Date. 22-02-2023
Page No. ______

Algorithm :

Algorithm is a combination of finite seque-
-nce to solve any problem.

It is combin.

Properties of Algorithm :

1.) It should produce atleast one o/p.
2.) It should take either 0 or more.
3.) It should produce o/p after finite step.
4.) It is Independent to the programming language.
5) Algorithm should be deterministic.
↳ on ambiguous.

Steps required to construct an algorithm:

- Problem defination
- Design algorithm
  - → Divide & conquer
  - → Backtracking
  - → Dynamic Programing
  - → Greedy techniques.
  - → Branch & Bond.
-
- Flowchart

Date: 26-02-23
Page No.

4) main()

Time complexity – $O(n)$

```
main()
{
   i = 1;
   while (i<n)
   {  x = y+z;
      i++;
   }
```

5.) main()

Time Complexity – $O(\log_2 n)$

```
main()
{  i = 1;
   while (i<n)
   {  x = y+z;
      i = 2i;
   }
}
```

$2^0 \to 1$

$2^1 \to 1$

$2^2 \to 1$

$2^3 \to 1$

$\vdots$

$2^k \to 1$

$2^k = n$

$\log_2 2^k = \log_2 n$

$k = \log_2 n$

6) main()

```
{  while (n>1)
   {  n = √n
   }
}
```

c=a+b

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

$n = \frac{n/2}{2}$

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

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

$2k = n$

$\log_2 2k = n$

$k = 2n$

$\log_2 k = \log_2 2n$

$n = \log_2 k$

$n = (k)^{1/2}$

$\log_2 n = \frac{1}{2} \log_2 (k)$

$2 \log_2 n =$

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

$\log_2 n^{1/2} = \log_2 1$

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

$\log_2 n^k = \log_2 1$

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

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

$\log_2 n = 2^k$

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

$\boxed{k = \log \log_2 n}$

$(n)^{1/2} = \sqrt{n}$

$(\sqrt{n})^{1/4}$

$(n)^{1/8}$

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

$(n)^{1/2^k} = 1$

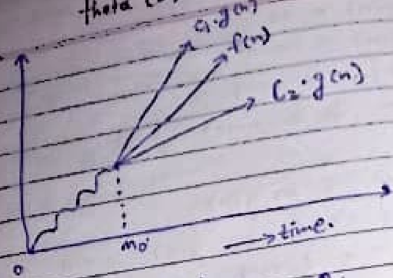
$\frac{1}{2^k} \log n = 1$

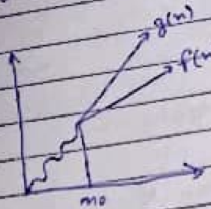
$\log n = 2^k$

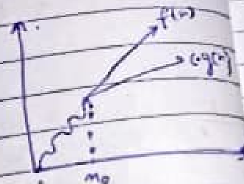
$(\log_2 \log_2 n = k)$

Date.
Page No.

theta (θ)







omega (Ω)

Big (O)

Ex. 1 $f(n) = n$

$g(n) = n + 10$

$f(n) \leq c_1 \cdot g(n)$

$n \leq c_1 (n+10)$, $c = 1$, $n_0 = 1$

$f(n) \geq c_2 \cdot g(n)$

$n \geq c_2 \cdot (n+10)$, $c = 1/2$, $n_0 = 10$

True

Date 8-06-2023
Page No.

Ques 1: $f(n) = n$

$g(n) = n^2$

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

$f(n) \leq c_1 g(n)$

$f(n) \geq c_2 g(n)$

$n \leq c_1 n^2$ ✓

$n \geq c_2 n^2$ ✗

$\therefore f(n) \neq \theta g(n)$

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

$g(n) = n^2$

$f(n) = o\, g(n)$?

$f(n) \leq c \cdot g(n)$ → Tidest upper bound

$f(n) \leq c\, n^2$, True. (TUB)

$n^2 \leq c\, n^2$

$\left.\begin{matrix} n^3 \\ n^4 \\ \vdots \\ n^n \end{matrix}\right\}$ → upper bound.

$n^2 \geq c n^2$ → Tidest lower bound

→ increasing TC

$o(1) < \log\log n < \sqrt{\log n} < \log_2 n < \sqrt{n} < n < n\log_2 n$, $< n\sqrt{n} < n^2 < n^3$ ..... polynomial $< 2^n$ .....

exponential

{ x = y + z ;

}

}

}

}

i	j	k
1	$(1)^2$	$\log_2 n$
n	$(n)^2$	$\log_2 n$

$n^3 \log_2 n$.

Que: main()

{ for (i = 1 ; i ≤ n ; i = 5i) $\log n$ (T = log log n)

{

for (j = 1 ; j ≤ n^2 ; j = 2j)

{

for (k = n ; k ≥ 1 ; k = $\sqrt{k}$)

{

x = y + z ;

} } } }

i	j	k
1	1	$\log \log_2 n$
5	2	$\log \log_2 n$

$n = 5^k$ $\quad 5^k \quad 2^k = \log_2 n \quad \log \log_2 n$

$k = \log_2 n$

$\log_5 n \cdot \log_2 n \cdot \log \log_2 n$

Date 1 March 2023

Page No. ______

Asymptotic Notations

(1.) Big (O) Notation → worst.

2.) Omega Ω → Best

3.) Theta θ → Avg.

>> Big (O) Notation :

f(n) = O g(n)

$\boxed{f(n) \le c \cdot g(n)}$, c = is any constant.

c > 0

$n_0 > 0$ such that $\forall n$, $\{n \ge n_0\}$

Ex: 1. f(n) = n + 10

g(n) = n

n + 10 = O(n)

n + 10 ≤ cn

n = 10 , c = 2

$\boxed{20 \le 20}$

2. f(n) = n , g(n) = n

n = O(n)

n ≤ c.n , c = 1

{n ≤ n}

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

$2^n > n^{\log n}$

$n < \log_n^{\log n}$

(ii) $2^n < n! < n^n$

$n > \log n^2$

$< \log_n^{\log n}$

$< \log n$

- $n > \log n$ taking log both side.

$\log_2 n \geq \log n$

let $n = 2^{100}$

$2^{100} > \log 2^{100}$

$2^{100} > 100$

- $\log_2^n \leq \log n^n$

$\{n < n \log_2 n\}$

(iii) $1000 n \log n = O\left(\frac{n \log n}{1000}\right)$

$1000\, n \log n < c \frac{n \log n}{1000}$ ~~false~~ T

(iv) $\sqrt{\log n} = O(\log \log n)$

$\sqrt{\log n} < c(\log \log n)$ → F

(v) If $(0 < x < y)$

then $n^x = O(n^y)$

$n^x = c\, n^y$ $x = 2$

$n^2 < c\, n^4$ $y = 4$

↳ T

(vi) $2^n \neq O(n^k)$ where k is constant.

$2^n < c n^k$

↳ T

(vii) $n^2 2^{3\log_2 n} = \Theta n^5$ → $n^{3\log_2 2}$

$f(n) \leq c_1 \cdot g(n)$ $(n \geq n_0)$

$f(n) \geq c_2 \cdot g(n)$ $(n \geq 0)$

$n^2 2^{3\log_2 n} \leq c_1 n^5$

$2\log_2 n + 3\log_2 n \leq c_1 5\log_2 n.$

$5\log_2 n \leq c_1 5\log_2 n$ → T

$n^2 2^{3\log_2 n} \geq c_2 n^5$ → T

$5\log_2 n \geq c_1 5\log_2 n$

↳ T

(viii) $4^{\log_2 n} = \Theta n^3$ → F

(ix) $\frac{4^n}{2^n} = o(2^n)$ → T

$2^{2n-n} = o(2^n)$ O✓, Θ✓, Ω✓

$(2^n) = o(2^n)$

(x) if $f(n) = og(n)$ then $2^{f(n)} = o(2^{g(n)})$ → F

$f(n) = 2n$, $g(n) = n$

$2^{2n} \le c.\frac{4^n}{4^{2n}}$

$2^{2n} \le c\,2^n$, $n = 4$

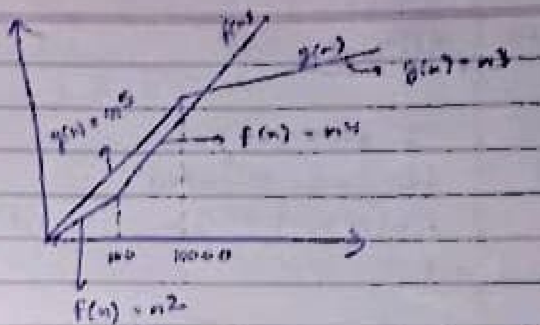
$2^8 \le 1.4$

O×, ω✓, Ω×, O×, θ×

xi) $f(n) = \begin{cases} n^2 & 0 < n < 100 \\ n^4 & n \geq 100 \end{cases}$

$g(n) = \begin{cases} n^5 & 0 < n < 10000 \\ n^3 & n \geq 10000 \end{cases}$

find out relation b/w f(n) & g(n)



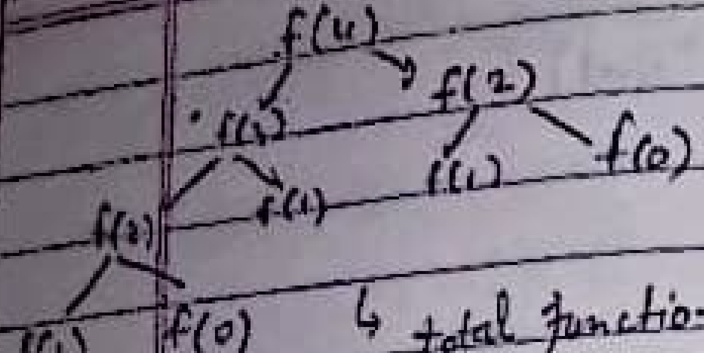
Properties of Asymptotic Notation:

(1) Reflexive: $f(n) = Of(n)$
$f(n) = \Omega f(n)$
$f(n) = \Theta f(n)$
$f(n) \neq of(n)$
$f(n) \neq \omega f(n)$

(2) Symmetric: if $f(n) = og(n)$
then $g(n) \neq of(n)$
only theta will pass

(3) Transitive: $f(n) = og(n)$
$g(n) = oh(n)$
then $f(n) = oh(n)$

(4) Multiplicative: if $f(n) = og(n)$
then $f(n) \cdot f(n) = o(h(n) \cdot g(n))$.



↳ total function call = 2^4

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

Ques3: Write a recurrence relation and recursive program to find gcd of two +ve no

g(2, 4)

g

(2, 2) → out

gcd = 2, 4

2

gcd (100, 200) =

gcd (23, 25) =

gcd (0, 0) = ∞

gcd (0, 10) = 0

```
gcd(int m, int n)
{
```

Date ________
Page No. ________

Divide and Conquer.

Recursion: function calling itself is called recursion.

```
if n=0 || n=1
    return n
else
    return n*f(n-1)
```
→ factorial

- solving big problems using smaller problem.
- In execute recursive program we use stack data structure.
- Every recursive program must have termination cond^{n} else it will go ∞.

- In recursive programs from one function call to another parameter value will only change but not the no. of parameters.

Ques1: write recursive program and recursive sol^{n} to find product of two (+)ve numbers a & b.

```
if n=0
    return 0
else n=1
```
X

```
else
    return n*f(n-1)
```

mul (2, 5) (a = 2, b = 5)

```
if a=0 || b=0
    return 0
else
    return a+f(a, b-1)
```
⇒ recursive program

⇒ O(B)

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

Ques2: write a recursive program and recursive func^{n} to find nth FIBONACCI no.

0 1 1 2 3 5 . . . . .

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

```
fib (a, b)
{
    if (n==0 || n==1)
        return n
    else
        return fib(n-1) + fib(n-2).
```
⇒ O(n)

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

$\Theta(n^{q}-1)$ $\quad n^3$

$=$.

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

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

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

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

Master Class :-

Master Theorem :-

Case 1: $f(n) = O(n^{\log_b a - \epsilon})$

where ϵ is a constant.

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

$n^{\log_2 (2)^3}$

$n^{\log_2 2^3}$

$\Rightarrow n^3$

$n^2 < n^3$

Case 2: $f(n) = \Omega(n^{\log_b a + \epsilon})$

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

Case 3: $f(n) = \Theta(n^{\log_b a})$

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

$T(n) = \overset{\text{\# of subproblems}}{a}T\left(\frac{n}{b}\right) + \overset{\text{time of conquer \& combine}}{f(n)}$

$\hookrightarrow$ size of problem

where a, b are two positive constants.

$a \geq 1$

$b > 1$

f(n) is a positive function.

$f(n) = 8T(n/2) + n^2 \Rightarrow$ it lies under case

$T(n) = 2T(n/2) + n^2 \Rightarrow$ it lies under case

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

$n^{\log_2 1} = 1^{\log_2 n} = 1$ (case

$(c = 1) \Rightarrow$ same multiply by $\log n$

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

Que: $T(n) = \underset{\downarrow \times}{0.5}\, T(n/2) + n^2$

$T(n) = 7T(n/2) - n\log n$

$\hookrightarrow f(n) = O$

$\times$

Date
Page No.

Ques:) $T(n) = 2T(n/2) + n\log n$

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

Because difference isn't polynomial.

(By ~~di~~

Extended

If log is comming

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

$f(n) = \Theta(n^{\log_b a} \cdot \log n^k)$ where $K \geq 0$

- If recurrence relation contain root operator.

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

(1) assume $n = 2^k$

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

S

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

$\Theta(\log k)$

$n = 2^k$

$\log n = k \log 2$

$\{k = \log_2 n\}$

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

$= 2T(2^{k/2}) + \log_2 k$

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

$= k\log k$

$= \log n \log\log n$

24 March 2023

Linear Search : Array may be sorted may not be.

20	62	70	19	5	10	1

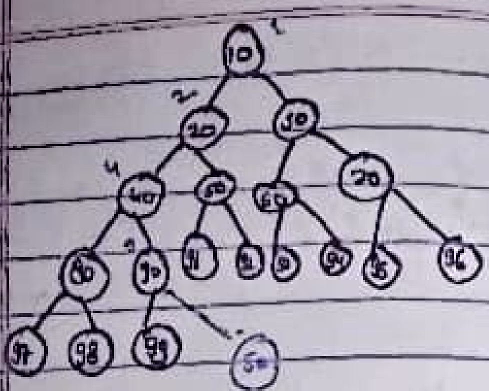
Best case TC $= \Omega(1)$

Worst case TC $= O(n)$

Avg Cas TC $= \Theta(n)$

Binary Search :

	Graph	Tree (DPCA)
1.	$G(V, E)$	$T(V, E)$
2.	No root element present in the graph.	Root element is present.
3.	Graph may be connected or disconnected.	connected.
4.	Graph may contains cycle	Tree can not cycle.



array :- { 10, 20, 30, 40, 50, 60, 70, 80, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99 }

insert = 5

insert last → to maintain property of complete Binary tree.

compare 5 with ⇒ (90, 40, 20, 10) root.

parent = $\left\lfloor \frac{n}{2} \right\rfloor_{floor}$ = $\left\lfloor \frac{19}{2} \right\rfloor$ = 9

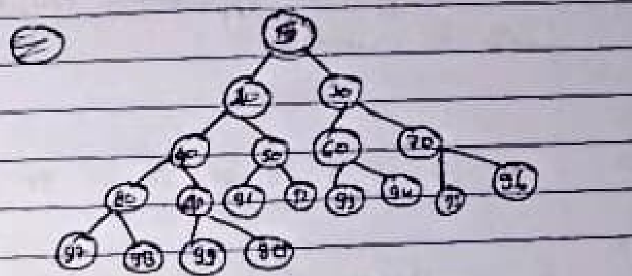
$\left\lfloor \frac{9}{2} \right\rfloor$ = 4

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

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

Best case = $O(1)$
worst case = $O(\log_2 n)$ = Average case.

min deletion in heap :-

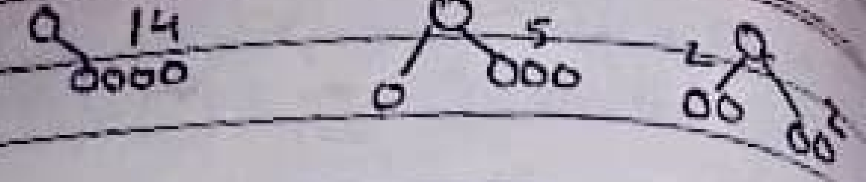


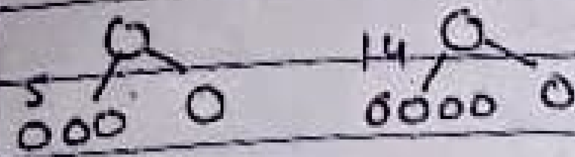
· deletion always takes place from top.

deletion of 1 element in min heap or max heap = $O(\log n)$

Best case = $O(1)$ ⇒ $O(n)$
worst = $O(\log_2 n)$ ⇒ $n \log n$

n=5

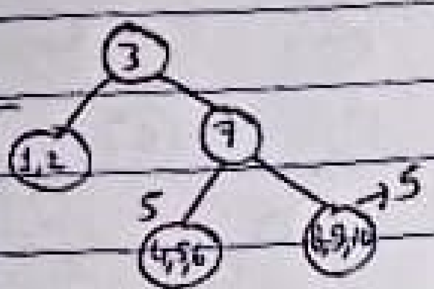




= 42 BST

n=6

- Root is 3 and right is 7



$= 2 \times 5 \times 5$

$= 50$

$$\text{Total no of BST} = \frac{{}^{2n}C_n}{n+1}$$

$$\#\text{ of BST} = \sum_{i=1}^{n} \left(BST(i-1) \times BST(n-i) \right)$$

$$n=3 \quad = \frac{{}^{6}C_3}{} = \frac{6!}{3!\,3!} = \frac{6 \times 5 \times 4 \times 3!}{3! \times 3!}$$

$$= \frac{20}{3+1} = \underline{5}$$

Date.
Page No.

Quick Sort (a, i, j)

```
{
  if ( i == j )
     return (a[i]);
  else
  {
    r = partition (a, i, j);
    QuickSort (a, i, r-1);
    QuickSort (a, r+1, j);
    return (a);
  }
}
```

Recurrence Relation :

$$\begin{cases} a[i] & ; \ n = 1 \\ n + T(r-i) + T(j-r) & \text{otherwise} \end{cases}$$

100 90 80 70 60

60 90 80 70 100

$\Rightarrow n + T(0) + T(n-1)$

$= T(n-1) + n$

$= O(n^2) \Rightarrow$ worst case

- If the array is sorted quick sort not preferable.

Ques 1: In quick sort the sorting of n numbers the $(n/4)^{th}$ smallest element is selected as pivot element using order of n time then what is + will be the time complexity of quick sort algorithm.

Ans: $n + n + T(n/4) + T(3n/4)$

$= n \log_{4/3} n$

Ques 2: In quick sort the sorting of n numbers the $(n/5)^{th}$ element is selected as pivot using order of $O(n^2)$ TC, what will be the worst case time complexity of quick sort.

Pivot Partition

$n^2 + n + T(0) + T(n-1) \Rightarrow$ Pivot smaller

OR

$n^2 + n + T(n-1) + T(0) \Rightarrow$ largest

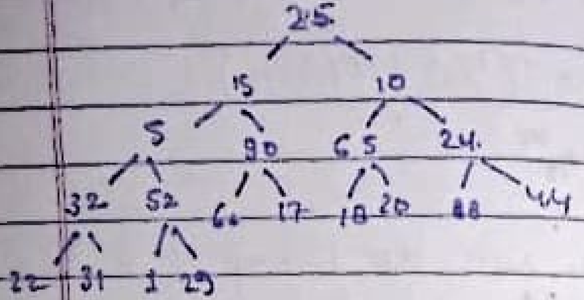
$n^2 + n + T(n-1)$

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

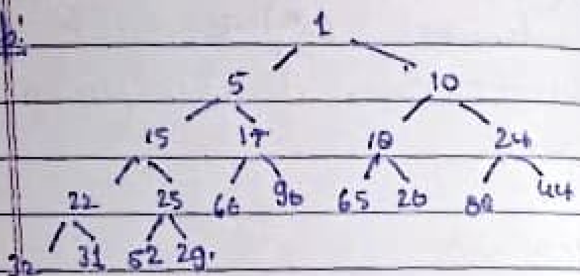
$O(n^3)$

Ques: Create a min heap.

25, 15, 10, 5, 90, 65, 24, 32, 52, 66, 17, 18, 20, 88, 44, 22, 31, 1, 29.



min heap:



↳ O(n)

Build heap (a, n)
{
for (i = [n/2] to 1)
minheapifydown (a, i) ;
}

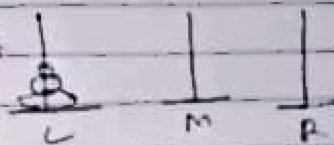
- Construct min heap using Build heap function → O(n)

Date
Page No.

(i) delete one by one and add at end.
↳ TC = O(nlogn)

Tower of Hanoi : $2^n - 1$ (n = no. of disk)

for → 3



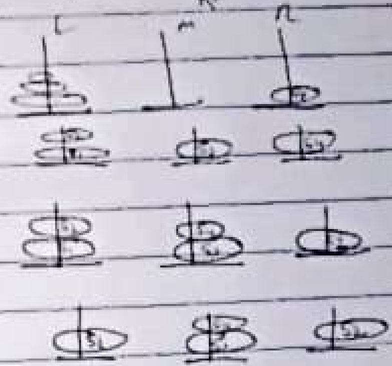
L-R → L-M → R-M
→ L-R → M-L → M-R
→ L-R

for-4

$S_1 > S_2 > S_3$

Step → 15.

L-R
L-M
R-M
L-R



step 3 L → M, 1 L → R, 3 M → R

20 → disk.

Quick Sort :-

1. It is based on divide and conquer technique.
2. It is Inplace.
3. It is not stable algorithm.
4. In all practical applications quick sort will be used.

50 20 15 5 10 45 70 80 90

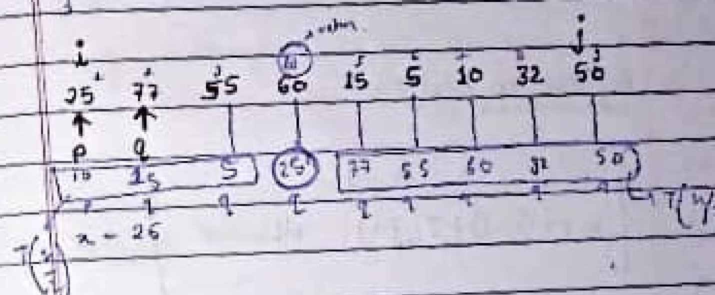
- If we change pivot place everytime it is called as randomized quick sort.
- If we choose the pivot place as same everytime it simply called as quick sort.
- It doesn't include merge step.

Partition Algorithm :

```
Partition (a, i, j)
{
    x = a[i] ;
    p = i ;
```



```
for (q = i+1 ; q ≤ j ; q++)
{
    if (a[q] ≤ x)
    {
        p = p+1 ;
        swap (a[p], a[q]) ;
    }
}
swap (a[i], a[p]) ;
return (p) ;
}
```



Time complexity = $O(n)$

Recurrence Relation: $O(1) + O(n) + T(n/2) + T(n/2)$

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

$= n \log n$

Quick sort algorithm :

Date
Page No.

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

Merge Sort:

Merging two sorted sub array.

Merge algorithm:-

A —
q - mid of array
r - last index
p - starting index
A - array.

```
merge (A, P, q, r)
{
  n1 = q-P+1 ;
  n2 = r-q ;
Let L[1....n1+1] and R[1....n2+1] be
  two arrays.
for (i = 1 to n1)
  L[i] = A[P+i-1]
for ( J=1 to n2)
   R[J] = A[q+J]
i=1 , J=1
for (k=P to r)
if(L[i] ≤ R[J])
  A[k] = L[i]
  i= i+1
else  A[k] = R[J]
      J = J+1
```

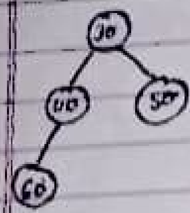
Date
Page No.

```
mergesort (A,P,r)
{
  if (P<r)
    q = P+r/2
  mergesort (A,P,q)
  mergesort (A,q+1,r)
  merge (A,P,q,r)
```

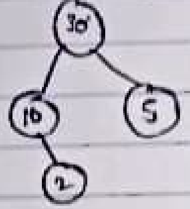
TC = nlogn.

Heap Sort :

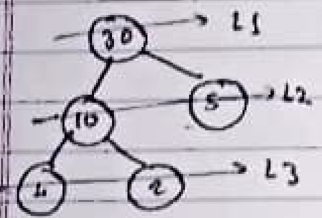
- Complete Binary tree

- 

→ min Heap.

- 

⇒ Not complete BT.

- 

⇒ max heap.

store heap in the arrays :-

30	10	5	1	2
1	2	3	4	5

Left child of i^{th} child = $2i$

eg. $2(2) = 4$ index = 1 (Left child)

Right child = $(2i+1)$

total no of elements = n

then total leaf nodes = $\left[\frac{n}{2}\right]_{\text{ceil value}}$ $\cdot [\frac{5}{2}] = 3$

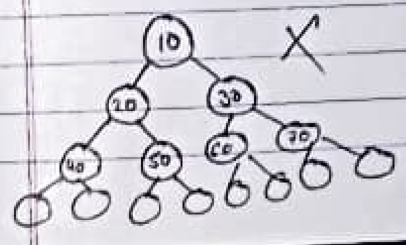
internal nodes = $\left[\frac{n}{2}\right]_{\text{floor value}}$ = $[\frac{5}{2}]$ =

Min Heap tree: In the given complete binary tree at every node root is minimum or equal comparing its children.

Max heap tree: In the given complete binaree tree root is maximum or equal.

Insertion in min heap :

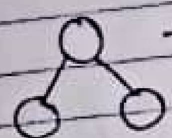
Insertion always takes place at L so that complete binary tree should be disturbed.



5. Graph may be directed. Tree is always directed.

• Every Tree is Graph but Graph isn't tree

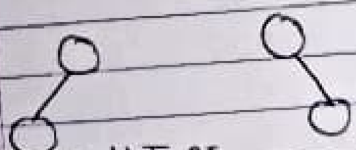
• Binary Tree has atmost 2 child.



→ strict BT or Full BT.

height = Level - 1

func^n calling – Pre order
" execution – Post order



→ Not complete BT

complete BT (Left → Right)

Total no of nodes = $2^n - 1$
n = Level

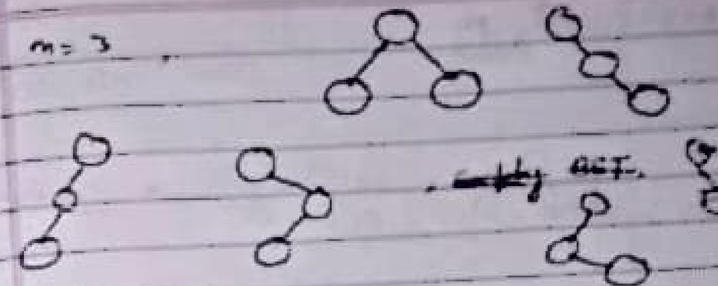
total nodes at n^{th} level = 2^{n-1}

Ques: Find the no of Binary search tree where no of nodes are 1,2,3,4,5 ....10 Root is 3 & right is 7.

Total BST if n is no of nodes = 23

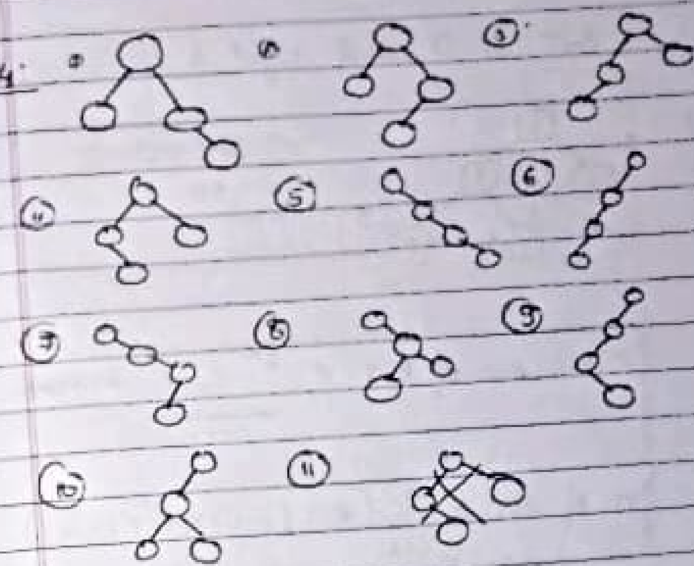
If n = 0 BST = 1 (Empty BST)

n = 3

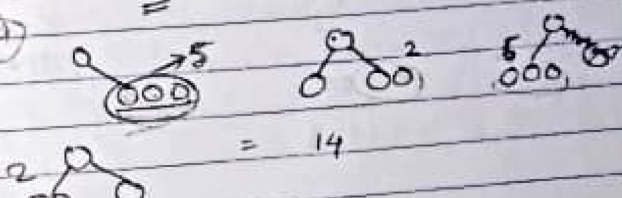


empty BST.

n = 4



(14 BST)



= 14

Page No.

(5.) **Additive :**

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

$\&\&$ $e(n) = O h(n)$

$f(n) + e(n) = O \max(g(n), h(n))$

17 March 2023

(1.) Consider : the following Relⁿ

$f(n) = n^{1+\sin n}$

$g(n) = n$

Find the relation among $f(n)$ & $g(n)$.

$n = (0, 90, 270, 360)$

$f(n) = n^{1+0} = n = 0$

$g(n) = n = 0$

$f(n) = n^{1+\sin 90} = n^2 = (90)^2$

$g(n) = n = 90$

$f(n) = n^{1+(-1)} = n^0 = 1$

$g(n) = n = 270$

$f(n) = n = 360$

$g(n) = n = 360$

Date

Page No.

n	$f(n)$	$g(n)$
$n = 0$	0	0
90	90°	90
180	180	180°
270	1	270
360	360	360

Both the function are now comparable

(2.) Consider : the following Relⁿ

$f(n) = n^{2+\sin n}$

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

Find the relation among $f(n)$ & $g(n)$

n	$f(n)$	$g(n)$
0	0	0
90	90^3	1
180	$(180)^2$	$1/180$
270	270	1

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

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

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

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

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

$\cancel{T(n-k) = T(n-(k-1)) + \log(n-k)}$

$\vdots$

$= T(n-k) + \log(n-k+1) + \log(n-k+2) + \ldots \log n$

$n-k=1$

$= 1 + \log(2) + \log(3) + \ldots + \log n$

$= 1 + \log(2 \cdot 3 \cdot \ldots n)$

$= 1 + \log(n!)$

$n! \approx n^n$

$= 1 + \log n^n$

$= 1 + n \log n$

$= (n \log n)$

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

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

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

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

$\vdots$

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

(3) $f(n) = n \Rightarrow f(n)$ is $O(n)$
$g(n) = n-10$

$f(n) = O(n-10)$
$n \le (n-10) \cdot c$

$c = 1$

$n \le n-10$
$(n+10) \ge 8 \Rightarrow$ for $c = 1$ not

for $c = 2$

(4) $f(n) = n^2$
$g(n) = n$
$f(n) = O g(n) \rightarrow f(n) \ne O g(n)$
$n^2 \le c \cdot n$

$n = 10$
$, c = 1$

» Omega Notation (Ω)

$f(n) = \Omega\, g(n)$
$f(n) \ge c \cdot g(n)$
$c > 0$
$n_0 > 0 \quad \forall \ (n \ge n_0)$

EX (1) $f(n) = n+10$
$g(n) = n$
$n+10 \ge \Omega g(n)$
$n+10 \ge c \cdot n$
↳ always true.

Ex 2: $f(n) = n$
$g(n) = n-10$
$f(n) \ge g(n)$
$n \ge c \cdot (n-10) \Rightarrow$ True.

Ex 3: $f(n) = \Omega g(n)$
$f(n) = n \quad , \quad g(n) = n+10$
$n \ge c \cdot (n+10) \Rightarrow$ True. $c = 1/2$
$n_0 = 10$

Ex 4: $f(n) = n \quad , \quad g(n) = n^2$
$f(n) = \Omega g(n)$
$n \ge c \cdot n^2 \Rightarrow$ X

Theta (Θ) :

$f(n) = \Theta(g(n))$
$f(n) \le c_1 \cdot g(n) \quad (n \ge n_0)$
$f(n) \ge c_2 \cdot g(n) \quad (n \ge 0)$

Ex 1 $f(n) \qquad g(n)$
$n = \Theta n$
$n \le c_1 n$
$n \ge c_2 n$

$f(n) = \Theta g(n)$
$f(n) \le c_1 \cdot g(n)$
$f(n) \ge c_2 \cdot g(n)$

Page No.

Ques) while (i ≤ n)

{

$n = \frac{n}{2}$; → $O(1)$ X

}

Ques) A (n)

{ if ($n \le 2$) return

else

return ($n\sqrt{n}$)

; ↳ $\boxed{\log \log_2 n = k}$

Ques) A (n)

{ for (i=1 ; i≤n; i++)

{ j=n;

while (j > 1) ⇒ $O(n \log_2 n)$

{ j = j/2 ;

}

⇒ $\frac{2^k n}{2^k} = 1$

$2^k n = 2^k$

n = 1 $\log_2 2^k n = \log_2 2^k$

Date

Page No.

Ques) A(n)

{ for (i=1 ; i≤n ; i++) → n

{

for (j=1 ; j≤i ; j++) → n

{

for (k=1 ; k≤133 ; k++)

{

z = x+z ;

}

}

}

}

i	j	k
1	1	133 → 1×133
2	1	133 } 2×133
	2	133 }
n	n	→ n×133

$O(n)$ 133 (1 + 2 + 3 + ... n)

$133 \cdot \frac{n(n+1)}{2}$

$= \frac{133 n^2 + 133 n}{2} = \frac{133}{2} n^2$

$= O(n^2)$

Ques) A(n)

{ for (i=1 ; i<n ; i++)

{ for (j=1 ; j ≤ i^2 ; j++)

{ for (k=1 ; k≤n ; k=2k)

- Analysis.

Analysis: two types of analysis.

① Apriory
② Apostiory.

Apriory:

1.) It is independent on programming language

2.) It is giving same answer in every system.

3.) It will give approximate answers.

Apostiory:

① It is dependent on programming language.

2) It is giving diff. answer in every system.

3.) It will give exact answers

Date
Page No.

1.) main ()

Time complexity = O(1)
counts no of operations

```
{
  a = a+b;
}
```

2.)

Time complexity = O(1+1)
= O(n)

```
main ()
{
  x = y+z
  for (i=1; i<=n ; i++)
  {
    x = y+z;
  }
}
```

3.)

```
main ()
{
  x = y+z ; -> O(1)
  for ( i=1 to n)
  {
    x = y+z;  -> O(n)
  }
  for (i=1 to n) -> O(n)
  { for (j=1 to n) -> O(n)
    { =
    }
  }
```

Time complexity
= O(1)+O(n)+O(n²)
= O(n²)