Unt-1 KCS-503

e : __/__/__

★ Algorithm :- It is a well defined computational processes that takes some values or set of value as input and produce some value or set of value as output.
An algorithm is a set of sequence that transform input into output.

★ Data structure :- Data structure is used to store & compress data & in order to facilitate ~~data~~ access modification no single data structure work for all purpose.

★ Complexity of an algo :- complexity of an algo is defined as:-

(i) Time complexity
(ii) space "

(i) Time complexity : Time required to transform i/p into o/p

(ii) Space complexity :- How much space or memory is required to tranform i/p into o/p.

★ Growth of function :- In order of growth of running time of an algo gives a simple characterisation of the algo from efficiency & also allow us to compare the relative performance & alternative algo.

★ Asymptotic Notation :- The notation that we use to describe the asymptotic running time of an algo is defined in terms of functions whose domains are the set of natural Numbers (N).

Such notations are convinent for describing the worst time, running time of process T(n) : which is individually defined only input size.

Θ - Notation :-

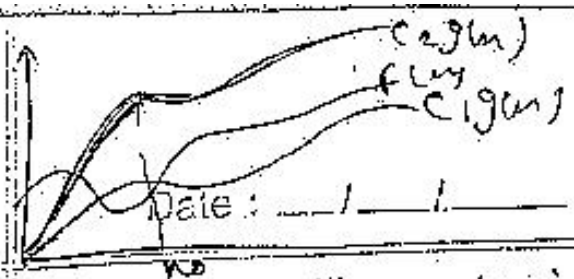
for a given function g(n) we denote it by Θ(g(n)). The set of funs Θ(g(n))

Θ(g(n)) = { f(n) : There exist a constant $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$

The fun f(n) ∈ set Θ(g(n)) if there exist (+)ve constant c_1 & c_2 such that it can be sand wich b/w $c_1 g(n)$ & $c_2 g(n)$ for significantly large n.

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

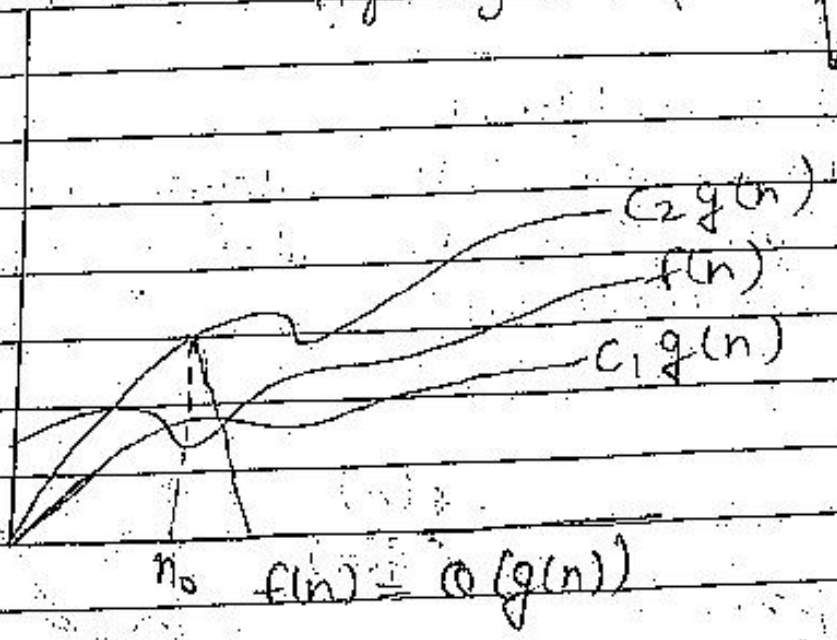
if ∀ (+)ve constants n_0 c_1, c_2 such that to the right of f_0 the value of f(n) always lie b/w $c_1 g(n)$ & $c_2 g(n)$.



Date : __/__/__

Page No.: _____

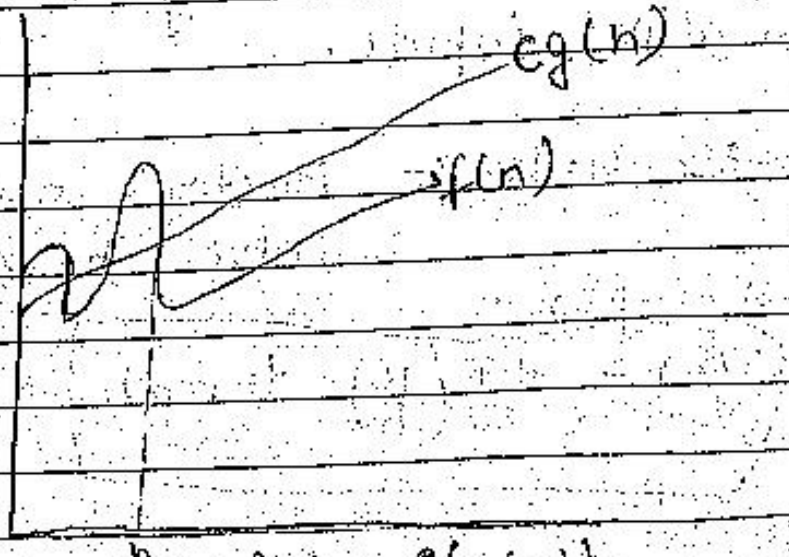
Θ - Notation decide bound.
Tightly bound



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

★ Big-oh Notation:- Gives upperbound for a function within a constant factor we write $f(n) = O(g(n))$ if there exists a positive constants n_0, c,

such that to the right of n_0 the value of $f(n)$ on or below $cg(n)$.

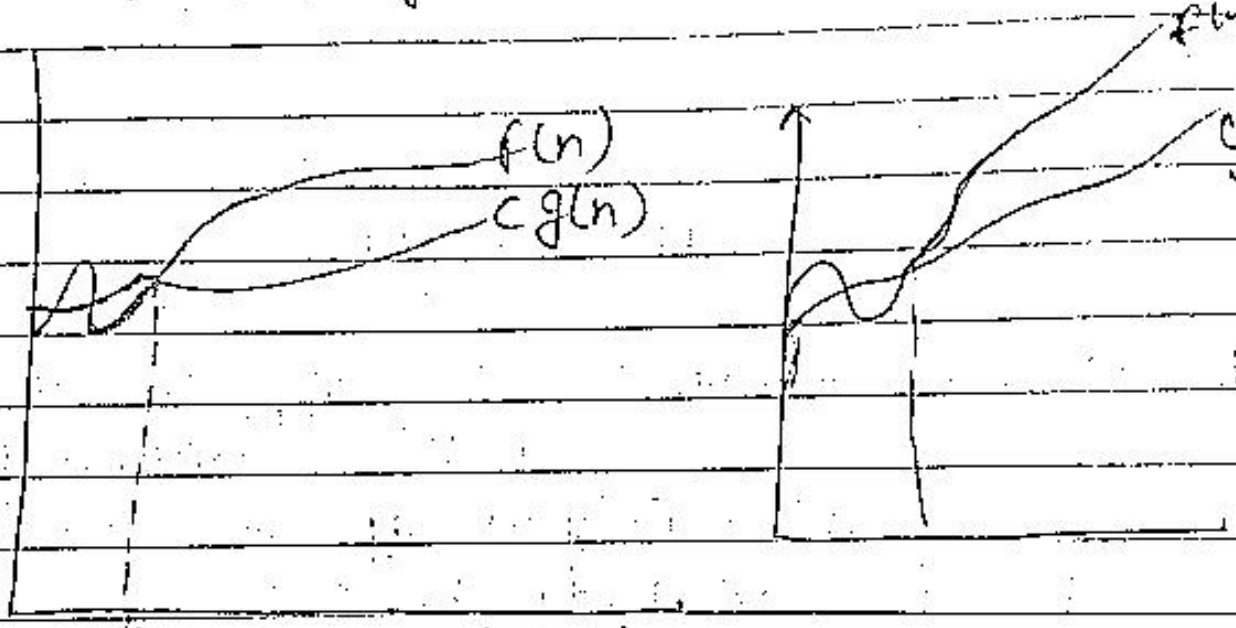


$f(n) = O(g(n))$

* Omega notation :- It gives lower bound of a function, we write

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

if n_0, c are (+)ve integers such that to right of n_0 the value of $f(n)$ lies on or above $g(n)$



* Recurrences :- Recurrences is an equation or enequality that describe its function in terms of its values or smaller inputs.

Master's Method :- Master's method solves the recurrences of the form :

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \qquad a, b > 1$$

Date : __/__/____ Page No.: ____

There are three cases in master's method

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

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

case(ii) (a) if $f(n) = \Theta(n^{\log_b a} \log^k n)$ then

$$\boxed{T(n) = n^{\log_b a} \log^{k+1} n}$$

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

$$\boxed{T(n) = n^{\log_b a} \log^{k+1} n}$$

Case (iii)

If $f(n) = \Omega(n^{\log_b a + \epsilon})$

for some constant $\epsilon > 0$ then

$$\boxed{T(n) = \Theta(f(n))}$$

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

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

Compare, $a=2, b=2, f(n) = n\log n$

$n^{\log_b a} = n^{\log_2 2} = n^1 = n$

$f(n) = n\log n$

case (ii) (b) $f(n) = \Theta(n^{\log_b a} \log^k n)$

$n^{\log_b a}$, $\log^k n$

If $K = 1$

then $f(n) = n \log^1 n$

$= n \log n$

$T(n) = n^{\log_b a} \log^{k+1} n$

$= n \log^{k+1} n$

$\boxed{T(n) = n \log^2 n}$

Ans

Ques $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \lg n$

Solve:-

$\rightarrow T(n) = aT(n/b) + f(n)$

$\lg n = \log_2 n$

$T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \log_2 n$

$m = \lg n$

$m = \log_2 n$

$\boxed{2^m = n}$ put the value of n

$T(2^m) = 2T(\lfloor 2^{m/2} \rfloor) + \log_2(2^m)$

Date : ___/___/___ Page No.: ___

$= 2T(\lfloor 2^{m/2} \rfloor) + m\log_2(2)$

$T(2^m) = 2T(\lfloor 2^{m/2} \rfloor) + m$

Let, $S(m) = T(2^m)$ $\quad S(m/2) = T(2^{m/2})$

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

$a = 2, \; b = 2, \; f(m) = m$

$m^{\log_b a} = m^{\log_2 2} = m^1$

Case 2(i) $\quad T(m) = m^{\log_b a} \log m$

$T(m) = m^{\log_2 2} \log m$

$T(m) = m \log m$

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

$-$ → Case 1
$=$ → Case 2 (i)
$+$ → Case 3

Page No.:

Date : / /

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

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

$f(n) = n^2, \; a = 4, \; b = 3$

$n^{\log_b a} = n^{\log_3 4} = n^{1.26}$

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

Applying case(3)

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

$\boxed{T(n) = \Theta(n^2)}$ Ans

Ques-4 $T(n) = 9T\left(\frac{n}{3}\right) + n^3$

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

$a = 9, \; b = 3, \; f(n) = n^3$

$n^{\log_b a} = n^{\log_3 9} = n^{\log_3 3^2}$

$\log_3 3^1 = 1 \quad = n^{2 \log_3 3} = n^2$

compare to $f(n)$ {add 1}

case apply III rd

Date : / / Page No.:

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

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

Ques-5 $T(n) = 2T(n/2) + n$

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

$\Rightarrow a=2,\ b=2,\ f(n) = n$

$n^{\log_b a} = n^{\log_2 2} = n^1 = n$

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

$\boxed{T(n) = n \log n}$ Ans

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

Compare with below formulae:-

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

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

Find → $n^{\log_b a} = n^{\log_2 2} = n$

Apply case II-(a) →

$$T(n) = n \log_b a \log n$$

$$T(n) = n \log_b a \log n$$

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

SUBSTITUTION METHOD

1→ Guess the form of the solution.
2→ Verify by induction
3→ Solve for constants

Date : __/__/____ Page No.: ____

→ Substitution method & Guess the form

method ↓ ↙ Guess form of soln

1 → $2n^2 + \Theta(n) = \Theta(n^2)$

2 → $T(n) = T(n-1) + n$

soln = $\Theta(n^2)$

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

4 → $T(n) = 2T\left(\left\lceil \frac{n}{2} \right\rceil\right) + n$ $a = b$

then soln = $n \log n$

5 → $T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor + 17\right) + n$ $a = b$

soln = $n \log n$

6 → $4T\left(\frac{n}{2}\right) + n$ if $a = b^2$

then soln = $\Theta(n^2)$

Q Find the upper bound soln of the given recurrance →

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

soln

→ Guess the form of the soln → $O(n^3)$

Now → Steps of solving →

1. Suppose the soln of given recurrance is $O(n^2)$

Date : __/__/____ Page No.:

2. Now we will verify the guess soln. by induction →

$T(n) = O(n^2)$ which is upper bound

So,

$T(K) \leq cK^2$

So we should prove that,

$T(n) \leq cn^2$

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

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

$= cn^2 + n$

if c = constant

$c = 1$

$\boxed{T(n) \leq n^2 + n}$

13 | Aug | 2019

Iteration Method :-

Q. $T(n) = 2T\left(\frac{n}{2}\right) + n$ → $T(n/2)$

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

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

$= 2^2\left[2T\left(\frac{n}{8}\right) + \frac{n}{4}\right] + 2n$ (replace T(n/2) → n/2)

Date : __/__/____ Page No.: ____

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

After k iteration

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

Sub problem size is
1 after $\frac{n}{2^k} = 1$ (maximum value)

$\therefore k = \log_2 n \quad = 2^k = n$

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

$= nc + n \log n \quad \{c = T(1)\}$

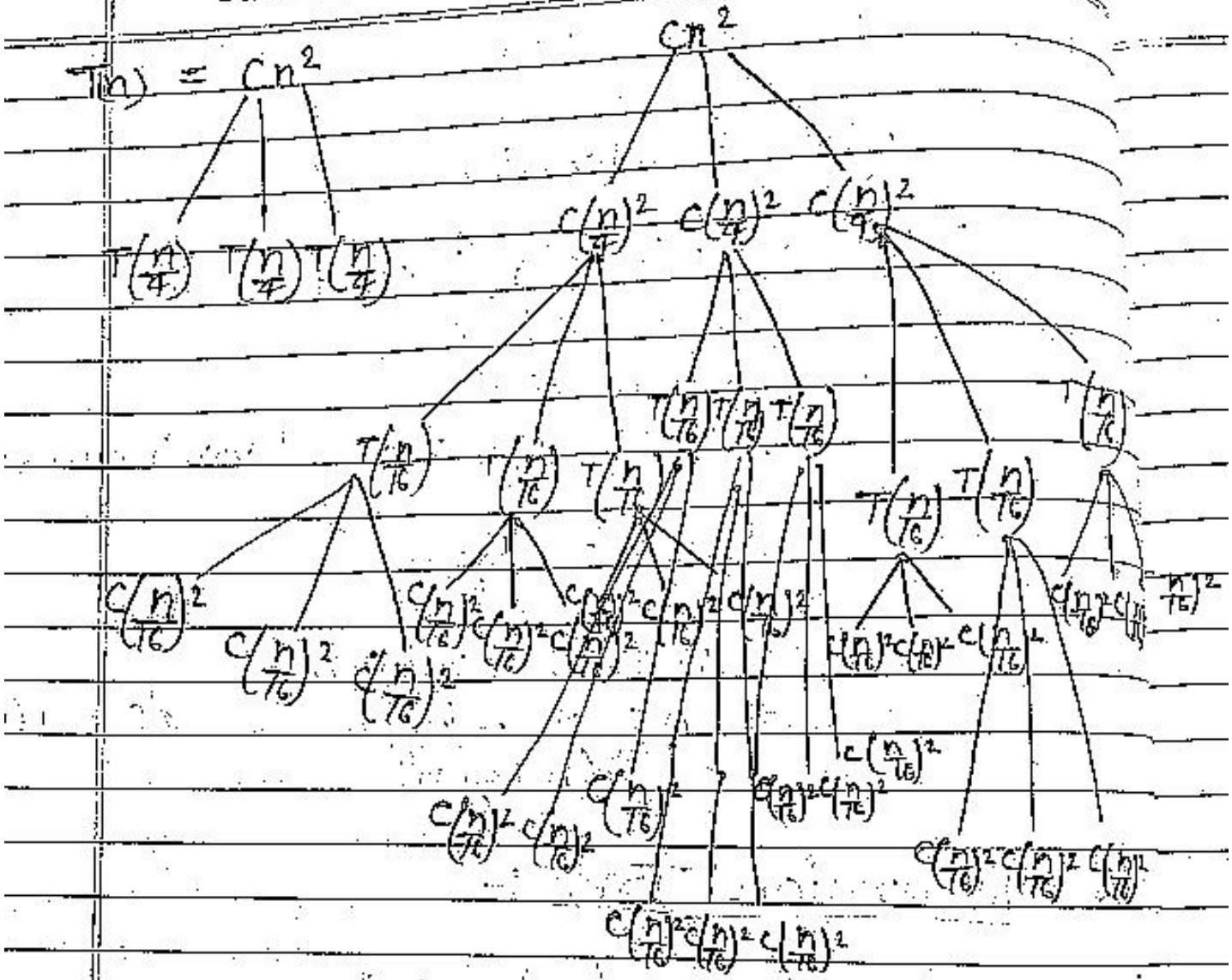
$\therefore = O(n \log n)$

★ Recursion Tree :-

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

Page No.:

Date : __/__/____



Sub problem size at depth

$$i = \frac{n}{4^i}$$

Sub problem size is one

when $\frac{n}{4^i} = 1 \Rightarrow i = \log_4 n$

So no of levels $= 1 + \log_4 n$

no of nodes at depth $i = 3^i$

cost of each node at depth $i = c\left(\frac{n}{4^i}\right)^2$

Date : __/__/____ Page No.: ____

cost of each level at depth i = $3^i c\left(\frac{n}{4^i}\right)^2 = \left(\frac{3}{16}\right)^i cn^2$

$$T(n) = \sum_{i=0}^{\log_4 n} cn^2 \left(\frac{3}{16}\right)^i$$

OR

$$T(n) = \sum_{i=0}^{\log_4 n - 1} cn^2 \left(\frac{3}{16}\right)^i + \text{cost of last level}$$

at last level $\Rightarrow c^{3\log_4 n} \Rightarrow cn^{\log_4 3}$

$$T(n) = cn^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i + cn^{\log_4 3} \Rightarrow O(n^2)$$

Sorting :-

- Insertion
- Shell sort
- Merge Sort
- Quick sort
- Heap sort

Sorting in a linear time

- Counting Sort
- Radix sort
- Bucket Sort

Date : / /

★ INSERTION SORT (A)

```
(1) for j=2 to length [A]
(2)     do key ← A[j]
(3)     Insert A[j] into a sorted sequence
                                A[1 -- --- j-1]
(4)     i ← j-1
(5)     while i>0 and A[i]>key
(6)         do A[i+1] ← A[i]
(7)             i ← i-1
(8)     A[i+1] ← key
```

Page No.:

Date:

Q- A =

1	2	3	4	5	6
5	2	4	6	1	3

Solution

length A = 6

(i) j = 2 to 6

key ← A[2] = 2

i ← j - 1 = 2 - 1

i ← 1

while i > 0 & A[i] > key

1 > 0 & A[1] > 2

1 > 0 & 5 > 2 True

A[i+1] ← A[i]

A[2] ← A[1]

∴ I value of A[1] transfer to value of A[2]

1	2	3	4	5	6
2	5	4	6	1	3

j ← i - 1

i ← 1 - 1

i ← 0

A[i+1] ← key

A[0+1] ← 2

A[1] ← 2

(ii) j = 3 to 6

Key ← A[3] = 4

i ← 2

2 > 0 & A[2] > 4

2 > 0 & 5 > 4 True

A[3] ← A[2] {value Transfer}

A[3] ← 5

1	2	3	4	5	6
2	4	5	6	1	3

Date : / /

Page No.:

$i \leftarrow 2-1$

$i \leftarrow 1$

$A[2] \leftarrow$ Key

$A[2] \leftarrow 4$

(iii) $j = 4$ to 6

Key $\leftarrow A[4] = 6$

$i \leftarrow 3$

$3 > 0$ & $A[3] > 6$

$3 > 0$ & $5 > 6$ false

$A[4] \leftarrow$ key

$A[4] \leftarrow 6$

1	2	3	4	5	6
2	4	5	6	1	3

(iv) $j = 5$ to 6

Key $\leftarrow A[5] = 1$

$i = 4$

$4 > 0$ & $A[4] > 1$

$4 > 0$ & $6 > 1$ True

$A[5] \leftarrow A[4]$ {Transfer value}

1	2	3	4	5	6
2	4	5	1	6	3

$i \leftarrow 3$

$A[4] \leftarrow 1$

(v) $j = 6$ to 6

Key $\leftarrow A[6] = 3$

$i \leftarrow 5$

$5 > 0$ & $A[5] > 3$

$5 > 0$ & $6 > 3$ True

$A[6] \leftarrow A[5]$ {value Transfer}

1	2	3	4	5	6
2	4	5	1	3	6

$i \leftarrow 4$

$A[5] \leftarrow 5$

similary as solving above sorted array

$j = 2$ to 6 continue

$\lfloor \rfloor$ → floor

Page No.:

Date : __/__/____

Best time complexity of insertion sort = $O(n)$
Avg = $O(n^4)$
Worst = $O(n^2)$

★ Shell Sort :-

```
(1)  {
(2)    d ← ⌊(n+1)/2⌋
(3)    for (i ← d to i >= 0; i ← i/2) do
(4)    {
(5)      for (j ← i; j ← n-1; j++) do
(6)      {
(7)        value ← a[j]
(8)        k ← j - i
(9)        while (k >= 0 && value < a[k])
(10)       {
(11)         a[k+i] ← a[k]
(12)         k ← k - i
(13)       }
(14)       a[k+i] ← value
(15)     }}}
```

Or

It is generalization of insertion sort.

(i) Compare each element that are distinct a part rather than adjacent.

where distance is $\lfloor \frac{n}{2} \rfloor$ and n is number of element in an array.

(iii) In different pass distance is reduced to 1.

(iv) When pass is completed at distance 1 we get sorted array.

for ex:-

n = 9 15 is compare with 24 because distance is 4

distance $\left|\frac{9}{2}\right| = \lfloor 4.5 \rfloor = 4$

Pass-1

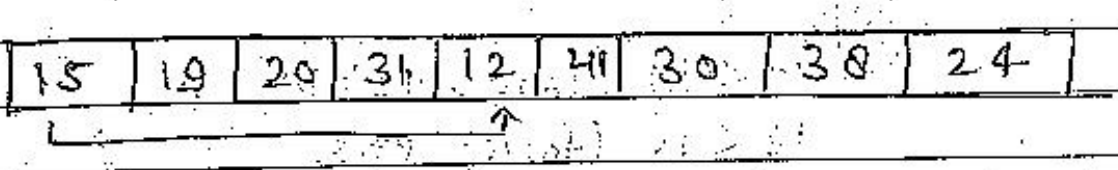
15 < 24 (True) 15 = 15

19 < 41 (True) 19 = 19

20 < 30 (True) 20 = 20

38 < 31 (false) 31 = 38 & 38 = 31 (exchange)

24 < 12 (false) 12 = 24 & 24 = 12 (exchange)



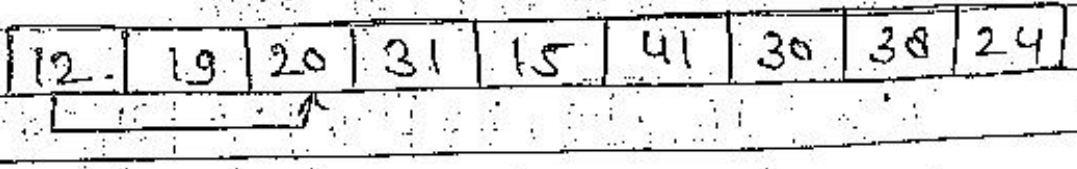
15 < 12 false 12 = 15 & 15 = 12 (exchange)

19 < 41 True 19 = 19

20 < 30 True 20 = 20

31 < 38 True 31 = 31

12 < 24 True 12 = 12



Pass-2 Date : __/__/__ Page No.: ____

Now gap = $\frac{\text{distance}}{2} = \frac{4}{2} = 2$

12 < 20 (True) 12 = 12
19 < 31 (True) 19 = 19
20 < 15 (false) → 20 = 15 & 15 = 20 (exch)
31 < 41 (True) → 31 = 31
15 < 30 (True) → 15 = 15
41 < 38 (false) 38 = 41 & 41 = 38
30 < 24 (false) 24 = 30 & 30 = 24

12	19	15	31	20	30	24	41	30

Pass 3 Now,
gap 2 = $\left\lfloor \frac{\text{gap 1}}{2} \right\rfloor = \left\lfloor \frac{2}{2} \right\rfloor = 1$

gap = 1
12 < 19 (True) 12 = 12
19 < 15 (false) exch
15 < 31 (True)
31 < 20 (false) exch
20 < 38 (True)
38 < 24 (false) exch
24 < 41 (True)
41 < 30 (false) exch

12	15	19	20	24	30	31	38	41

↳ Sorted Array

Date : __/__/____ Page No.: ____

★ Quick SORT :-

(1) If $p < r$
q = Partition (A, p, r)
Quick SORT $(A, P, q-1)$
Quick SORT $(A, q+1, r)$

Partition (A, P, r)

```
1  x ← A[r]
2  i ← p-1
3  for (j ← p to r-1)
4     do if A[j] ≤ x
5        then i ← i+1
6  Exchange A[i] ↔ A[j]
7     exchange A[i+1] ↔ A[r]
8     return i+1
```

Basic condition
conquer
technique

Ques

	1	2	3	4	5	6	7	8 = r
j=0	2	8	7	1	3	5	6	4

$x = A[r] = A[8] = 4$

$i = p-1$

$i = 1-1 = 0$

p → first index
r → last index

for $j = 1$ to 7

★ for $j = 1$: If $A[1] \le 4$ True
$2 \le 4$
$i = i+1 = 1$
exchange $A[1] \leftrightarrow A[1]$

	1	2	3	4	5	6	7	8
	2	8	7	1	3	5	6	4

Date : __/__/____ Page no.

★ for $j = 2$

if $A[2] \leq 4$

$8 \leq 4$ false

2	8	7	1	3	5	6	4

i j

★ for $j = 3$

if $A[3] \leq 4$

$7 \leq 3$ false

1	2	3	4	5	6	7	8
2	8	7	1	3	5	6	4

i j

★ for $j = 4$

if $A[4] \leq 4$

$1 \leq 4$ True

$i = i + 1 = 2$

$A[2] \leftrightarrow A[4]$

$8 \leftrightarrow 1$

2	1	7	8	3	5	6	4
1	2	3	4	5	6	7	8

★ for $j = 5$

if $A[5] \leq x$

$3 \leq 4$ True

$i = 2 + 1$

$i = 3$

Date : __/__/____ Page No.: ____

exchange $A[i] \leftrightarrow A[j]$

$A[3] \leftrightarrow A[5]$

exchange 7 with 5

2	1	3	8	7	5	6	4
1	2	3	4	5	6	7	8

★ for $j = 6$

$A[6] \leq x$

$5 \leq 4$ false

★ for $j = 7$

$A[7] \leq x$

$6 \leq 4$ false

★

$A[i+1] \leftrightarrow A[r]$

$A[3+1] \leftrightarrow A[8]$

$A[4] \leftrightarrow A[8]$

exchange $8 \leftrightarrow 4$

2	1	3	4	7	5	6	8
p		$q-1$		$q+1$			r

A A

return $i+1$ return $i = 4$

$3+1$

Similarly we will get the sorted array.

Date : / /

Quick Sort algo →

If $p < r$

$q =$ Partition (A, p, r)

Quick Sort $(A, p, q-1)$

Quick sort $(A, q+1, r)$

Date : __/__/____ Page No.:

Merge sort algo:

- Based on divide & conquer technique
- worst case time complexity = $O(n \log n)$
- It divides the array into equal halfs and then combines them in a sorted manner.

algo :-

```
M (L, R, A)
{
nL ← length (L)
nR ← length (R)
while (i < nL && j < nR)
{
if (L[i] <= R[j])
{
A[k] ← L[i]
i ← i+1
}
else
{
A[k] ← R[j]
j ← j+1
}
k ← k+1
}
```

Quicksort complexity		Mergesort	Heapsort
worst :-	$O(n^2)$	worst :- $O(n \log_2 n)$	worst :- $O(n \log_2 n)$
Best :-	$O(n \log_2 n)$	Avg :- $O(n \log_2 n)$	Best :- "
Avg :-	$O(n \log_2 n)$	Best :- $O(n \log n)$	Avg :- "

Date : __/__/____ Page No.: ____

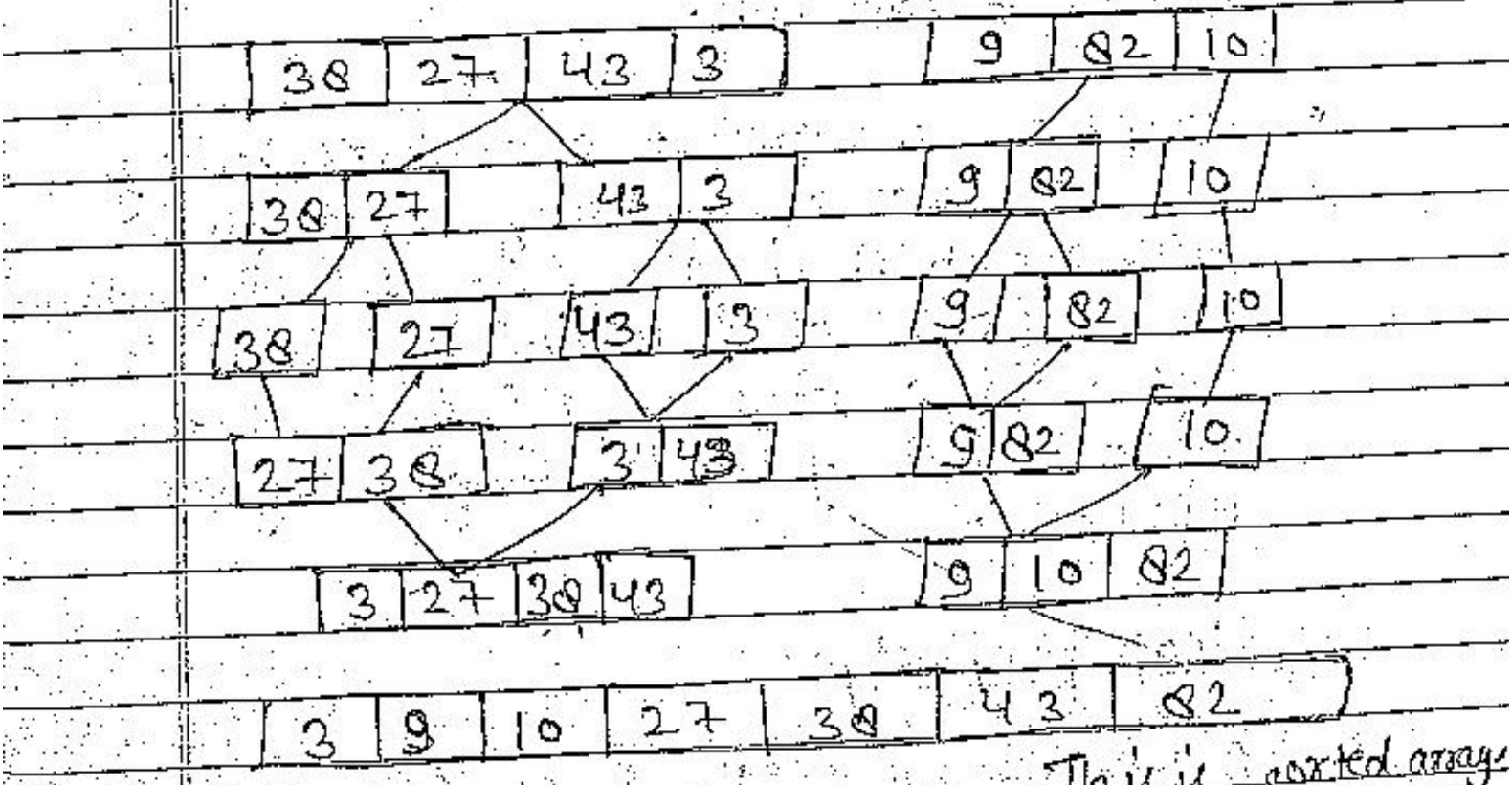
★ Merge Sort :-

Ques-

1	2	3	4	5	6	7
38	27	43	3	9	82	10

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

$\lfloor \frac{n}{2} \rfloor = \lfloor \frac{7}{2} \rfloor = \lfloor 3.5 \rfloor = 3$



This is sorted array

★ every element of left hand side is compare to every element of right hand side.

☆ Heap sort :-

- Heapify
- Build heap

Heap can be seen as a complete binary tree. Heap is usually independent as an array. In it we consider a complete binary tree as a sequential representation of binary tree. A Heap is defined as max heap. The value of each node is less than or equal to the value of root node.
Min heap is defined as value of each node is greater than or equal to the value of root node.

- Heaps :- To represent a complete binary tree as an array, root gives A[1] and A[i] is the node at i^{th} position. Now, parent of i is $\lfloor A[i/2] \rfloor$ exact is left child of node or A[2i] is the right child of node, ~~i.e.~~ i gives A[2i+1].

- Heap property :-

A parent of I i] should always greater than or equal to A[i].

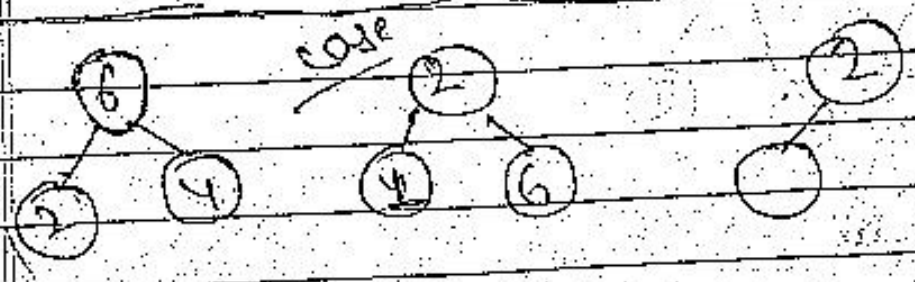
Date : __/__/____ Page No.: ____

- **Heap height :-** Height of a node in the tree is equal to the no. of edges on the longest downward path to a leaf.

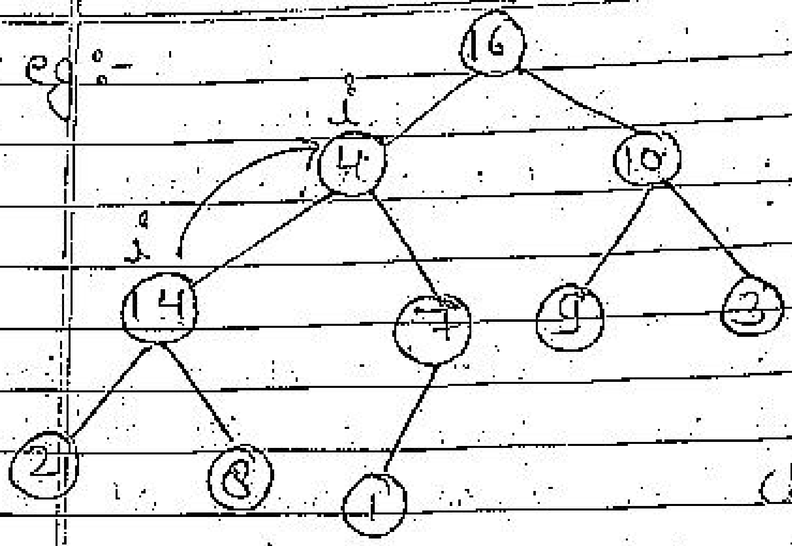
- **Heapify :-**
Heapify operation maintains the heap property.
for eg → Suppose a given i with children l & R and two subtree rooted at l & R and subtree violet the subtree.

★ **Algo for max. Heapify :-**

```
MAX-HEAPIFY (A, i, n)
1. l ← left (i)
2. R ← Right (i)
3. if l ≤ n and A[l] > A[i]
4.     then largest ← l
5.     else largest ← i
6. if R ≤ n and A[R] > A[largest]
7.     then largest ← R
8. if largest ≠ i
9.     then exchange A[i] ↔ A[largest]
10.        max Heapify (A, largest, n)
```

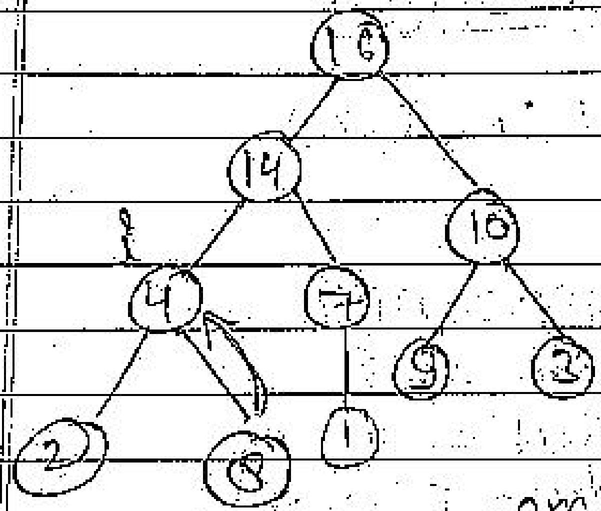


for eg:-

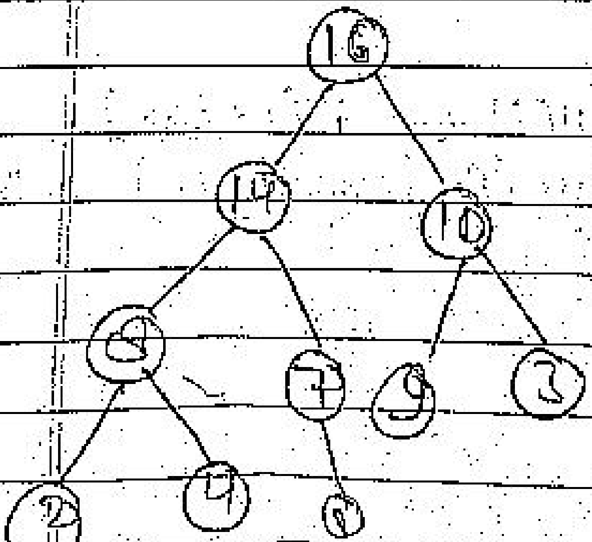


check parent node from left & right node
{means parent node greater hona chahiye ~~are~~ apne left right childs se}

exchange 14 ⟷ 4



exchange 8 ⟷ 4



Date : __/__/____ Page No.: ____

★ Build heap (): -

```
BUILD-MAX-HEAP(A)
{
    A.heap-size = A.length
    for (i = LA.length/2 down to 1)
    MAX-HEAPIFY (A, i);
}
```

24/Aug/2018

Que-1

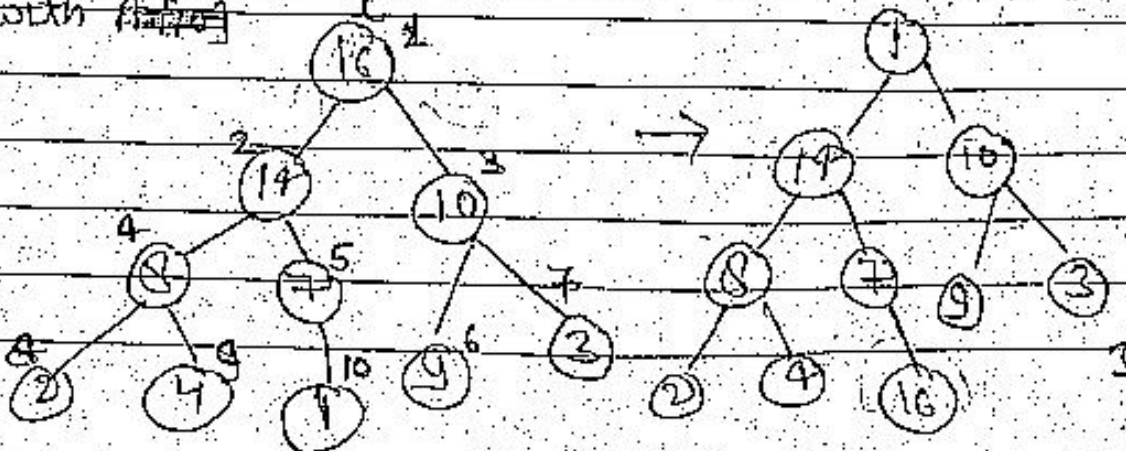
16	4	10	14	7	9	3	2	8	1

```
1. Heap Sort(A)
2. {
3.   BUILD-MAX-HEAP(A);
4.   for (i = A.length down to 2)
5.   {
6.     exchange (A[1] with A[i])
7.     A.heap size = A.heap size - 1;
8.     MAX-Heapify (A, 1);
9.   }
10. }
```

i = 10 to 2

1 exchange A[1] with A[i]

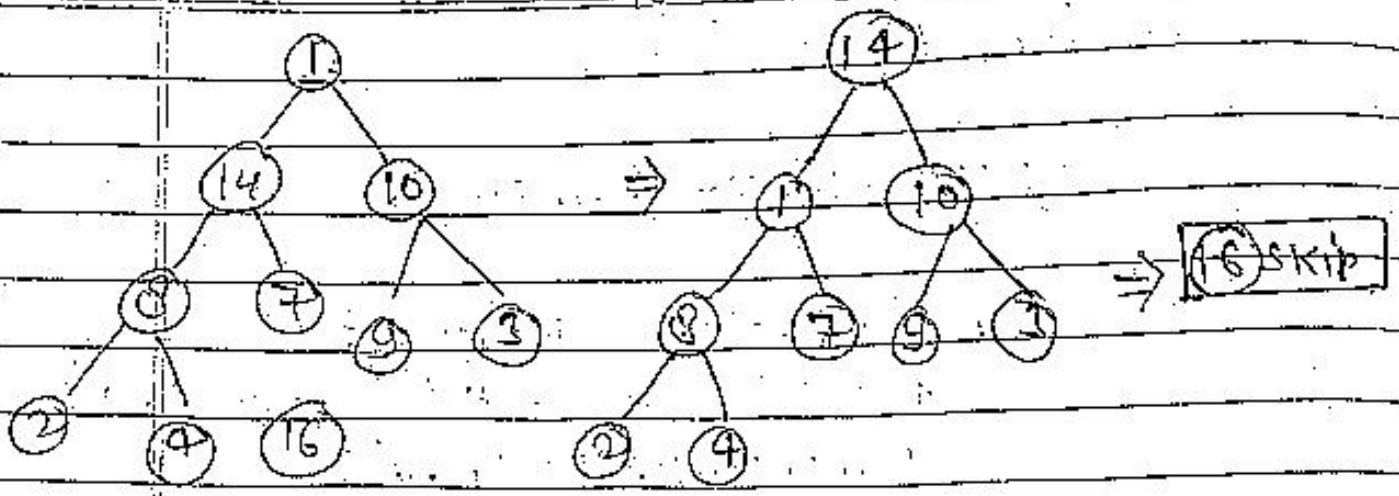
exchange is with

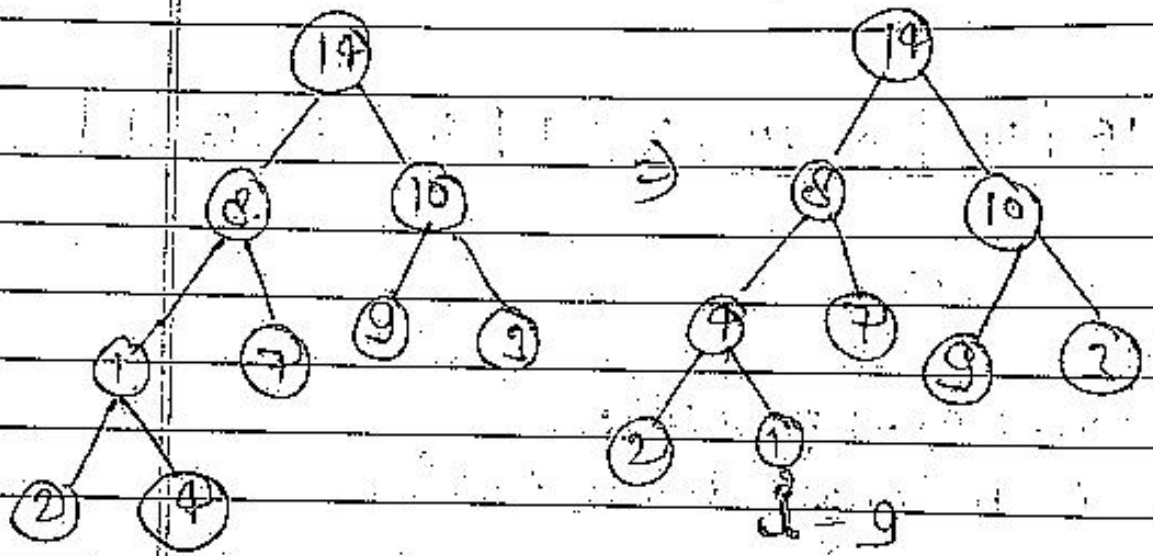


heap size = 10

Date : __/__/____

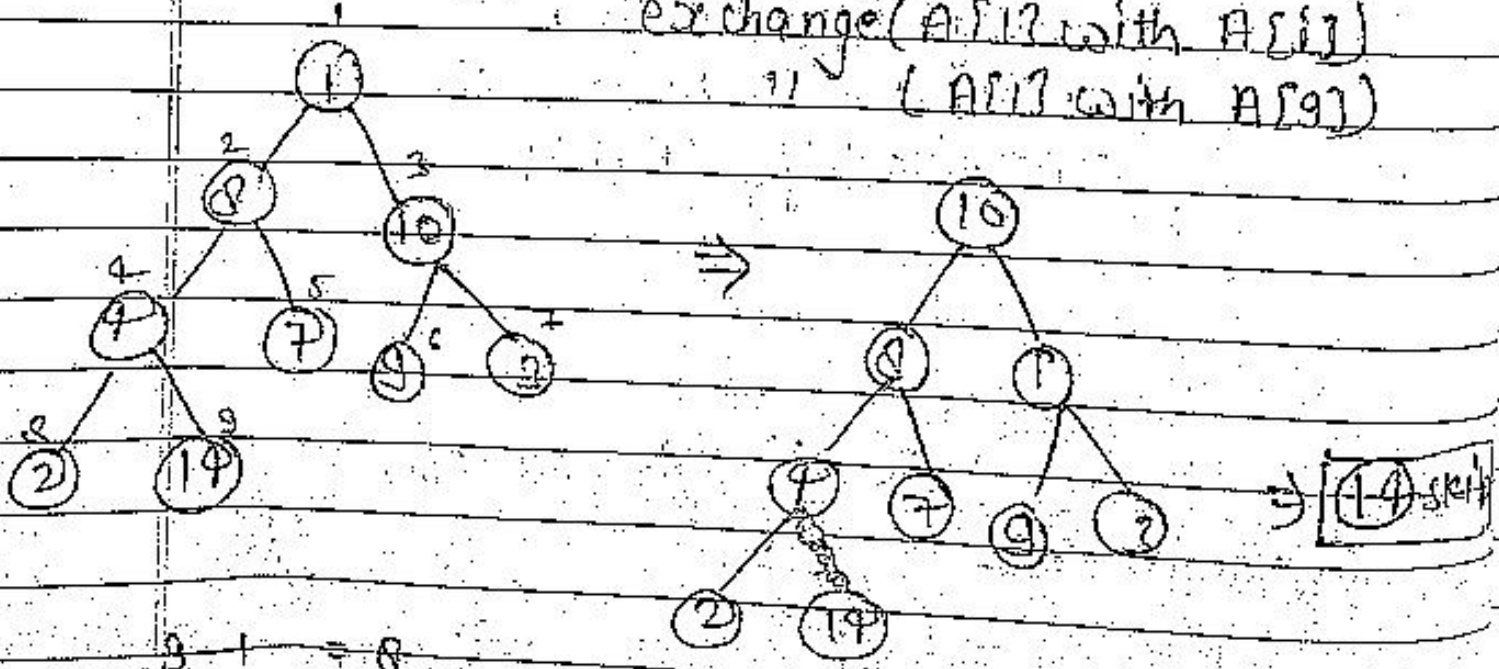
10 − 1 = 9 — 16 → skip



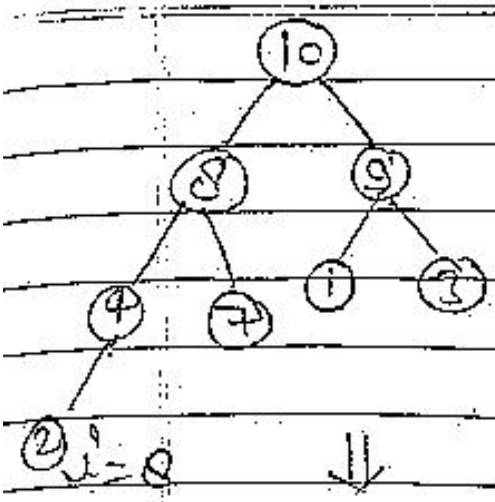


exchange(A[1] with A[i])

" (A[1] with A[9])

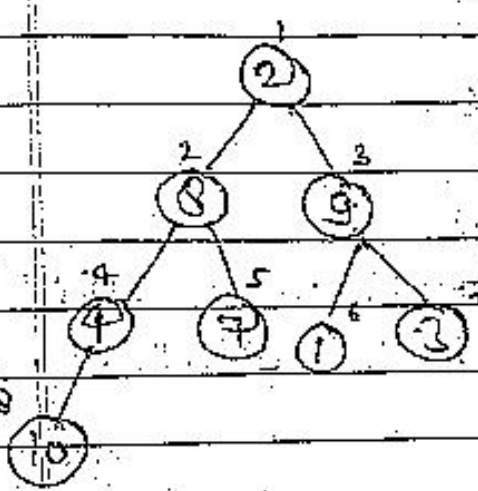


Date : __/__/____ Page No.: ____



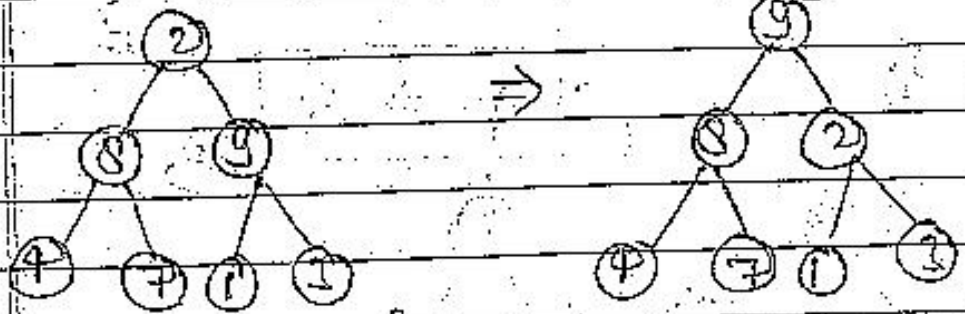
(14) skip

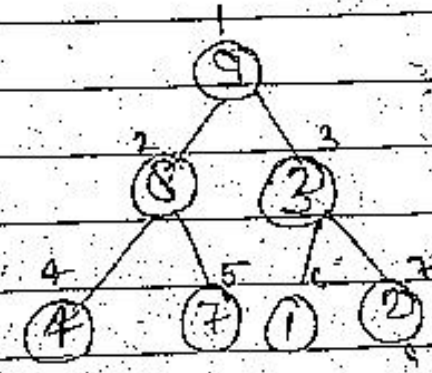
exchange (A[1] with A[i])
" (A[1] with A[9])



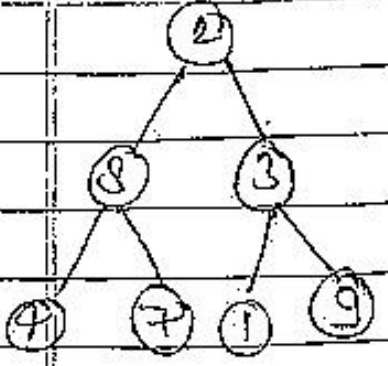
A.heap size = A.heap size - 1;

skip (10)

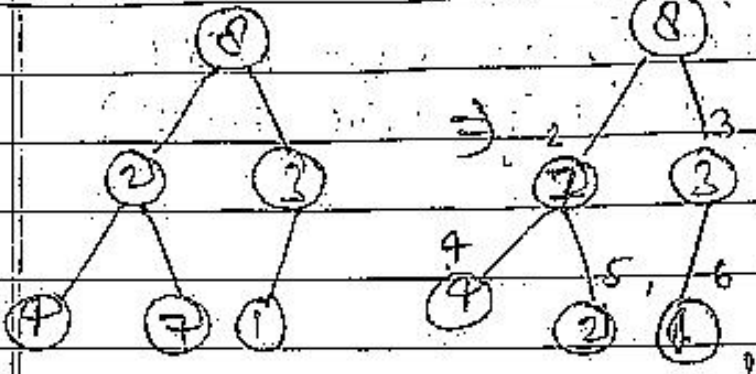




j=7 exchang A[1] ↔ A[i]
A[1] ↔ A[7]

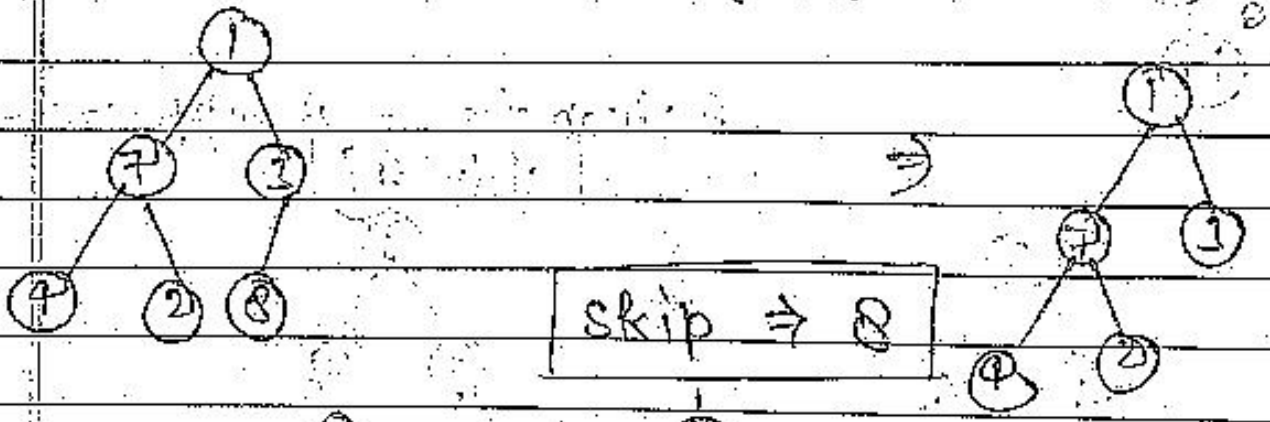


skip ⇒ 9

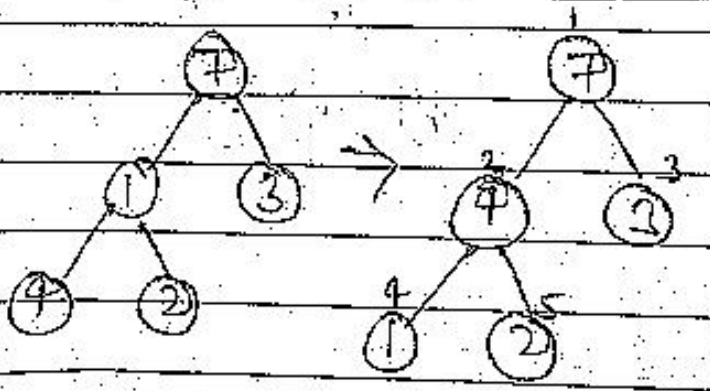


j = 6

exchange A[1] with A[1]
" A[1] with A[6]



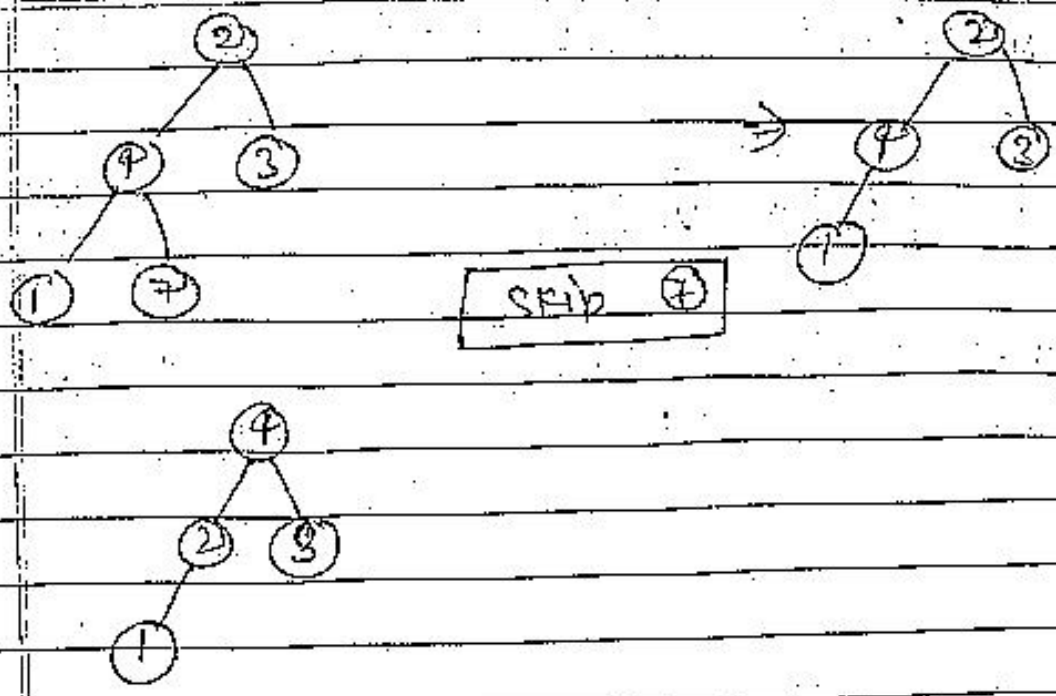
skip ⇒ 8



j = 5

exchange A[1] with A[1]
" A[1] with A[5]

Date : __/__/____ Page No.: ____



similarly procedure will repeat for {4, 3, 2, 1} then we will find sorted array ←

1	2	3	4	7	8	9	10	14	16

★ Sorting in a linear time :-

↳ Counting

↳ Radix

↳ Bucket

COUNTING SORT (A, B, K)

```
(1) for i ← 0 to K
(2)     do C[i] = 0
(3) for j ← 1 to length [A]
(4)     do C[A[j]] ← C[A[j]] + 1
(5) for i ← 1 to K
(6)     do C[i] ← C[i] + C[i-1]
(7) for j ← length [A] down to 1
(8)     do B[C[A[j]]] ← A[j]
(9)        C[A[j]] ← C[A[j]] - 1
```

Page No.:

Date : __/__/____

	1	2	3	4	5	6	7	8
Que- A =	2	5	3	0	2	3	0	3

(a) not given

i =	0	1	2	3	4	5
C[i] =	2	0	2	3	0	1

lowest = 0
Highest = 5

j = 1 to 8
i = 1 to 5

Step-5 for i to 5 = 1 to 5

i = 1

C[1] ← C[1] + C[1-1]

C[1] ← C[1] + C[0] ⇒ 0+2 = 2

C[1] ← 2

2	2	2	3	0	1
0	1	2	3	4	5

i = 2

C[2] ← C[2] + C[1]

C[2] ← 4

2	2	4	3	0	1
0	1	2	3	4	5

i = 3

C[3] ← C[3] + C[2]

C[3] ← 7

2	2	4	7	0	1
0	1	2	3	4	5

$\Rightarrow x$

Date : __/__/____ Page No.: ____

$j = 4$

$C[4] \leftarrow C[4] + C[3]$

$C[4] \leftarrow 7$

2	2	4	7	7	1
0	1	2	3	4	5

$j = 5$

$C[5] \leftarrow C[5] + C[4]$

$C[5] \leftarrow 8$

C =	2	2	4	7	7	8
	0	1	2	3	4	5

Step-7 :-

$j \leftarrow 8$ to 1 :

Step-8 : $B[C[A[j]]] \leftarrow A[j]$

	0	1	2	3	4	5	6	7
B =	0	0	2	2	3	3	3	5

$B[C[A[8]]] \leftarrow A[8]$

$B[C[3]] \leftarrow A[8]$

$B[7] \leftarrow A[8]$

Put the value of $A[8]$

$\therefore B[7] \leftarrow 3$ in $B[7]$

Step-9 $C[A[j]] \leftarrow C[A[j]] - 1$

$C[A[8]] \leftarrow C[A[8]] - 1$

$C[A[8]] \leftarrow C[3] - 1$

$C[A[8]] \leftarrow 7 - 1$

$C[3] \leftarrow 6$ is

Date : __/__/__

j ← 7 to 1

B[C[A[7]]] ← A[7]
B[C[0]] ← A[7]
B[2] ← A[7]
B[2] ← 0

C[A[7]] ← C[A[7]] − 1
C[0] ← C[0] − 1 = 2 − 1
C[0] ← 1

j ← 6 to 1

B[C[A[6]]] ← A[6]
B[C[3]] ← A[6]
B[6] ← 3

C[A[6]] ← C[A[6]] − 1
C[3] ← C[3] − 1 = 6 − 1
C[3] ← 5

j ← 5 to 1

B[C[A[5]]] ← A[5]
B[C[2]] ← A[5]
B[4] ← 0

C[A[5]] ← C[A[5]] − 1
C[2] ← C[2] − 1 = 4 − 1
C[2] ← 3

Date : __/__/____ Page No.: ____

j ← 4 to 1

B[C[A[4]]] ← A[4]
B[C[3]] ← A[4]
B[1] ← 0

C[A[4]] ← C[A[4]] - 1
C[3] ← C[3] - 1 = 1 - 1
C[3] ← 0

j ← 3 to 1	j ← 1 to 1
B[C[A[3]]] ← A[3]	B[C[A[1]]] ← A[1]
B[C[2]] ← A[2]	B[C[2]] ← A[1]
B[2] ← 3	B[3] ← 2
C[A[3]] ← C[A[3]] - 1	C[A[1]] ← C[A[1]] - 1
C[3] ← C[3] - 1	C[2] ← C[2] - 1
C[3] ← 5 - 1	C[2] ← 3 - 1
C[3] ← 4	C[2] ← 2

j ← 2 to 1

B[C[A[2]]] ← A[2]
B[C[5]] ← A[5]
B[4] ← 5

C[A[5]] ← C[A[5]] - 1
C[5] ← C[5] - 1
C[5] ← 9 - 1
C[5] ← 7

Now final the value of C & B array :

	0	1	2	3	4	5
C =	0	2	2	4	7	7

	1	2	3	4	5	6	7	8
B =	0	0	2	2	3	3	3	5

Ans

90

Date : __/__/____

★ **Radix sort**:- It is sort to algorithm used to sort numbers. The sort the no from LSB to MSB digit.

RADIX SOR (A, d)

(1) for $i \leftarrow 1$ to d

(2) do use any stable sort to sort array A on digit.

d → digit

MSD		LSD	for ① $i = 1$ to ③	
Ques 3	2	9	7 2 0	7 2 0
4	5	7	3 5 5	3 2 9
6	5	7	4 3 6	4 3 6
8	3	9	4 5 7	8 3 9
4	3	6	6 5 7	4 5 7
7	2	0	3 2 9	6 5 7
3	5	5	8 3 9	3 5 5

3 2 9
3 5 5
4 3 6
4 5 7
6 5 7
7 2 0
8 3 9

worst :- n log n
Best :- n
Avg :- n

91

Date : 28 / Aug / 2018

→ Bucket sort :-

(1) n ← length [A]
(2) for i ← 1 to n
(3) do insert A[i] into list
B [1.n A[i]]
(4) for i ← 0 to n-1
(5) do sort B[i] with insertion sort
(6) Concatenate the list B[0], B[1], ---- B[n-1] together in a order.

Ques-

	A		B
1	.78	0	
2	.17	1	→ .12 → .17
3	.39	2	→ .21 → .23 → .26
4	.26	→ 3	→ .39
5	.72	4	
6	.94	5	
7	.21	6	→ .68
8	.12	7	→ .72 → .79
9	.23	8	
10	.68	9	→ .94

Now
step-6 "Concatenate the list B,

.12	.17	.21	.23	.26	.39	.68	.72	.78	.94

Page No.:

Date : __/__/____

★ Question :-

(1) Explain algorithm & complexity of algorithm.
(2) What do you mean by growth of the function.
(3) Solve the recurrence relation using Master's Method.
(4) solve the recurrence relation using substitution method
(5) Explain shell sort with example.
(6) Explain Quick sort with example & what is the complexity of quick sort in best, average & worst sort.
(7) Explain Counting sort and solve the array using counting sort.
(8) What is the property of heap sort and solve the given array heap sort.
(9) What do you mean by sorting in a linear time and what are the different algo with it.