_1, b_2, b_3 \ldots\ldots b_n)$

LCS

$x = (A, B, C)$

$2^3 \Rightarrow$ { } | {A, B} | {A, B, C}

{A}

{A, C}

{B}

{B, C}

{C}

(1) Find all possible subsets to 1st subsequence $x(1 \cdots m)$

$\Rightarrow 2^m$

(2) For every subset of x check that subset is there in y (or) not. $\nearrow n$

$\Downarrow$

$O(2^m * n)$

using Brute force: $O(2^m * n) \Rightarrow$ Exponential time

$\Downarrow$

(In technical term means algo is very slow)

\# $x = (\overset{1}{A}, \overset{2}{A}, \overset{3}{A}, \overset{4}{A}, \overset{5}{A})$

$y = (\underset{1}{A}, \underset{2}{A}, \underset{3}{A}, \underset{4}{A}, \underset{5}{A})$

longest common subsequence $(5,5) = 1 + LCS(4,4)$

$LCS(4,4) = 1 + LCS(3,3)$

$LCS(3,3) = 1 + LCS(2,2)$

$LCS(2,2) = 1 + LCS(1,1)$

$LCS(1,1) = 1 + \underline{LCS(0,0)} \rightarrow 0$

$LCS(n,n) = 1 + LCS(n-1, n-1)$ if $x[n] == y(n)$

eg: $x = (A, A\ A\ B\ A\ A\ A)$

$y = (A\ A\ A\ A\ A\ A\ A)$

Sol^n: $LCS(7,7) = 1 + LCS(6,6)$

$\Downarrow$

$1 + LCS(5,5)$

$\Downarrow$

$1 + LCS(4,4)$

$\Downarrow$

$LCS(3,4)$

b ←

$\Downarrow$

$1 + LCS(2,3)$

$\Downarrow$

$1 + LCS(1,2)$

$\Downarrow$

$1 + \cancel{LCS(0,1)}$

0

eg:

1 2 3 4 5 6 7

$x = (B, B, B, B, A, A, A)$

$x = (B, B, B, A, A, A, A)$

1 2 3 4 5 6 7

$LCS(7,7) = 1 + LCS(6,6)$

$\Downarrow$

$1 + LCS(5,5)$

$\Downarrow$

$1 + LCS(4,4)$

$\Downarrow$

$LCS(4,3)$

b ←

$\Downarrow$

$1 + LCS(3,2)$

$\Downarrow$

$1 + LCS(2,1)$

$\Downarrow$

$1 + \cancel{LCS(1,0)}$

0

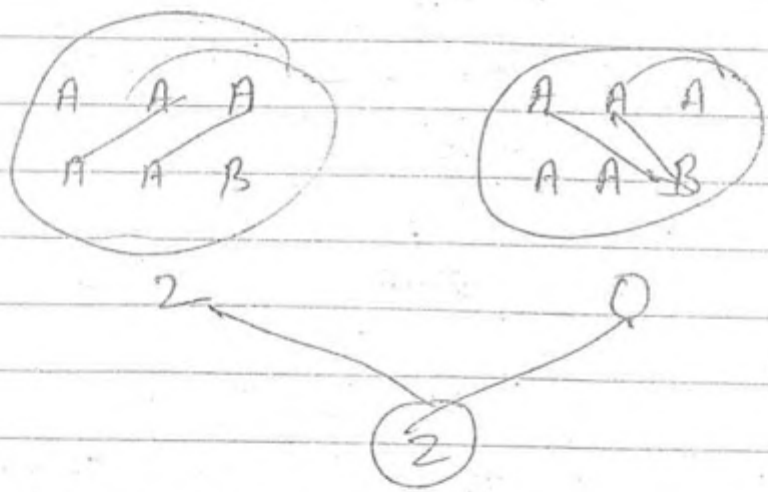
eg $x =$ A B A B

$y =$ B A B D

If match is not found then dont decrement both.

$$LCS(i,j) = \begin{cases} 0 & \text{if } i=0 \text{ (or) } j=0 \\ 1 + LCS(i-1, j-1) & \text{if } x[i] == y[j] \\ \max\begin{pmatrix} LCS(i-1, j) \\ LCS(i, j-1) \end{pmatrix} & \text{if } x[i] \neq y[j] \end{cases}$$

eg



Optimal Solution

(i) Optimal Solution of original problem contains optimal sol^n of subproblem.

\# Write a reccurcive "c" Program to LCS(i,j

```
LCS(i,j)
{
  if(i==0 (or) j==0)
      return(0);
  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);
    }
  }
}
```

exponential
$O(2^n)$

worst

eg

Find LCS for the following Problem.

	1	2	3	4
x =	A	B	B	A
y =	B	A	A	B
	1	2	3	4

Soln

$$LCS(4,4) = \max\begin{Bmatrix} LCS(3,3) \\ LCS(4,3) \end{Bmatrix}$$

$$= 2$$

$$LCS(3,4) = 1 + LCS(2,3)$$

$$LCS(2,3) = \max\begin{Bmatrix} LCS(1,3) \\ LCS(2,2) \end{Bmatrix}$$

$$LCS(1,3) = 1 + LCS(0,2)$$

$$LCS(0,2) = 0$$

$$LCS(2,2) = \max\begin{Bmatrix} LCS(1,2) \\ LCS(2,1) \end{Bmatrix}$$

$$LCS(1,2) = 1 + LCS(0,1)$$

$$LCS(2,1) = 1 + LCS(1,0)$$

(0,1)

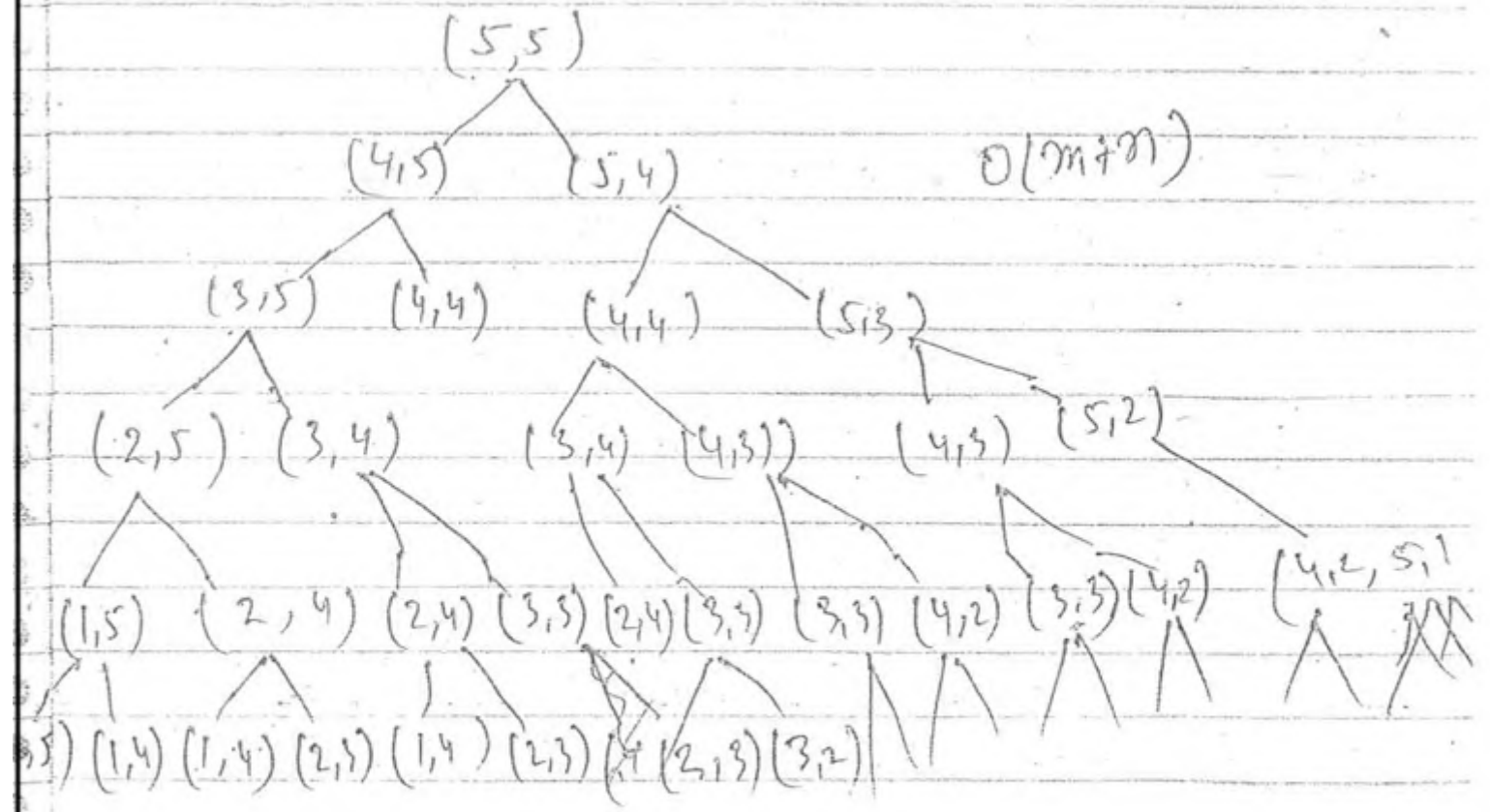
$$LCS(4,3) = 1 + LCS(3,2)$$

$$LCS(3,2) = \max \begin{Bmatrix} LCS(2,2) \\ LCS(3,1) \end{Bmatrix}$$

$$LCS(3,1) = 1 + LCS(2,0)$$

eg $x = (A, A, A, A, A)$ (1 2 3 4 5)

$y = (B, B, B, B, B)$ (1 2 3 4 5)



$O(m+n)$

$3 \Rightarrow 2^3 - 1$

$2^{m+n} = 2^m \cdot 2^n$

2. Overlaping Subproblem ⇒ In most of the recursive program small no of distinct fun^n call are present and more no of repeated subproblem. So Dynamic programming solves only distict subproblem.

Using recursion (Without DP)

⇒ $m+n$ leveled are.

⇒ 2^{m+n} ✓

Using recursion (with DP)

$m=4$		$n=5$	
5	8,5	2,5	1,5
4	3,4	2,4	1,4
3	3 3	2,3	1,3
2	3 2	2,2	1,2
1	3 1	2,1	1,1

⇒ $m \times n$

⇒ $O(m,n)$ $O(1)$

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

Quadratic

Drawback

Dynamic Programmic take more space to store table for every sub problem.

I saw answer this previously.

```
Memorization - LCS(i, j)
{
if (i==0 or (j==0)
    {Table[i,j]=0
    return(Table(i,j)
else
{
if [x[i] == y[j]]
{
if (Table[i-1, j-1] ≠ nil)
return (1+Table[i-1, j-1])
else
{
Table[i-1, j-1] = LCY(i-1, j-1)
return (1+Table[i-1, j-1])
}
}
```

O(mn)

Table

	1	2	3	4	5
1					
2					
3					
4					

Contents are in Table

initially all value of table are nil

Contents are in 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] = LCS(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	∞	0	∞	∞	∞
1	0	∞	1	∞	∞
2	∞	1	∞	2	∞
3	∞	∞	2	∞	3
4	∞	∞	∞	3	∞

initially all are ∞

$$LCS(4,4) = \max\{LCS(3,4), LCS(4,3)\}$$

$$LCS(\overset{3}{3},4) = 1 + LCS(\overset{2}{2},3)$$

$$LCS(\overset{2}{2},3) = 1 + LCS(\overset{1}{1},2)$$

$$LCS(\overset{1}{1},2) = 1 + LCS(0,1)$$

$$LCS(\overset{3}{4},3) = 1 + LCS(\overset{2}{3},2)$$

$$LCS(\overset{2}{3},2) = 1 + LCS(\overset{1}{2},1)$$

$$LCS(\overset{1}{2},1) = 1 + LCS(1,0)$$

While computing LCS Table we can not say in which order we are computing.

eg:

(2) 0/1 Knapsack Problem

$n = 3$

objects.	ob_1	ob_2	ob_3
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) manully

9

eg 2 $n = 3$

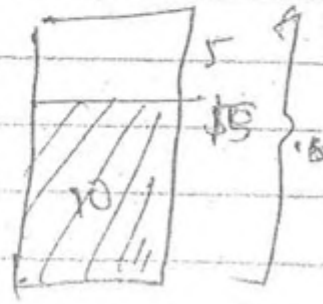
objects	obj_1	obj_2	obj_3
Profit	50	29	33
weight	10	6	7

$M = 15$

$\frac{50}{10} = 5$

$\frac{29}{6} = 4.8$

$\frac{33}{7} = 4.7$



~~profit~~ 50

manually $M = 15$

In DP there is no shortcut beoz it is not give a optimal soln

P $n = 5$

object	ob_1	ob_2	ob_3	ob_4	ob_5
weight	3	1	2	4	6
Profit	7	2	1	6	12

$M = 10$

Sol^n

$$0/1KS(M,N) = \begin{cases} 0 & \text{if } M=0 \text{ (or) } N=0 \\ 0/1KS(M,N-1) & \text{if } M < W[N] \\ \max\{0/1KS(M,N-1),\ 0/1KS(M-W[N],N-1)+P(N)\} & \end{cases}$$

M = Capacity of knapsack
N = no of object

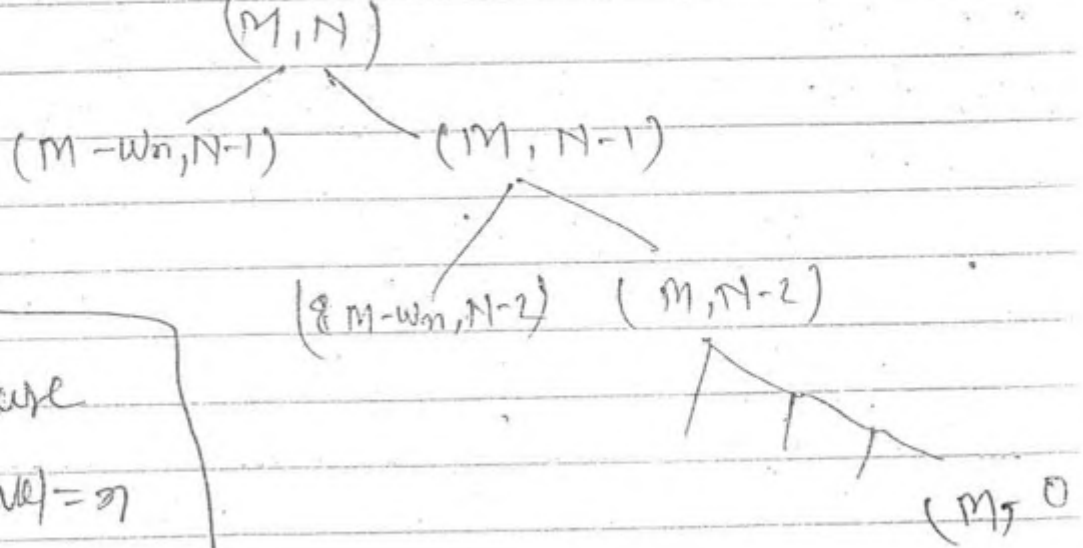
M, N	0/1KS
10 5	
4 4	12
4 3	7
4 2	2
3 1	2

Recursive Program for 0/1 Knapsack

```
01KS(M,N)
{
if ( M==0 (or) N=0 )
   return (0);
else
{
   if (M< W[N])
      return(01KS( M, N-1))
   else
return (max( 01KS( M-W[N] ,N-1) + P[N]
             01KS(M , N-1) +0 ))
}
}
```

Time complexity $O(2^n)$

(A)



In worst case

Level = n

Note ⇒ In the 0/1 Knapsak Program (recursive) Step (A) gives worst case behaviour i.e. every time two possibilities like them 2 times. So binary tree 'n' levels.

So time complexity of 0/1 Knapsack = $O(2^n)$

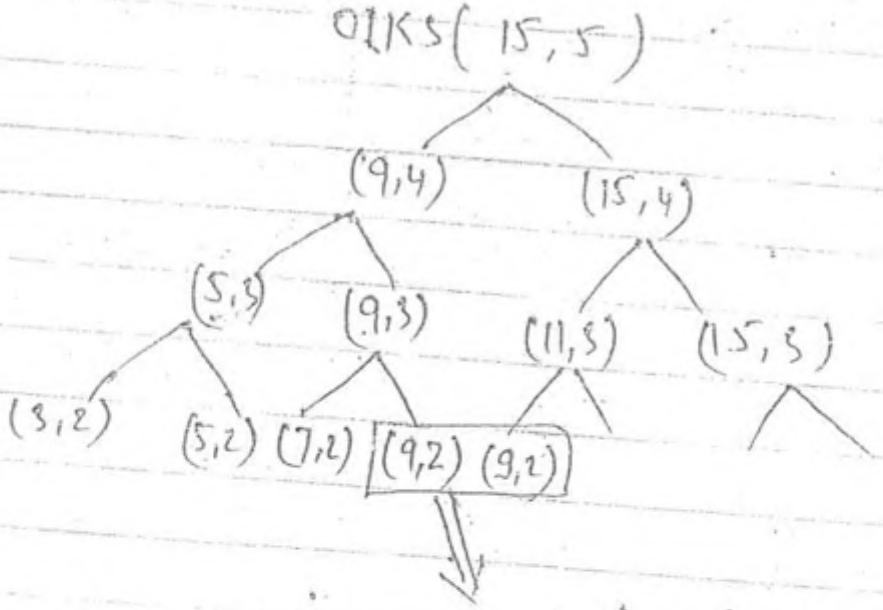
P

M = 15

n = 5

object	Ob_1	Ob_2	Ob_3	Ob_4	Ob_5
weight	3	1	2	4	6
Profit	7	2	1	6	12

Soln



Overlaping Subproblem.

Can be solved only one time in D.P

8

How many distinct subproblems 0/1 KS(M,N) having.

M = 15

N = 5

(15,5)			
(15,4)	(14,4)	(13,4)	12 , 11 . 10 - - - - 0
(15,3)	(14,3)	(13,3)	
(15,2)	(14,2)	(13,2)	
(15,1)	(14,1)	(13,1)	
(15,0)	(14,0)	(13,0)	

the no of distinct subproblem are.

$$0/1KS(M,N) = (M+1)*(N+1)$$

$$= M*N$$

larger

= Exponential

= N.P Complete.

Dynamic Programming 0/1KS

Using Dynamic Programming

```
Memoization - OLKS (M, N)
{
if ( M==0 (or) N==0)
  { return (0);
    table [M,N] = 0
else
{
  if ( M< w[N])
    {
      if ( table [M, N-1] ≠ Nil )
        return ( table [M, N-1])
      else
        {
          table [M, N-1] = OLKS (M, N-1)
          return ( table [M, N-1]
        }
    }
  else
  {
    if ( table [M, N-1] ≠ Nil &&
                table [M - w[N], N-1] ≠ Nil )
return ( max { table [M, N-1]
               table [M - w[N], N-1] + PN] } )
```

```
else
{
    if table[M, N-1] == NIL)
        table[M, N-1] = 01KS(M, N-1)
    if(table(M-W[N], N-1] == NIL
        Table[M-W[N], N-1) = 01KS(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)

$S = (10, 50, 30, 80, 70, 40, 100)$

$\quad\quad\quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7$

$M = 210$

$S_1 = \{30, 80, 100\}$

$S_2 = \{70, 40, 100\}$

$S_3 = \{10, 30, 70, 100\}$

Recurrence Relation

$$SOS(N, M) = \begin{cases} 0 & \text{if } M = 0 \\ \phi & \text{if } N = 0 \\ SOS(N-1, M) & \text{if } W[N] > M \\ \max \begin{cases} SOS(N-1, M - W[N]) \cup W[N] \\ SOS(N-1, M) \end{cases} & \text{if } W[N] < M \end{cases}$$

Max

mplete binary tree with
n-levels $\Rightarrow O(2^n)$

NM

0

4 0 0U70U40U100

7, 210

6, 110

5, 70

4, 0

SOS(N-1), M-W(N)] U W[N])

= 0U70U40U100

210

Using Dynamic Programming

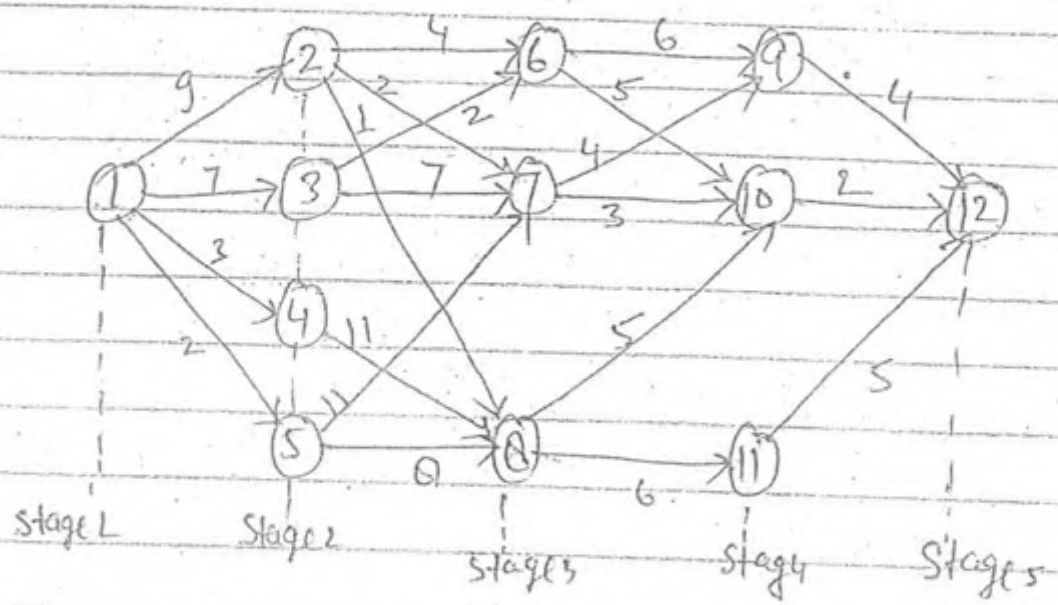
MXN ⇒ O(MN)

⇓

larger

↳ Exponential.

Multi Stage Graph.



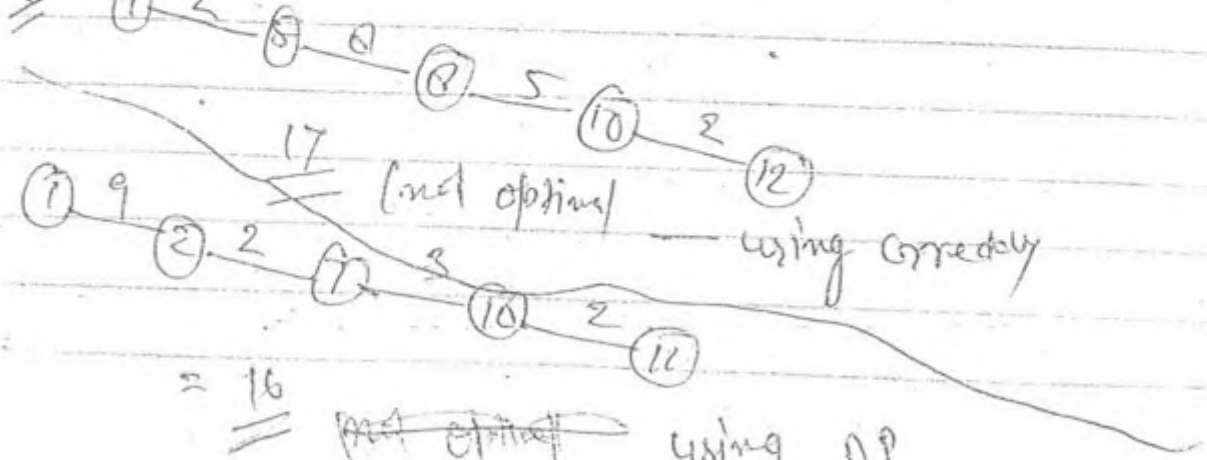
MSG(1,1)

MSG(2,5)

MSG(3,7)

MSG(Si, Vi) = The minm cost required from stage Si to destination VJ.

eg



$O(V+E)$

$$MSG(1,1) = \min \begin{cases} c(1,2) + MSG(2,2) \\ c(1,3) + MSG(2,5) \\ c(1,4) + MSG(2,4) \\ c(1,5) + MSG(2,5) \end{cases}$$

16

$$MSG(2,2) = \min \begin{cases} c(2,6) + MSG(3,6) \\ c(2,7) + MSG(3,7) \\ c(2,8) + MSG(3,8) \end{cases}$$

$$MSG(3,6) = \min \begin{cases} c(6,9) + MSG(4,9) \\ c(6,10) + MSG(4,10) \end{cases}$$

$$MSG(4,9) = \min \{ c(9,12) \}$$

$$MSG(4,10) = \min \{ c(10,12) \}$$

$$MSG(3,7) = \min \begin{cases} c(7,9) + MSG(4,9) \\ c(7,10) + MSG(4,10) \end{cases}$$

5

$$MSG(3,8) = \min \begin{cases} c(8,10) + MSG(4,10) \\ c(8,11) + MSG(4,11) \end{cases}$$

$$MSG(4,11) = c(11,12)$$

$$MSG(2,3) = \min \begin{cases} c(3,6) + MSG(3,6) \\ c(3,7) + MSG(3,7) \end{cases}$$

$$MSG(2,4) = c(4,8) + MSG(3,8)$$

$$MSG(2,5) = \min \begin{cases} c(5,7) + MSG(3,7) \\ c(5,8) + MSG(3,8) \end{cases}$$

(1) 1 → 2 → 7 → 10 → 12 (edge costs 9, 2, 3, 2)

$= 16$

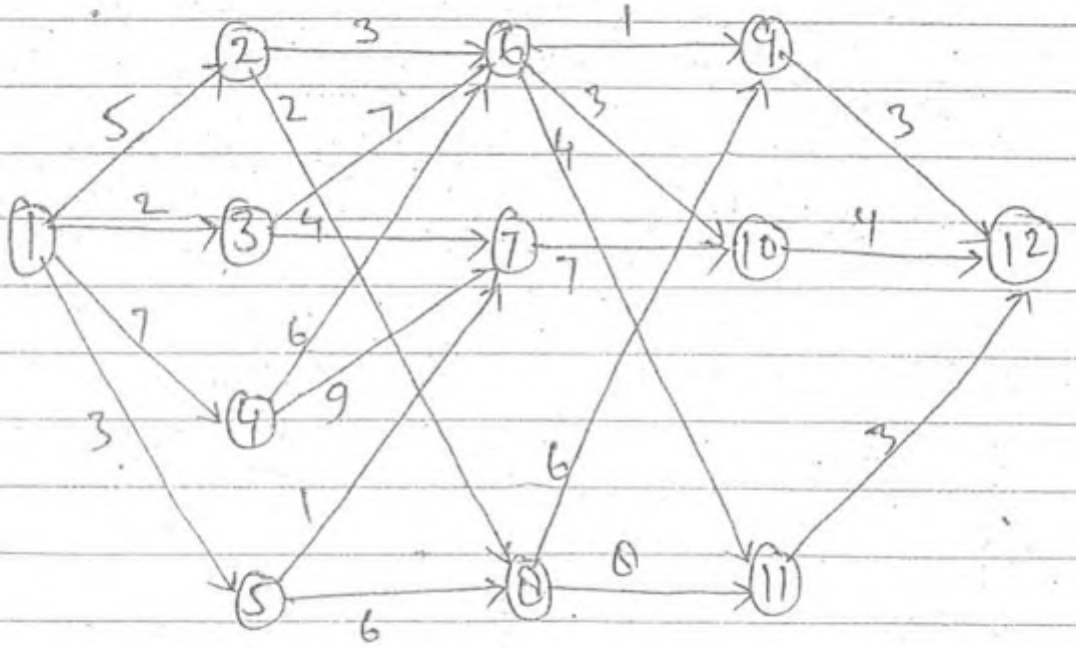
(2) 1 → 3 → 6 → 10 → 12 (edge costs 7, 2, 5, 2)

$= 16$

Note- Every edge is covered only once.
At every vertex covered only once.

> no of fun^{n} call = no of vertex

P //



$$MSG(1,1) = \min \begin{cases} c(\overrightarrow{1,2}) + MSG(\overrightarrow{2,4}) \\ c(\overrightarrow{1,3}) + MSG(\overrightarrow{2,3}) \\ c(\overrightarrow{1,4}) + MSG(\overrightarrow{2,4}) \\ c(\overrightarrow{1,5}) + MSG(\overrightarrow{2,5}) \end{cases}$$

5, 7, 2, 11, 7, 10, 3, 12

12 //

$$MSG(2,2) = \min \begin{cases} c(2,4) + MSG(3,6) \\ c(2,8) + MSG(3,8) \end{cases}$$

$$MSG(3,6) = \min \begin{cases} c(6,9) + MSG(4,9) \\ c(6,10) + MSG(4,10) \\ c(6,11) + MSG(4,11) \end{cases}$$

$$MSG(3,8) = \min \begin{cases} c(8,9) + MSG(4,9) \\ c(8,11) + MSG(4,11) \end{cases}$$

$$MSG(2,3) = \min \begin{cases} c(3,6) + MSG(3,6) \\ c(3,7) + MSG(3,7) \end{cases}$$

$$MSG(3,7) = c(7,10) + MSG(4,10)$$

$$MSG(2,4) = \min \begin{cases} c(4,6) + MSG(3,6) \\ c(4,7) + MSG(3,7) \end{cases}$$

$$MSG(2,5) = \min \begin{cases} \underset{I}{C(5,7)} + \underset{II}{MSG(3,7)} \\ C(5,8) + MSG(3,8) \end{cases}$$

$$\underset{12}{\quad} \qquad\qquad 6 \qquad\qquad 9$$

Recurrence Relation

$$MSG(S_i, V_J) = \min \begin{cases} C(V_J, D) & \text{if } S_i = S_f - 1 \\ C[V_J, K) + MSG(S_i + 1, K) & \\ \forall K \in S_{i+1} & \\ (V_J, K) \in E & \end{cases}$$

D = Destination.

S_f = Final stage.

Eular Graph && Hamiltonian Graph

Path: Sequence of zero (or) more edge

Open Path: $V_S \neq V_E$

Starting vertex is not equeal to ending vertex.

Closed Path: $V_S = V_E$

Starting vertex is equal to ending vertex.

Eular Path: In the given ~~Graph~~ path every edge of the given graph is covered exactly once.

Eular Circuit: It is an Eular path in which $V_S = V_E$ is called Eular Circuit.

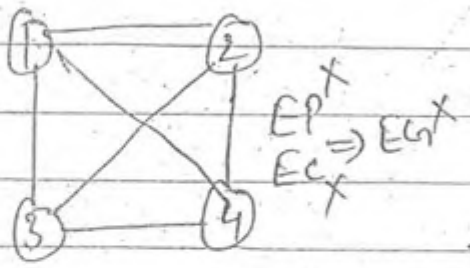
Hamiltonian Path: In the given path every vertex of the given graph is covered exactly once.

Hamiltonian Cycle: It is closed Hamiltonian Path.

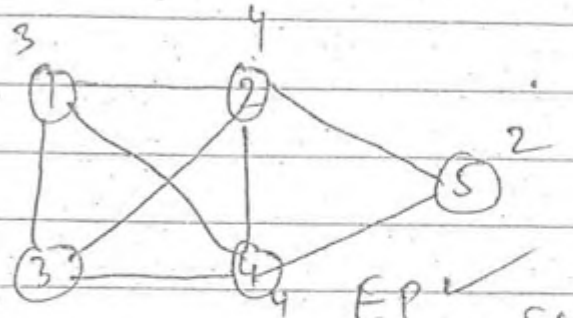
⇓

Hamiltonian Graph.

Q. Consider the following Graph.

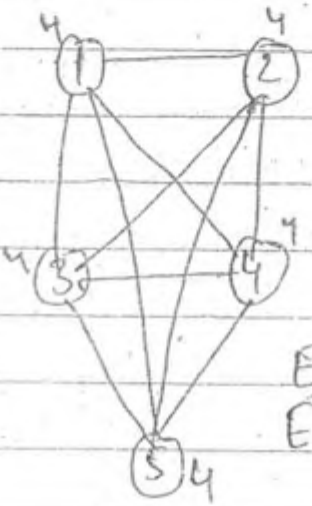


EP ✗
EC ✗
⇒ EG ✗



EP ✓
EC ✗
⇒ EG. ✗

① — ③ ✓
③ — ① ✓
④ — ∝
⑤ — ∝
② — ✗



EP ✓
EC ✓
⇒ EG ✓

In the given graph every vertex degree is even then that graph contain Eular path and Eular Circuit are possible then graph is called Eular Graph.

To check a graph is Eular or not $O(E)$

$= O(V^2)$

P class Problem
bcoz Polynomial time complexity

Note In the Given Graph Contain Exatly two vertex with odd degree. Then too that graph. Eular path is possible but Eular circut is not possible.

To that Eular path one of the odd degree vertex is starting other is ending

Note
In the Given Graph contain more then two vertise with odd degree. Even Eular Path is not possible.

Which one of the fallowing is true.

a) $G \Rightarrow |V| = 10, |E| = 10$ is EG.

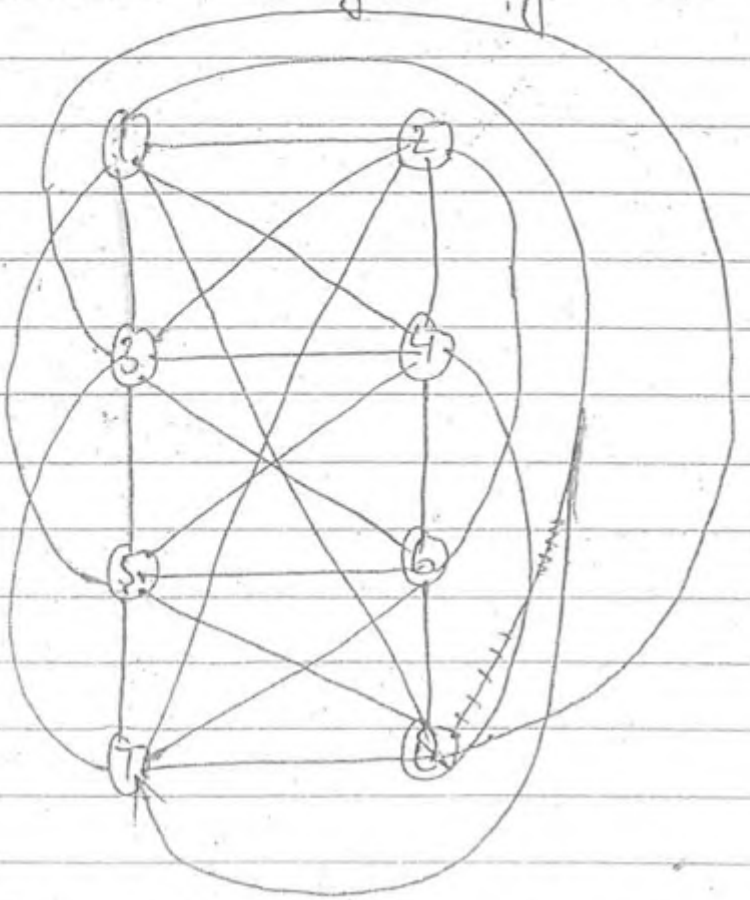
b) $G \Rightarrow |V| = 50, |E| = 100$ is EG.

c) $G \Rightarrow |V| = n, |E| = \frac{n(n-1)}{2}$ when n is even is EG.

d) $G \Rightarrow |V| = n, |E| = \frac{n(n-1)}{2}$ when n is odd is E.G.

Every vertex is even degree.

Q. Consider the following Graph.

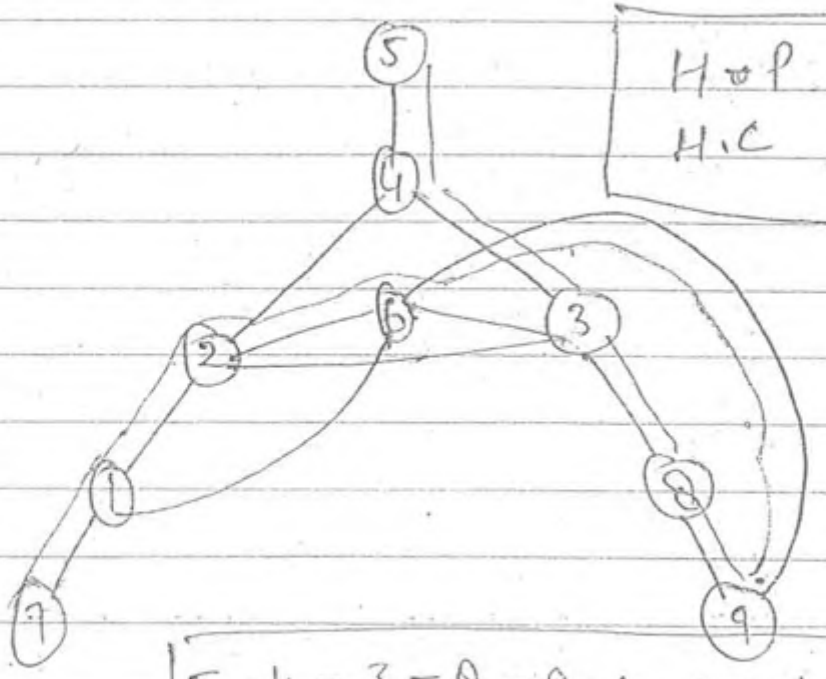


H.P ✓ ⇒ H.G ✓

H.C ✓

1-2-4-6-8-7-5-3-1

Q. The following Graph is H.G or not



H.P ✓ H.G ✗

H.C ✗

5-4-3-8-9-6-2-1-7

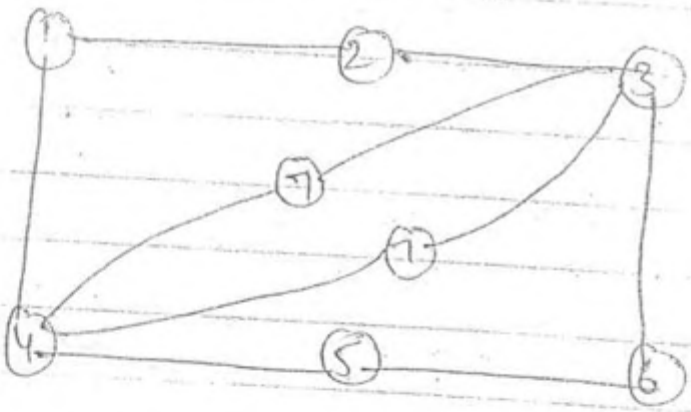
3Sat — 3000

NP Complete

P-class.

Note: To check Given Graph is H.G or not there is not polynomial time.

2. Which one of the following is true.?



EG ✓

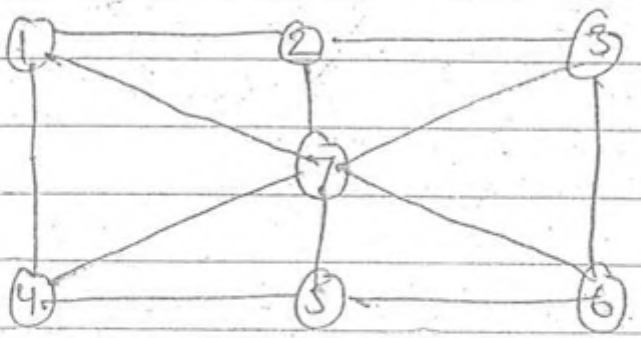
a) G - both E & H

b) ~~G~~ G - E but not H

c) G - H but not E

d) both are not

P// Which one of the following is true?



EC X
EP X
EG X

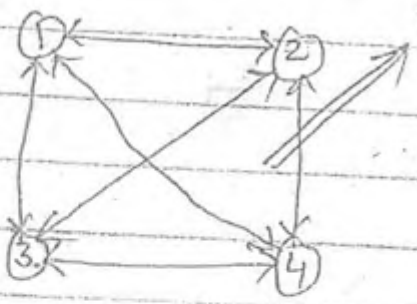
a) G — both E & H

b) G — E but not H

c) G — H but not E

d) both are not

Travelling Sales Person (Hamiltonian tour)



$O(2^n) \Rightarrow$ Exponential (NP-Complete)
(Becoz no overlaping problem)

Travell adject vertex exactly once and the 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

If $V = \{1,2,3,4\}$
initially source = 1
TSP = (1, (2,3,4))

$TSP(s, V-\{s\})$

= The min cost required to go from s to all other remaing vertices in V exactlly once & coming back to source (may be s)

$TSP(1, \overset{35}{(2,3,4)}) = \min \begin{cases} \overset{10}{c(1,2)} + TSP(\overset{25}{2, (3,4)}) \\ \overset{15}{c(1,3)} + TSP(3, \overset{25}{(2,4)}) \\ \underset{20}{c(1,4)} + TSP(\underset{23}{4, (2,3)}) \end{cases}$

Ans 35

$$\overset{25}{TSP(2,(3,4))} = \min \begin{cases} \overset{9}{c(2,3)} + \overset{20}{TSP(3,(4))} \\ \underset{10}{c(2,4)} + \underset{15}{TSP(4,(3))} \end{cases}$$

$$\overset{20}{TSP(3,\{4\})} = \underset{12}{c(3,4)} + \underset{8}{c(4,1)}$$

$$\overset{15}{TSP(4,\{3\})} = \overset{9}{c(4,3)} + \overset{6}{c(3,1)}$$

$$\overset{25}{TSP(3,(2,4))} = \min \begin{cases} \overset{13}{c(3,2)} + \overset{10}{TSP(2,\{4\})} \\ \underset{12}{c(3,4)} + \underset{13}{TSP(4,\{2\})} \end{cases}$$

$$\overset{10}{TSP(2,\{4\})} = \overset{10}{c(2,4)} + \overset{0}{c(4,1)}$$

$$\overset{13}{TSP(4,\{2\})} = \overset{8}{c(4,2)} + \overset{5}{c(2,1)}$$

$$\overset{23}{TSP(4,(2,3))} = \min \begin{cases} \overset{8}{c(4,2)} + \overset{15}{TSP(2,(3))} \\ \underset{9}{c(4,3)} + \underset{10}{TSP(3,(2))} \end{cases}$$

$$\overset{15}{TSP(2,(3))} = \overset{9}{c(2,3)} + \overset{6}{c(3,1)}$$

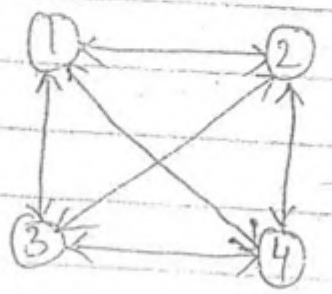
$$\overset{10}{TSP(3,\{2\})} = \overset{13}{c(3,2)} + \overset{5}{c(2,1)}$$

Using Greedy

$$(1) \xrightarrow{10} (2) \xrightarrow{9} (3) \xrightarrow{12} (4) \xrightarrow{8} (1)$$

$$= 39$$

P Consider the following Graph.



	1	2	3	4
1	0	5	10	7
2	6	0	11	5
3	8	5	0	6
4	9	4	11	0

Source = 1

$$TSP(1,\{2,3,4\}) = \min \begin{cases} c(1,2) + TSP(2,\{3,4\}) \\ c(1,3) + TSP(3,\{2,4\}) \\ c(1,4) + TSP(4,\{2,3\}) \end{cases}$$

$TSP(1,\{2,3,4\}) = 26$

$TSP(2,\{3,4\}) = 24$

$TSP(3,\{4\}) = 15$

$TSP(4,\{3\}) = 19$

$$TSP(2,\{3,4\}) = \min \begin{cases} c(2,3) + TSP(3,\{4\}) \\ c(2,4) + TSP(4,\{3\}) \end{cases}$$

$c(2,3) = 11$, $TSP(3,\{4\}) = 15$, $c(2,4) = 5$, $TSP(4,\{3\}) = 19$

$$TSP(3,\{4\}) = c(3,4) + c(4,1)$$

$c(3,4) = 6$, $c(4,1) = 9$

$$TSP(4,\{3\}) = c(4,3) + c(3,1)$$

$c(4,3) = 11$, $c(3,1) = 8$

$$TSP(3,\{2,4\}) = \min \begin{cases} c(3,2) + TSP(2,\{4\}) \\ c(3,4) + TSP(4,\{2\}) \end{cases}$$

$$TSP(2,\{4\}) = c(2,4) + c(4,1)$$

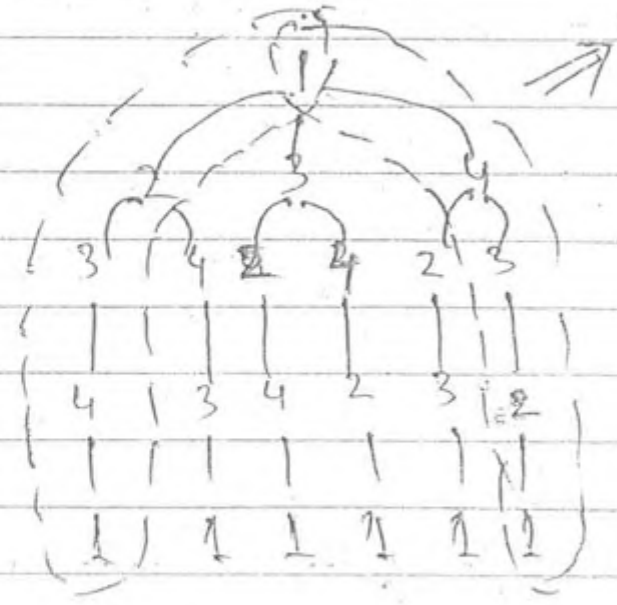
$$TSP(4,\{2\}) = c(4,2) + c(2,1)$$

$$TSP(4,\{2,3\}) = \min \begin{cases} c(4,2) + TSP(2,\{3\}) \\ c(4,3) + TSP(3,\{2\}) \end{cases}$$

$$TSP(2,\{3\}) = c(2,3) + c(3,1)$$

$$TSP(3,\{2\}) = c(3,2) + c(2,1)$$

Hence DP = 26



$O(n^2(2^n))$ // exponential.

Recurrence Relation

$$TSP(s, V') = \min_{1,\{2,3,4\}} \begin{cases} c(s, \text{source}) & \text{if } V' = \phi \\ c(s,K) + TSP(K, V'-K) & \forall K \in V' \\ & (s,K) \in E \end{cases}$$

Dry Run

$TSP(s, V')$	$c(s,K)$	$TSP(K, V'-K)$
$1, \{2,3,4\}$	1,3	3 $\{2,4\}$
$2\ \{2,4\}$	3,4	4 $\{2\}$
$4, \{2\}$	1,2	2 $\{\phi\}$.
$2, \phi$		

(2) Matrix chain Multiplication (MCM)

$A_{2\times3}$, $B_{3\times2}$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

$C = A * B$

$2\times2 \quad 2\times3 \quad 3\times2$

$= \begin{bmatrix} — & — \\ — & — \end{bmatrix}_{2\times2}$

eg $A_{5\times10}$ $B_{10\times20}$

$C = A * B$

$C_{5\times20} \Rightarrow 100$

$5 \times 20 \times 10$

$\Downarrow$

1000

$A_{m\times n}$ $B_{n\times p}$

$C = AB$

$C_{m\times p} \Rightarrow n$

$\Rightarrow mnp$

Let $A_{n\times n}$, $B_{n\times n}$ Matrix them

if $C = A \times B$

then C size is $C_{n\times n}$

to get every element of c n-multiplied

total multiplication are $= n^2 \times n$

$\Rightarrow n^3$

eg $A_{5\times10}$, $B_{10\times5}$, $C_{5\times20}$

(AB)C ⇒ 750

ABC, how many min^{m} no of multiplications are required then to find

Soln (AB)C

$A_{5\times10}$, $B_{10\times5}$, $C_{5\times20}$

$(AB)_{5\times5}$ $C_{5\times20}$

$((AB)C)_{5\times20\times5}$

= 250
500
750

$A_{5\times10}$, $B_{10\times5}$, $C_{5\times20}$

$A_{5\times10}$ $BC_{10\times20}$

$(ABC)_{5\times20\times10}$

= 1000
1000
2000

eg $A = \begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix}$, $B = \begin{Bmatrix} 2 & 3 \\ 4 & 5 \end{Bmatrix}$, $C = \begin{Bmatrix} 5 & 4 \\ 5 & 6 \end{Bmatrix}$.

(A(BC))

$A = \begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix}$, $BC \begin{Bmatrix} 21 & 26 \\ 37 & 46 \end{Bmatrix}$

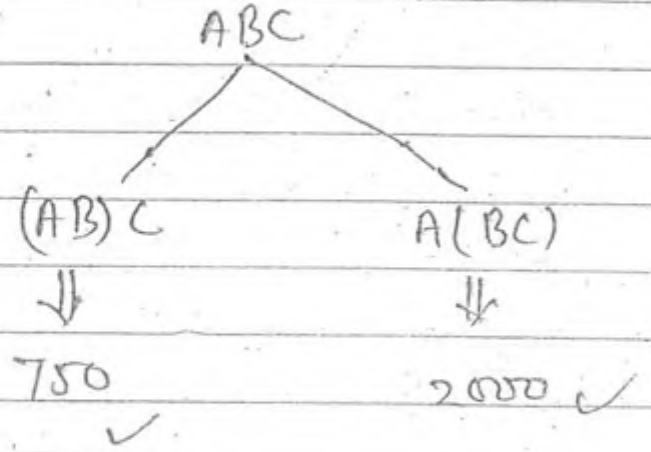
$A(BC) = \begin{Bmatrix} 95 & 110 \\ 211 & 262 \end{Bmatrix}$

((AB)C)

$AB = \begin{bmatrix} 10 & 13 \\ 22 & 29 \end{bmatrix}$ $C = \begin{bmatrix} 5 & 4 \\ 5 & 6 \end{bmatrix}$

$(AB)C = \begin{bmatrix} 95 & 110 \\ 211 & 262 \end{bmatrix}$.

Note – Matrix multiplication satisfied associative property

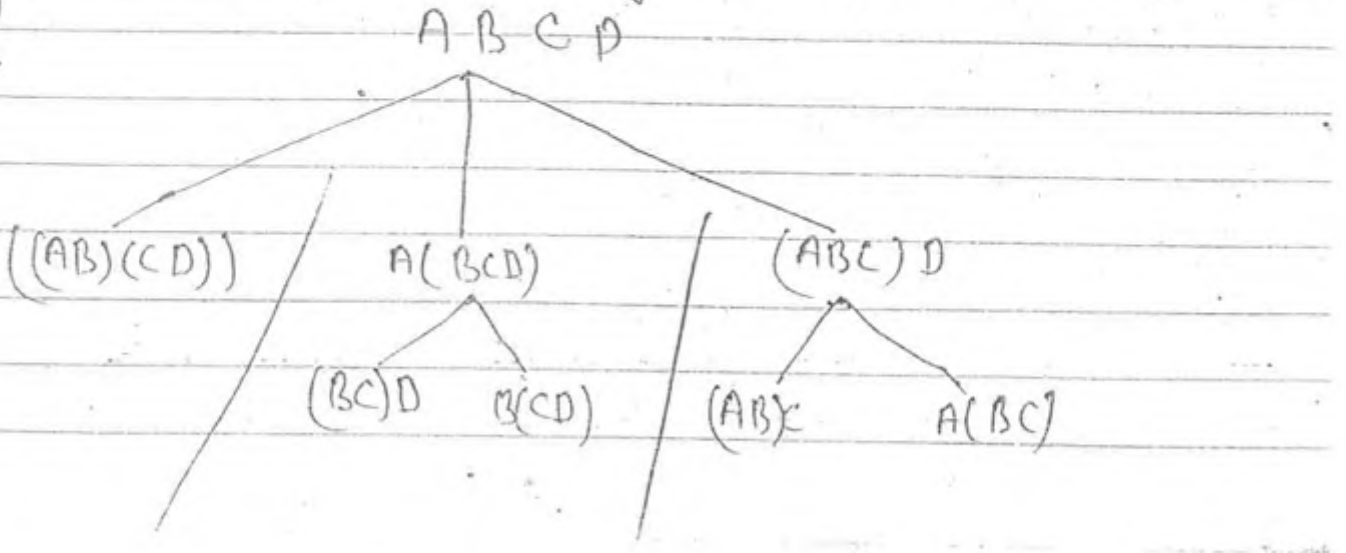


Note – In Matrix chain multiplication we dont want resultant matrix after multiplicy n matrix.

In MCM our goal is to multiply chain of n matrices we how to find out min no of multiplication required.

eg $A_{2\times5}$, $B_{5\times10}$, $C_{10\times5}$, $D_{5\times2}$ to find ABCD how many min^{m} no of multiplication required?

Sol^{n}



Q2 ((AB)(CD))

(AB) $2 \times 10 \times 5 = 100$

(CD) $10 \times 2 \times 5 = 100$

200

(ABCD) $2 \times 2 \times 10 = 40$

$\Rightarrow$ 100
100
40
240

(BC)D

$BC_{5 \times 5 \times 10} = 250$

$(BC)_{5 \times 5} D_{5 \times 2} = 50$

= 250
50
300

B(CD)

$CD_{10 \times 2 \times 5} = 100$

$B_{5 \times 10} (CD)_{10 \times 2} = 100$

= 100
100
200

(AB)C

$AB_{2 \times 10 \times 5} = 100$

$(AB)_{2 \times 10} C_{10 \times 5} = 100$

= 100
100
200

A(BC)

$BC_{5 \times 5 \times 10} = 250$

$A_{2 \times 5} (BC)_{5 \times 5} = 50$

= 250
50
300

$A((BC)D) = A_{2\times5} \times (BCD)_{5\times2} \Rightarrow (ABCD)_{2\times2\times5} = 20$

$= 320$

$A(B(CD)) \Rightarrow 220$

$((AB)C)D = 220$

$(A(BC))D = 320$

$((AB)(CD)) = 240$ $\quad$ $(((AB)C)D) = 220$

$(A((BC)D)) = 320$ $\quad$ $((A(BC))D) = 320$

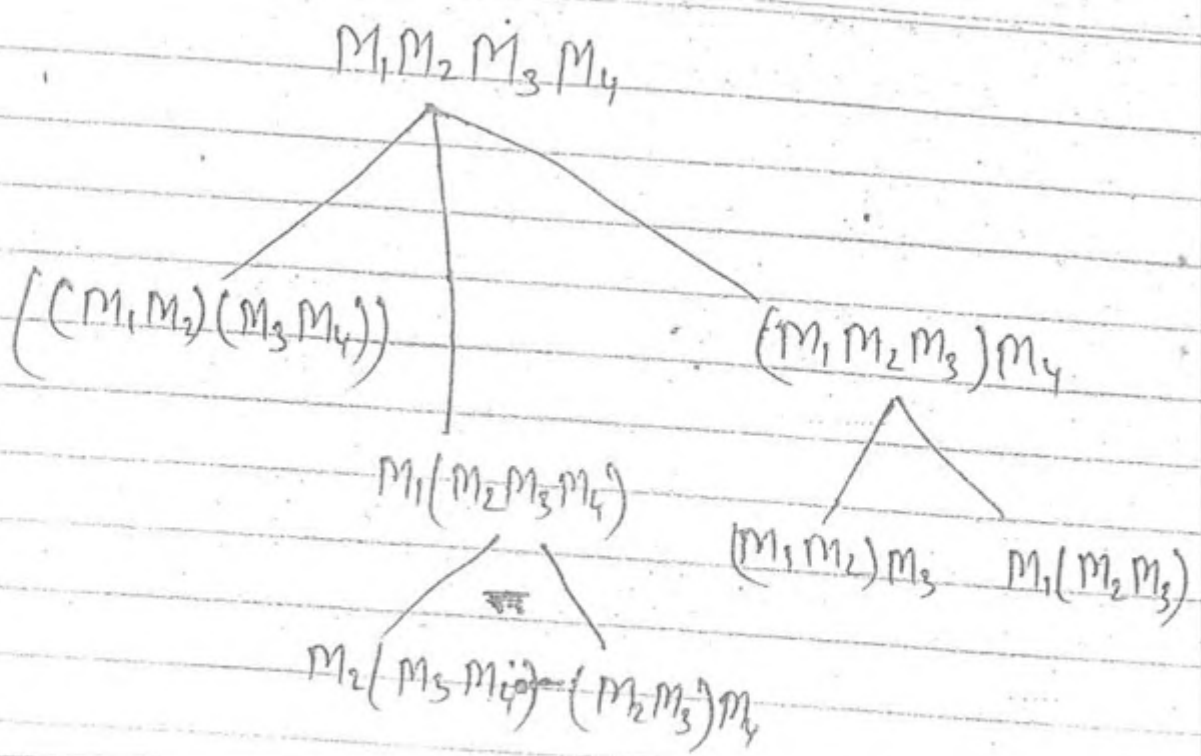
$A(((B(CD)) = 220$

\# Four matrices M_1, M_2, M_3, M_4 of dimension $p\times q$ $q\times r$, $r\times s$, st respectively can be multiplied in several ways with diff no of mopety if $p=10$, $q=100$, $r=20$, $s=5$, $t=80$, Hence min^m no of multiplicat. M_1 to M_4 is required

a) 2,40,000 $\quad$ c) 19,000 ✓

b) 44,000 $\quad$ d) 2,50,000

Sol^n



$(M_1 M_2)$ $(M_3 M_4)$
10×100 100×20 20×5 5×80

$M_1 M_2 = 10 \times 20 \times 100 = 20,000$

$M_3 M_4 = 20 \times 80 \times 5 = 8000$

$((M_1 M_2)(M_3 M_4)) = 10 \times 80 \times 20 = 16000$
10×20 20×80

$= 20000$
8000
16000
44000

$M_2(M_3 M_4)$

$M_3 M_4 = 8000$ ✓

$M_2(M_3 M_4)$

$100 \times 20 \quad 20 \times 80$

$= 100 \times 80 \times 20$

$= 160000$ ✓

$M_1(M_2(M_3 M_4))$

$10 \times 100 \quad 100 \times 80$

$= 10 \times 80 \times 100$

$= 80000$ ✓

$= 2,40,000$

$(M_2 M_3) M_4$

$M_2 M_3$

$100 \times 20 \quad 20 \times 5$

$= 100 \times 5 \times 20 = 10000$ ✓

$(M_2 M_3) M_4$

$100 \times 5 \quad 5 \times 80$

$= 100 \times 80 \times 5$

$= 40000$ ✓

$M_1((M_2 M_3) M_4)$

$10 \times 100 \quad 100 \times 80$

$= 10 \times 80 \times 100$

$= 80000$ ✓

$= 1,30,000$

$(M_1 M_2) M_3$

$M_1 M_2 = 20,000$ ✓

$(M_1 M_2) M_3$

$10 \times 20 \quad 20 \times 5$

$= 10 \times 5 \times 20$

$= 1000$ ✓

$((M_1 M_2) M_3) M_4$

$10 \times 5 \quad 5 \times 00$

$= 10 \times 00 \times 5$

$= 4000$ ✓

$= 25000$

$M_1 (M_2 M_3)$

$M_2 M_3 = 10,000$

$M_1 (M_2 M_3)$

$10 \times 100 \quad 100 \times 5$

$= 10 \times 5 \times 100$

5000 ✓

$(M_1 M_2 M_3) M_4$

$10 \times 5 \quad 5 \times 00$

$= 10 \times 00 \times 5$

$= 4000$ ✓

$= 19000$ ✓

#

ABCD | ABC

$A(BCD)$ | $A(BC)$

$(AB)(CD)$ | $(AB)C$

$(ABC)D$

Recurrence Relⁿ for MCM

$$MCM(i,j) = \min \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \le k < j} \{ MCM(i,k) + MCM(k+1,j) + \text{NMRTMTTM} \} \end{cases}$$

no of multiplication required to multiply the total multiplication

K=1 A (BCD)

K=2 (AB) (CD)

K=3 (ABC) D

eg Q $A_{2\times5}, B_{5\times2}, C_{2\times5}, D_{5\times2}$ find MCM of 1 to 4. matrix is equal to

Sol^n

$$MCM(1,4) = \min \begin{cases} MCM(1,1) + MCM(2,4) + 40 \\ MCM(1,2) + MCM(3,4) + 20 \\ MCM(1,3) + MCM(4,4) + 40 \end{cases}$$

$$= \min \begin{cases} 0 + MCM(2,4) + 40 \\ 20 + MCM(3,4) + 20 \\ 40 + MCM(1,3) + 0 \end{cases}$$

40

20

40

$\sqrt{O(n)}$

40

$$MCM(2,4) = \min \begin{cases} \overbrace{MCM(2,2)}^{0} + \overbrace{MCM(3,4)}^{20} + 20 \\ \underbrace{MCM(2,3)}_{50} + \underbrace{MCM(4,4)}_{0} + 50 \end{cases}$$

$O(n)$

20

$$MCM(3,4) = \underbrace{MCM(3,3)}_{0} + \underbrace{(4,4)}_{0} + 20$$

(CD)

C 2x5 D 5x2

2x2x5 = 20

B (CD)

5x2 2x2

5x2x2 = 20

50

$$MCM(2,3) = \underbrace{MCM(2,2)}_{0} + \underbrace{MCM(3,5)}_{} + 50$$

(BC)

B 5x2 C 2x5

5x5x2 = 50

(BC)D

5x5 5x2

5x2x5 = 50

A (BCD)

2x5 5x2

ABCD

2x2x5 ⇒ 20

20

$MCM(1,2) = \underbrace{MCM(1,1)}_{0} + \underbrace{MCM(2,2)}_{0}$ 20

$(AB)_{2\times2\times5} = 20$

$(AB)_{2\times2}\ (CD)_{2\times2} = 0$

$MCM(1,5) = \min\begin{cases} \underbrace{MCM(1,1)}_{0} + \underbrace{MCM(2,3)}_{50} + 50 \\ MCM\underbrace{(1,2)}_{20} + \underbrace{MCM(3,5)}_{0} + 20 \end{cases}$

40

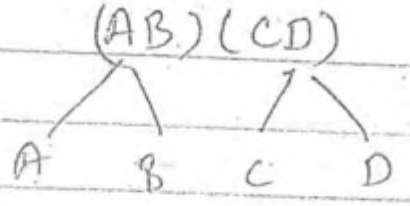
$A(BC)$	$(AB)(C)$
$2\times5 \quad 5\times10$	$2\times2 \quad 2\times5$
$10\times5=50$	$10\times2=20$

$(ABC)\ (B)$

$2\times5 \quad 5\times2$

$= 2\times2\times5$

$= 20$ ₹



How many different fun^n calls.

	1	2	3	4
1		✓	✓	✓
2	X		✓	✓
3	X	X		✓
4	X	X	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 fun^n Call we have to find min^m

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

	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 ⟹ $((E+V)\log V)$

② Bellman-Ford ⟹ $O(VE)$

⇓

idea

① Dijkstra algo × |V|

⇓

$V * ((E+V)\log V)$

⇓

$O((VE+V^2)\log V)$

```
for(i=1, i≤n, i++)
{
Dijkstraalgo(v,i)

}.
```

② Bellman-Ford × |V|

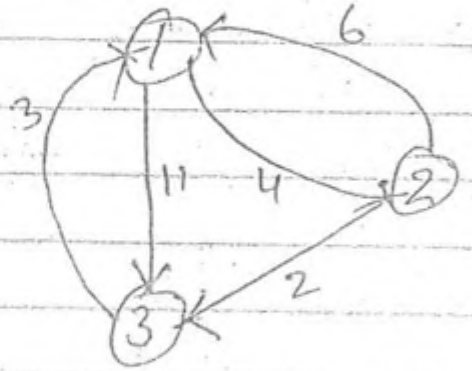
⇓

$|V| \times (VE)$

⇓

$O(V^2E)$

All pair shortest Path (SP)



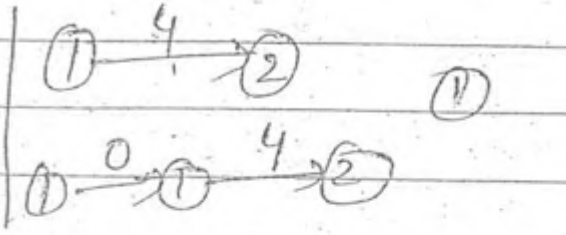
$A^K(i, j)$ = The minm cost required to go i to j. By taking Not greater than intermediate vertices numbers.

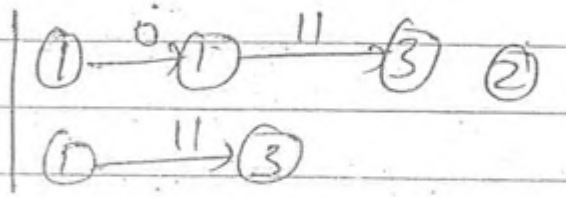
$A^K(i,j)$ ⇒ The minm cost required to go from i to j. By taking intermediate vertex number not greater than K.

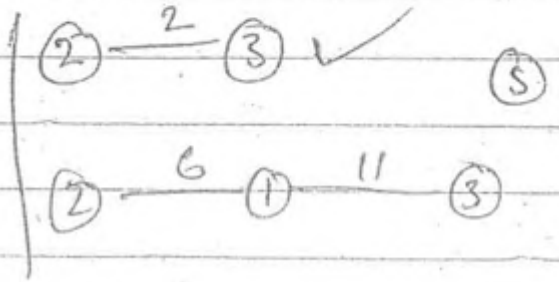
Without SP

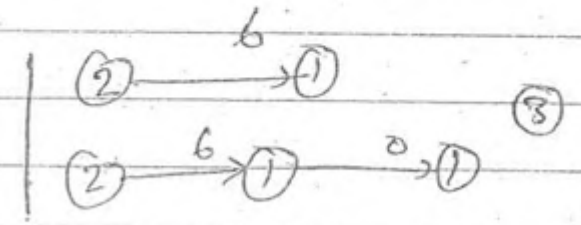
A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

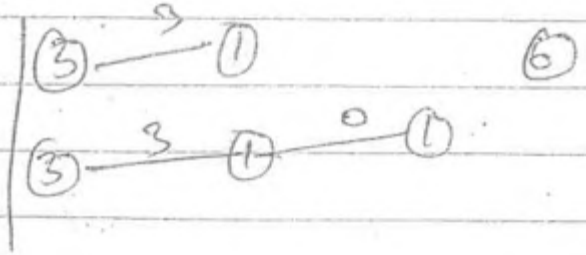
A	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

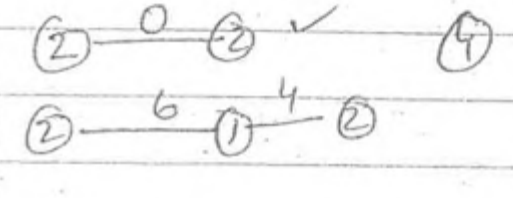


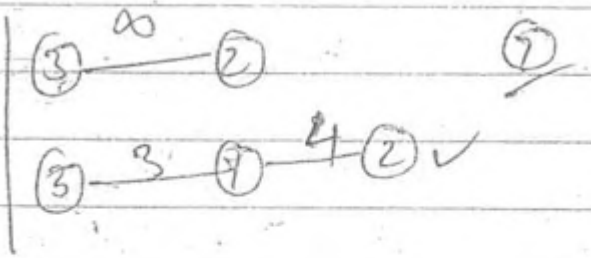






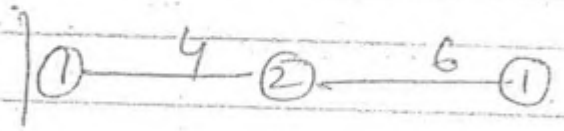


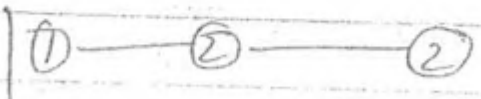




A^2	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

A^3	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0



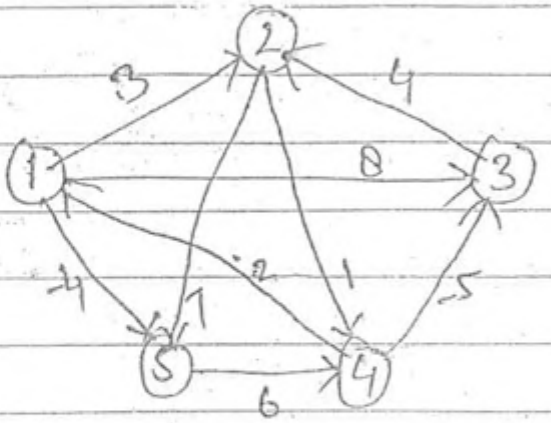


Floyd-Warshall's All pairs shortest path algo

$$A^K(i,j) = \min\left\{ \underbrace{A^{K-1}(i,j)}_{\text{Previous value}}, \underbrace{A^{K-1}(i,K) + A^{K-1}(K,j)}_{\text{via K.}} \right\}$$

$$A^3(1,3) = \min\left\{ A^2(1,3), A^2(1,3) + A^2(3,2) \right\}$$

P/



Soln

A^0	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

A^1	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

A^2	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	4	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

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	8	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	8	5	1	6	0

```
for ( k=1 , k<=|V| , k++ )
    for ( i=1 , i<=|V| , i++ )
        for ( j=1 , j<=|V| , j++ )
```

$$A^k(i,j) = \min\left\{ A^{k-1}(i,j),\ A^{k-1}(i,k) + A^{k-1}(k,j) \right\}$$

$$\Downarrow$$

$$O(n^3)$$

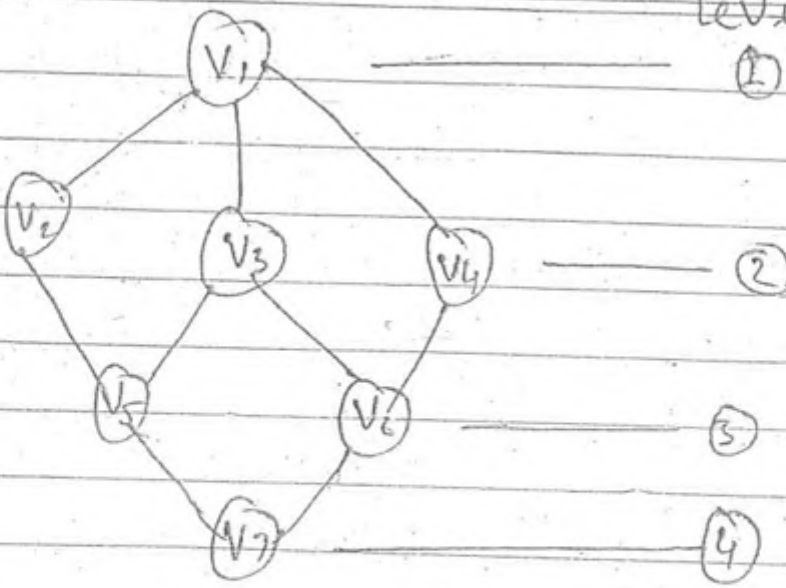
22-7-11

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 ≠ ϕ)
{
  v = delete (q)
for all w adajacent to v;
  {
  if (w is not visited)
    {
      visited (w) = 1;
      add (w, q);
    }
  }
}
}
```



ω (V2, V3, V4) adjacent to u (V1)

ω (V1, V5) adjacent to u (V2)

(i) The data structure we are using BFT is Queue.

(ii) BFT nothing but visited level by level becoz of random the BFT is called as level order Traversal

V1 | V2 V3 V4 | V5 V6 | V7
1 | 2 | 3 | 4

(iii) the time complexity of BFT:

$O(V+V^2)$

$O(V+E)$

$O(V+E)$

(1) V_1 V_2 V_3 V_4 V_5 V_6 V_7

(2)

(3)

find 4 - different answers if you start from V_3 in BFT.

(1) V_3 V_1 V_5 V_6 V_2 V_4 V_7

(2) V_3 V_6 V_5 V_1 V_7 V_4 V_2

(3) V_3 V_5 V_1 V_6 V_2 V_7 V_4

(4) V_3 V_5 V_6 V_1 V_2 V_7 V_4

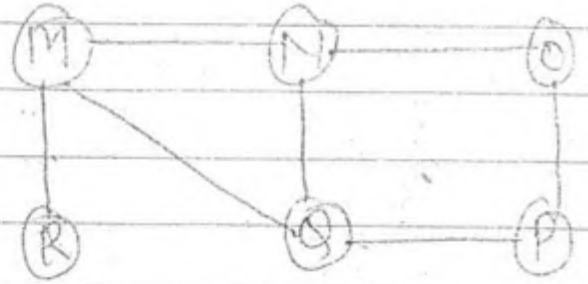
V_3 V_1 V_5 V_6 V_4 V_2 V_7

find 2-different answers if start from V_7 in BFT.

(1) V_7 V_5 V_6 V_2 V_3 V_4 V_1

(2) V_7 V_6 V_5 V_3 V_4 V_2 V_1

Q Consider the following graph.



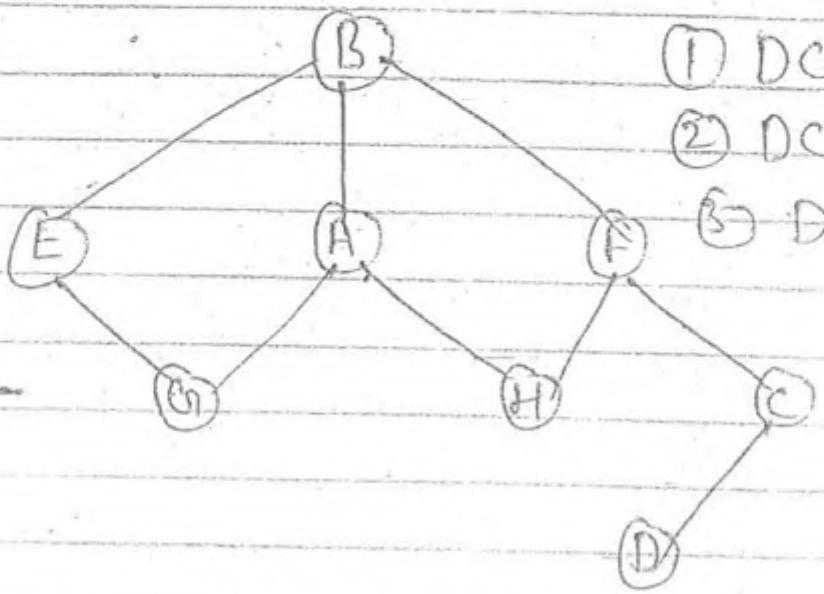
BFT applying on the above graph which one of the following is one possible order of visiting nodes in BFT.

(a) MNOPQR b) NQMPOR

c) QMN PRO d) Q MNPOR

Q. Consider the following Graph.

Find 3 - different BFT starting from D



(1) DCFBHEAG

(2) DCFHBAEG

(3) DCFBHAEG

(1) D C F BH E A G

(2) D C F HB A E G

(3) D C F BH A E G

Q. Consider undirected Graph G.

Let a BFT of G be done starting from a node r.

Let d(r,u) and d(r,v) be the length of the shortest paths from r to u and r to v respectively in G.

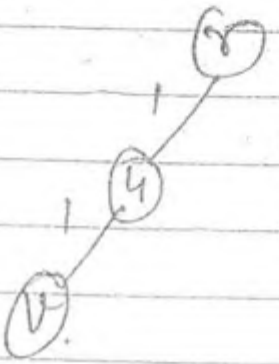
if u ~~visited~~ is visited before v during BFT, which one of the following statements are correct.

a) $d(r,u) < d(r,v)$

b) $d(r,u) > d(r,v)$

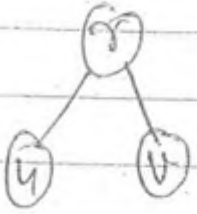
c) $d(r,u) \leq d(r,v)$

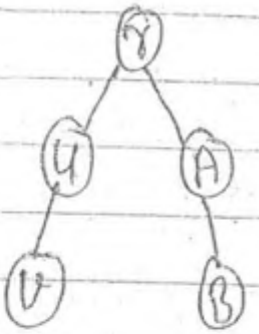
d) $d(r,u) \geq d(r,v)$



$d(r,u) = 1$

$d(r,v) = 2$

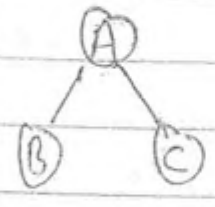




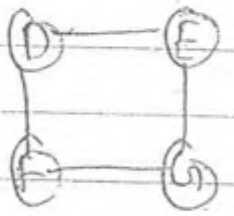
$r \cup A \vee B$

$d(r,u) \leq d(r,v)$

G



$V_1 + E_1$



$V_2 + E_2$

$V_3 + E_3$

$G_1(A, B, C, D, E, F, G, H, I)$

$\Rightarrow O(V+E)$

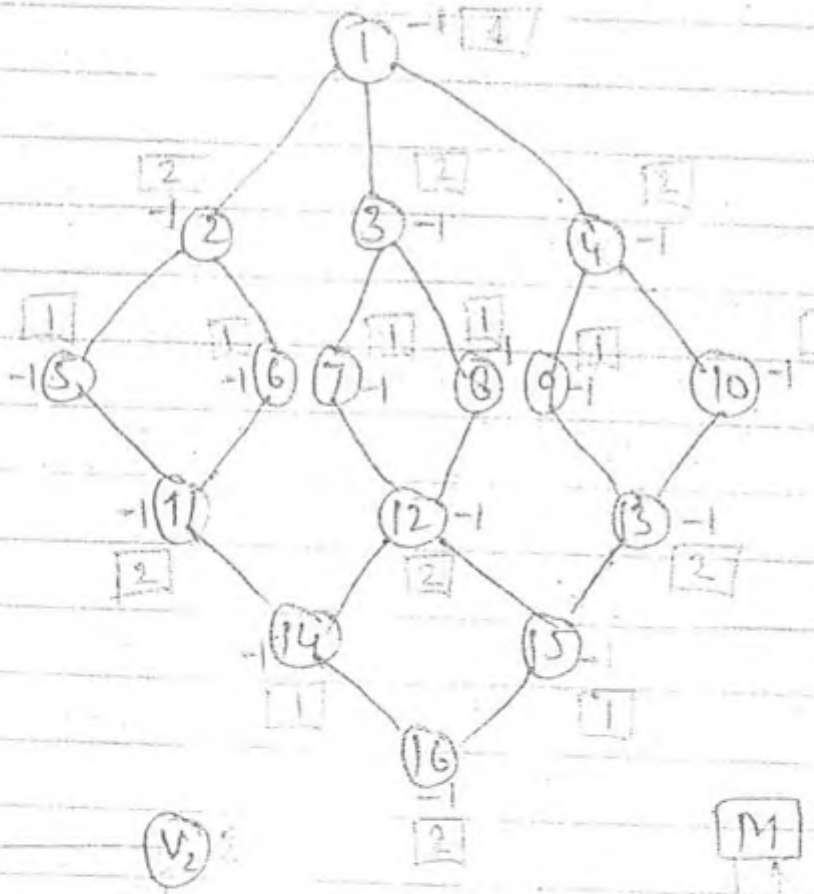
Applications of BFT

(1) Using BFT algo we can check given graph is completed or not.

(2) Using BFT algo we can find out no of connected component in the given graph.

(3) Using BFT algo we can find out given derived graph contain cycle or not.

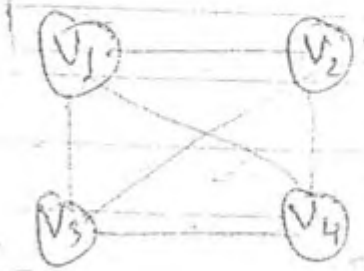
(4) Using BFT algo we can find out given graph is bipatite graph or not



V = MUN

M = 1, 5, 6, 7, 8, 9 10, 14, 15

N = 2, 3, 4, 11 12, 13, 16.

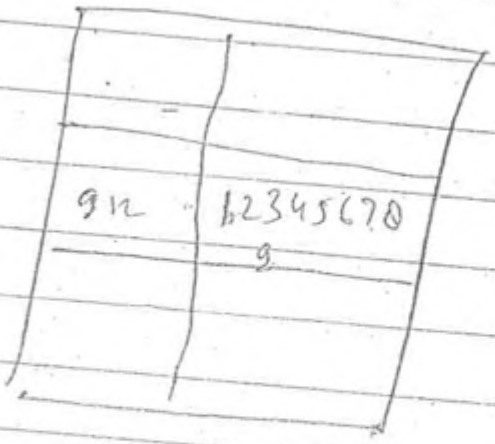


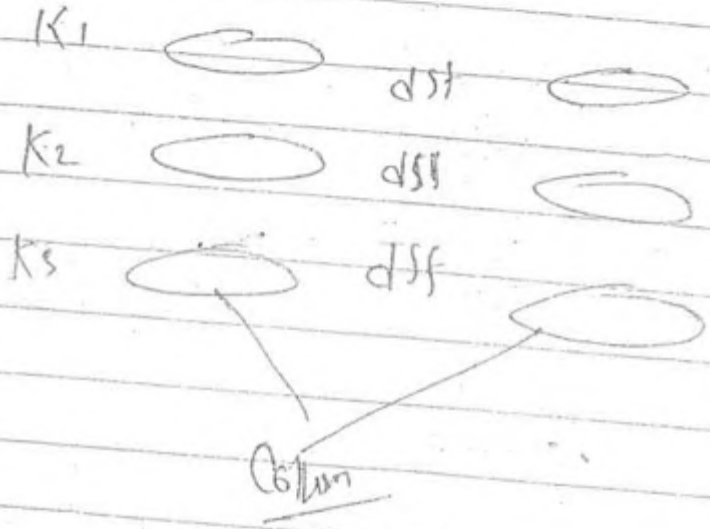
Fold boundary Method

Key = 123456789

M = 1000

123
789
912





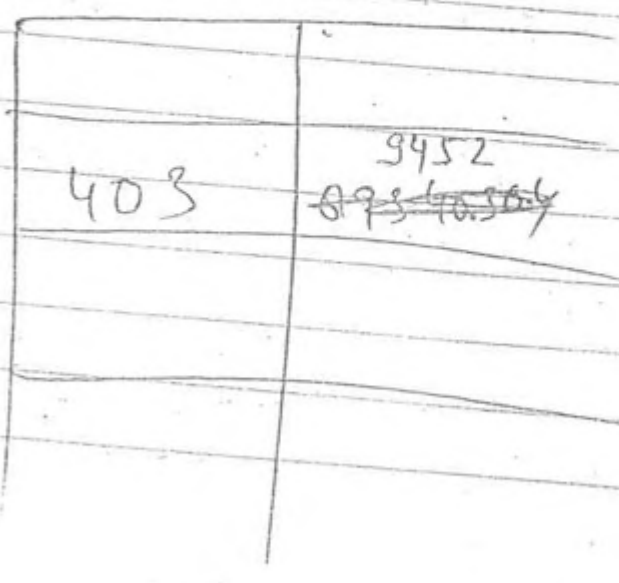
Mid Square

M = 1000

Key = 9452

$(9452)^2$

89310304



Digit Extraction Method (Function)

m = 1000

eg. 379452 = 394
1 3 4

121267 = 112

570045 = 380

112	121267
394	379452

Using digit extraction method selected digit are extracted from the key and use as the hash address.

$K_1 = a_1\ a_2\ a_3\ a_4 \ldots\ldots a_{10}$
$K_2 = b_1\ b_2\ b_3\ b_4 \ldots\ldots b_{10}$
$K_3 = c_1\ c_2\ c_3\ c_4 \ldots\ldots c_{10}$

Collision

Folding Method

(1) Fold Shift Method

Key = 123 456 789

m = 1000

123
456
789
1368

136
8
144

144	123456789

bcoz result always contain 1st P-number of bit (P-LSB bit)

If the given no of m is prime no than no of collission will be s.

Given no is prime number and not base to power of 2 or to is always preferable.

eg

① $M = 2^P \alpha$

result always contain P-bit.

② M is Prime.
It is advisable.

③ M is prime
not close to power of 2
is good choice.

Types of Hashfunction

1. Division modulo Method.
2. Digit extraction Method. (Truncation)
3. Folding Method.
4. Mid Square

(1) Division Modulo Method ⇒

Hash Table Size = M

Key

Key Modulo M.

eg.

Hash Table Size = 10

Key = 625

(5) = 625 mod 10

eg

$m = 2^4$

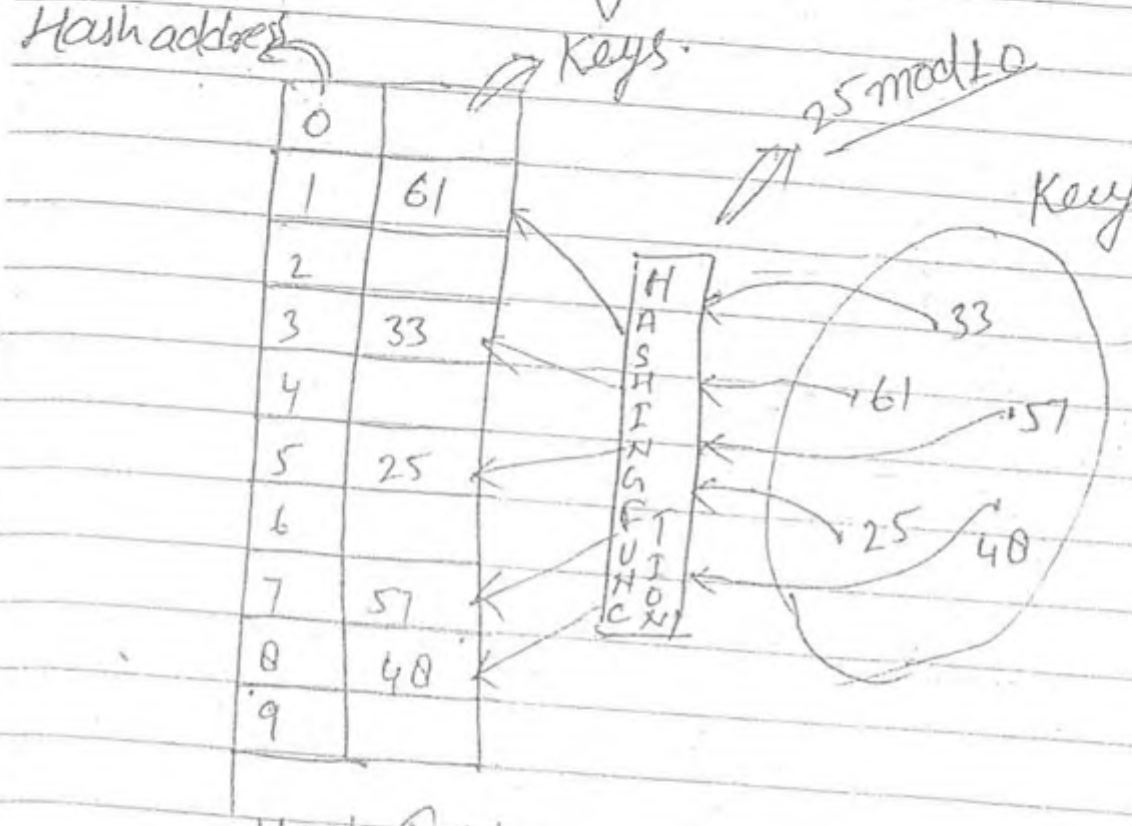
Key = 15

15 modul 16 = 15

(1) In division modulo method M should not be powers of 2 means powers of can not 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.



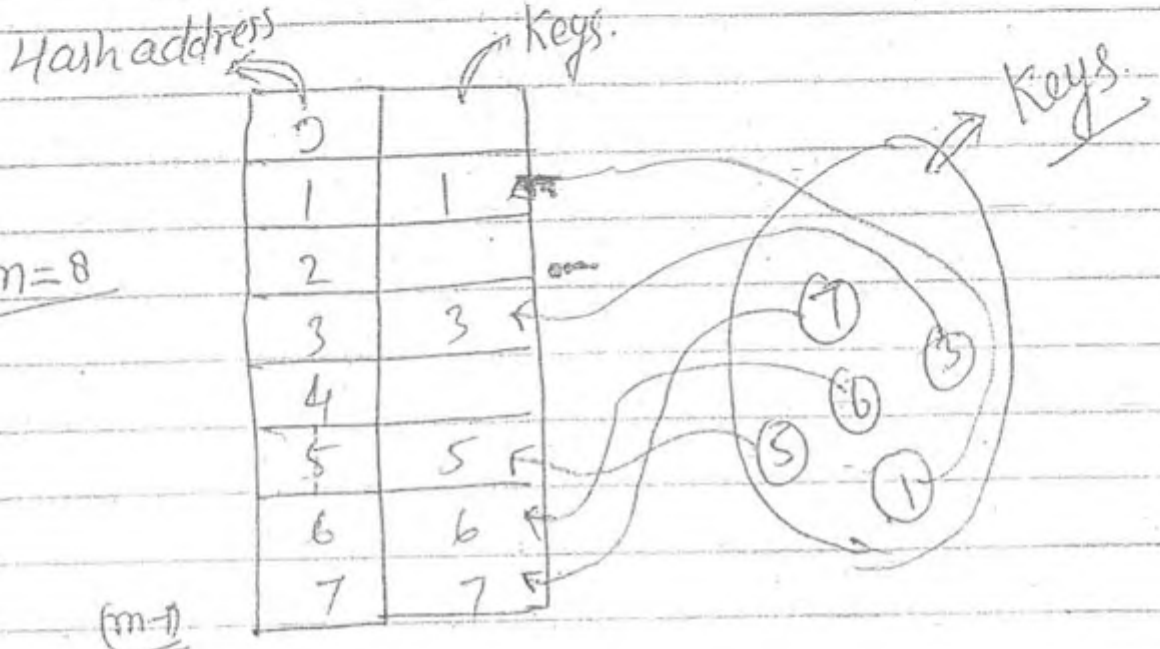
Producing the same Hash address that is already produce is called Collission

The purpose of hash function is reducing the given key into one of the Hash function.

Hashing

(1) It is one of the searching Technic.
(2) Average ⇒ O(1)

Direct address Table



Hash Table

(1) In direct address / Hashing Table Key is the address without any manipulation

(2) Even Though no of Keys are very less but one of the Keys may contains 64 bit.

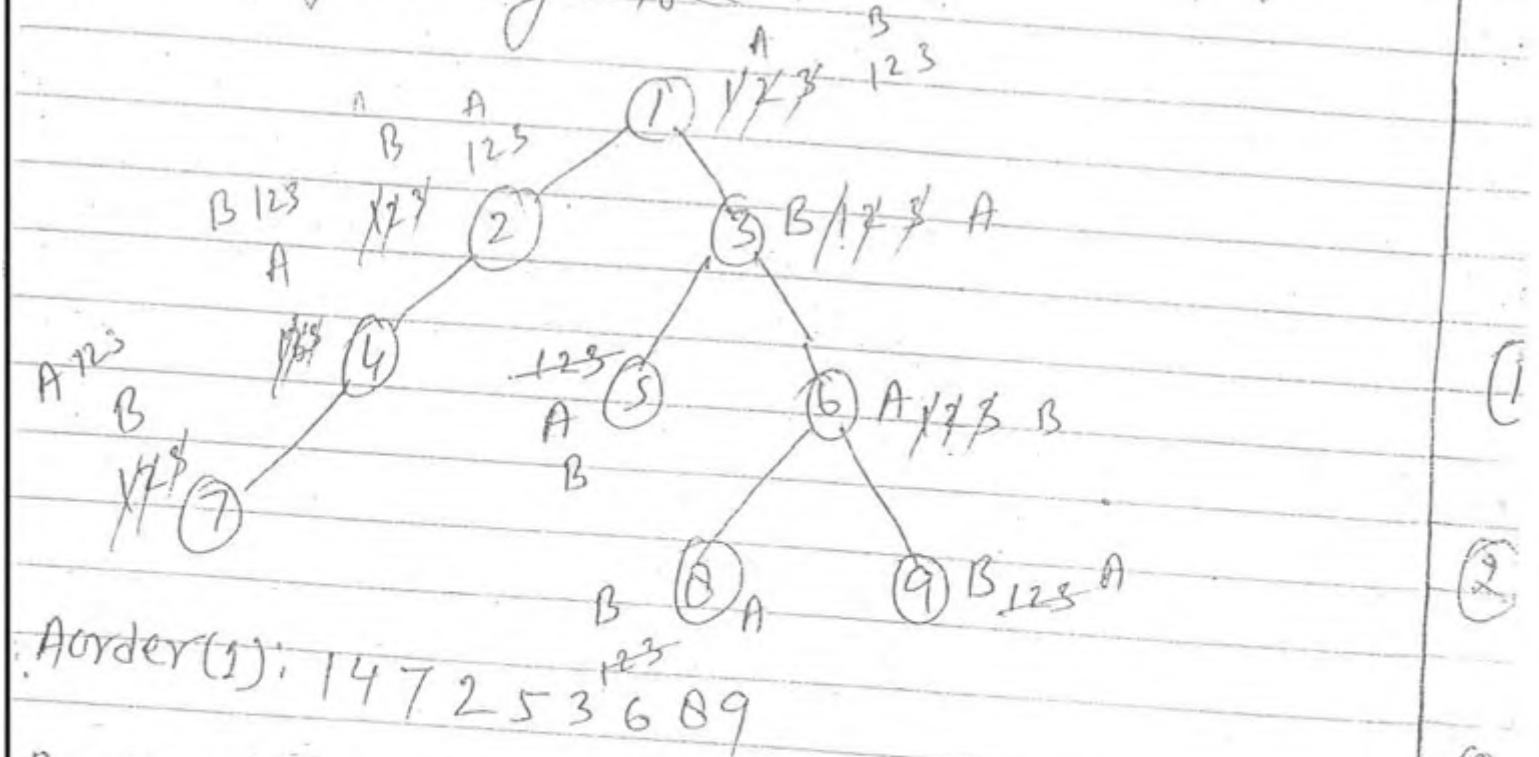
(3) The range of Keys determines size of the Hash Table.

Consider the following C Program

```
Border(root)
{
  if (root != null)
  {
    Aorder(root->left);
    Printf(root->data);
    Aorder(root->right);
  }
}
```

```
A Order(root)
{
  if(root != null)
  {
    Printf(root->data);
    Border(root->left);
    Border(root->right;
  }
}
```

What will be the output of the above program for the following tree.



Aorder(1): 1 4 7 2 5 3 6 8 9

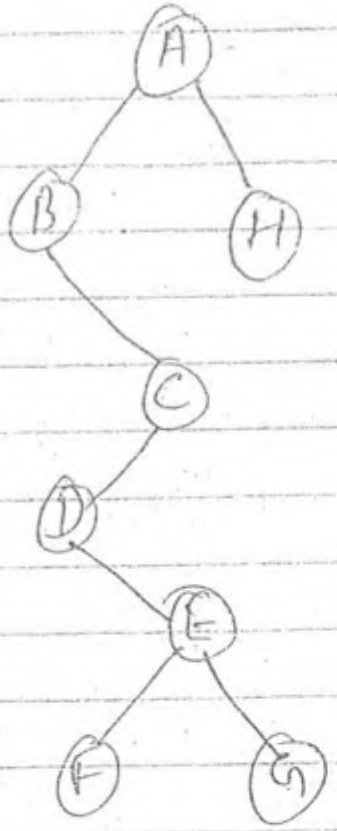
Border(1): 2 7 4 1 3 5 8 6 9.

2. 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)
  }
}
```

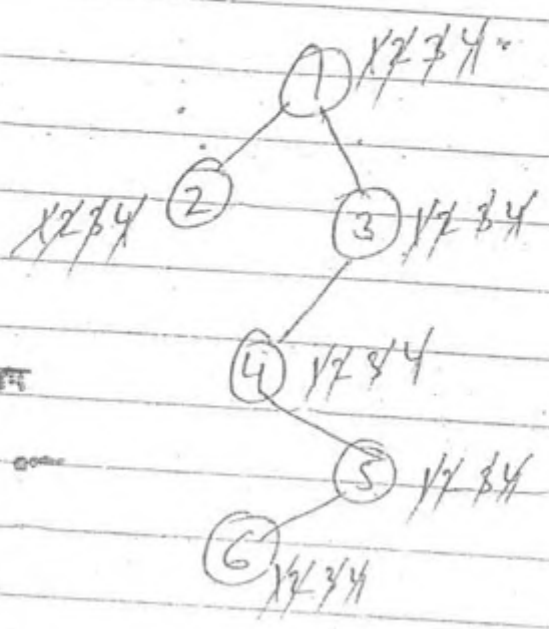
Sol^n

A B B C D D E F F F
E G G G E D C C B A
H H H A



A B B

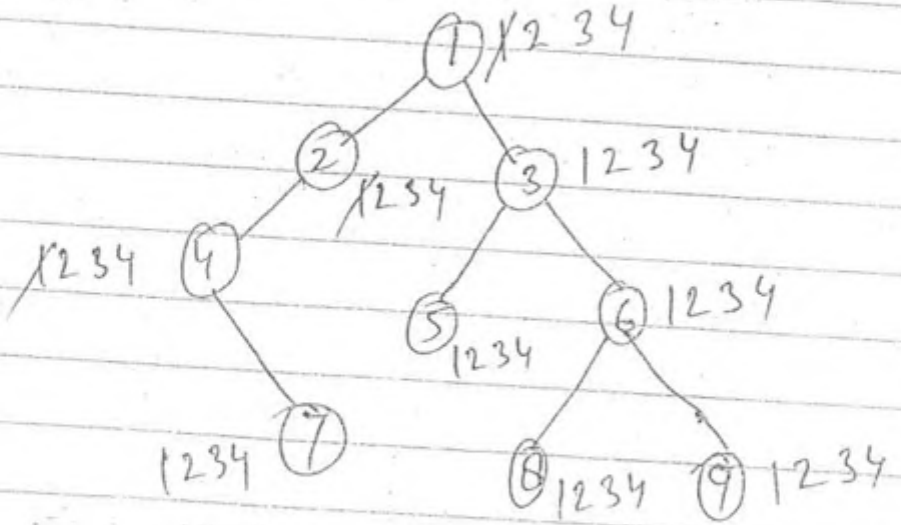
(3) Printf (root->data)
(4) Do (root -> right)
}
}



Soln

1 2 2 1 3 4 4 5 6 6 5 3

Q. What will be the o/p for the above code of the following tree.

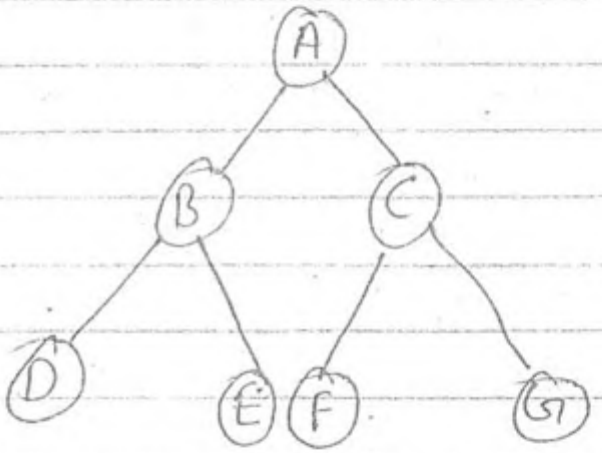


Soln

1 2 4 4 7 7 2 1 3 5 5 3 6 0 8 6 9 9

#

```
ABC ( Root)
{
  if (root != null)
  {
    ① ABC (root → right)
    ② Printf( root → data)
    ③ ABC ( root → left)
  }
}
```

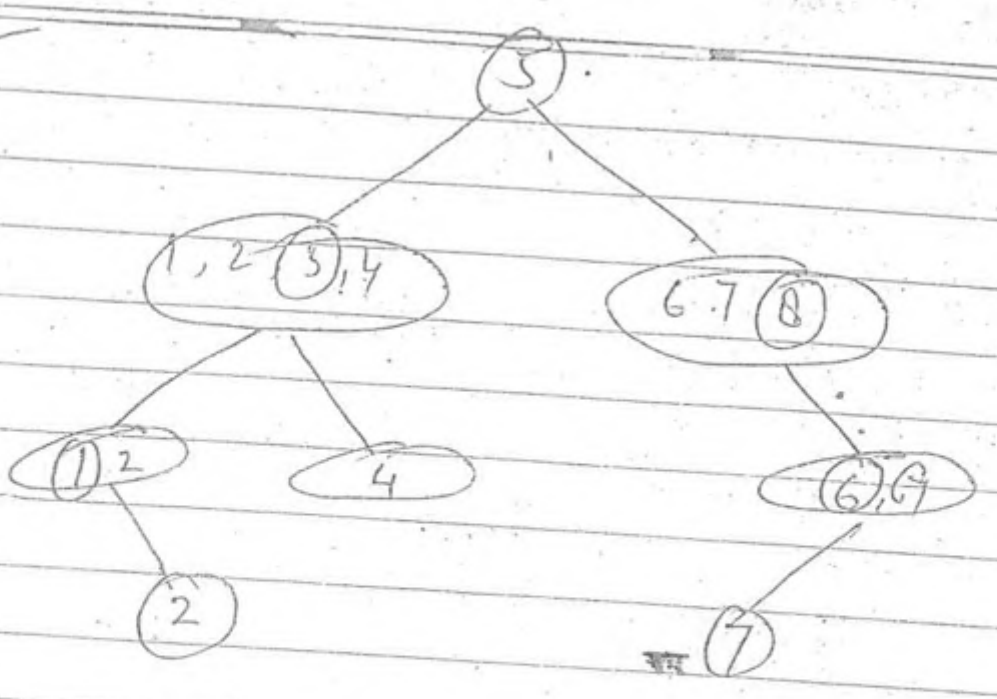


G, C F, A, E B, D

P What will be the output for the following program.

```
Do (root)
{
  if (root != null)
  {
    ① Printf (root → data)
    ② Do (root → left)
```

50)



Post order — 2, 1, 4, 3 7 8 6 5

Note - In order travels of binary search tree will given always ascending order of element

For every element of PreOrder or PostOrder, apply Linear search to find LST and RST in InOrder.

O(n) × O(n)

⇓

$O(n^2)$

PreOrder - Postorder

eg PreOrder - A B D E C F G

Postorder - D E B F G C A

⇓

(A) Unique binary tree not possible.

Note - Using PreOrder and InOrder we can construct unique binary tree.

Using Postorder — InOrder ⇒ Unique binary tree.

Using PreOrde - Postorder ⇒ Not Unique.

Q

A binary Search Tree Contain following PreOrder

PreOrder ⇒ 5, 3, 1, 2, 4, 6, 8, 7

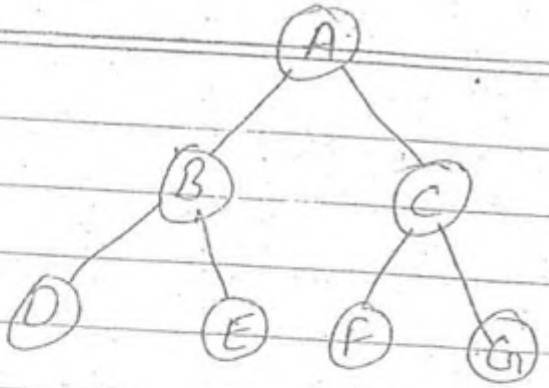
PostOrder ⇒

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

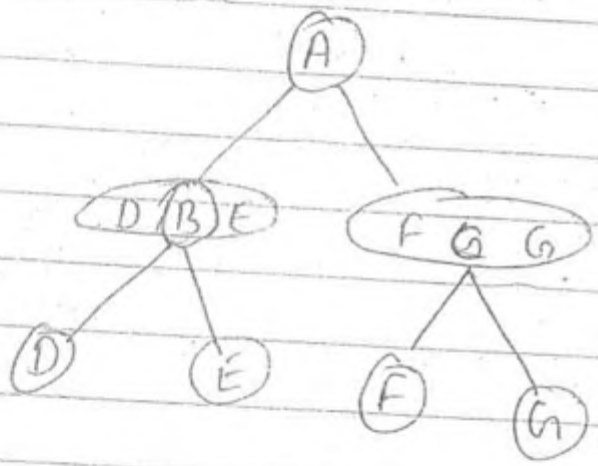
d) 2, 1, 4, 3, 7, 8, 6, 5



PreOrder - A B D E C F G

InOrder - D, B, E A F C G.

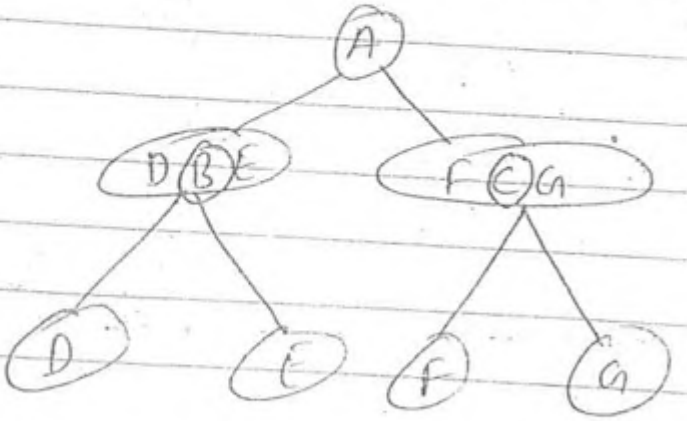
⇓



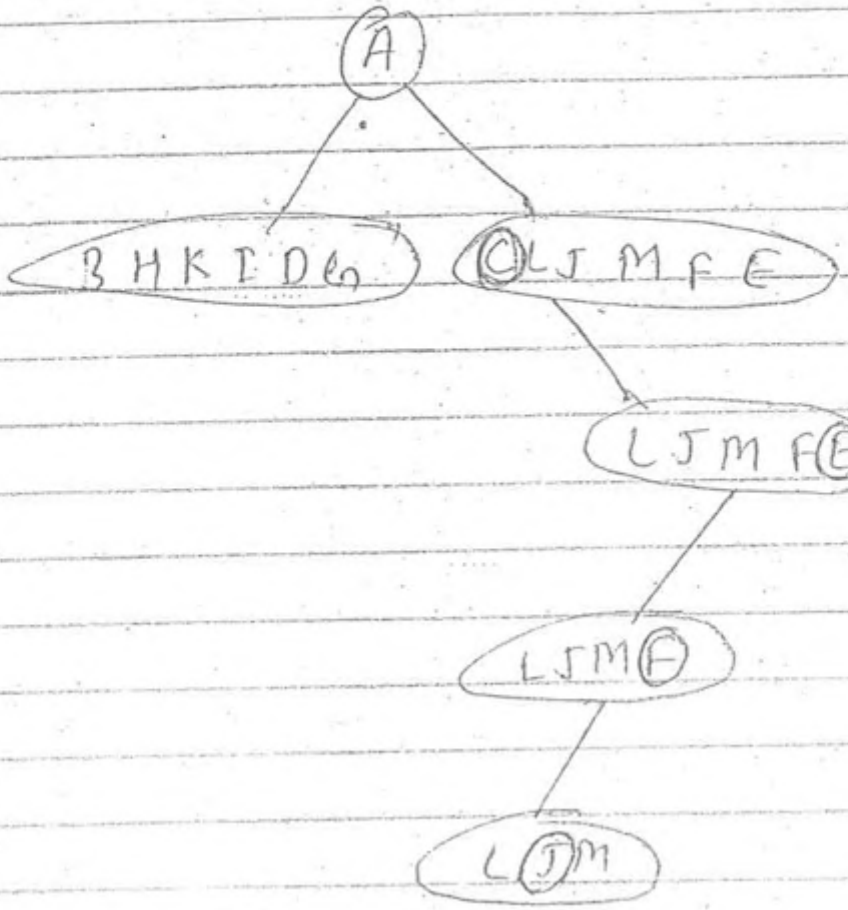
Unique binary tree possible

PostOrder - D E B F G C A

InOrder - D B E A F C G.



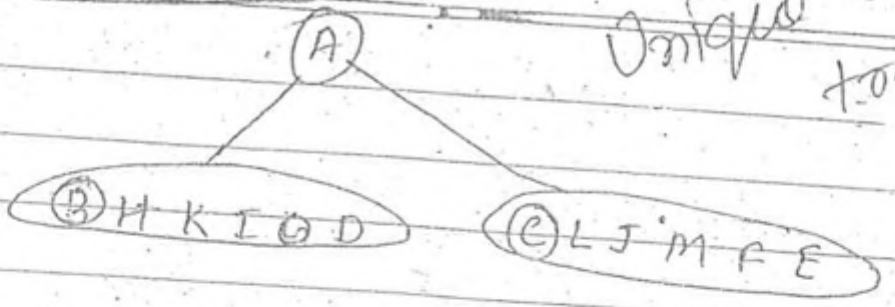
PostOrder - InOrder



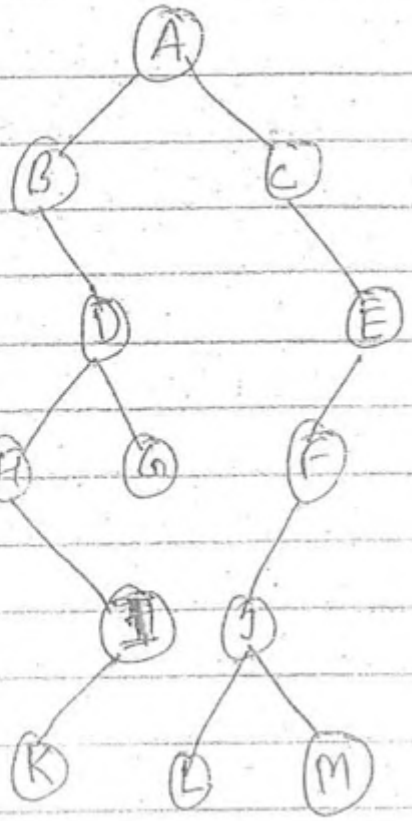
Unique binary tree Possible.

PreOrder - Inorder

Unique binary tree possible



P



PreOrder - (R, LST, RST) ⇒ A, B, D, H, I, K, G, C, F, J, L, M

PostOrder - (LST, RST, Root) ⇒ K, I, H, G, D, B, L, M, J, F, E, C, A

InOrder - B, H, K, I, D, G, A, C, L, J, M, F, E

Postorder

```
Postorder ( Root )
{
  if (root != null)
  {
    Postorder ( Root → left )
    Postorder ( Root → right )
    Printf ( Root → data )
  }
}
```

O(n)

InOrder

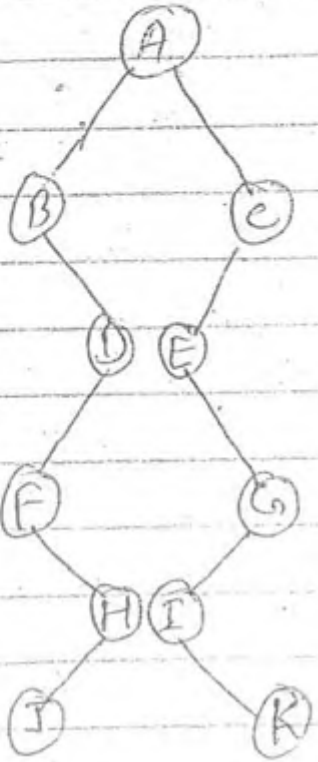
```
InOrder ( root )
{
  if ( root != null )
  {
    InOrder ( Root → left );
    Printf ( root → data );
    InOrder ( root → right );
  }
}
```

O(n)

23/01/11

Monday

Q. Find Preorder, Inorder and Postorder for the following binary Tree.



PreOrder – A, B, D, F, H, J, C, E, G, I, K.

Postorder – J, H, F, D, B, K, I, G, E, G, A.

InOrder – B, F, J, H, D, A, E, B, I, K, G, & C

PreOrder

```
PreOrder(Root)          → O(n)
{
  if (root != null)
  {
    Printf(root → data)
    Preorder(root → left)
    Preorder(root → right)
  }
}
```

① int u
② u = 10
↳ int u = 10

u
10
1000

u	10
&u	1000

reference operator

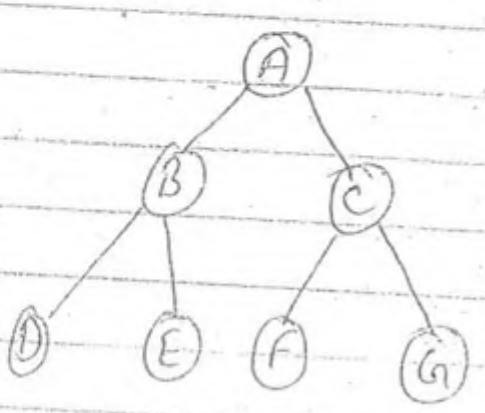
*(&u) 10

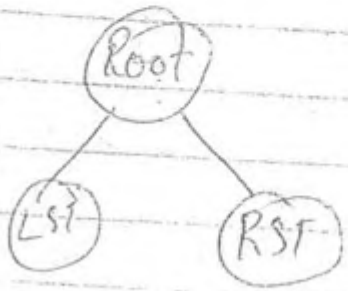
↳ dereference operator.

Tree Traversal (Unique)

1. InOrder - (LST, Root, Right Sub Tree (RST))
2. PreOrder (Root, LST, RST)
3. Post Order (LST, RST, Root)

eg



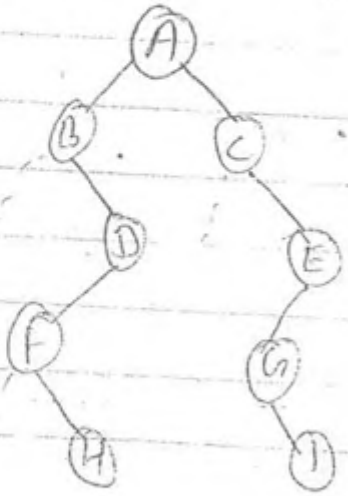


PreOrder - A, B, D, E, C, F, G

Postorder - D, E, B, F, G, C, A

InOrder - D, B, E, A, F, C, G

eg Find Preorde, Postordr, and Inorder in the following tree:



PreOrder - A, B, D, E, H, C, F, G, I

Postorder - H, F, D, B, I, G, E, C, A

InOrder - B, F, H, D, A, I, G, C, E

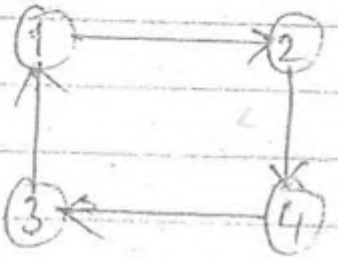
every strongly connected graph is also weekly connected.
but every weekly connected graph. not be strongly connected.

graph

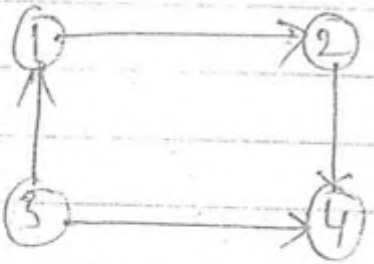
Directed — undirected

undirected ⇓ Connected / disconnected.

Directed → Strongly Connected / weakly 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 then graph is called weakly connected

Biconnected Component :—

A graph is said to be Biconnected component if it is contains zero articulation point.

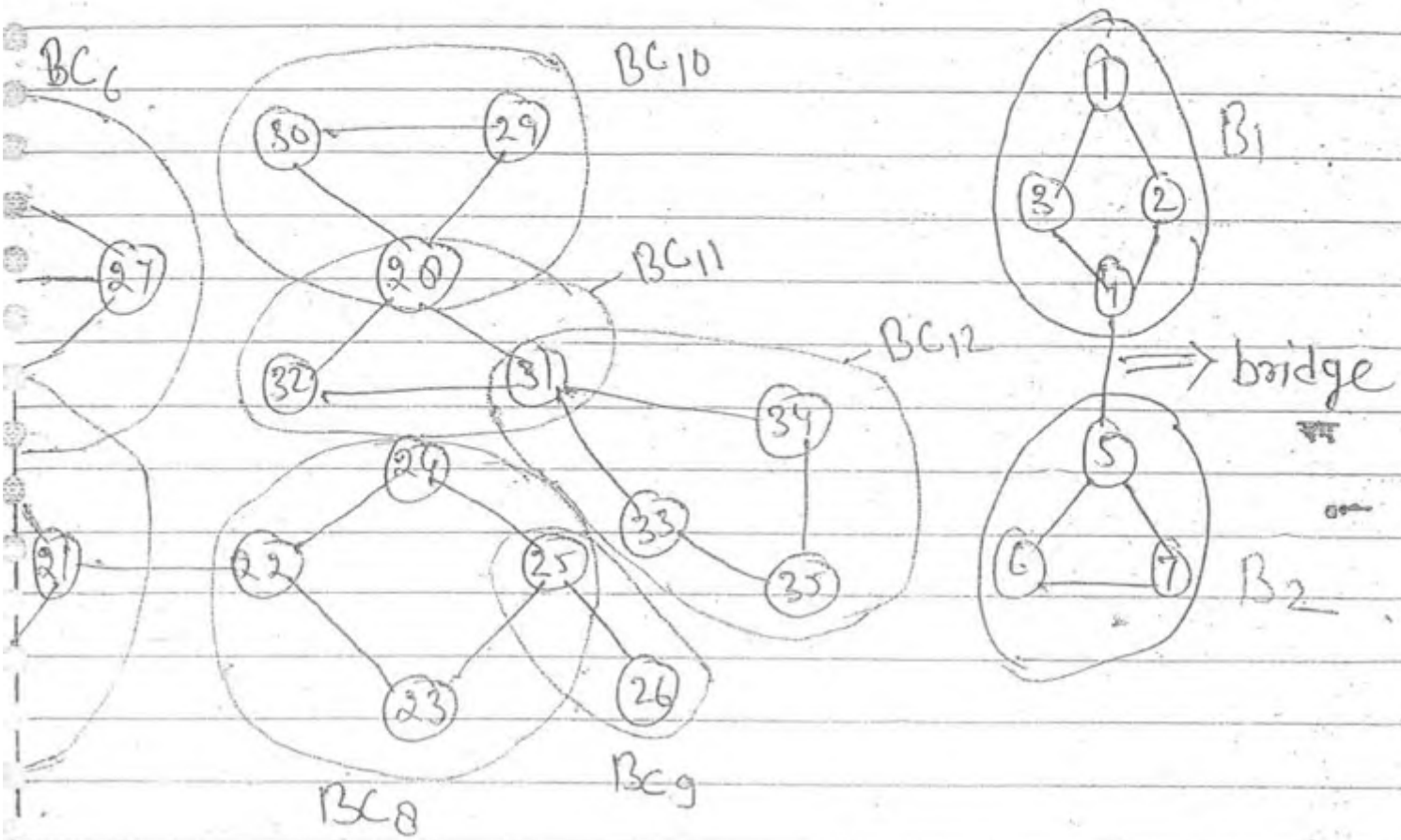
Note —

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.

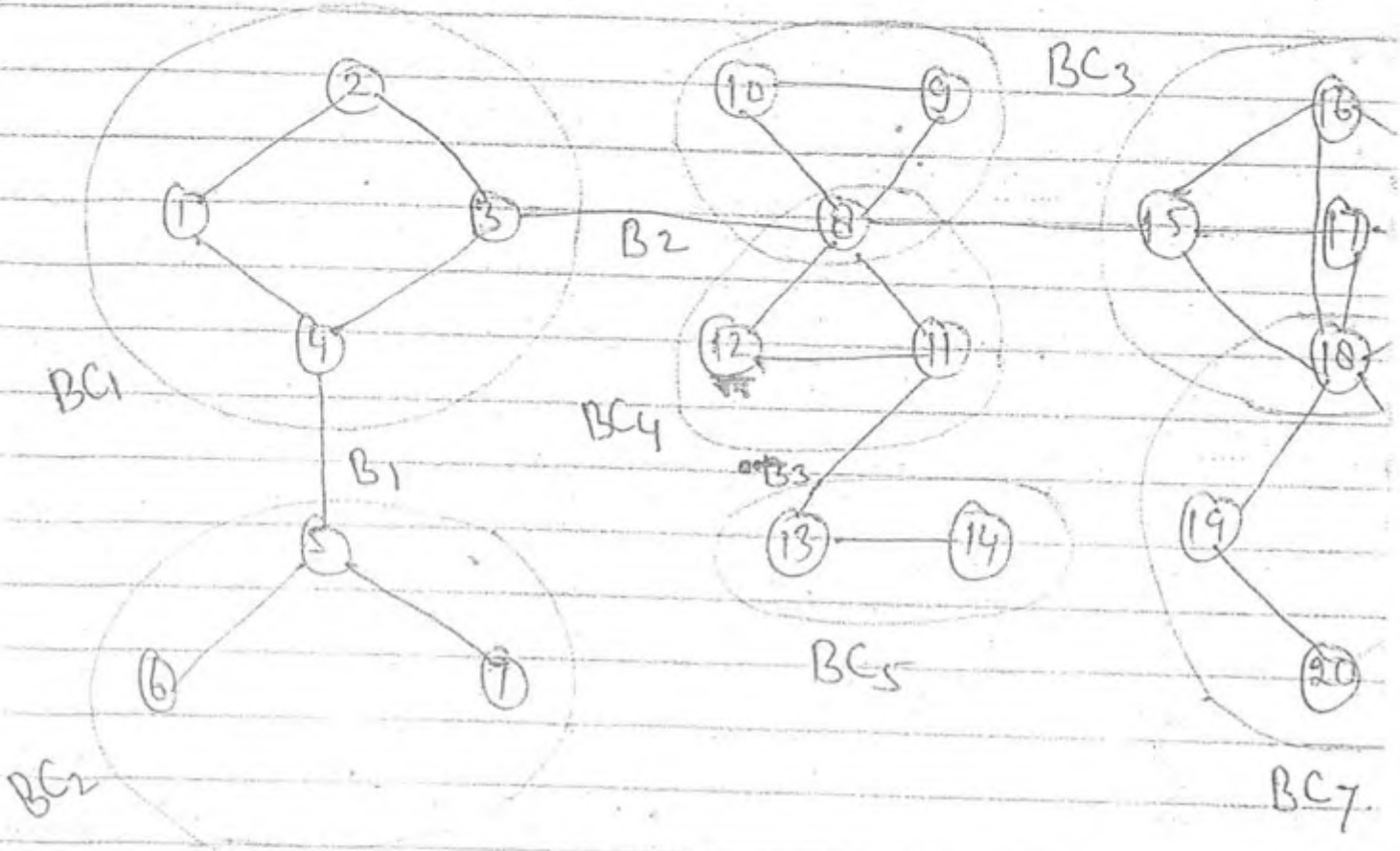


Using DFT alg we can find out bridge also.

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.

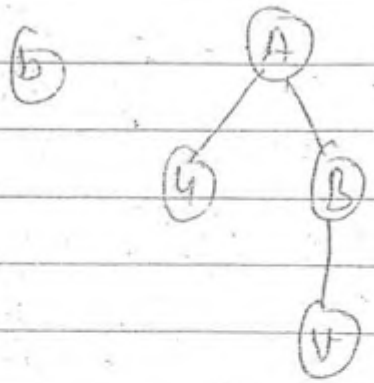
Q/ Find out the no of articulation point of following graph.

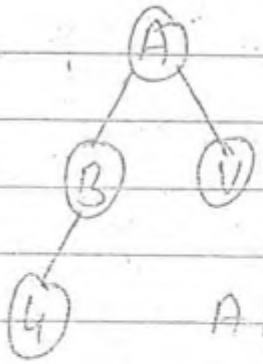


articulation point ⇒ 14
bridge ⇒ 0.

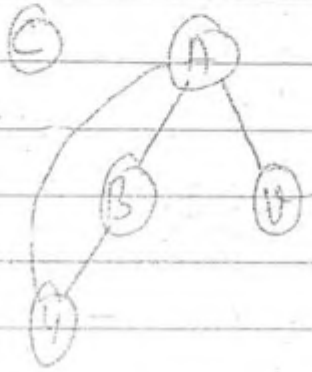
Note ① In a particular computer network the number of routers, which are behaving like a articulation point.

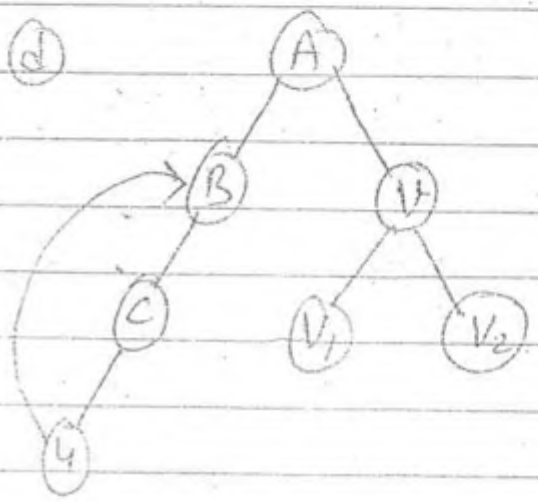
② Using DFT by we can find all possible articulation point in given graph.



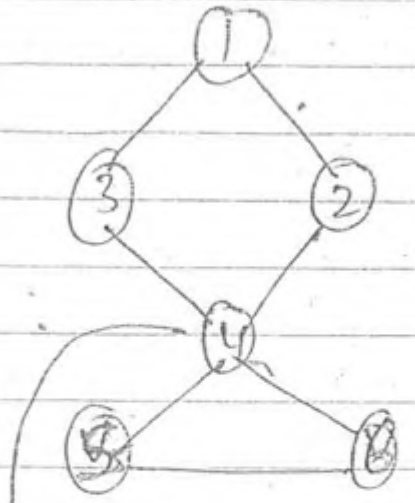


A B 4 V





Articulation Point



→ 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.

Sol^n

(i) A, B, E, G, F, D, C, H

A, B, F, D, C, H, E, G

(ii) A, B, E, G, F, C, H, D

(iii) A, D, F, C, H, B, E, G.

Let G be undirected Graph

Consider a DFT of g and Let T be the resulting DFT seach tree. Let u be a vertex in G and v be the first new vertex visted after visiting u in the traversal.

Which one of the following is true?

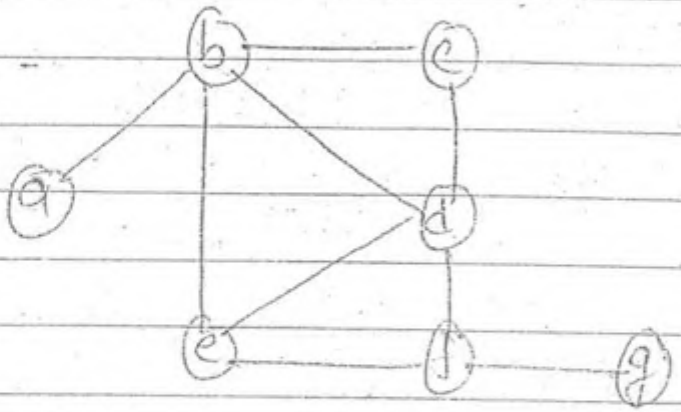
a) {u, v} must be an edge in G

b) if {u, v} is not an edge in G, then {u, v} must have same parent

c) if (u, v) is not an edge in g the v is leaf node in G.

d) If {u, v} is not an edge in G then u is leaf in T

Q. Consider the following graph.



a possible DFT for above graph is.

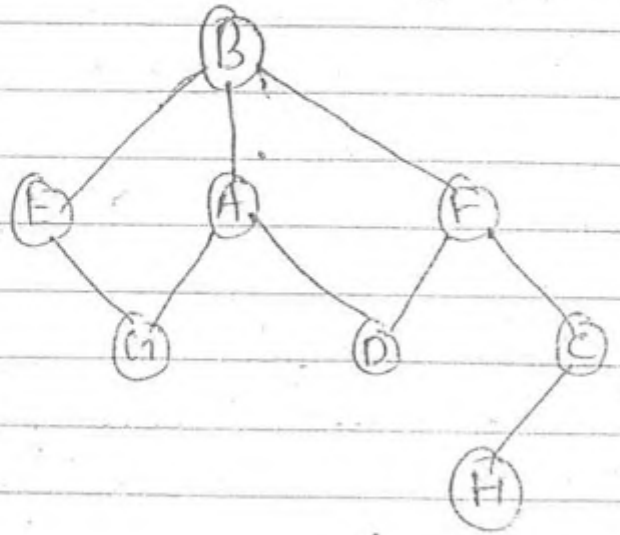
a) b, a, e, c, d, f, g.

b) d, b, a, e, f, c, g.

c) b, c, d, e, f, a, g.

~~d)~~ d, e, b, a, c, f, g.

Q. Consider the following Graph.



find 3 - different DFT starting from A.

Q// $V_1, V_5, V_7, V_8, V_6, V_2, V_3, V_4$

find DFT by considering always ascending order starting from V_3

$V_3, V_1, V_2, V_6, V_8, V_7, V_4, V_5$

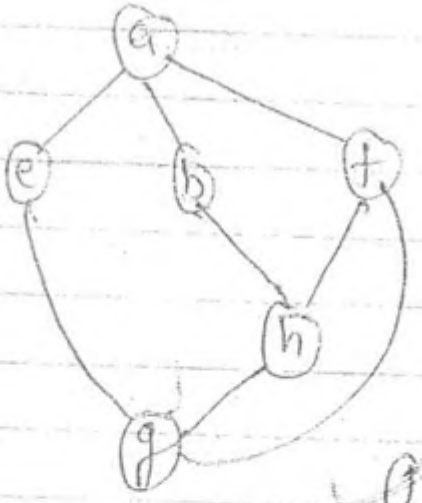
find 3 DFT starting from V_8.

$V_8, V_6, V_2, V_1, V_3, V_4, V_7, V_5$

$V_8, V_7, V_5, V_4, V_1, V_4, V_3, V_6, V_2$

$V_8, V_6, V_3, V_1, V_5, V_7, V_4, V_2$

Q//



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 ⇒ DFT

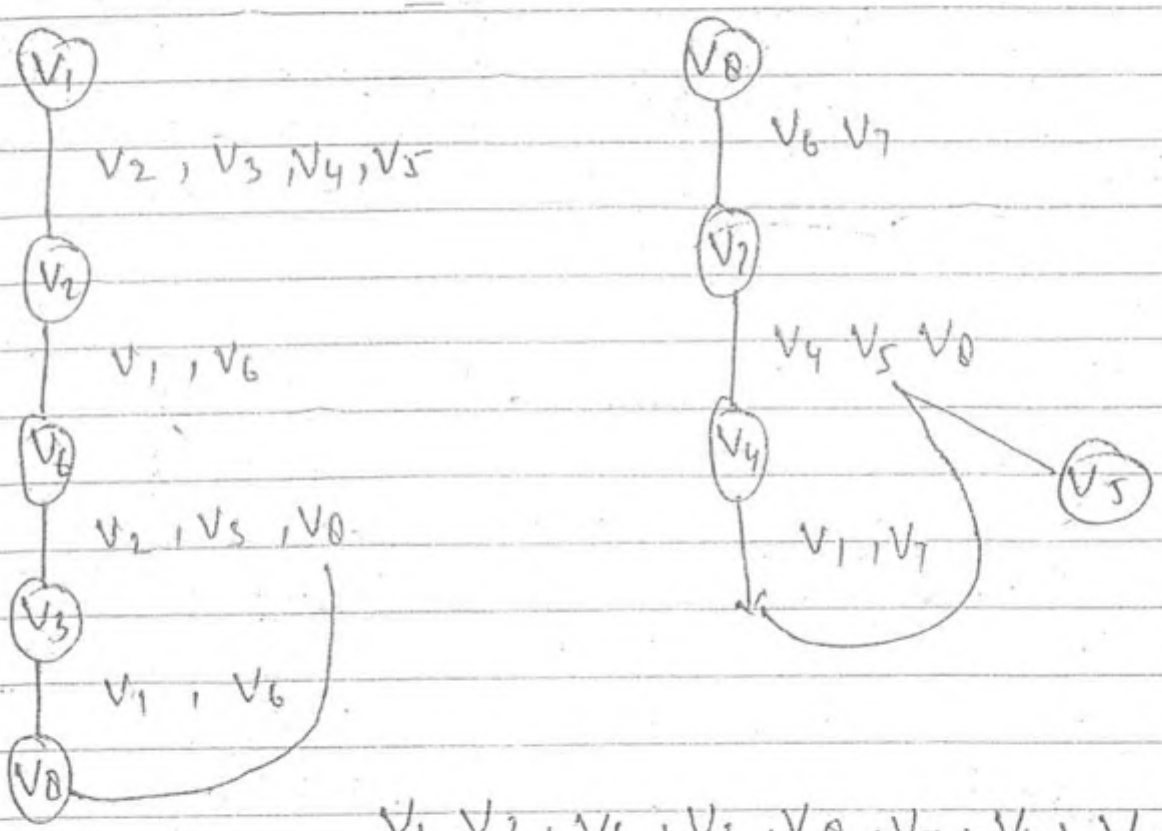
for every vertex called DFT

⇒ $O(V^2+V)$

$O(V+E)$

(I) The ds we are using the DFT is stack

(II) The time complexity of DFT $O(V+E)$



$V_1, V_2, V_6, V_3, V_0, V_7, V_4, V_5$

DFT (V_2)

① ~~①~~

② $w = V_1, V_6$

③ ~~③~~

DFT (V_6)

① ①

② $w = V_3 \cdot V_0$

③ ③

DFT (V_3)

① ①

② $w = V_1, V_6$

DFT (V_0)

① ①

② $w = V_6, V_7$

③ ~~③~~

④ DFT (V_7)

① ①

② $w = V_4, V_5 V_0$

③ ③

④ DFT (V_4)

① ①

② $w = V_1, V_7$

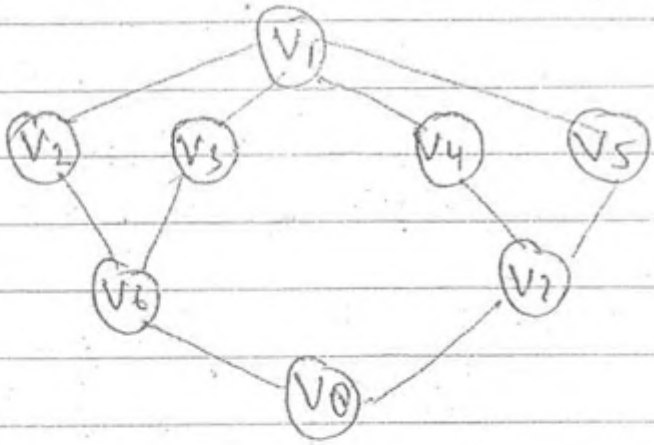
DFT (V_5)

① ①

② ③ $w = V_1, V_7$

DFT

```
DFT(V)
{
1. visited(V) = 1
2. for all w adjacent to V
   {
3.   if (w is not visited)
4.     DFT(V);
   }
}
```

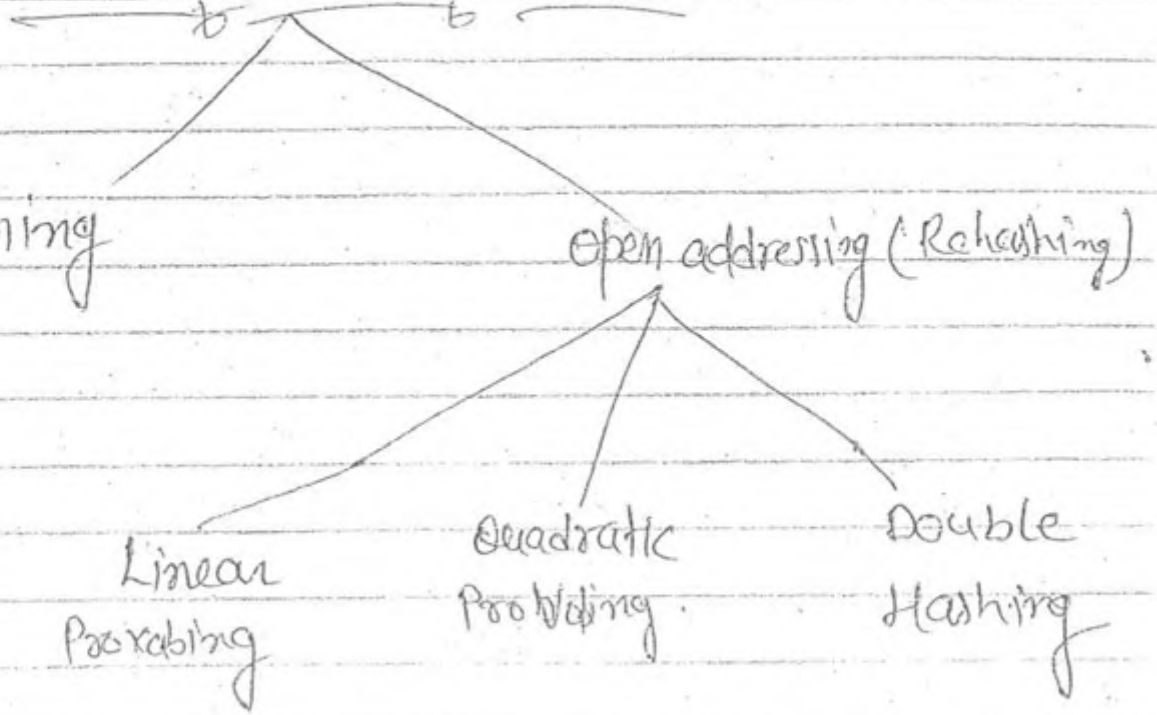


$V_1\ V_2\ V_6\ V_3\ V_0\ V_7\ V_4\ V_5$

DFT(V_1)

```
1.
2. w = V2 V3 V4 V5
3.
4. DFT(V2)
```

Collision Resolution Technic

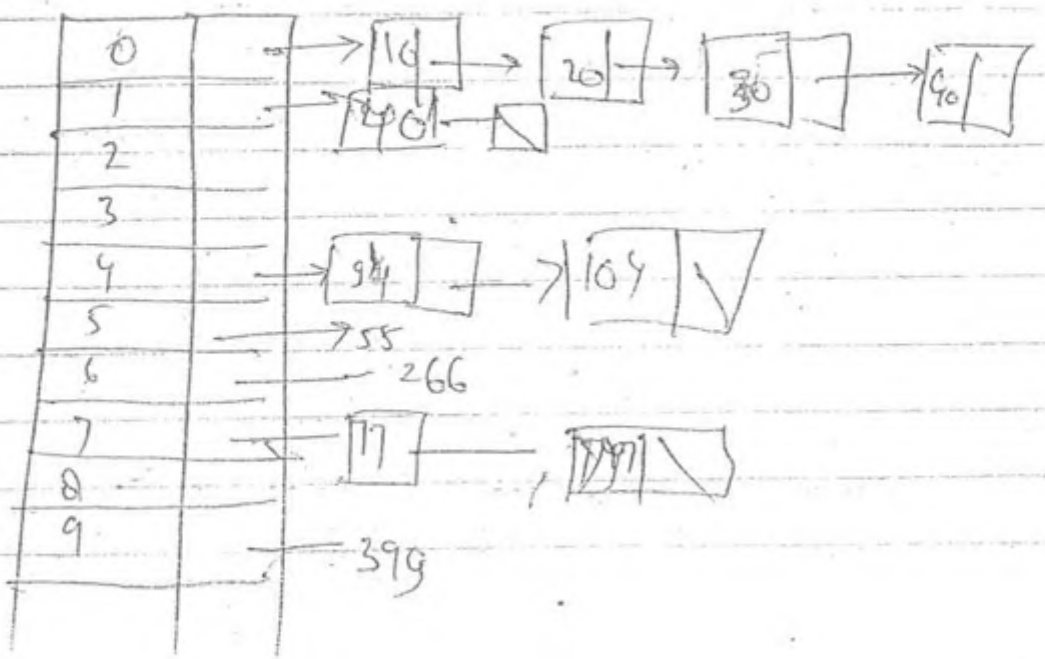


Collission - Resolution by Chaining

Key: 10 77 94 399 401 505 104 266 747

Hash function, Key mod 10

Collission Resolution Technic = Chaining



Drawback

1) We are taking extra space also to store & when collision occurs, ~~instead using a variable space~~. Even table have empty space.

2) In the worst case is be having linear searching.

3) Unlimited collisions will be handled by this method.

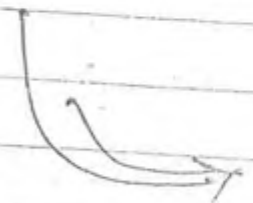
1) Does not required pre-knowledge of How many are comming into hash.

If the element are store in hash table using chaining, we can delete these element easily whenever not required.

Open Addressing

No storage require for linked list.

If I hash to particular slot in hash table, it already fulled. 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	
	9	
h(369, m-1)		

Linear Probing

Keys: 25, 30, 77, 55, 62, 78, 99, 52, 101, 41,

Hash fun^n = Key modulo 10

Collision Resolution technique = Linear Probing.

$h'(K) = K \bmod m$

$h(K, i) = (h'(K) + i) \bmod m$

$\forall\ i \in \{0, 1, 2, 3 - - - - m-1\}$

0	30
1	101
2	62
3	52
4	41
5	25
6	55
7	77
8	78
9	99

$h'(25) = 25 \bmod 10$
$= 5$
$h(25,0) = (h'(K) + 0) \bmod 10$
$= 5$

$h'(30) = 30 \bmod 10$
$= 0$
$h(30,0) = (0+0) \bmod 10$
$= 0$

$h'(77) = 77 \bmod 10$
$= 7$
$h(77,0) = (7+0) \bmod 10$
$= 7$

$h'(55) = 5$
$h(55,0) = (5+0) \bmod 10$
$= 5$
$h(55,1) = (5+1) \bmod 10$
$= 6$

$h'(62) = 2$
$h(62,0) = 2$

$h'(52) = 2$
$h(52,0) = (2+0) \bmod 10$
$= 2$
$h(52,1) = (2+1) \bmod 10$
$= 3$

$h'(41) = 1$
$h(41,0) = (1+0) \bmod 10$
$= 1$
$h(41,1) = (1+1) \bmod 10$
$= 2$
$h(41,2) = (1+2) = 3$
$h(41,3) = (1+3) \bmod 10$
$= 4$

Q. Let $h(K) = K \bmod 7$

Keys : 29, 36, 16, 30

Collision Resolution Technique :- linear probing.

While calculating the no of collision inserting the given key.

0	
1	29
2	36
3	16
4	30
5	
6	

$h'(K) = K \bmod 7$

$h'(29) = 29 \bmod 7 = 1$

$h(29,0) = (1+0) \bmod 7$

$= 1$

Collision = 4

Q Keys : 12, 10, 15, 2, 3, 25, 5 and is are inserted into an initially empty, hash table length 10. Using linear probing

hash fun^n = K mod 10

0	
1	10
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

= Hash Table size $m = 11$

hash fun" is given as follows.

```
h(Key)
{
  int x;
  x = (Key+5) * (Key+5);
  x = x/16;
  x = x + Key;
  x = x % 11;
  return x;
}
```

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

Key = 43

$x = 48 \times 48;$

$x = (48 \times 48)/16 = 48 \times 3$

$x = (48 \times 3) + 48 ; = 187$

$x = 0$

return (0)

Key = 23

$x = 28 \times 28$

$x = 49$

$x = 49 + 23 = 72$

$x = 6$

Key = 1

$x = 36$

$x = 36/16 = 2$

$x = 2 + 1 = 3$

$x = 3 \% 11 = 3$

return (3)

Q A Hash table of length 10 uses open addressing with hash fun^n $h(K) = K \bmod 10$
CRT = linear Probing.

After inserting 6 values into an empty hash table, the table as shown below.

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

① Which 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.

1) How many different insertion sequence of the key values using the ~~same~~ same hash fun^n and linear probing result in the hash table shown above.

a) 10 , b) 20 c) 30 d) 40

Soln

46, 34, 42, 25		52, 33

$4! = 24$

34 42 23 52 46 33

$3! = 6$

total $= 24 + 6 = 30$

Note

① In linear probing we have probing one slots after another..

② If an empty slots then long runs of occupied slots increase average searching time. this problem is known as.. primary clostering.

* In the worst case linear probing will take O(n) time. to reach for particular element.

* In linear probing deleting is difficult. ie delete one element it creates problem to others.

Quadratic Probing

$h'(K) = K \bmod M$

$h(K,i) = (h'(K) + C_1 \cdot i + C_2 i^2) \bmod M$

$\forall i \in (0,1,2,3 \ldots m-1)$

h fun^n = Key mod 10

Keys = 10, 15, 38, 49, 20
30, 25

0	10
1	35
2	20
3	
4	
5	15
6	30
7	25
8	38
9	49

C · R · T = QP

$c_1 = 1$, $c_2 = 1$

$0 = 10 \bmod 10$

$h(10,0) = 0$

$5 = 15 \bmod 10$

h

$h'(30) = 30 \bmod 10$

$= 0$

$h(30,0) = (h'(30) + c_1 \cdot 0 + c_2 \cdot 0^2) \bmod 10$

$= (0 + 0 + 0) \bmod 10$

$= 0$

$h'(20)$

$0 = 20 \bmod 10$

$h(20,0) = (h'(20) + 0 + 0^2) \bmod 10$

$= 0$

$h(20,1) = (h'(20) + 1 \cdot 1 + 1 \cdot 1^2) \bmod 10$

$= 0 + 1 + 1$

$= 2$

$h'(30)$

$0 = 30 \bmod 10$

$h(30,0) = (h'(30) + 0 + 0^2) \bmod 10$

$= 0$

$h(30,1) = (h'(30) + 1 \cdot 1 + 1 \cdot 1^2) \bmod 10$

$0 + 1 + 1$

$= 2$

$h(30,2) = (h'(30) + 1 \cdot 2 + 1 \cdot 2^2) \bmod 10$

$= (0 + 2 + 4) \bmod 10$

$= 6$

$h'(40) = 40 \bmod 10$

$= 0$

$h(40,0) = (0 + 0 + 0^2) \bmod 10$

$= 0$

$h(40,1) = (0 + 1 + 1^2) \bmod 10$

$= 2$

$h(40,2) = (0 + 3 + 2^2) \bmod 10$

$= 6$

$h(40,3) = (0+3+3^2) \bmod 10$

$= 12 \quad 2$

$h(40,4) = (0+4+4^2) \bmod 10$

$= 0$

$h(40,5) = (0+5+5^2) \bmod 10$

$= 0$

$h(40,6) = (0+6+6^2) \bmod 10$

$= 2$

$h(40,9) = (0+9+9^2) \bmod 10$

$= 0$

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

$h'(35) = 35 \bmod 10$

$= 5$

$h(35,0) = 5 + 0 + 0^2$

$= 5$

$h(35,1) = 5 + 1 + 1^2$

$= 7$

$h(35,2) = (5 + 2 + 2^2)$

$= 1$

Key 45

$h'(45) = 45 \text{ and } 10$

$= 5$

$h(45,0) = 5 + 0 + 0^2$

$= 5$

$h(45,1) = 5 + 1 + 1^2$

$= 7$

$h(45,2) = 5 + 2 + 2^2$

$= 1$

$h(45,3) = 5 + 3 + 3^2$

$= 7$

If two keys having same starting hash address. they both will follow same path. this problem is known as Secondary cluster bcoz of secondary clustering average searching little bit increases.

Keys = 10, 22, 31, 4, 15, 10, 17, 88, 59

$M = 11$

$c_1 = 1$, $c_2 = 3$.

$h'(k) = k$; $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod M$

C.R.T = Q.P.

=

0	22
1	59
2	
3	88
4	4
5	
6	17
7	10
8	15
9	31
10	10

✓ $h(10,0) = (10 + 0 + 0) \bmod 11$

$= 10$

✓ $h(22,0) = (22 + 0 + 0) \bmod 11$

$= 0$

✓ $h(31,0) = (31 + 0 + 0) \bmod 11$

$= 9$

✓ $h(15,0) = (15 + 0 + 0) \bmod 11$

$= 4$

$h(15,1) = (15 + 1 + 1^2) \bmod 11$

$= 0$

✓ $h(10,0) = (15 + 0 + 0) \bmod 11$

$= 7$

$h(17,0) = (17 + 0 + 0) \bmod 11$

$= 6$

$h(00+0) = (00 + 0 + 0) \bmod 11$

$= 0$

$h(00,1) = (00 + 1 + 3) \bmod 11$

$= 4$

$h(00,2) = (00 + 2 + 3 \cdot 2^2)$

$= (102) \bmod 11) \quad = 5$

$h(59,0) = (59 + 0 + 0) \bmod 11$

$= 4$

$h(59,1) = (59 + 1 + 3) \bmod 11$

$= 63 \bmod 11 \quad = 8$

$h(59,2) = (59 + 2 + 3 \cdot 2^2) \bmod 11$

7

$h(59,3) = (59 + 3 + 3 \cdot 3^2) \bmod 11$

1

L.P > Q.P

more searching less searching

Double Hashing (Good Hashing Technic)

L.P

⇓

Primary Clustring

⇓

Quadratic Probing

⇓

S.C

⇓

Double Hashing

$$h(K,i) = (h_1(K) + i \cdot h_2(K)) \bmod m$$

$$\forall i \in \{0, 1, 2, 3, \ldots, m-1\}$$

$$h_1(K) = K \bmod m$$

$$h_2(K) = 1 + (K \bmod (m-1))$$

eg

$M = 10$

Keys = 10, 25, 14, 16, 20, 30, 35, 45, 65.

0	10
1	65
2	35
3	20
4	14
5	25
6	16
7	45
8	30
9	

$h_1(10) = 10 \bmod 10$

$= 0$

$h(10,0) = (0 + 0 \cdot h_2(K)) \bmod 10$

$= 0$

$h_1(25) = 25 \bmod 10$

$= 5$

$h_1(25,0) = (5 + 0 \cdot h_2(K)) \bmod 10$

$= 5$

$h_1(14) = 14 \bmod 10$

$= 4$

$h_1(14,0) = (4 + 0 \cdot h_2(K)) \bmod 10$

$= 4$

$h_1(16) = 16 \bmod 10$

$= 6$

$h(16,0) = (16 + 0 \cdot h_2(K)) \bmod 10$

$= 6$

$h_1(20) = 20 \bmod 10$

$= 0$

$hi(20,0) = (0 + 0 \cdot h_2(5)) \bmod 10$

$= 0$

$h_2(20) = 1 + (20 \bmod (9))$

$= 1 + 2$

$= 3$

$h(20,1) = (0 + 1 \cdot 3) \bmod 10$

$= 3$

$h_1(30) = (0 + 0 \cdot h_2(30)) \bmod 10$

$= 0$

$h_2(30) = 1 + 30 \bmod (9)$

$= 1 + 3 = 4$

$h(30,1) = (0 + 1 \cdot 4) \bmod 10$

$= 4$

$h(30,2) = (0 + 2 \cdot 4) \bmod 10$

$= 8$

$h_1(35) = 35 \bmod 10$

$= 5$

$h_2(35) = 1 + 35 \bmod 10$

$= 1 + 0 = 9.$

$h(35,0) = (5 + 0 \cdot h_2(K)) \bmod 10$

$= 5$

$h(35,1) = (5 + 1 \cdot 9) \bmod 10$

$= 4$

$h(35,2) = (5 + 2 \cdot 9) \bmod 10$

$= 3$

$h(35,3) = (5 + 3 \cdot 9) \bmod 10$

$= 2$

$h_1(45) = 45 \bmod 10$

$= 5$

$h_2(45) = 1 + 45 \bmod 9$

$= 1$

$h(45,0) = (5 + 0 \cdot 1) \bmod 10$

$= 5$

$h(45,1) = (5 + 1 \cdot 1)$

$= 6$

$h(45,2) = (5 + 2 \cdot 1)$

$= 7$

* In linear probing whenever collission is occured who will always add # by 1. It does not matter what key. it is bcoz of that region average searching time increase.

* In Quadritic probing whenever 2 keys having same starting hash address we will always add same Quadratic eqn to both the keys which will leads to secondary clouster.

* In double hashing 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 genrated by the second hash function.

* In double hashing S.C is completly eleminating bcoz we are adding 2 different random no for both the keys

27/07/011

eg

$M = 13$, $h_1(K) = K \bmod M$, $h_2(K) = 1 + K \bmod 11$

29, 69, 72, 98, 10, 50

$h(K, i) = (h_1(K) + i\,h_2(K))$

$h_1(K) = 29 \bmod 13$

$h_1(29) = 3$

$h(29, 0) = (3 + 0 \cdot h_2(K))$

$= 3$

$h(69, 0) = (4 + 0 \cdot h_2(K))$

$= 4$

$h(72, 0) = (7 + 0 \cdot h_2(K))$

$= 7$

$h_1(92) = 98 \bmod 13$

$= 7$

$h_2(92) = 1 + 98 \bmod 11$

$= 1 + 10$

$= 10$

$h(98, 0) = (7 + 0 \cdot h_2(K)) \bmod 13$

$= 7$

$h(98, 1) = 7 + 1(10)$

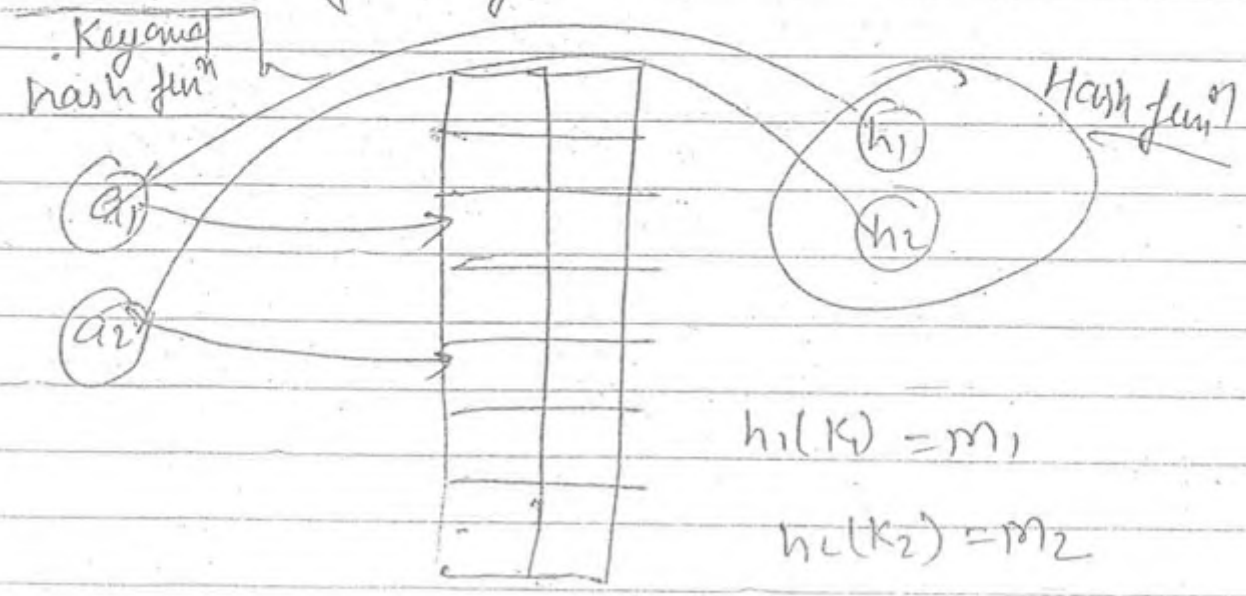
$= 10 \bmod 13$

$= 5$

0	
1	
2	
3	29
4	69
5	98
6	
7	72
8	
9	
10	10
11	50
12	

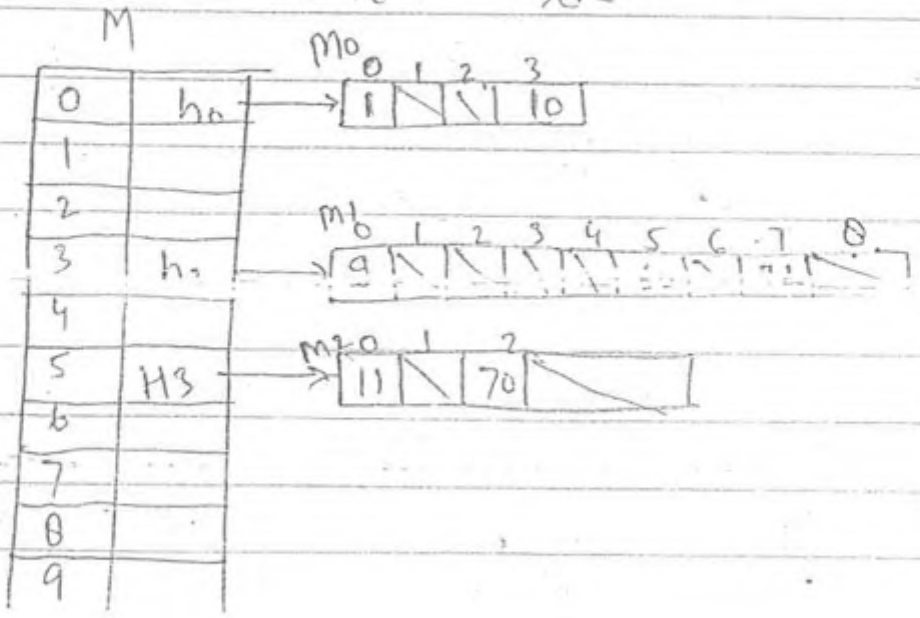
Weakness of Hashing

For any choice of hash fun^n there exist bad set of keys that all hash to same slot.



The sol^n to this problem pick hash fun^n randomly, which is known as Universal Hashing.

Perfect Hashing



$h(1) = 0$ $\quad h_0(1) = 0$

$h(10) = 0$ $\quad h_0(10) = 3$

$h(9) = 3$ $\quad h_2(9) = 0$

$h(60) = 3$ $\quad h_2(60) = 5$

$h(72) = 3$ $\quad h_2(72) = 7$

$h(11) = 5$ $\quad h_3(11) = 0$

$h(70) = 5$ $\quad h_3(70) = 2$

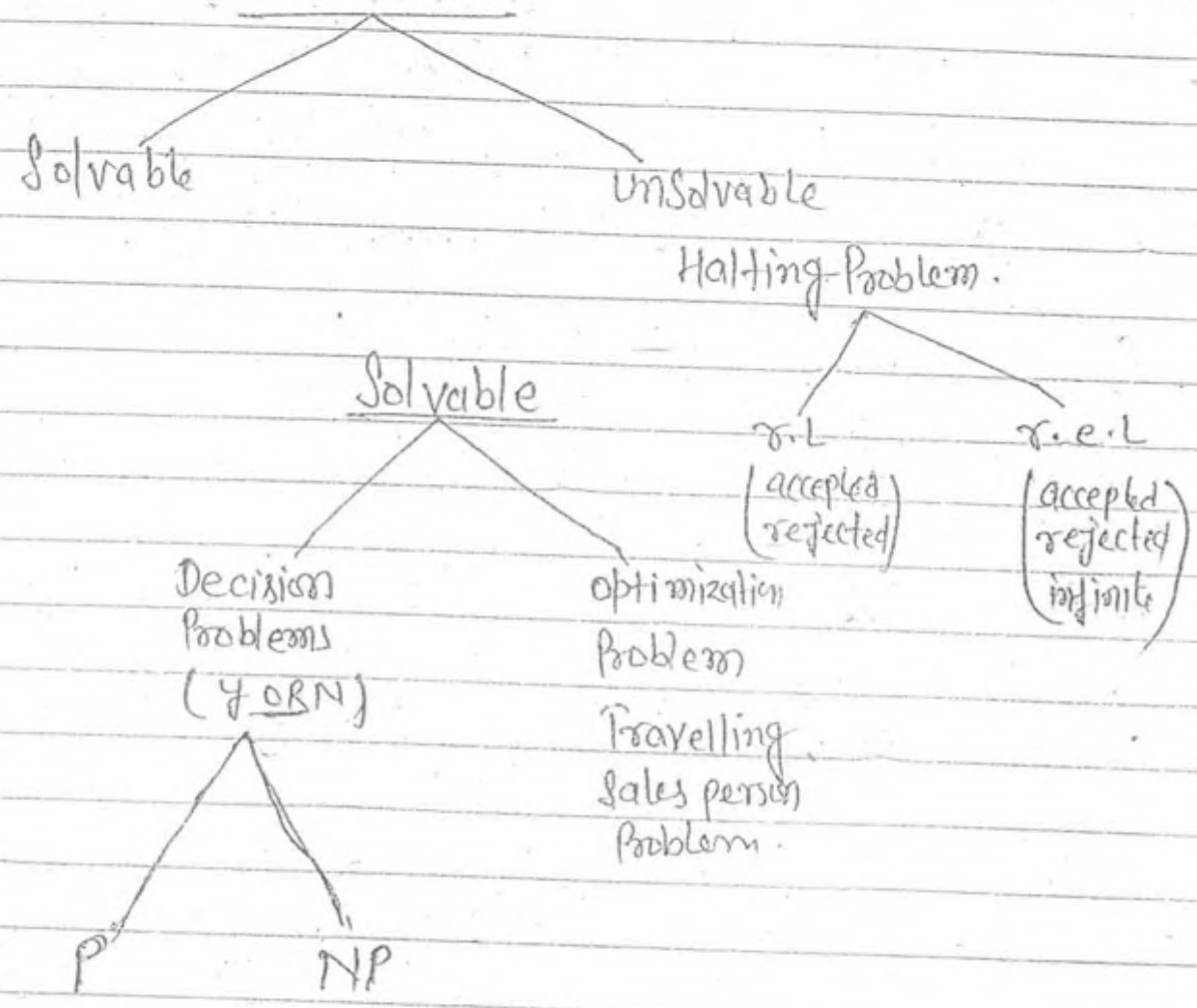
Note—

Instead of making linked list of key hasing to slot i we can use small seconday hash function with small table size.

* Total no of collision in second level of hashing is $\max^{m}(Y_2)$ no collision.

* The worst case time complexity to search an element is $O(1)$.

P, NP, NPC, NPH



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 polynomials 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) Eular Graph $\Rightarrow \theta(V^2)$
(5) Prim's algo $\Rightarrow (E+V) \log V$
(6) Merge Sort $\Rightarrow \theta(n \log n)$
(7) LCS $\Rightarrow O(n^2)$
(8) All pair shortest Path $\Rightarrow O(n^3)$
(9) Matrix chain multiplication $\Rightarrow O(n^3)$
(10) Binary Search $\Rightarrow$

NP- Class Problem

$\Longrightarrow$ A Problem P is in NP iff there exist non deterministic time polynomide time algo to solve that problem.

~~Set of NP Class~~ :-

```
ND Sorting ( a , n )
{
  for( i = 1 to n)  → with guess.
  {
    find ith smallest element
  }
}
```

$\Rightarrow O(n)$

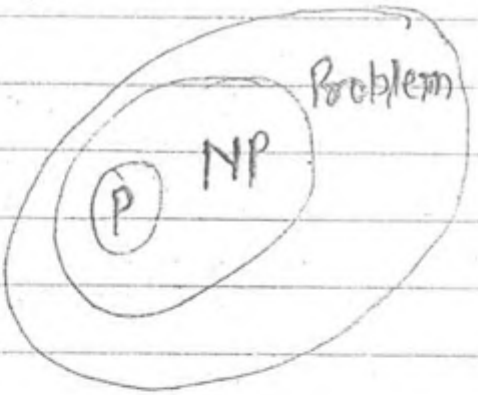
without guess to solve a problem in polynomial time is called P-class. and without guess is called NP class

Set of NP-class

1. Travelling Sales Person.
2. Hamiltonian
3. 0/1 Knapsack
4. Sum of Subsets.
5. Linear Seach.
6. Quick Sort
7. Merge Sort
8. Eular Graph.

Every P class problem is NP class problem.

$P \subseteq NP$



$P \subseteq NP$

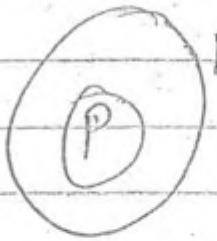
OR

$P \neq NP$

OR

$P = NP$

P & NP



7 - (1-million dollar)

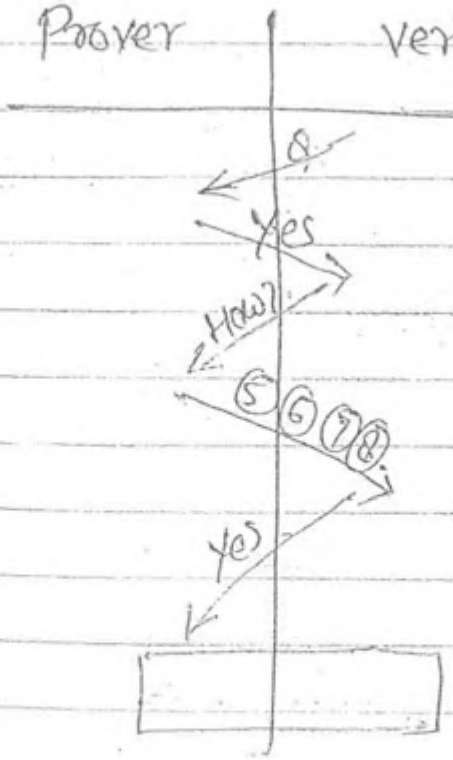
⇓

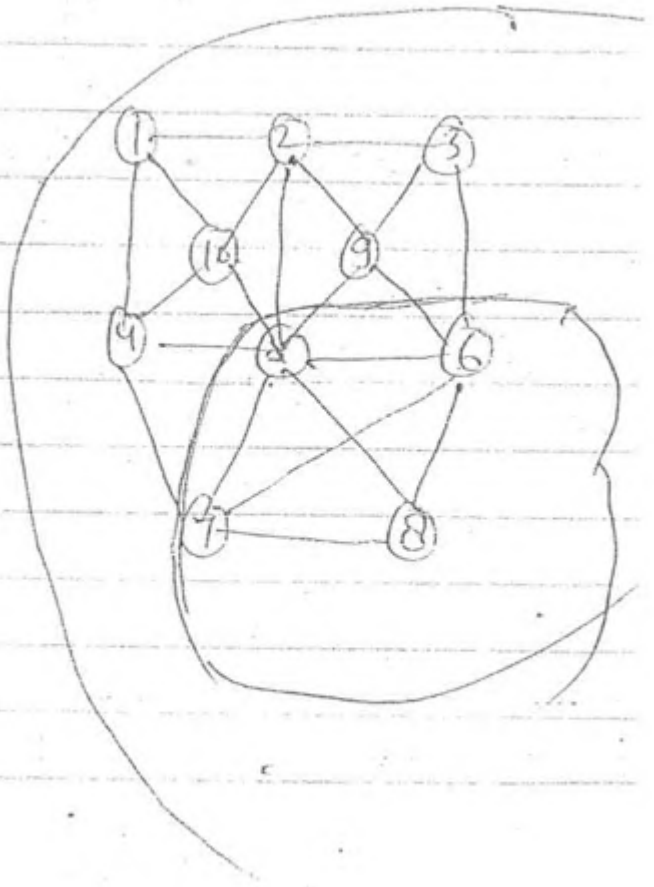
P ⊂ NP

NP

I/P : Grap G(V, E) ; K

Q : Does G - Contain K-Clique.





P-Class Problem ⇒ Solving entire problem with polynomial time. is called P-Class Problem.

NP-Class Problem ⇒

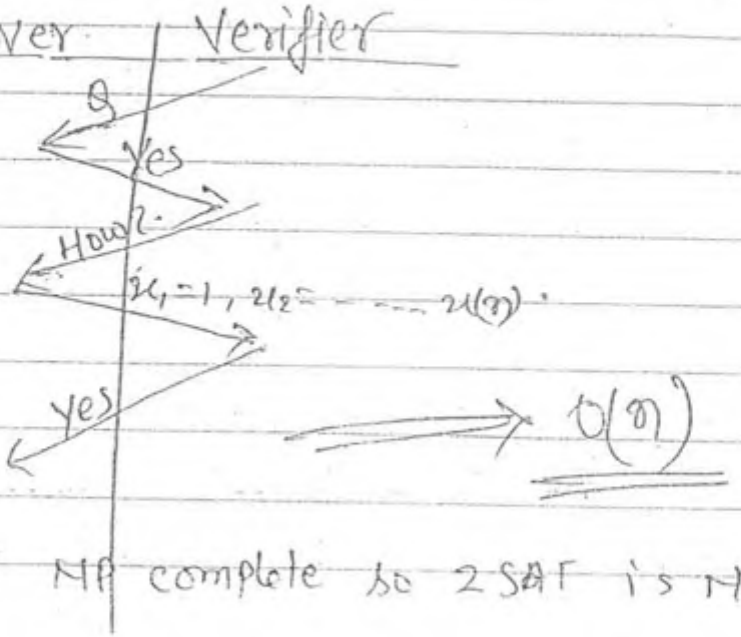
Verifying the given correct input in polynomial time is called NP-class problem.

Ex i/p : A boolean expression (CNF) and each term of CNF contain 3-variables.

eg. $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1' \vee x_2' \vee x_3') \wedge (x_1 \vee x_{100} \vee x_{50}) \wedge (x'_{99} \vee x_1' \vee x'_{1000})$

Q : Does the given boolean expression contain truth values for all variables $x_1 \dots x_n$ such that given expression becomes true ?.

Idⁿ



3 SAT is NP complete so 2 SAT is NP complete

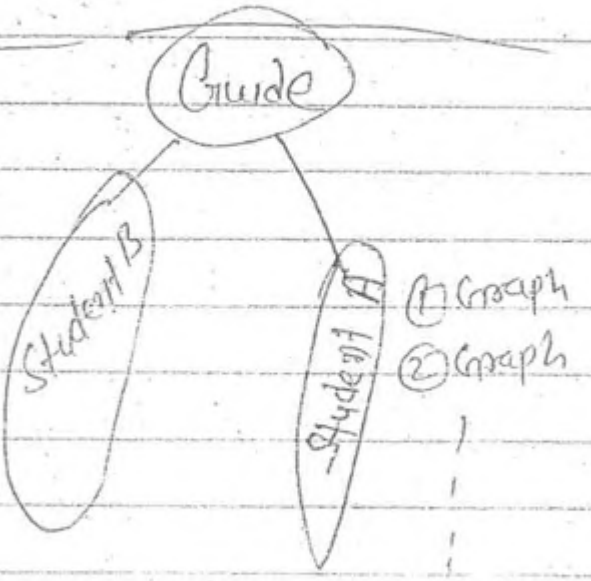
Polynomial time reduction

A : $bx + c$

B : $ax^2 + bx + c$

$\Downarrow$ Polynomial (exponential)

A : $ax^2 + bx + c$



A - NP-C

Let A - Known

B - Unknown.

If A is polynomially reduced to B

OR

$A \leq_p B$

then B is also known Problem

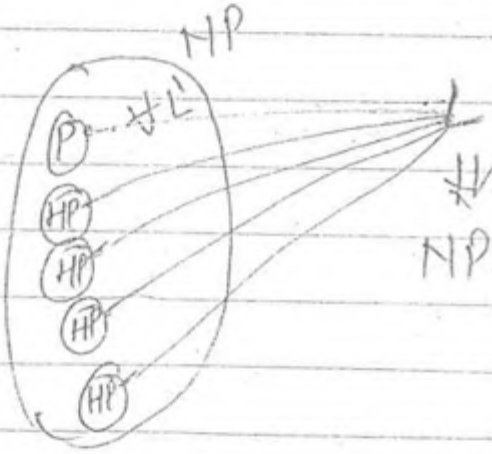
NP-Hard

Let L be the given problem which we want to prove NP Hard.

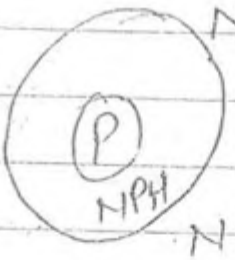
If L' is Polynomiall reduced to L for all L' belongs to NP then L is NPH

$L' \leq_p L$, $\forall L' \in NP$ then.

L is NPH



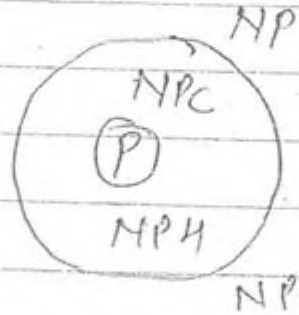
(My Problem L is as ~~least~~ Hardest as NP-Problems)



N-P Complete

A problem L is NP-complete iff.

1. $L \in NPH$
2. $L \in NP$



Note

(1) If L is NP-complete and
L is polynomialy reduced to A ie $L \leq_p A$
then
A is NPH

NPC →

(2) If L is P-class Problem and.
A is Polynomially reduced to L ie $A \leq_p L$
then
A is also P-class Problem

thm of $L \in NP$

$A \leq_p L$ then A is also NP-class.

(i) 3-SAT ⇒ NP Complete.

⇓

Vertex cover

K-Clique Set cover Sum of Subsets Problem Hamiltonian

Set cover → 2011 (3000)

Sum of Subsets Problem → 0/1 Knapsack.

Hamiltonian → Travelling Sales Person

⇒ If any NP Complete problem is polynomial time Solvable then $P = NP$

⇒ Any Problem in NP-Complte is not solvable in polynomial time. then $P \neq NP$

A, B, and C are three decision Problems and A is NPC then which one of the following is true.

a) $A \in NP$ (A-NP)

b) $\forall L \in NP, L \leq_p A$ (A-NPH)

c) If $C \in NP$ and $A \overset{NPC}{\leq_p} C \overset{NPH}{}$ then C is NPC (Note 1)

d) If $B \in NP$ and $B \leq_p A$ then B is NPC (Note 2)

Note: (1)

Q

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 " " " " "

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.

3-SAT to Π

NPC $\Rightarrow \Pi$ (Note 1)

$\Downarrow$

NPH

Π to 3-SAT (Note 2)

$\Pi \in NP$

Q

Let S be a NPC and Q and R be 2 other problem not known to be in NP.

If $S \overset{NPC}{\leq_p} R$ $\overset{NPH}{}$ && $Q \overset{NP}{\leq_p} S$ $\overset{NPC}{}$ then which one of the following is true?

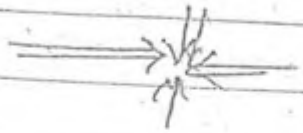
a) R is NPC

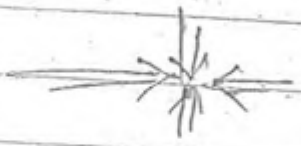
b) R is NPH

c) Q is NPC

d) Q is NPH.

OBST (Optimal Cost Binary Search Tree)





Any Comparision 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^2
Insertion ⇒	n^2	n	n^2
Bubble Sort ⇒	n^2	n^2	n^2

Sorting

- Comparision Based
  - Quick Sort, MergeSort
  - Heap sort
  - Selectionsort
  - Insertion sort
  - Bubble sort
- Non-Comparision 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≤K; i++)                    → O(K)
     c[i]=0;
② for( i=1; i≤n; i++)                    → O(n)
     c[a[i]]= c[a[i]] + 1;
③ for( i=1; i≤K; i++)                    → O(K)
     c[i]= c[i] + c[i-1];
④ for( J=n; J≥1; J--)                    → O(n)
   {
     B[c[a[J]]] = a[J];
     c[a[J]]= c[a[J]]-1;
   }
}
```

$O(n+K)$

$\Downarrow$

$O(n)$

K (0-5)

A:

2	5	3	0	2	3	0	3
1	2	3	4	5	6	7	8

① C: | 0 | 0 | 0 | 0 | 0 | 0 |
0 1 2 3 4 5

② C: | 2 | 0 | 2 | 3 | 0 | 1 |
0 1 2 3 4 5

C: | 2 | 0 | 2 | 3 | 0 | 1 |
0 1 2 3 4 5

// how many times i is present in the given array.

③ C: | 2 | 2 | 4 | 7 | 7 | 0 |
0 1 2 3 4 5

// how many elements are present such that all the elements are ≤ i

C: | 1 | 2 | 2 | 4 | 7 | 7 |
0 1 2 3 4 5

B: | 0 | 0 | 2 | 2 | 3 | 3 | 3 | 5 |
1 2 3 4 5 6 7 8

Note -

(1) Counting sort 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	0	0	0	0	0
0	1	2	3	4	5

(2) C:

0	0	0	0	0	0
0	1	2	3	4	5

C:

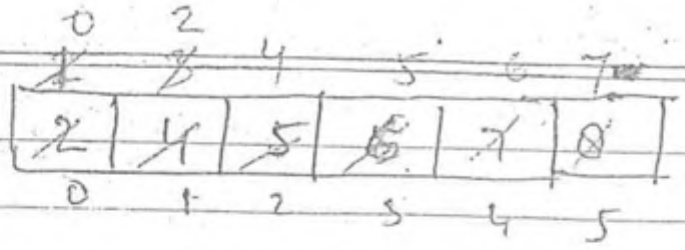
2	2	1	1	1	1
0	1	2	3	4	5

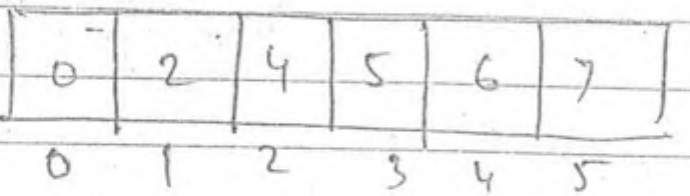
(3) C:

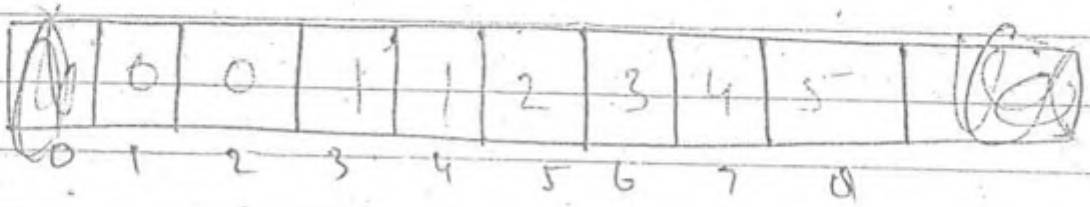
2	2	1	1	1	1
0	1	2	3	4	5

2	4	5	6	7	8
0	1	2	3	4	5

(4)







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 (1st LSB) : 420 451 911 143 375 825 786 137 007 569
(10n)

Pass 2 (2nd LSB) : 007 911 420 825 137 143 451 569 375 786
(10n)

Pass 3 (3rd LSB) : 007 137 143 375 420 451 569 786 825 911
(10n)

$\Rightarrow O(dn)$ dn = no of digit.

$\Rightarrow O(n)$

10n x 3

⇓

30n

⇓

O(n)

Eg

i/p : 329 457 657 839 436 720 355

1 2 3 4 5 6 7

Pass1 (1st LSB) : 720 355 436 457 657 329 839

⇒ n

Pass2 (2nd LSB) : 720 329 436 839 355 457 657

⇒ n

Pass3 (3rd LSB) : 329 355 436 457 657 720 839

⇒ n

Radix sort not in place but stable

O(dn)

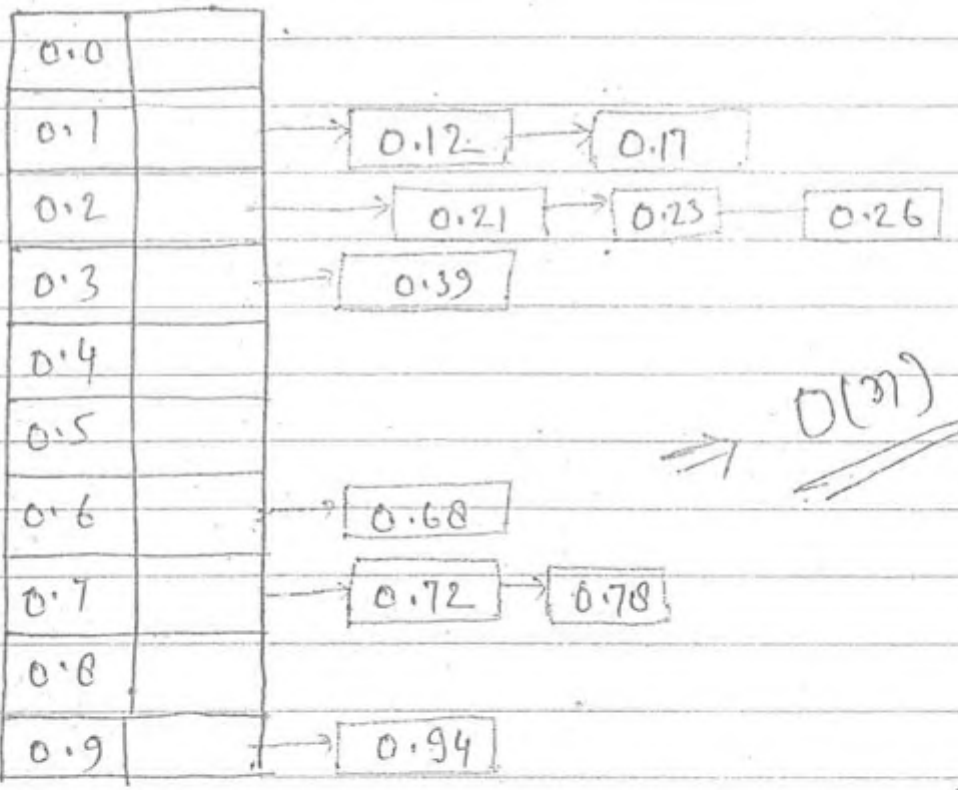
Bucket Sort (Assume uniform distribution)

Assume that all the elements are in the range [0, 1) ; [including 0 Excluding 1]

i/p $\frac{78}{100}$ $\frac{17}{100}$ $\frac{39}{100}$ $\frac{26}{100}$ $\frac{72}{100}$ $\frac{94}{100}$ $\frac{21}{100}$ $\frac{12}{100}$ $\frac{23}{100}$ $\frac{68}{100}$

: .78, .17, .39, .26, .72, .94, .21, .12, .23, .68

M = 10

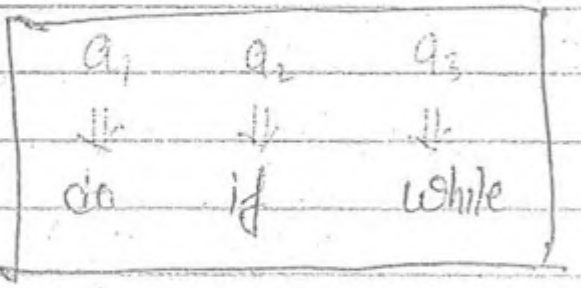


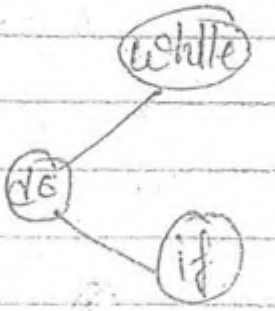
→ O(n)

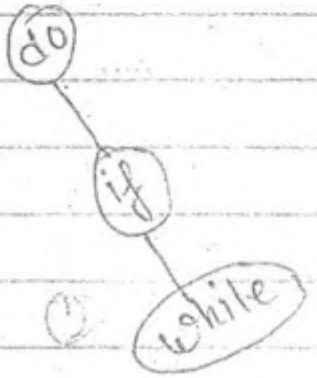
12	17	21	23	26	39	68	72	78	94

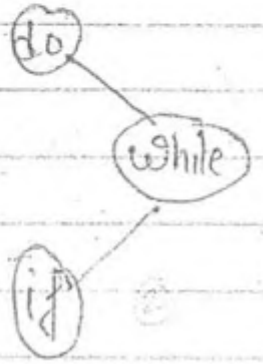
Bucket sort also not inplace

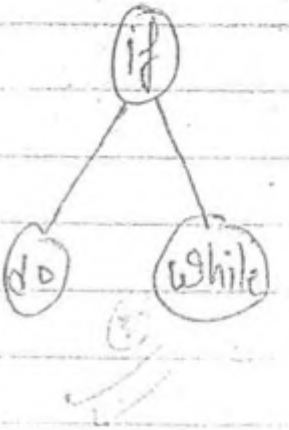
Optimal Cost Binary Search tree (OBST)

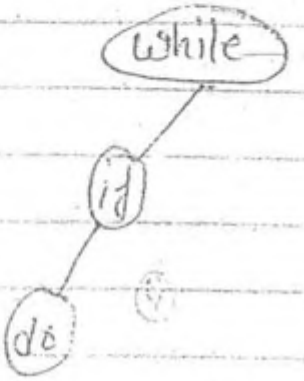


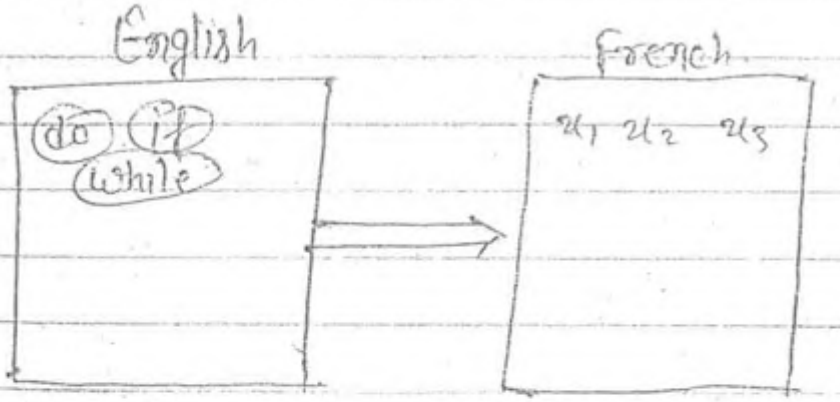












100

(1) do = 20 — 20 X 1 = 20
if = 50 — 50 X 2 = 100
while = 30 — 30 X 3 = 90
210

(2) $20 \times 1 = 20$
$50 \times 3 = 150$
$30 \times 2 = 60$
230

(3) $20 \times 2 = 40$
$50 \times 1 = 50$
$30 \times 2 = 60$
150

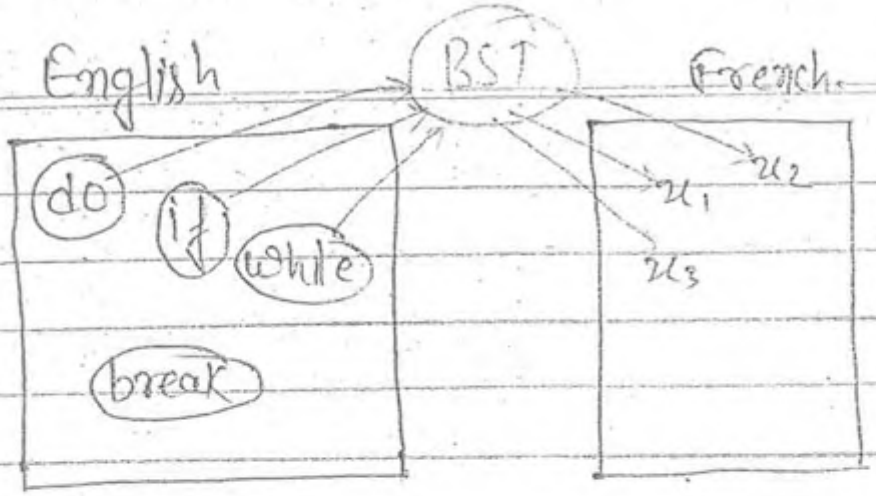
(4) $20 \times 3 = 60$
$50 \times 2 = 100$
$30 \times 1 = 30$
190

(5) $20 \times 2 = 40$
$50 \times 3 = 150$
$30 \times 1 = 30$
220

For the given 3 ~~Prob~~ Key words do, if, while with the frequency 20, 50, 30 respectively, 5 difference BST. In those 5 BST optimal one is (3)

Cost of a BST purly Based on.

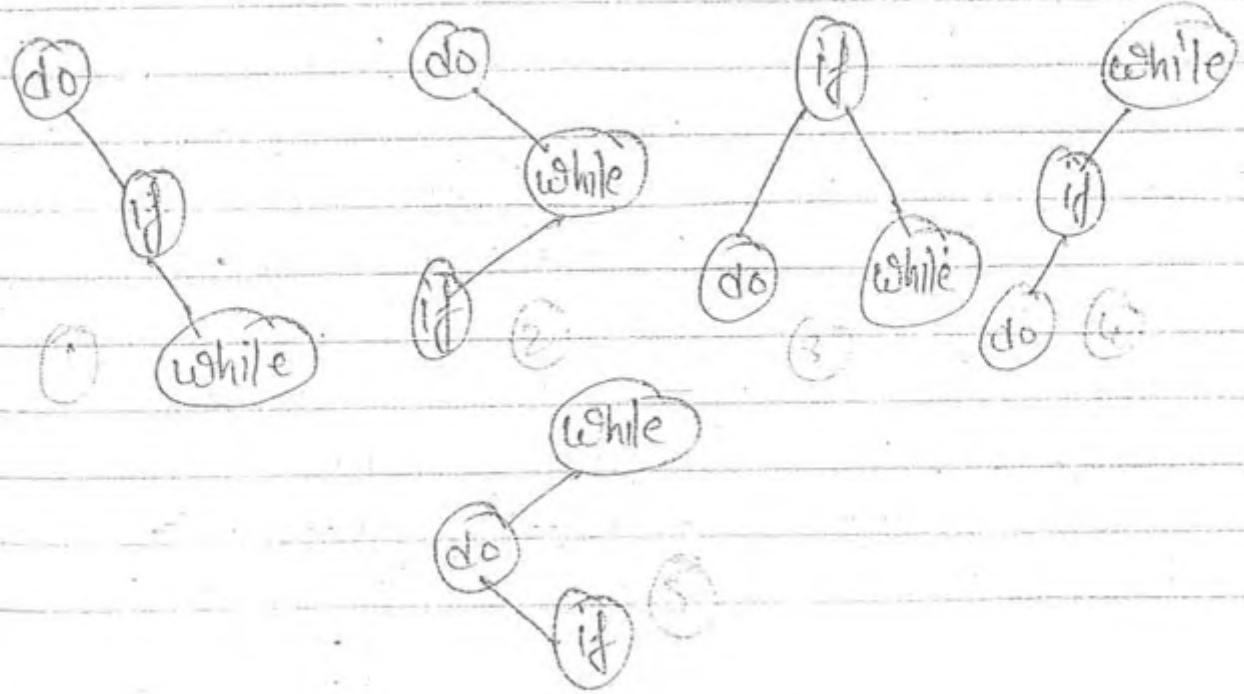
(i) no of leabels.
(ii) highest frequency Key words should be there near to root.



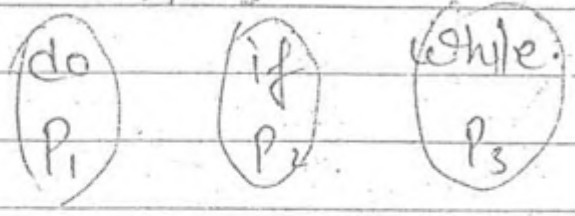
Cost of any BST going will be going to cost required are success full node ← cost requi for all unsuccess full node.

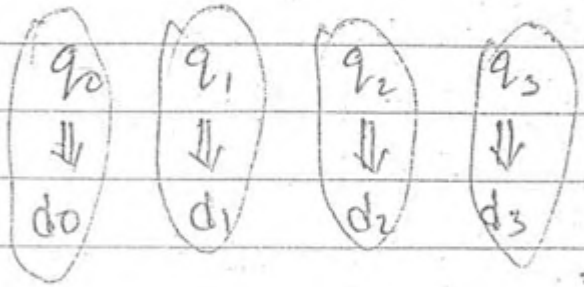
Cost(BST) = Cost(Successfull nodes) + Cost(unsuccessfulno

do, if, while



P// $n = 3$



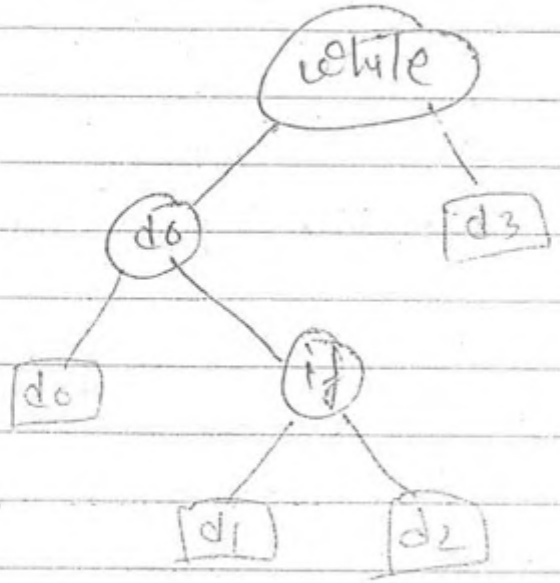


Cost (BST) = Cost (successfull node) + Cost (unsuccessfull node)

$$= \sum_{i=1}^{n} P_i \cdot l_i + \sum_{i=0}^{n} q_i \, l_i$$

$P_i \Rightarrow$ Probability of nodes.

$l_i \Rightarrow$ level of nodes.



Cost (SSN) = 20 X2 + 50 X3 + 30 X1

+

Cost (OSS) = $q_0 X3 + q_1 X4 + q_2 X4 + q_3 X2$

P

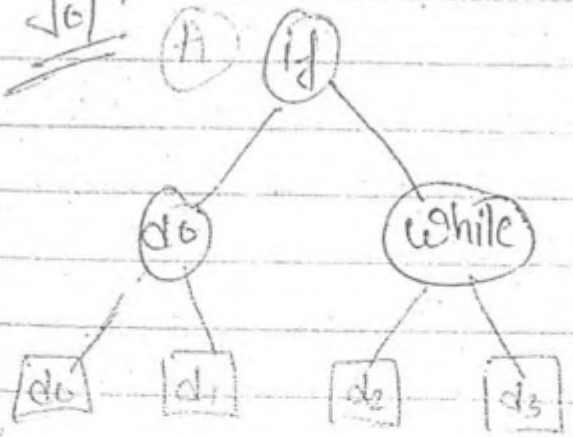
$n = 3$

$a_1, a_2, a_3 = (do, if, while)$

$(P_1 - P_3)$ 0.5 0.1 0.05

$(q_0 - q_3)$ 0.15 0.1 0.05 0.05

Soln



Cost (SSN) = 0.1 X1 + 0.5 X2 + 0.05 X

= .1 + 1.0 + .10 = 1.20

cost (OSS) = 0.15 X3 + 0.1 X3 + 0.05 X3

+ 0.05 X3

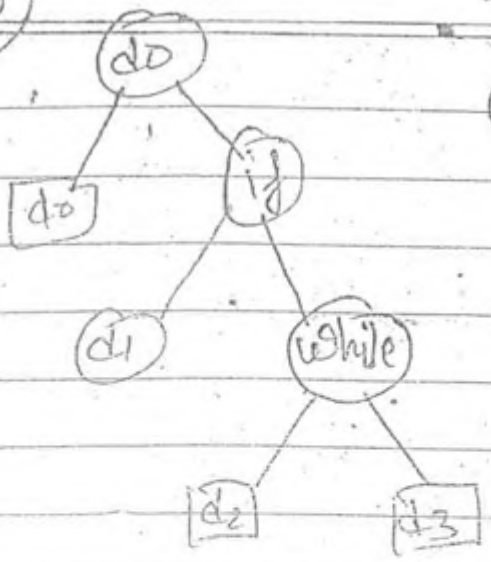
= .45 + .3 + .15 + .15

= 1.05

Cost (BST) = 1.20 + 1.05

= **2.25**

(B)



Cost(SSN) = 0.5 X1 + 0.1 X2 + 0.05 X3

= .5 + .2 + .15

= .85.

Cost (USN) = 0.15 X2 + 0.1 X3 + 0.05 X4

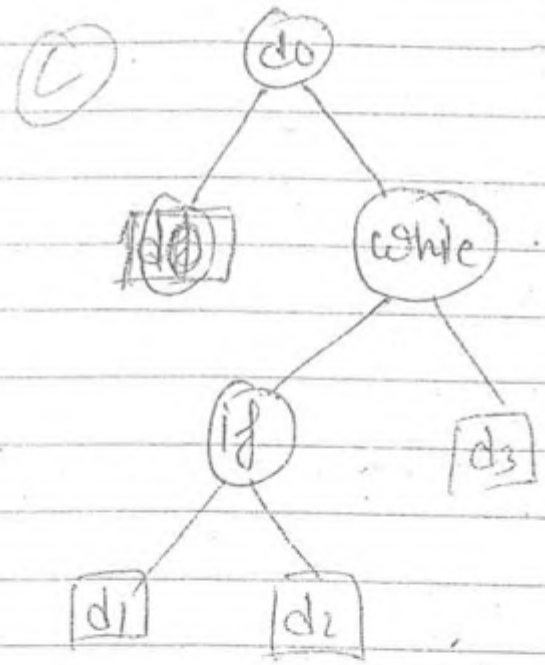
+ 0.05 X4

= .30 + .3 + .20 + .20

= 1.00

Cost(BST) = 1.05

(C)



Cost(SSN) = 0.5 X1 + 0.05 X2 + .1 X3

= .5 + .10 + .3

= .9

Cost (USN) = 0.15 X2 + 0.1 X4 + 0.05 X4

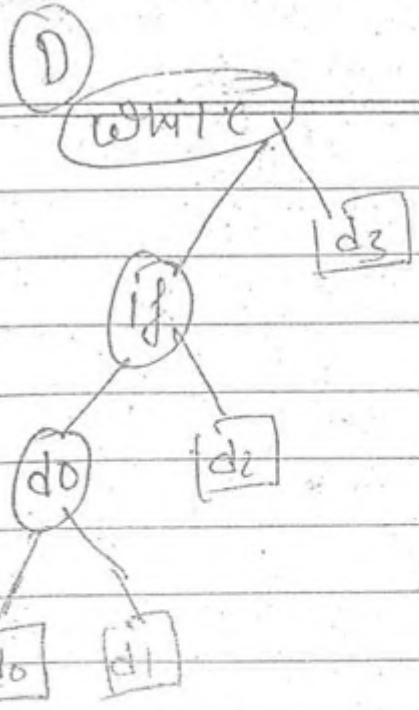
+ 0.05 X3

= .30 + .4 + .20 + .15

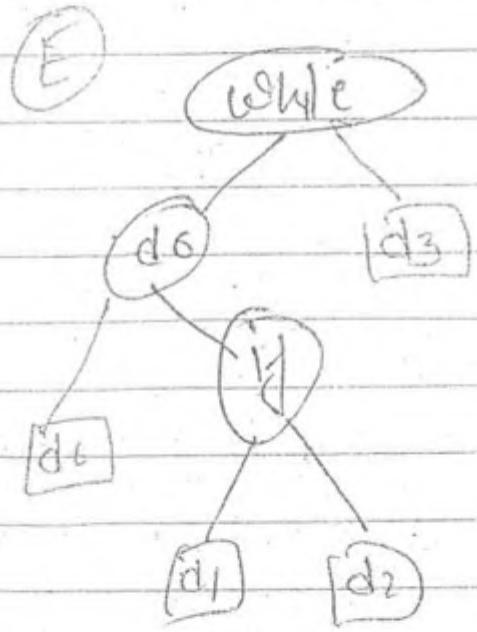
= 1.05

Cost(BST) = .9 + 1.05

= 1.95



Cost (BST) = $\frac{\text{sum}}{\text{num}}$ = 3.00



Cost (BST) = = 2.50

In the above 5 BST ~~tree be~~ tree B is optimal cost binary tree becoz higher probability node it self & and next higher probaibly 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.

Q//

$$W(i,j) = \sum_{K=i}^{j} p_i + \sum_{K=i-1}^{j} q_i$$

	do	if	while		
$p_1 - p_3$	0.5	0.1	0.05		
$q_1 - q_4$	0.15	0.1	0.05	0.05	
	q_0	q_1	q_2	q_3	

$e[i,j]$ = The min cost required for the optimal cost BST which contains i to j elements.

$$e[q_1,q_2,q_3] = \min \begin{cases} e[1,0] + e[2,3] + p_1 + W[1,0] + W[2,3] \\ e[1,1] + e[3,3] + p_2 + W[1,1] + W[3,3] \\ e[1,2] + e[4,5] + p_3 + W[1,2] + W[4,5] \end{cases}$$

D: ...

$$e[i,j] = \min \begin{cases} i \text{ as the root} \\ e[i,i-1] + e[i+1,j] + p_i + w(i,i-1) \\ \quad + w(i+1,j) \\ j \text{ as the root} \\ e[i,j-1] + e[j+1,j] + p_j + w(i,j-1) \\ \quad + w(j+1,j) \end{cases}$$

OR

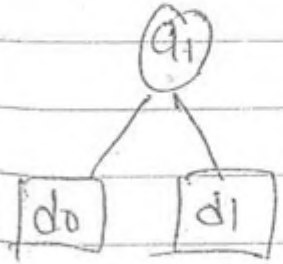
$$e[i,j] = \min \begin{cases} \text{for} \\ \dots \\ e[i,r-1] + e[r+1,j] + p_r + w(i,r-1) \\ \quad + w[r+1,j] \\ i \le r \le j \end{cases}$$

$n = 1$

a_1

p_1

$q_0\ q_1$



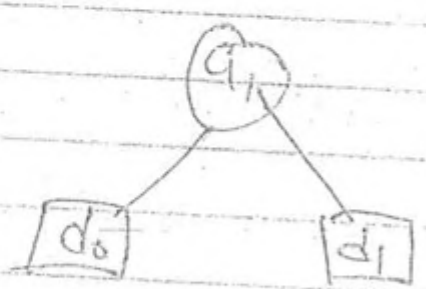
Q1

$n = 1$

$a_1 = d_0$

$P_1 = 0.5$

$q_0, q_1 = 0.05, 0.15$



$= 0.5 \times 1$
$+$
$0.05 \times 2 + 0.15 \times 2$
$= 0.9$

$$e[1,1] = \begin{cases} e[1,0] + e[2,1] + P_1 + w(1,0] + w[2,1] \\ 0.05 + 0.15 + 0.5 + 0.05 + 0.15 \end{cases}$$

$= 1.9$

D.A.A. 235