C- Programming & Data structure] - (SN)

classmate
Date
Page 1

- **Array** -

(i) Row major order -

→ ∞ [ ]$_{m \times n}^{row, col}$

m:n,

$A[i][j] = ((i \times n) + j) * size\ of\ element + Base\ add.$

→ (Index start form 0)

$A[i][j] = ((i-1) \times n + (j-1)) * size\ of\ element + Base.$

→ (Index start form 1)

(ii) Col mayor order -

m:n,

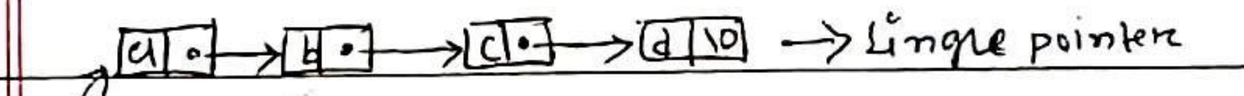
$A[i][j] = ((j \times m) + i) * size\ of\ element + Base.$

$A[i][j] = (((j-1) * m) + (i-1)) * size\ element + Base.$

- Structure -

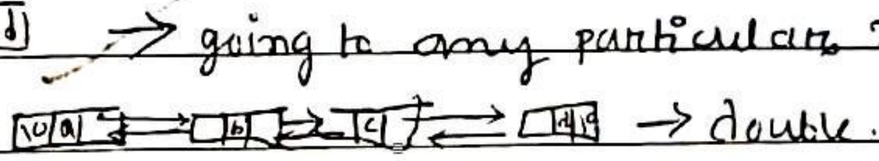
→ structure can be pass to a function as well as return by a function.

* • LinK List -



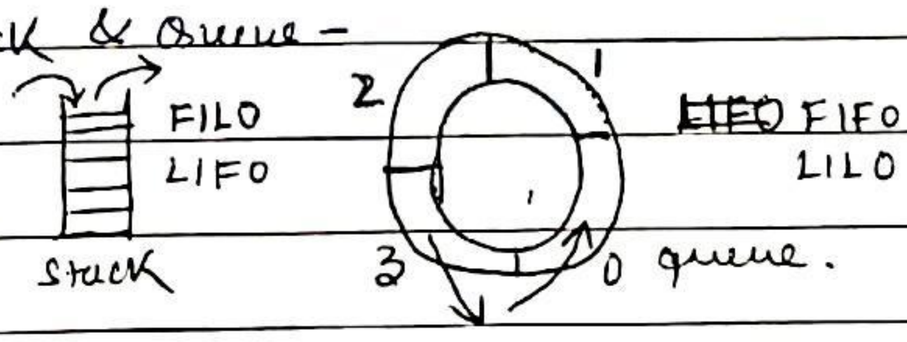
→ Singly pointer

→ going to any particular number structure (link-list) take O(n).

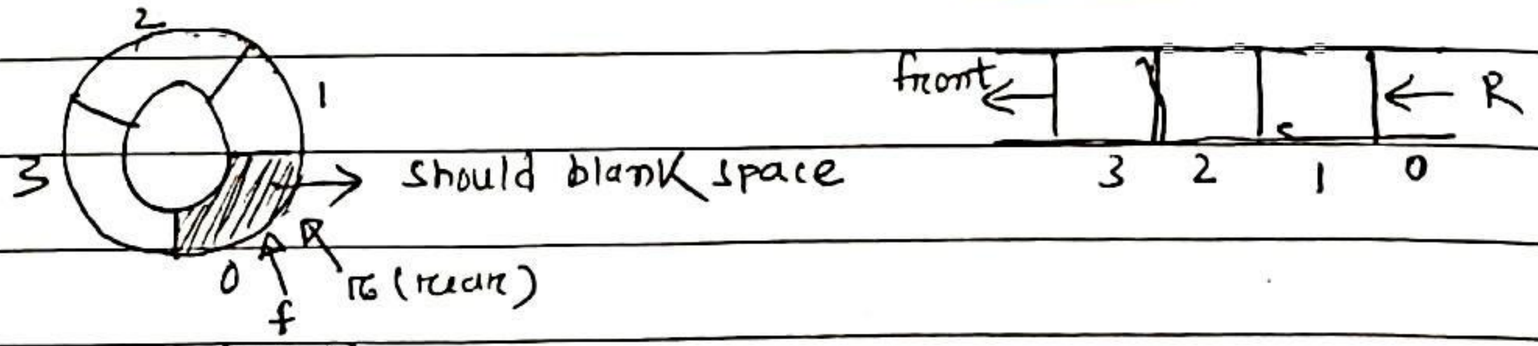


→ double pointer.

* • Stack & Queue -



→ In circular queue, if array size = n, then we use (n-1) amount of size.

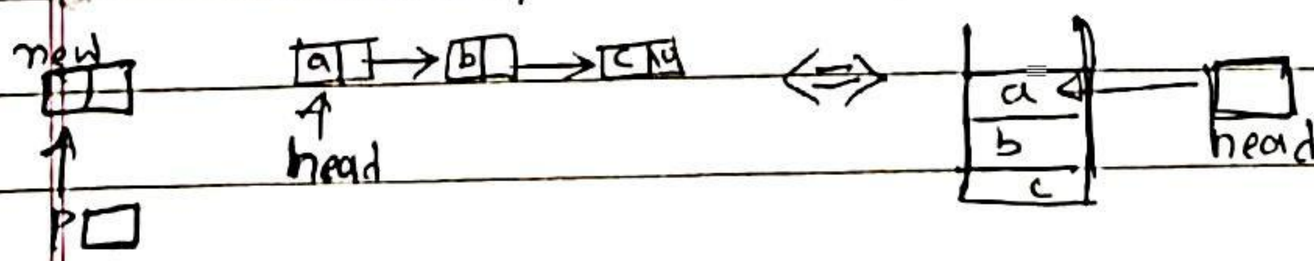


→ (rear+1) mod n == front [overflow] → full.

→ Queue rear == front [underflow] → empty.

→ stack PUSH & POP → O(1) Time complexity.

→ link list imp ~~wise~~ of stack:



- **Infix ⇒ postfix**

⇒ postfix related anything stack req.

ex: a*b+c — infix

⇒ ab*c+ — postfix

ex: a*b+(c-d) — infix

⇒ ab*cd-+ — postfix.

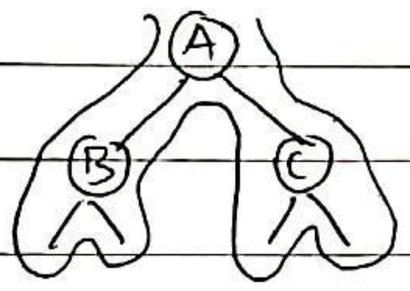
*** • Tree:

Methods of traversing —

→ In-order traversing. Left-Root-Right = BAC

→ pre-order " . Root-left-Right = ABC

→ post-order " . left-Right-Root = BCA



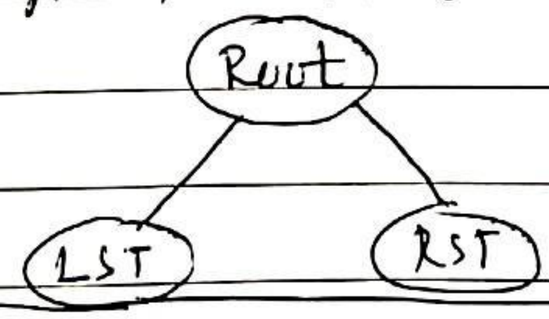
→ Time complexity ~~for~~ In/pre/post order traversal = O(n).

→ With 'n' node no. of Unlabelled Binary tree = $\frac{2^nC_n}{(n+1)}$.

→ With 'n' node, no. of labelled " " = $\frac{2^nC_C}{(n+1)} * n!_0$

→ With given pre, post and Inorder we always get only one ~~Unic~~ unique binary tree.

- Recursive program to count no. of node - O(n) - T.C & S.C

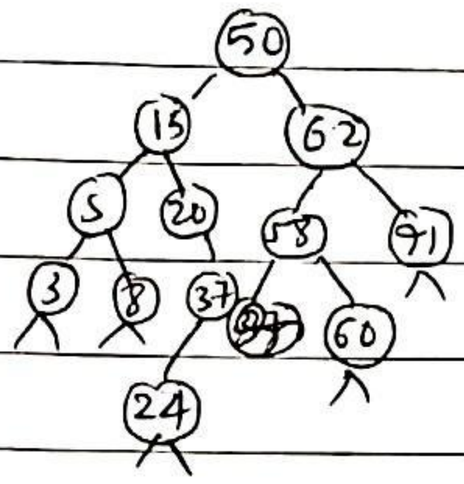



```
NN(T) = 1 + NN(LST) + NN(RST)
      = 0 ; if T is NULL
```

- Recursive pro to ~~find~~ count no. of leaves and non leaves = O(n)
- " " find height of tree = O(n) T.C & S.C.

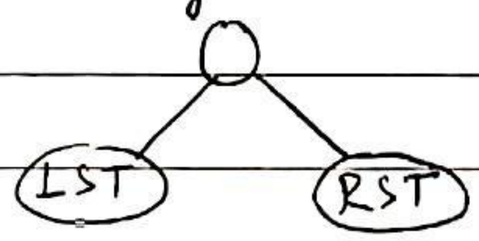
- BST (Binary Search tree) —

50, 15, 62, 5, 20, 58, 91, 3, 8, 37, 60, 24



→ Inorder traversal of a BST give sorted ordered element.
→ By doing deletion never increase hight of tree.
→ By " insertion hight may increase.
- Inorder successor = last element in Right sub tree.
- Inorder predecessor = greates element in left sub tree.

- AVL tree -
→ is an balencing search tree.



Balence factor = Hight(LST) − Hight(RST)
= (-1, 0, 1) OK
= -2 or 2 → unbalanced

* - Complexity Analysis -

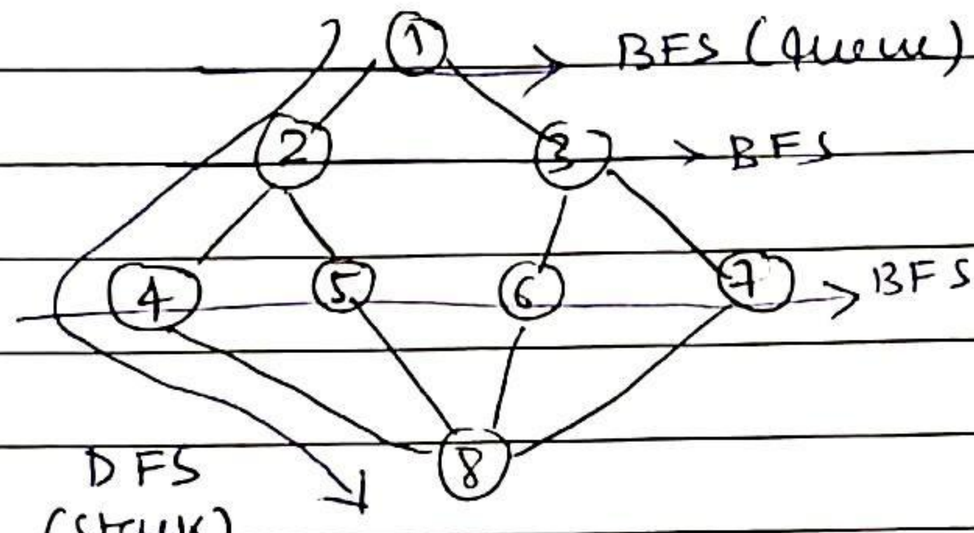
	BST	BBST (AVL)
Search	O(n)	O(logn)
Hight	O(n)	O(logn)
Insert	O(n)	O(logn)

→ Max node in AVL or normal tree = $2^{h+1} - 1$ h → hight.
→ Min node in AVL,
N(h) = N(h-1) + 1 + N(h-2) = (h+1).
→ some time use substitution method.

**
- Recursion - (stack tree or tree)
→ In recursion using stack or tree we can found o/p.
→ When function contain many lines then go with stack method.
→ " " " only recurrive function then go with tree method.
→ Q. Type (find o/p).

*. Graph -

- BFS (breadth first search) & DFS : Depth



→ ~~BFS Implem analysis on adjacency mtx Implemen~~

→ BFS Implementation	adj mtx	link list.
T.C	$O(V^2)$	$O(V+E)$
S.C	$O(V)$	$O(V)$ or $O(n)$

→ DFS Imp	Adj mtx	adj list.
T.C	$O(V^2)$	$O(V+E)$
S.C	$O(V)$	$O(V)$

→ Space & Time complexity same in both case BFS & DFS.