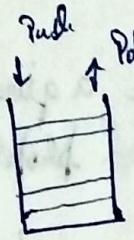
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Data Structure ⇒ Linear → Stack, Array, Linked List, Queue
⇒ Non Linear → Tree, Graph, Heap

☆ Stack ⇒ LIFO
⇒ Ex ⇒ Stack of book



☆ Basic Operation ⇒ ① PUSH (Insertion)
② POP (Deletion)

☆ Push Algorithm ⇒

① Check whether stack is full or not (TOP == size - 1).
② If full, display "Full stack" & can't insert
③ Else, increment top value by one (top ++) set value of [stack[top] = value]

☆ Pop algorithm ⇒

① Check whether stack is empty (top == -1)
② If empty, display "Stack is empty". Can't delete.
③ If not empty, then delete stack[top] & decrement top value (top --)

☆☆ Display elemt of stack algo ⇒

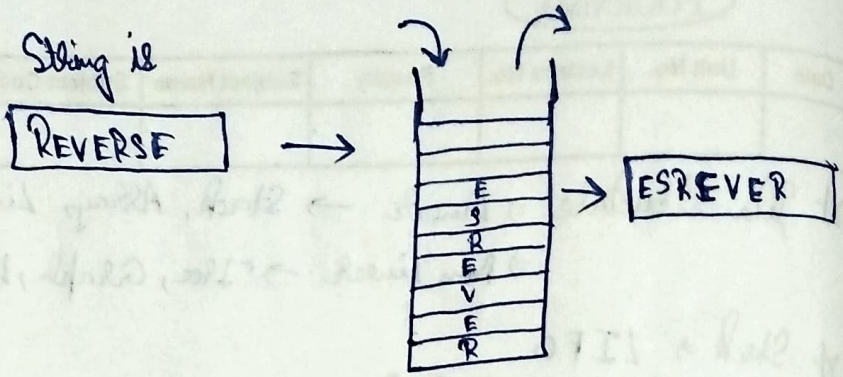
① check whether stack is empty (top == -1)
② If empty, display "Stack empty" & terminate function.
③ else, define a variable 'i' & initialize with top (top = size - 1).
Display stack[i] value & decrement i value by one (i--).
④ Repeat ③ step until ~~i == 0~~ (i != 0).

Main Ideas, Questions & Summary:

Library / Website Ref.:-

★ Application of stack =>

① Reversing String =>



② Checking whether a given arithmetic expression containing nested parenthesis is properly parenthesized.

Valid input	Invalid input
{}	{(}
({[]})	([(()])

③ Evaluating arithmetic expression. ④ Factorial calculation

★ Tower of Hanoi => Invented by French Mathematician
=> Puzzle involving usage of Stack.

[Diagram left]

• Rules =>

① One disc can be shifted at once
② Only top disc can be shifted
③ Large disc cannot be placed on smaller disc.
④ Only 3 Stacks can be used.

★★ Can be solve in $(2^n)-1$ moves.

n = no of disc

• Use of recursion is used.

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

★ Representation of Stack ⇒

① Using array → Static & dynamic array

② Using linked list

★ Stack using static array ⇒ size of array is fixed

• Steps to create an empty stack ⇒

① Include all header files which are required & define constant 'SIZE' with specific value.

② Create one-D array with fixed size.

③ Define 'top' variable with '-1' value. (int top = -1)

★
```
void push (int value){
    if(top == size-1)
        printf("stack is full");
    else {
        top ++;
        stack[top] = value;
        printf("successfully inserted");
    }
}
```

Main Ideas, Questions & Summary:

Library / Website Ref.:-

```
void pop(){
    if (top == -1)
        printf("Stack empty");
    else{
        printf("deleted: %d" stack[top]);
        top--;
    }
}
```

* Dynamic array to create stack => Use malloc function.

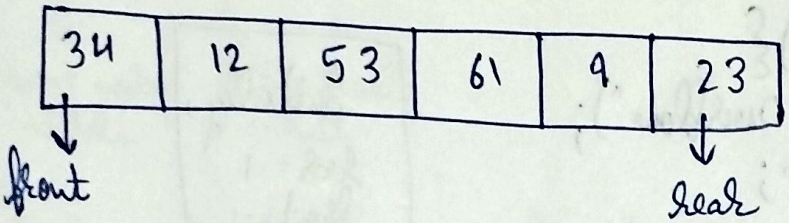
1. Allocate new space
2. Copy data to new space
3. Free old space

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Queue ⇒ First in First out (FIFO)

① new element is added at one end called Rear end.
② element delete from other end called front end.



☆ Operations on queue ⇒

① Insertion ⇒ Known as enqueue.
⇒ Placing item in queue.
⇒ Done from ~~rear~~ Rear.

② Deletion ⇒ Known as dequeue
⇒ Removing item fromqueue
⇒ Done from front

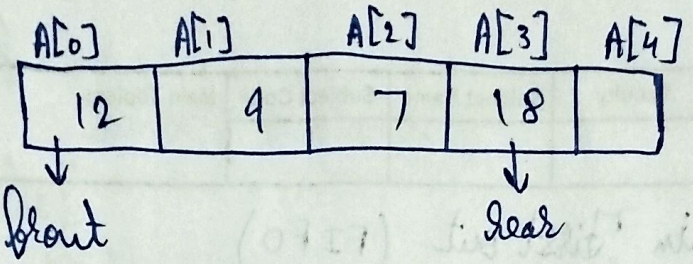
☆ Representation of queue ⇒

① Array
② Using stack

Main Ideas, Questions & Summary:

Library / Website Ref.:-

✡ Array representation of queue ⇒



✡ Algorithm for insertion (enque) ⇒

① if (rear == max-1){
printf("Overflow");
return 0;
}

Initially,
rear = -1
front = -1

if (rear == -1){
front = rear = 0;
}
else{
rear = rear+1;
}

✡ queue[rear] = num;

✡[More Pg 37]✡

✡ Algorithm for delete (~~enq~~ dequeue) ⇒

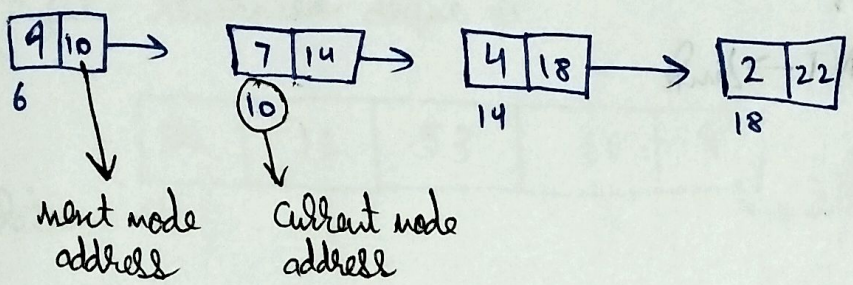
if (front == -1 || front > rear){
printf("underflow");
}
else{
val = queue[front];
front = front + 1;
}

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Linked Representation of queue =>

Front = 6
Rear = 18



☆ Algorithm to insert ~~enen~~ element using linked list =>

① Allocate memory for the new node & name it as TEMP

② set TEMP → data = NUM
set TEMP → link = NULL

③ if FRONT = NULL
FRONT = REAR = TEMP
else
REAR → link = TEMP
REAR = TEMP

④ EXIT

Initially,
FRONT = NULL
REAR = NULL

Main Ideas, Questions & Summary:

Library / Website Ref.:-

* Algorithm to delete using linked list ⇒

① if FRONT = NULL
write underflow
go to step ③

② Set TEMP = FRONT
FRONT = FRONT → Link
if FRONT = NULL
REAR = NULL

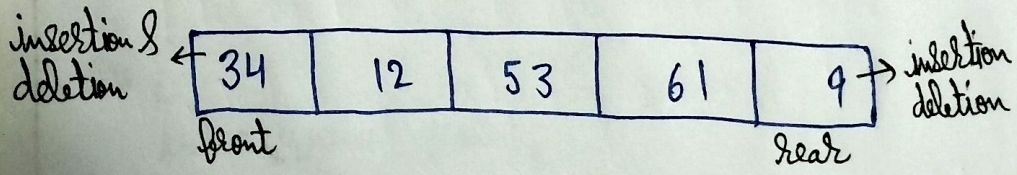
③ EXIT

* Type of queue ⇒

① Deque
② Circular queue
③ Priority queue

* Dequel ⇒

① Stands for double ended queue.
② Deletion & insertion can be done from any end.
③ Also known as head - tail linked list.

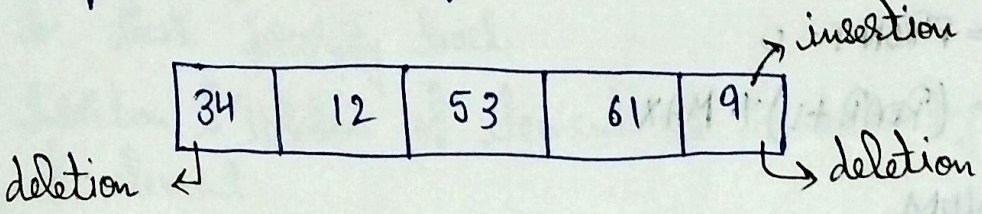


POORNIMA

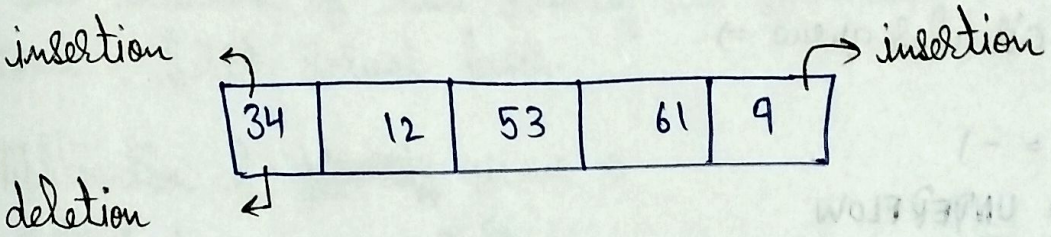
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Types of deque ⇒

① Input restricted deque ⇒



② Output restricted deque ⇒



☆ Operations on deque ⇒ ☆ [37] ☆ ☆ ☆

① insertFront()
② insertLast()
③ deleteFront()
④ deleteLast()

☆ Circular queues ⇒

① Use to remove drawback of ~~simple~~ simple queue.
② Also called "Ring buffer".
③ Both, front & rear pointers wrap around to beginning of array.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

* Algorithm to insert element in circular queue =>

① if FRONT = (REAR+1) % MAX
    write OVERFLOW
    go step ④

② if FRONT = -1
    REAR = FRONT = 0
  else REAR = (REAR+1) % MAX

③ CQ[REAR] = NUM

④ EXIT

* Delete in circular queue =>

① if FRONT = -1
    write UNDERFLOW
    go to ③

② if FRONT = REAR
    FRONT = REAR - 1
  else
    FRONT = (FRONT+1) % MAX

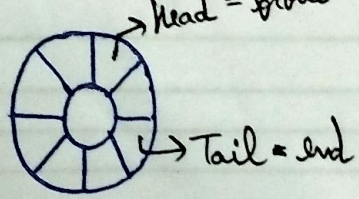
③ EXIT

* Application of circular queue =>

① Computer controlled traffic signal system

② CPU scheduling & memory management.

* [More Pg 27] *



POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Priority Queue ⇒ ☆[Pg 47]☆

(1) Collection of elements where elements are stored according to their priority level.

(2) Insertion & Deletion of element is decided by priority of element.

(3) Element of higher priority is processed first

(4) Two elements of same priority are processed on first-come - first served basis.

☆ Application of ~~priority~~ queue ⇒

(1) Cashier line in any store

(2) People on an escalator

(3) Job ~~Sc~~ scheduling

(4) Checkout at any book store

☆ Stacks	Queues
(1) Principle of LIFO	(1) Principle of FIFO
(2) Insert operation is 'push'.	(2) Insert operation is 'enqueue'.
(3) Delete operation is 'pop'.	(3) Delete operation is 'Dequeue'.
(4) Insertion & deletion take place from only one end.	(4) Both end are use for insertion & deletion
(5) We maintain one pointer (Top)	(5) Maintain 2 pointer (front & rear).

Main Ideas, Questions & Summary:

Library / Website Ref.:-

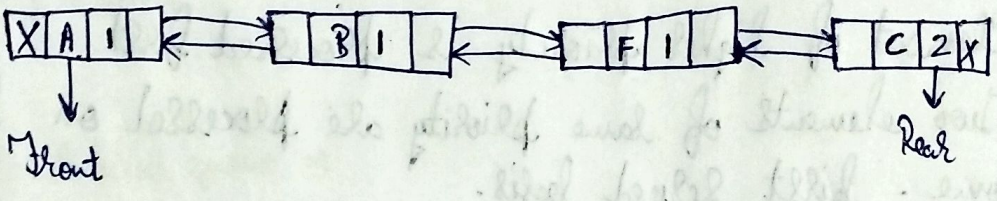
* Priority queue operations ⇒

① is_empty ⇒ check whether the queue has no elements.

② insert_with_priority ⇒ add element to queue with associated priority

③ full_highest_priority_element ⇒ remove element that has highest priority & return it.

* Priority queue using Linked list ⇒

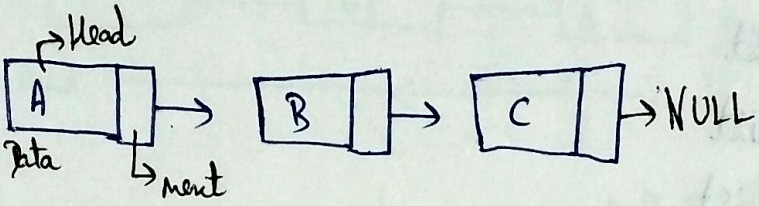


POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

★ Linked list ⇒

① Linear data structure
② Element stored at contiguous memory location.
③ Element are linked using pointers



• First node is called 'head'. If list is empty, head = NULL.
• Each node contain ⇒ (i) data
(ii) Pointer to next node

★ Advantages of linked list ⇒

① Reduce the access time.
② Stacks & queue can be easily executed.
③ Insertion & deletion can be done easily.
④ Dynamic in nature

★ Disadvantages of linked list ⇒

① Reverse traversing is difficult.
② Memory is wasted as pointer require extra memory.
③ No element can be accessed randomly.
④ We cannot do binary search easily.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

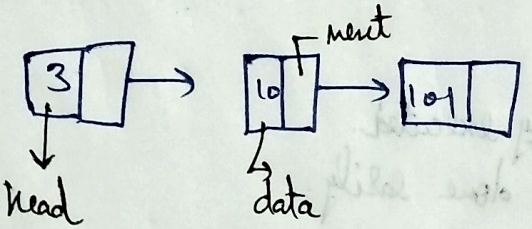
✱ Application of linked list ⇒

① Used to implement stacks, queue, graph.
② Dynamic memory allocation
③ Performing arithmetic operations on long integers
④ Music player
⑤ Image viewer

✱ Types of linked list ⇒

① Singly linked list
② Doubly linked list
③ Circular linked list

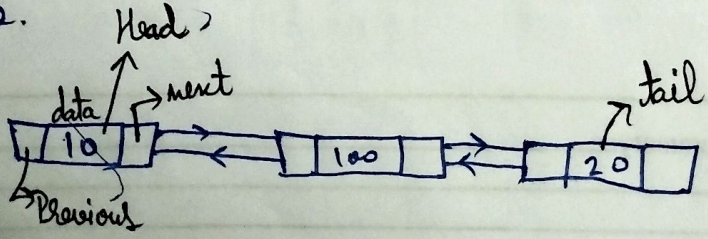
✱ Singly linked list ⇒ Contain nodes which have a data part as well as an address part (next) which points to the next node in sequence of nodes.



✱ Operations ⇒ Insertion
⇒ deletion
⇒ traversal

⇒ Peek = See value of topmost element.

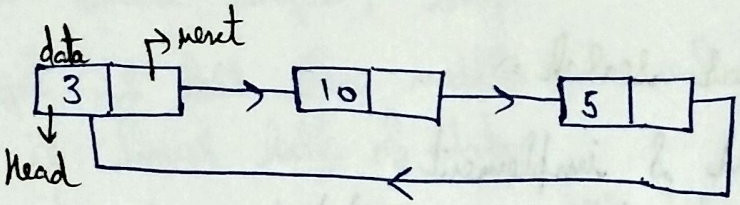
✱ Doubly linked list ⇒ Each node contain a data part & 2 address. one for previous node & one for next node.



POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Circular linked list ⇒ Last node of list holds the address of first node, forming circular chain.



☆ Array	Linked list
① Size of array must be specified at time of it's declaration	① Size is variable. It grows at runtime.
② Insertion & deletion take more time	② Insertion & deletion take less time
③ Use, static memory allocation	③ Use, dynamic memory allocation
④ Array can be 1, 2, 3].	④ It can be singly, doubly, circular linked list.
⑤ It uses index value	⑤ Uses pointer

Main Ideas, Questions & Summary:

Library / Website Ref.:-

* Linear Search =>

(1) It is a Sequential Search.

(2) Compare each element with the value is search & stop when element is found till the end of array.

(3) Searching is easy

* Advantage of linear search =>

(1) Easy to understand & implement

(2) It does not require data in sorted manner.

* Disadvantage of linear search =>

(1) Slower than other algorithm of search

(2) Poor efficiency as lot of comparisons is done in big file.

* Analysis of linear search =>

(1) Best case, target value found in first element. It takes less time.

(2) Worst case, target value at last element of array.

Search time $\propto$ length of array

(3) Average case, target value at middle of array

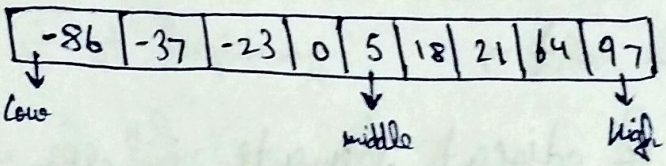
POORNIMA

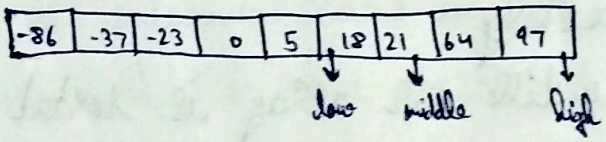
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

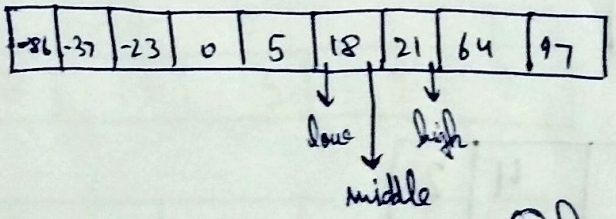
☆ Binary Search ⇒ Data should be sorted for this.

① Look at middle element of array.
② If you found target, Stop
③ If target is less than middle data value, new search region is lower half of data.
④ If target is greater than middle data value, new search region is higher half of data.
⑤ Continue from step ②

⑱ ⇒







⑱ found !!

Main Ideas, Questions & Summary:

Library / Website Ref.:-

★ Types of Sorting ⇒

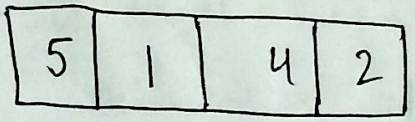
① Bubble Sort ✓
② Insertion Sort ✓
③ Selection Sort ✓
④ Quick Sort ✓
⑤ Heap Sort
⑥ Merge Sort ✓
⑦ Radix Sort ✓
⑧ Counting Sort ✓

★ Bubble Sort ⇒ Works by repeatly swapping the adjacent elements if they are in wrong order.

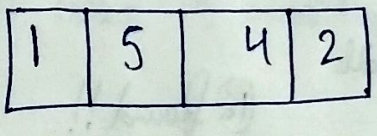
★ Algorithm of bubble sort ⇒

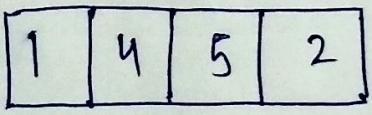
① Compare each pair of adjacent elements in array.
② Swap element if necessary
③ Repeat this process untill our array is sorted.

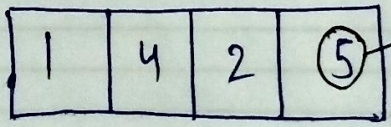
Ex ⇒



First iteration ⇒



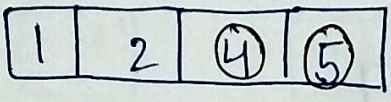




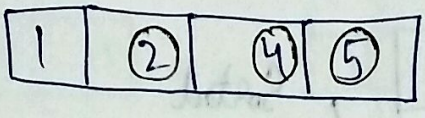
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

Second iteration ⇒



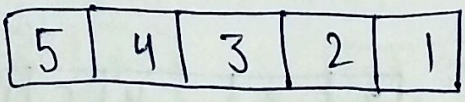
Third iteration ⇒



, Done

✱ Time complexity of bubble sort ⇒ $O(n^2)$

• Worst case ⇒



✱ Selection sort ⇒ Sort element by repeatedly finding the minimum element from unsorted part & putting it at beginning.

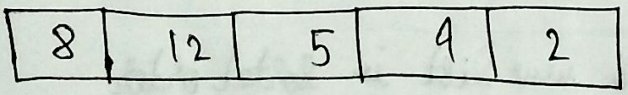
✱ Algorithm of selection sort ⇒

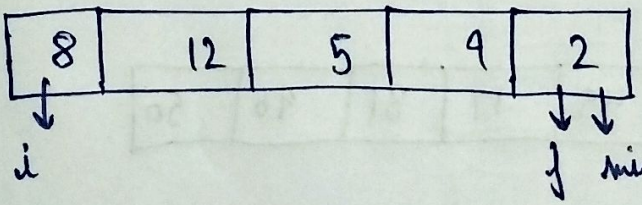
① Find the minimum value in list

② Swap it with the value in current position.

③ Repeat this process, until array is sorted.

Ex ⇒

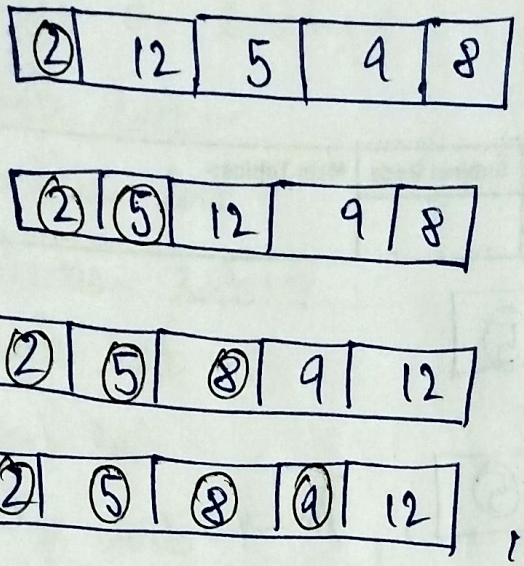




(Swap i & j)

Main Ideas, Questions & Summary:

Library / Website Ref.:-



, Sorted

★ Time complexity of selection sort ⇒ $O(n^2)$

• Example of worst case, 2 | 3 | 4 | 5 | 1

★ Merge Sort ⇒ It is a divide & conquer algorithm. It divides array in 2 halves, calls itself for the two halves and then merges the 2 sorted halves.

★ Algorithm for merge sort ⇒

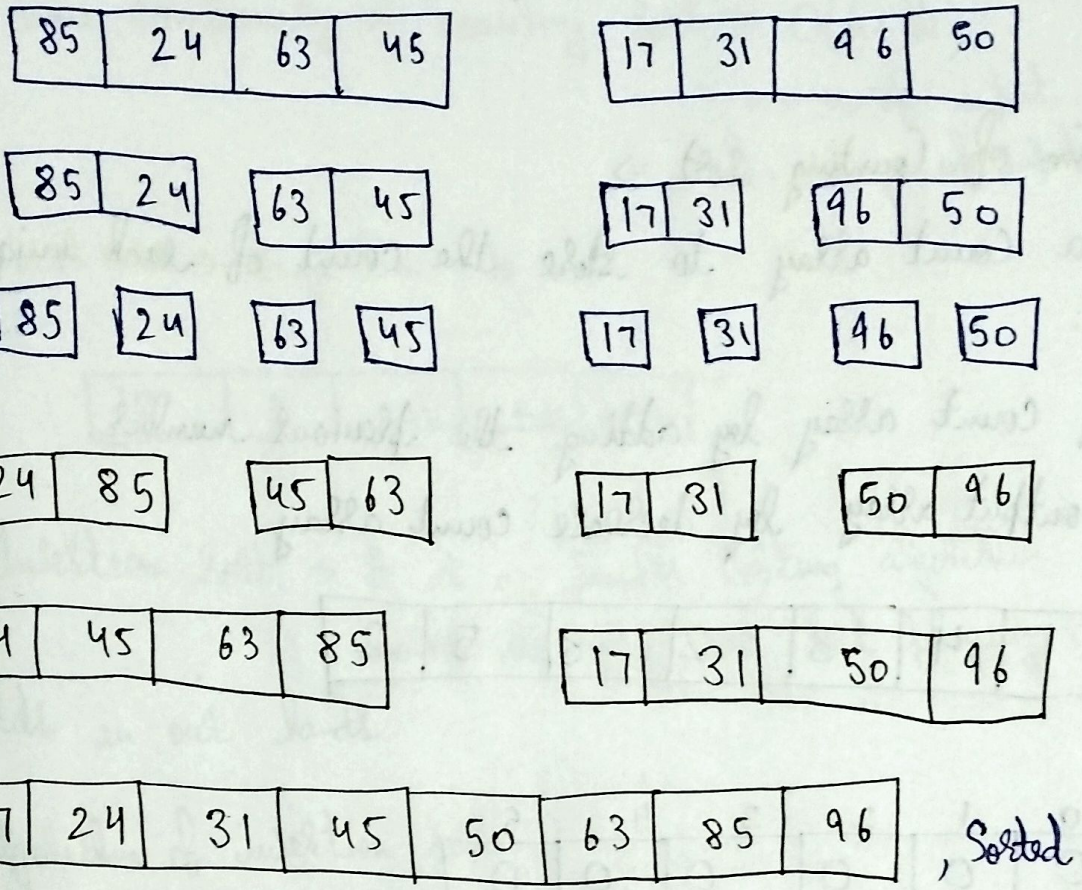
① Divide the list recursively into two halves until it can no more be divided.

② Merge the smaller list into new list in sorted order.

Ex ⇒ 85 | 24 | 63 | 45 | 17 | 31 | 96 | 50

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-



* Time complexity of Merge sort $\Rightarrow O(n * \log(n))$

Main Ideas, Questions & Summary:

Library / Website Ref.:-

✡ Counting Sort ⇒ It is based on keys between a specific range. It works by counting the number of ~~digits~~ objects having distinct keys values.

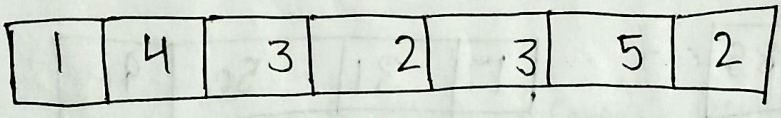
✡ Algorithm of Counting Sort ⇒

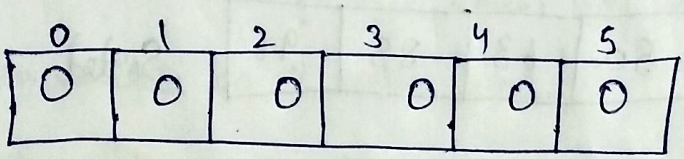
① Create a count array to store the count of each unique object.

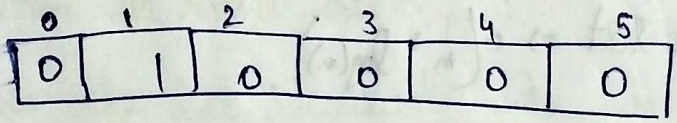
② Modify count array by adding the previous number.

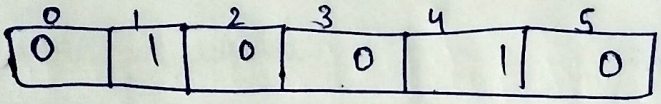
③ Create output array by decrease count array

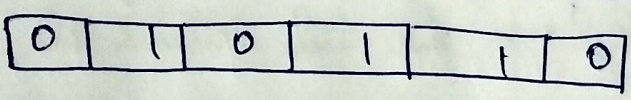
Ex ⇒

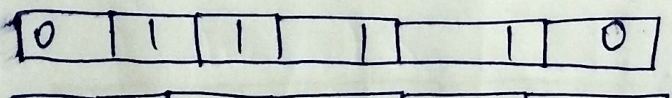


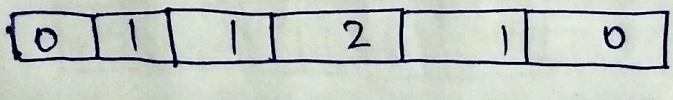


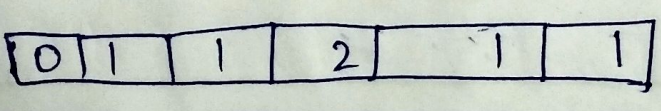


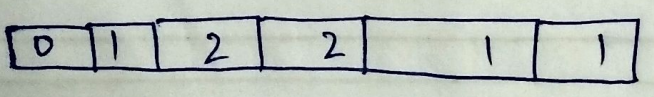


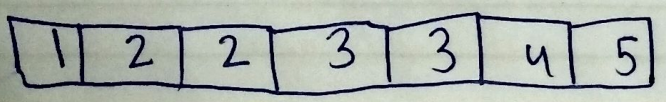










⇒ 

, Done

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

★ Time complexity of counting sort ⇒ $O(n+k)$

⇒ n = no of input element

⇒ k = range of input

• Worst case ⇒ Range b/w 1 to 10K

10	5	10k	5K	200

★ Insertion Sort ⇒ It is a simple sorting algorithm that works the way we sort playing cards in our hands.

★ Algorithm of insertion sort ⇒

① Compare each pair of adjacent element in list

② Insert element into the sorted list, until it occupies correct position.

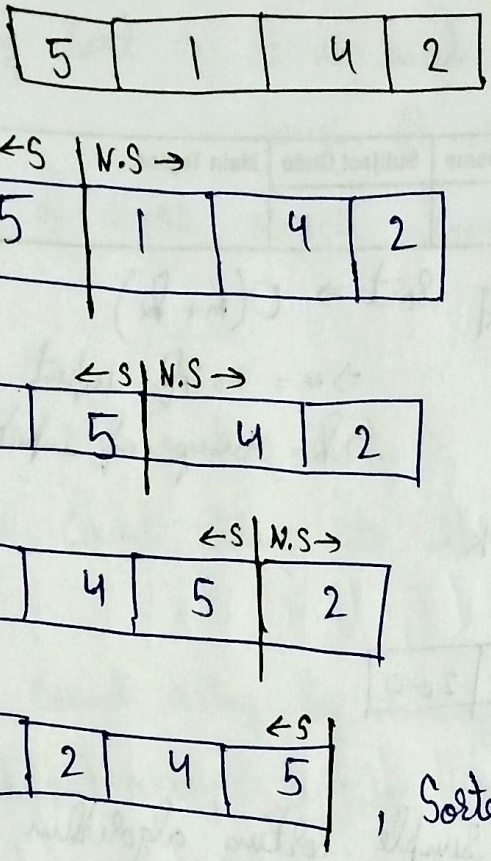
③ Swap 2 element if necessary

④ Repeat this process for all element until the entire array is sorted.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

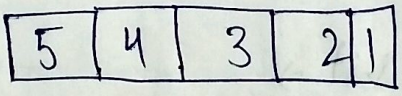
Ex ⇒



, Sorted

★ Time Complexity of insertion sort ⇒ $O(n^2)$

- Worst case ⇒



★ Radix Sort ⇒ It sort number by processing digits of each number either starting from LSD or MSD.

⇒ Use digit by digit sort starting from LSD to MSD

⇒ Use counting sort as subroutine to sort

★ Algorithm of Radix sort ⇒

① Take LSD of each element

② Sort list of element based on that digit.

③ Repeat the sort with next MSD.

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

Ex ⇒

170	045	075	090	802	024	002	066

Sorting (1st place) ⇒

170	90	802	2	24	45	75	66

Sorting (10's place) ⇒

802	2	24	45	66	170	75	90

Sorting (100's place) ⇒

2	24	45	66	75	90	170	802

, Done

☆ Time complexity of Radix sort ⇒ $O(n + k/d)$

⇒ n = no of elements
⇒ k = Range of input
⇒ d = no of digits

~~Quick~~

☆ ~~Radix sort~~ ⇒

☆ Quick sort ⇒ Based on divide & conquer algorithm.

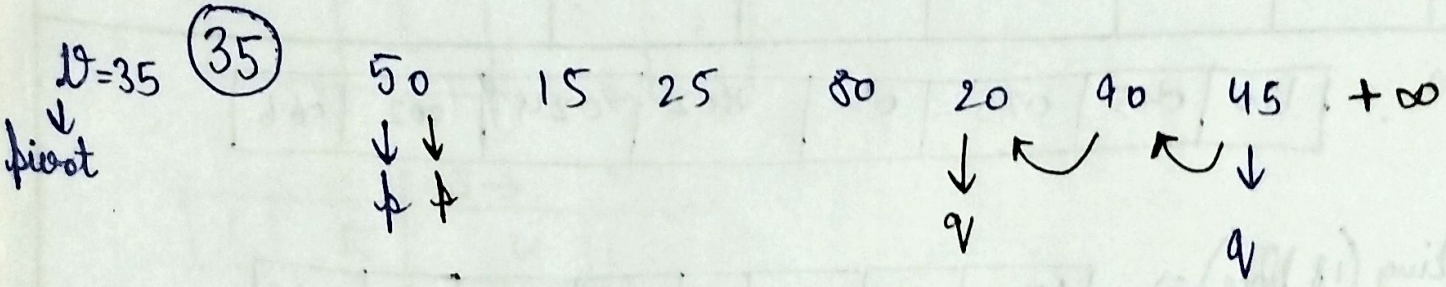
⇒ It picks an element as a pivot & partitions the given array around the picked pivot.

☆ Algorithm of quick sort ⇒

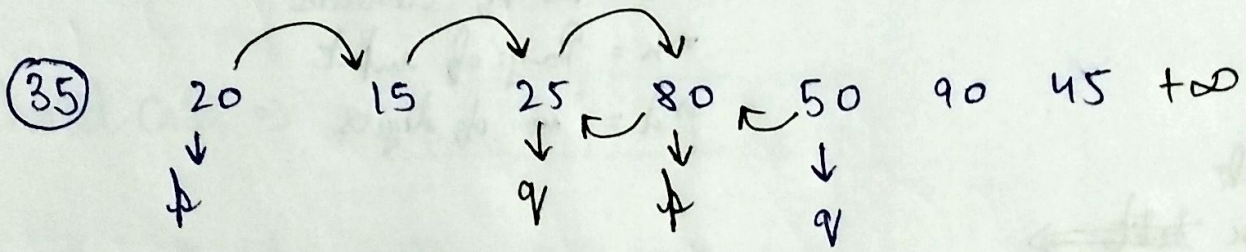
Main Ideas, Questions & Summary:

Library / Website Ref.:-

Ex ⇒ 35 50 15 25 80 20 90 45



- i ⇒ move RHS
  ⇒ Stop when find element greater than pivot
- j ⇒ move LHS
  ⇒ Stop when find element smaller than pivot
- After stop, swap i & j



- If i, j are at same position or cross each other then swap v, j

25 20 15 (35) 80 50 90 45 +∞

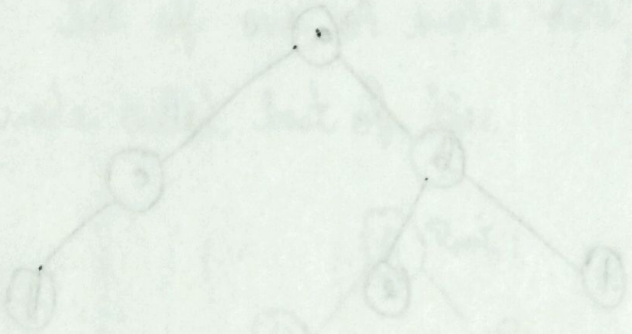
Apply quick sort — Apply quick sort

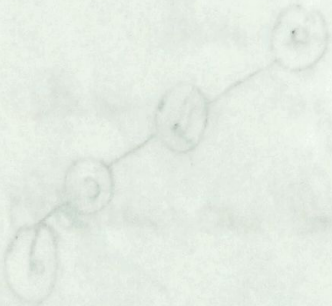
15 20 25 35 45 50 80 90 , Done

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

* Complexity of quick sort =>





Main Ideas, Questions & Summary:

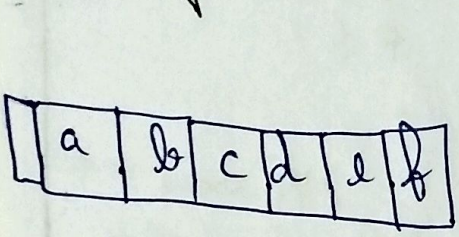
Library / Website Ref.:-

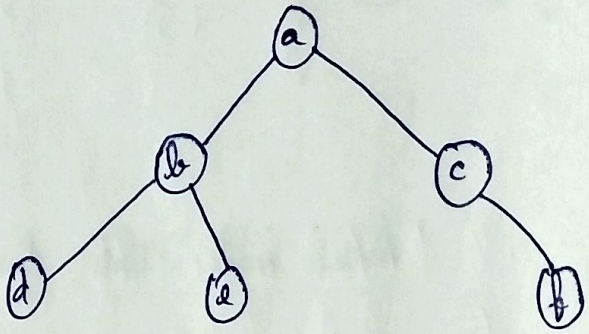
✱ Binary tree representation ⇒

① Array representation

② Linked representation

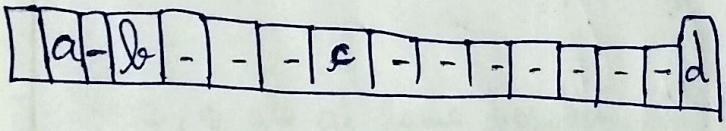
✱ Array representation ⇒

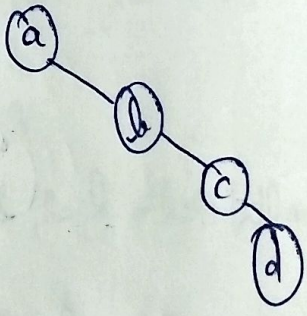




- An 'n' node binary tree needs an array whose length is b/w 'n+1' and '2^n'.

✱ Right skewed binary tree ⇒

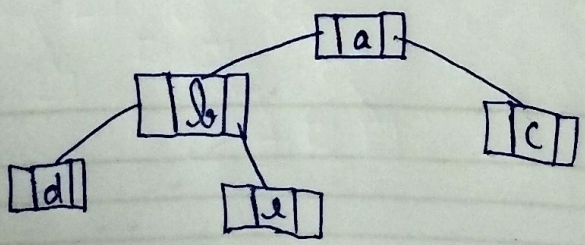




✱ Linked representation ⇒

① Each node is represented by Binary tree node

② Space required for 'n' node is n x (Space required by one node)

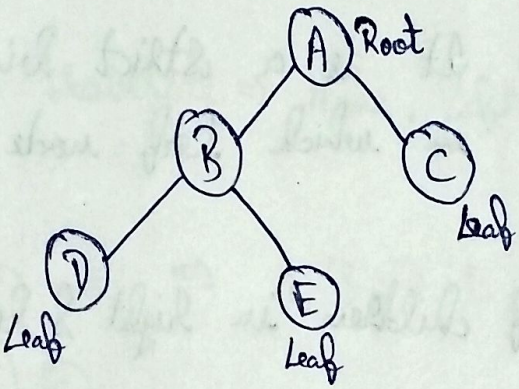


POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Tree ⇒ It is a hierarchical representation of a finite set of one or more data items such that :-

- Special node called root of tree
- Ex ⇒



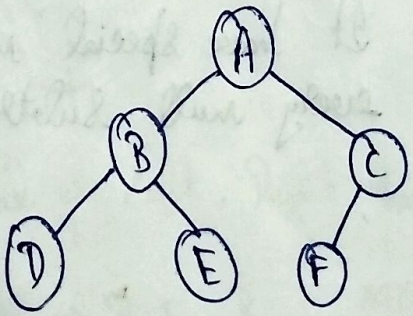
Level 0

Level 1

Level 2

☆ Binary tree ⇒ It is a tree, in which root can have maximum two children such that each of them is again a binary tree.

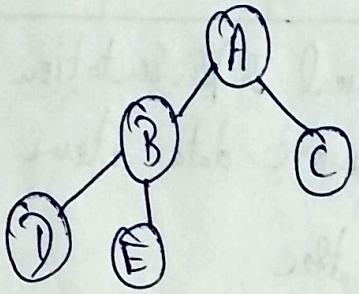
⇒ 0, 1, 2 children can be possible.



Main Ideas, Questions & Summary:

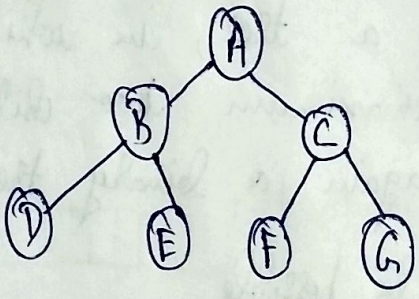
Library / Website Ref.:-

✡ Strict binary tree ⇒ Root can have exactly 2 or 0 children.

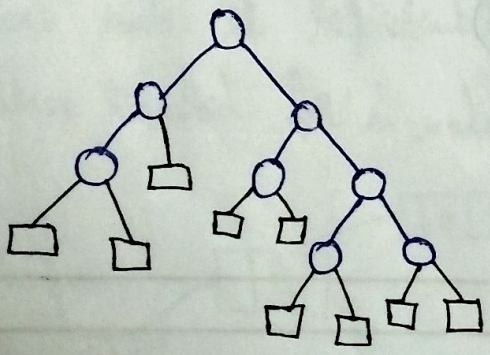


✡ Complete binary tree ⇒ It is a strict binary tree in which leaf node is at same level.

⇒ There are equal no of children in right & left subtree of every node.



✡ Extended binary tree ⇒ It have special node which replace every null subtree.



POORNIMA

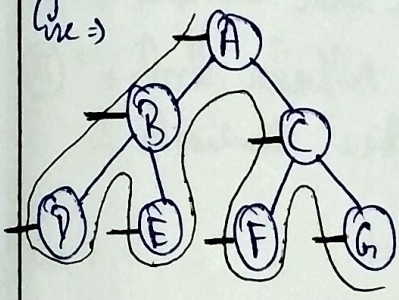
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Tree Traversal ⇒

1. Pre-Order
2. In-Order
3. Post-Order

☆ Pre-Order Traversal ⇒ Node, Left, Right

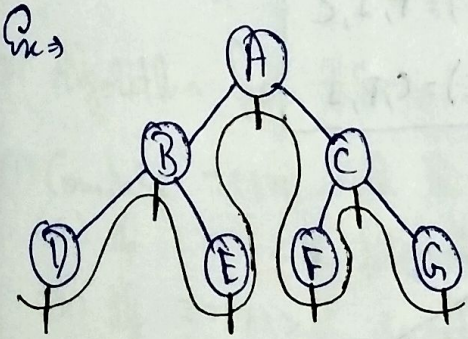
Eg ⇒



⇒ A, B, D, E, C, F, G

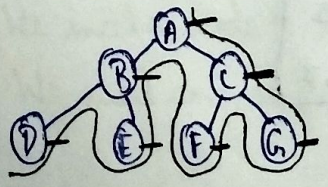
☆ In-Order Traversal ⇒ Left, Node, Right

Eg ⇒



⇒ D, B, E, A, F, C, G

☆ Post-Order Traversal ⇒ Left, Right, Node



⇒ D, E, B, F, G, C, A

Main Ideas, Questions & Summary:

Library / Website Ref.:-

✱ Construction of binary tree ⇒

Ex ⇒ Inorder = D, G, B, H, E, A, F, I, C

Preorder = A, B, D, G, E, H, C, F, I

① Find root ⇒ A (From Preorder)

② Find right & left part of root ⇒

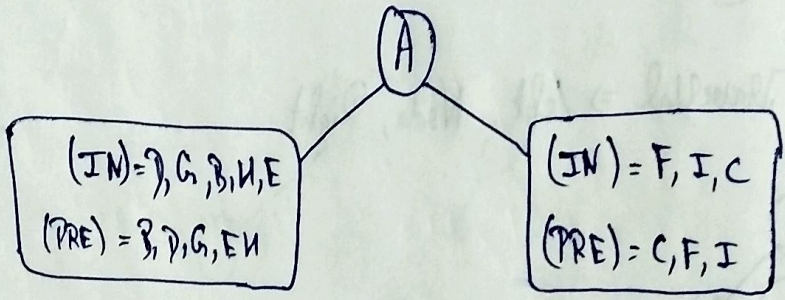
- Left part ⇒ (IN) = D, G, B, H, E

  (PRE) = B, D, G, E, H

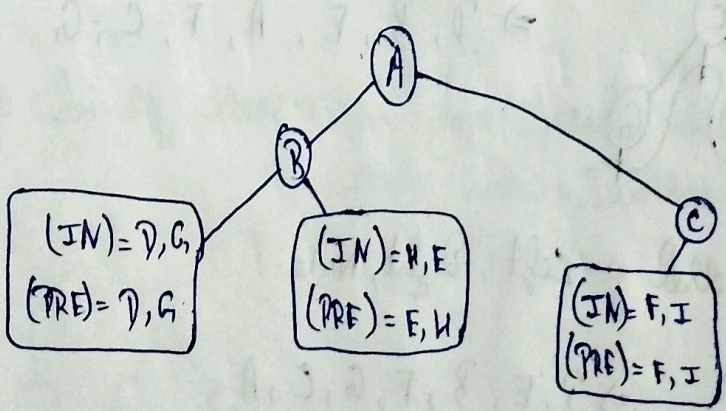
- Right part ⇒ (IN) = F, I, C

  (PRE) = C, F, I

⇒



⇒



⇒ Continue again

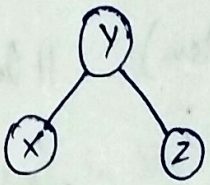
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Binary Search Tree (BST) ⇒ Time of search ∝ Height of tree

- The value at any node,

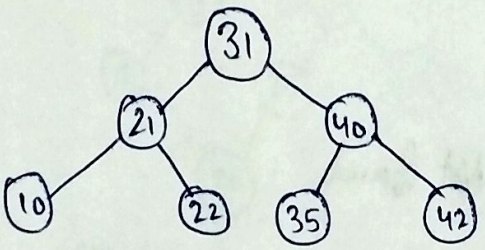
① Greater than left subtree
② Smaller than right subtree



$Y > X$
$Y < Z$

③ Search is fast
④ No duplicate node should be present
⑤ To check, whether the tree is BST or not, traverse tree in 'Inorder' & elements will be in ascending order.

Ex ⇒



☆ Algorithm for BST ⇒

① Compare ITEM with the root node N of tree ⇒
(a) If ITEM < N, proceed to left child of N.
(b) If ITEM > N, proceed to right child of N.

② Repeat ① until ⇒
(a) We meet a node N such that ITEM = N, i.e Search is successful
(b) We meet an empty sub tree, i.e Search is unsuccessful.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

✡ Iterative search in BST ⇒

```
search(){
    while(n != NULL){
        if(n->data == item)      // Found it
            return n;
        if(n->data > item)       // In left subtree
            n = n->lc;
        else                     // In right subtree
            n = n->rc;
    }
    return null;
}
```

✡ Recursive search in BST ⇒

```
search(node *n, info){
    if(n == NULL)                // Not found
        return n;
    else if(n->data == item)     // Found it
        return n;
    else if(n->data > item)       // In left subtree
        return search(n->left, item);
    else
        return search(n->right, item);   // In right subtree
}
```

POORNIMA

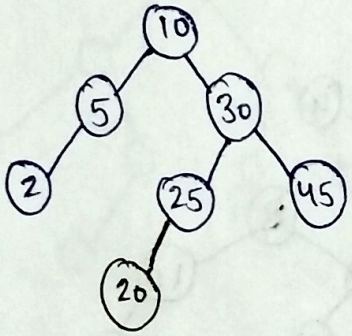
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Insertion in BST ⇒

• Algorithm ⇒

① Perform search for value X.
② Search will end at node Y (if X not in tree)
③ If X < Y, insert new leaf X as new left subtree for Y
④ If X > Y, insert new leaf X as new right subtree for Y

Ex ⇒ Insert 20,



☆ Deletion in BST ⇒

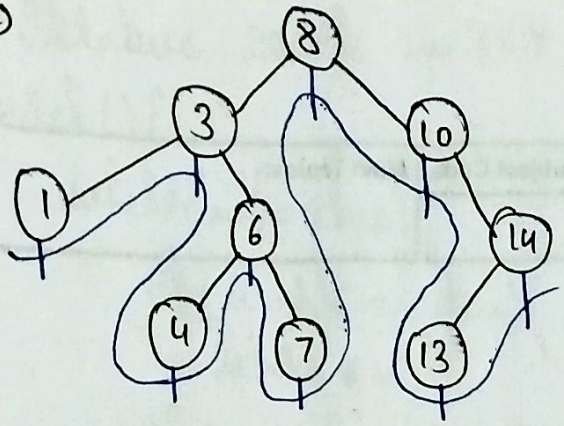
• Algorithm ⇒

① Perform search for value X.
② If X is a leaf, delete X.
③ Else, replace the ~~node~~ deleted node with "Inorder Pre" or "InorderPost".

Main Ideas, Questions & Summary:

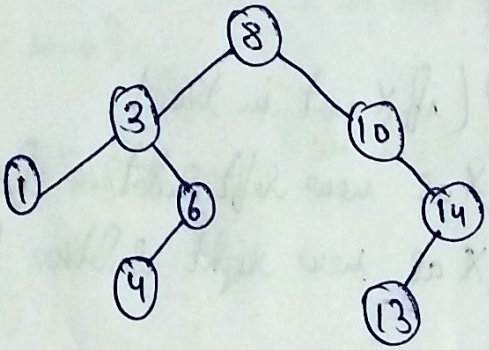
Library / Website Ref.:-

Ex ⇒

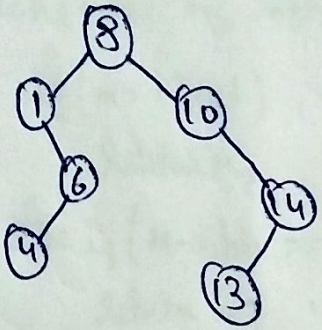


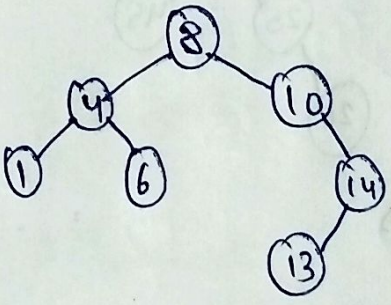
Inorder ⇒ 1, 3, 4, 6, 7, 8, 10, 13, 14

(i) Delete (7) ⇒



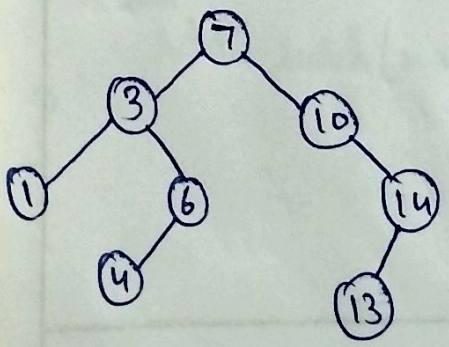
(ii) Delete (3) ⇒ Replace (3) with (1) or (4)

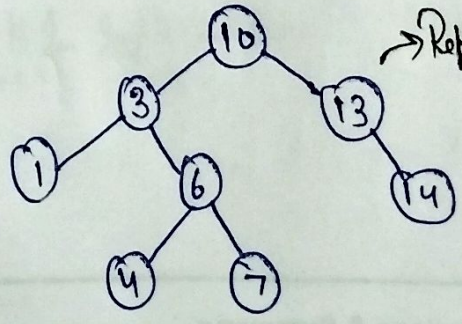




(iii) Delete (8) ⇒ Replace (8) with (7) or (10)

↳ Easy to replace





→ Replace (10) with (13)

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

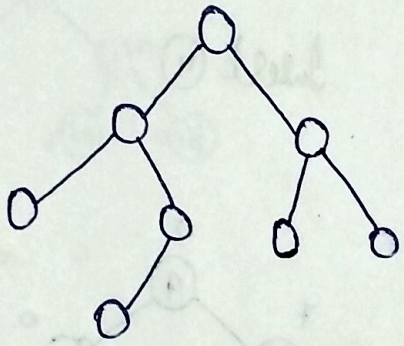
☆ AVL tree ⇒

① Height balanced BST

② Balanced factor = Height of Left Subtree - Height of Right Subtree

③ Height should be -1, 0, 1 at each node difference b/w Height of Left & Right Subtree.

Ex ⇒



☆ Rebalancing ⇒ When we insert or delete a node in tree, it losses it's AVL property. So rebalancing is done.

. It is done using Rotations.

☆ Rotations ⇒

① Single Rotation

② Double Rotation

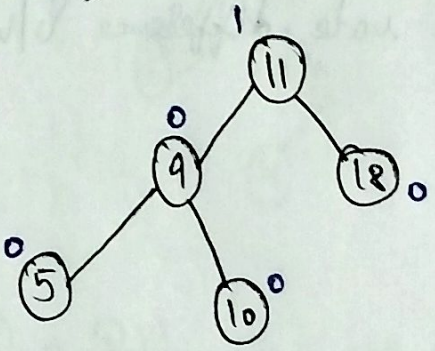
Main Ideas, Questions & Summary:

Library / Website Ref.:-

✡ Balancing a AVL tree after insertion ⇒

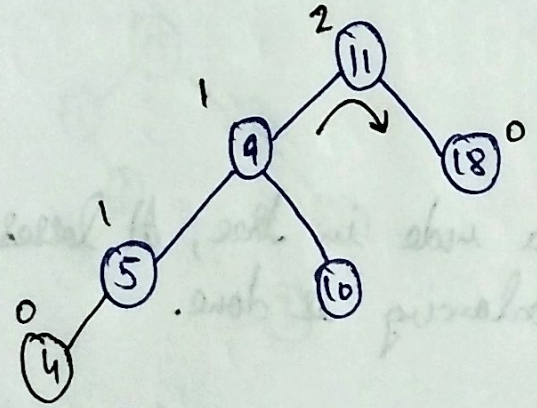
① LL insertion ⇒ Right Rotate w.r.t first imbalanced node.
② RR insertion ⇒ Left Rotate w.r.t first imbalanced node
③ LR insertion ⇒ Left Rotate than Right Rotate
④ RL insertion ⇒ Right Rotate than left Rotate.

✡ LL insertion ⇒

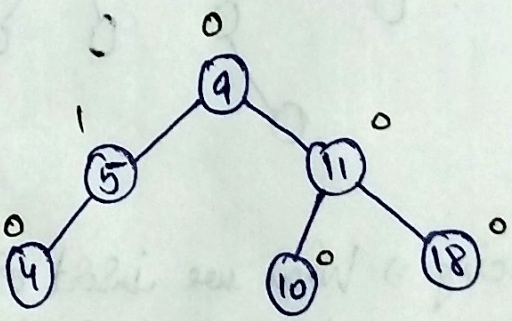


Insert ④

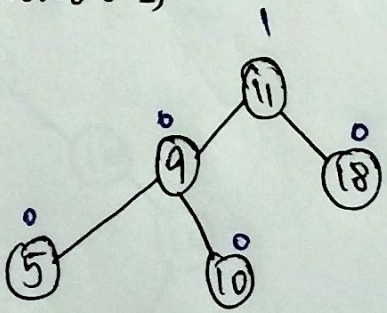
⇒



⇒



✡ ~~RR~~ LR insertion ⇒

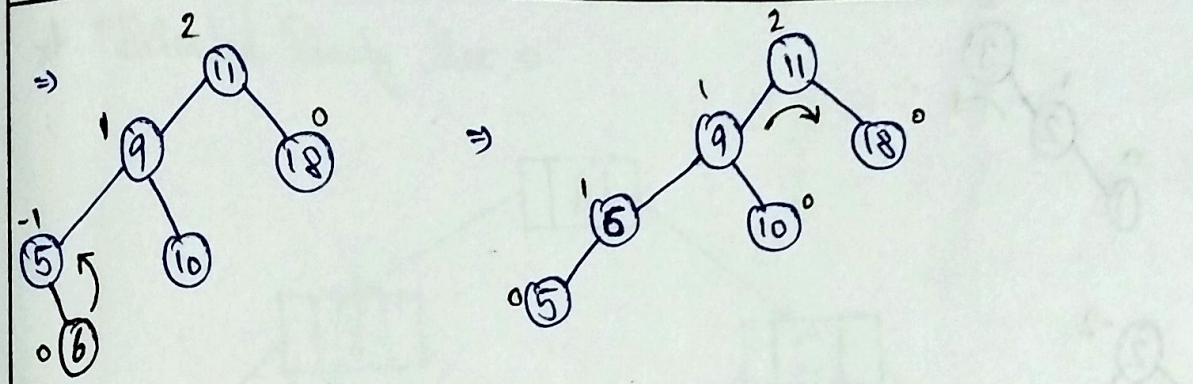


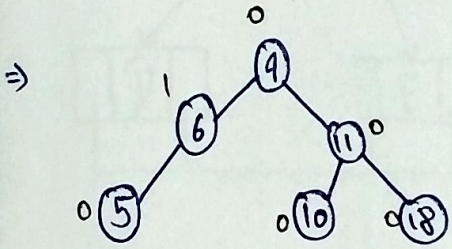
Insert ⑥

RR Left

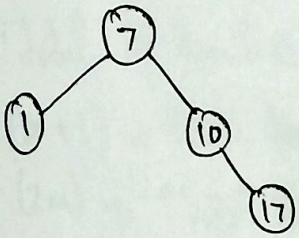
POORNIMA

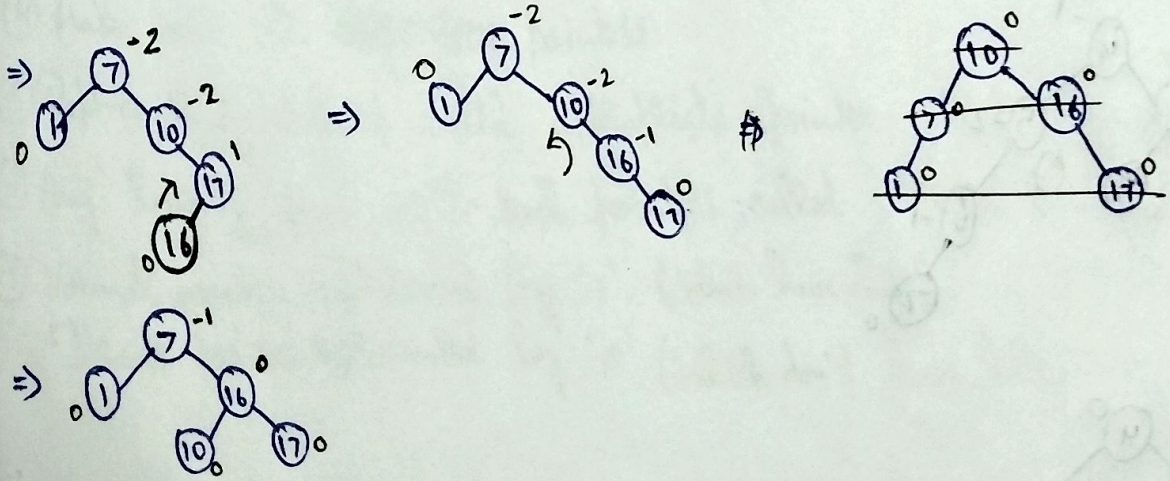
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-





★ RL insertion ⇒





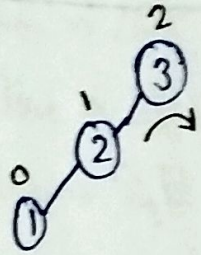
Main Ideas, Questions & Summary:

Library / Website Ref.:-

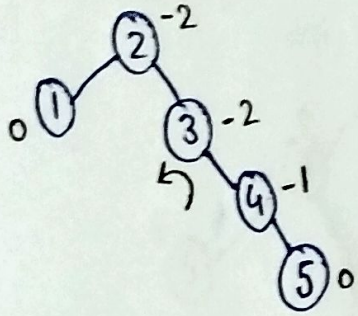
☆ Construct AVL tree ⇒

Q. ⇒ 3, 2, 1, 4, 5, 6, 7

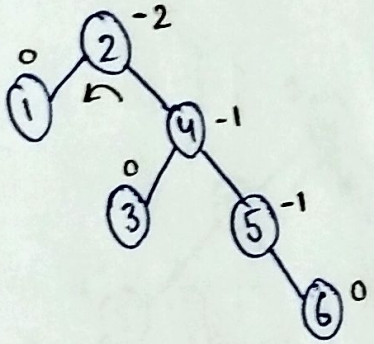
⇒



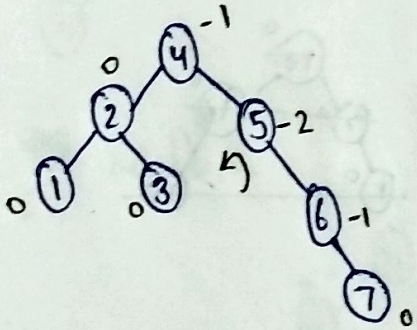
⇒



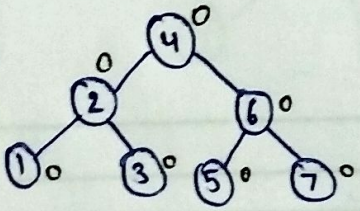
⇒



⇒



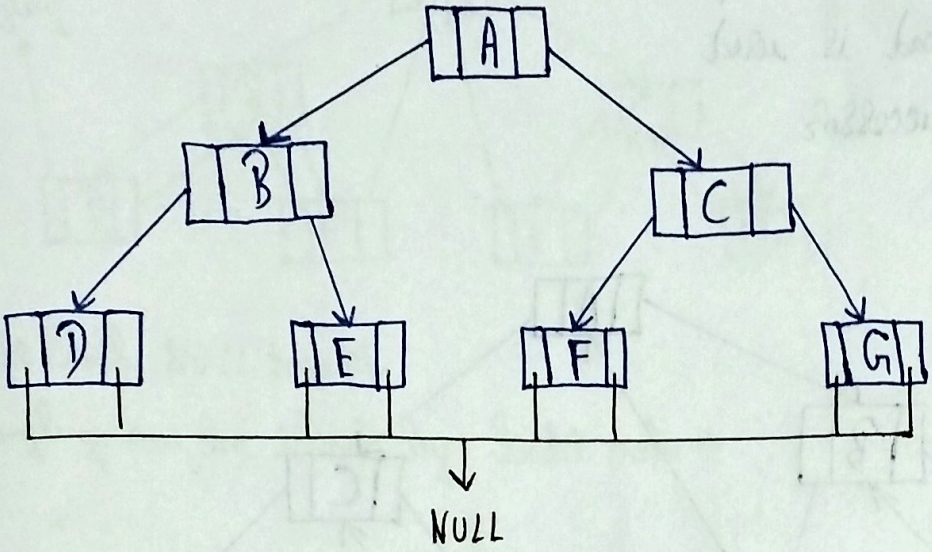
⇒



POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

★ Threaded Binary tree ⇒



① There are 8 NULL pointer & 6 actual pointers.

② Total 14 pointers

③ $(n+1)$ ⇒ NULL pointer ; n ⇒ no of nodes
$(2n)$ ⇒ Total pointers

④ Make use of these NULL pointers

⑤ Replace NULL pointers with appropriate pointer called "threads".

⑥ Any binary tree with such pointers called threaded binary tree.

⑦ Normal pointer represented by '1'. (Which have child)
Threads pointer represented by '0'. (Which don't have child)

Main Ideas, Questions & Summary:

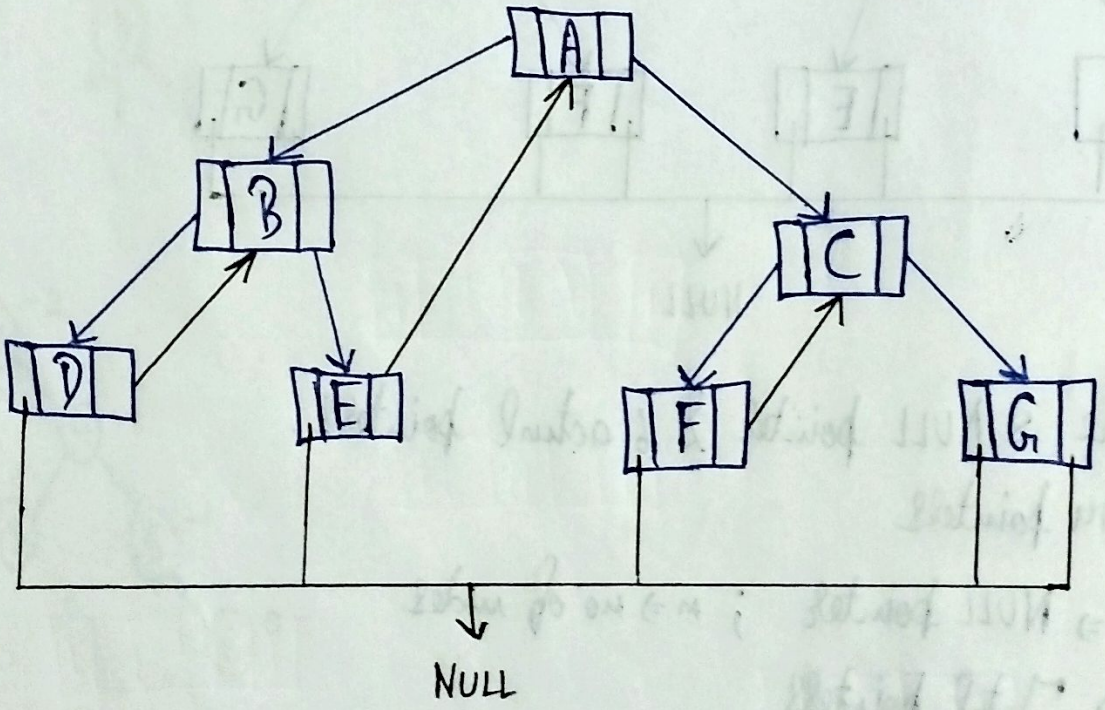
Library / Website Ref.:-

✡ Types of Threaded binary tree ⇒

① One Way
② Two Way / Double

✡ One Way Threaded binary tree ⇒

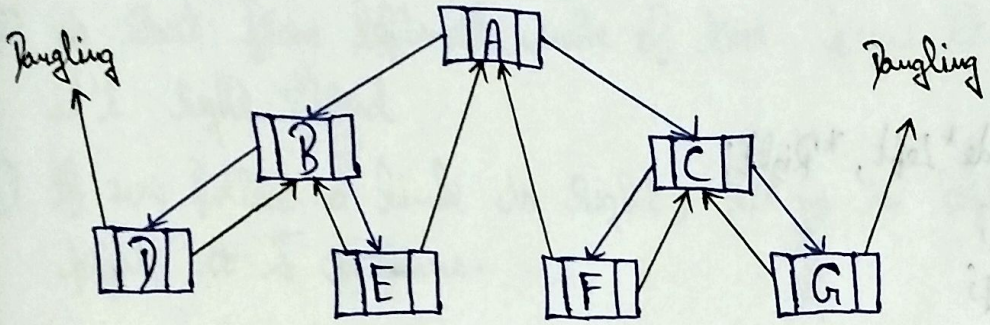
① Only right thread is used
② Use in-order successor



POORNIMA

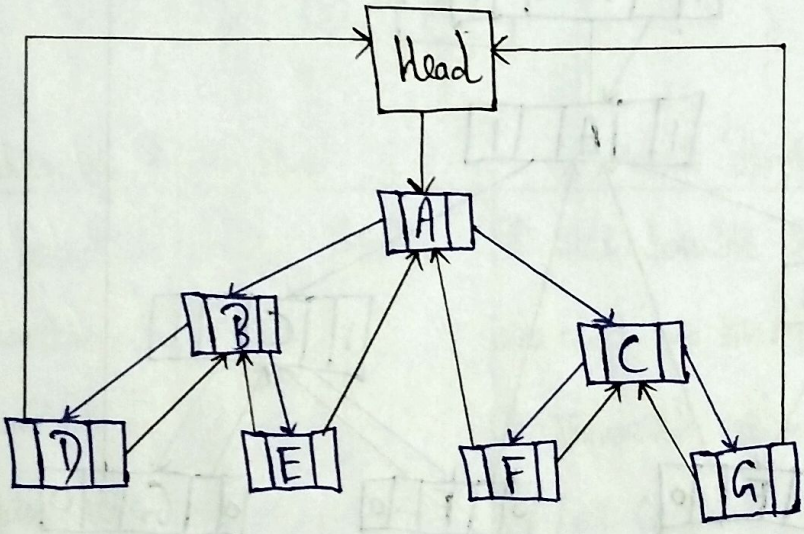
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Two Way / Double Threaded binary tree ⇒



① Use both NULL pointer.

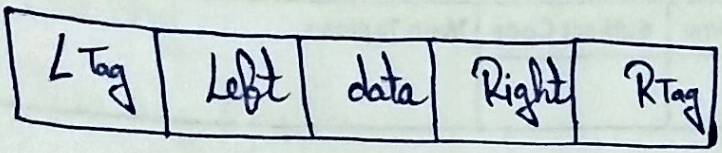
② Dangling can be solved by header node ⇒



Main Ideas, Questions & Summary:

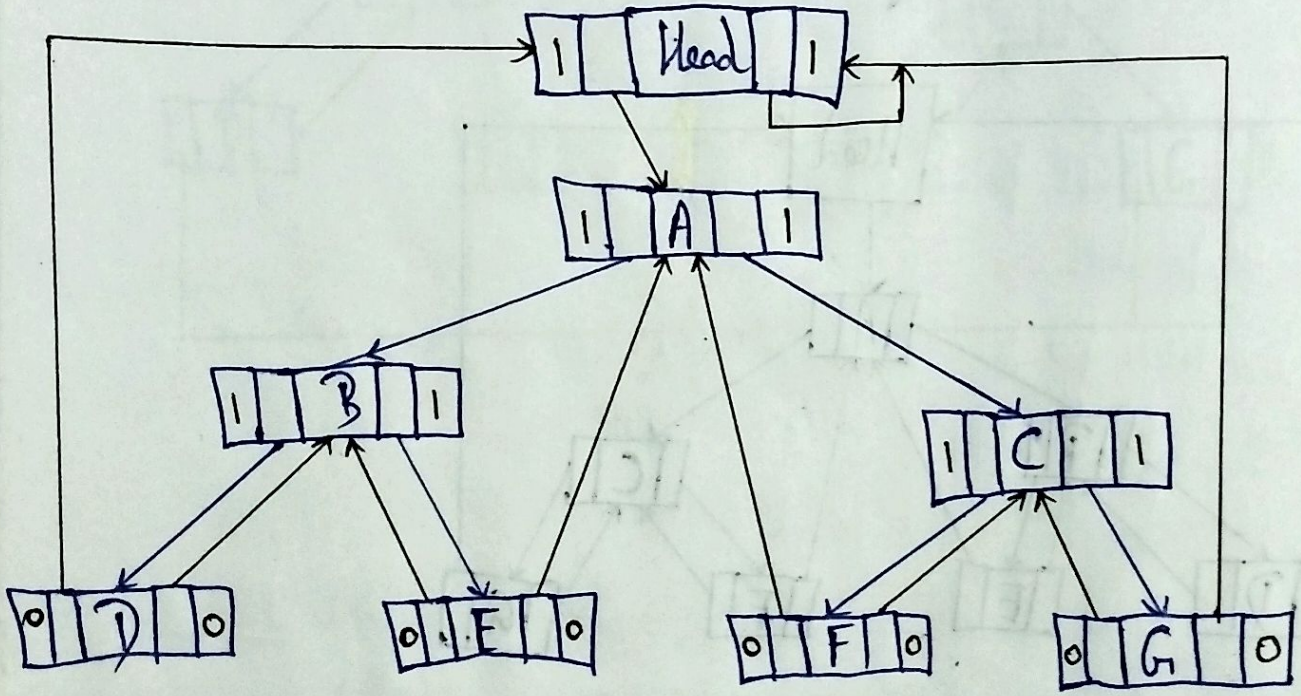
Library / Website Ref.:-

✡ Structure of Threaded Binary tree ⇒



⇒
```
Struct Node {
    int data;
    Struct Node* Left, *Right;
    bool LTag;
    bool RTag;
};
```

Ex ⇒



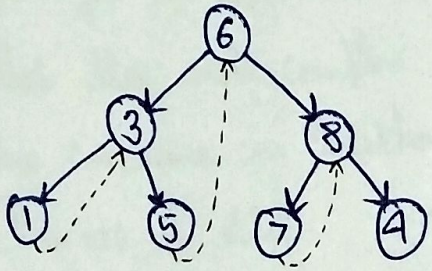
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Traversal of Threaded tree ⇒

1. We start from leftmost node of tree, print it & follow it's right thread
2. If we follow a link to right, we go to leftmost node, print it & continue.

Ex ⇒



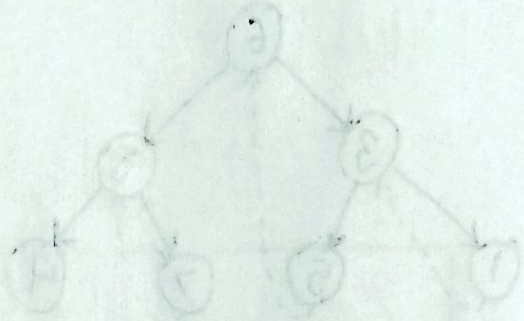
Output ⇒ 1, 3, 5, 6, 7, 8, 9

☆ Threaded Binary tree	Normal Binary tree
① NULL pointers used as thread	① NULL pointer remain NULL.
② No wastage of memory.	② We can't use NULL pointer, wasting of memory.
③ Traversal is easy.	③ Traversal is not easy
④ Complex structure	④ Less Complex
⑤ Insertion & Deletion takes more time	⑤ Take less time

Main Ideas, Questions & Summary:

Library / Website Ref.:-

☆ Insertion in Threaded binary tree ⇒



POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Advantages of threaded binary tree ⇒

(1) Backward traverse is possible

(2) Use less memory

(3) Applicable for most types of binary tree

(4) Node can keep record of it's root

☆ Disadvantages of threaded binary tree ⇒

(1) It makes tree more complex.

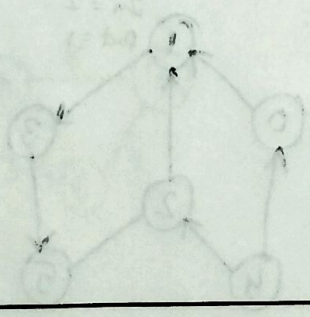
(2) More time consume in Insertion & deletion

(3) More chances of error.

☆ Application of threaded binary tree ⇒

(1) Same as binary tree

(2) Used in search & traverse based work.



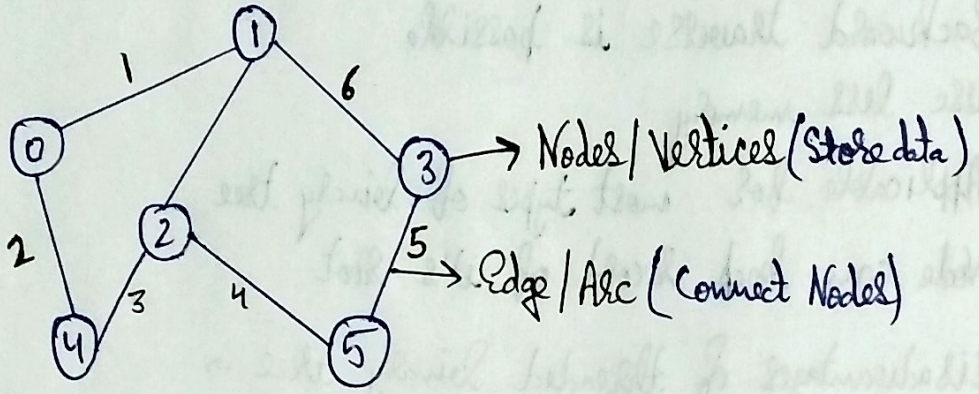
Main Ideas, Questions & Summary:

Library / Website Ref.:-

☆ Graph ⇒

① Non Linear data type

② Collection of nodes connected through edges.



③ Vertices (V) = {0, 1, 2, 3, 4, 5}

Edge (E) = {{0,1}, {0,4}, {1,2}, {1,3}, {2,4}, {2,5}, {3,5}}

④ Type of edge ⇒

(i) Directed edge ⇒ One way connection

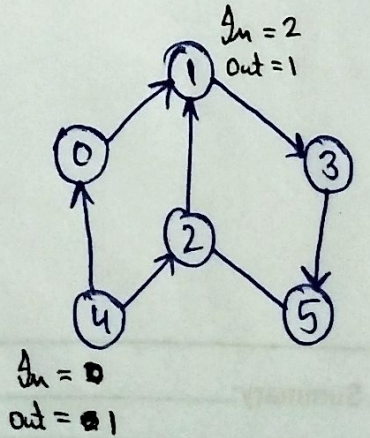
(ii) Unidirected edge ⇒ Two way Connection

⑤ Use in social networks, google maps

☆ Indegree & Outdegree of a node ⇒

↓ No of edges going out ⇄ Change

↓ No of edges coming in



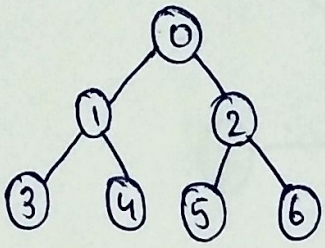
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

* Types of graphs ⇒

① Undirected graph ⇒ All edge is undirected

Ex ⇒



$V = \{0, 1, 2, 3, 4, 5, 6\}$

$E = \{(0,1)(0,2)(1,3)(1,4)(2,5)(2,6)\}$

② Directed graph ⇒ All edge is directed

Ex ⇒

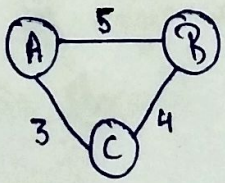
$V = \{0, 1, 2\}$

$E = \{<0,1> <1,0> <1,2>\}$

③ Weighted graph ⇒ Graph where each edge has an associated numerical value, called weight.

⇒ These may be directed or undirected.

⇒ Weight refer to cost of an edge



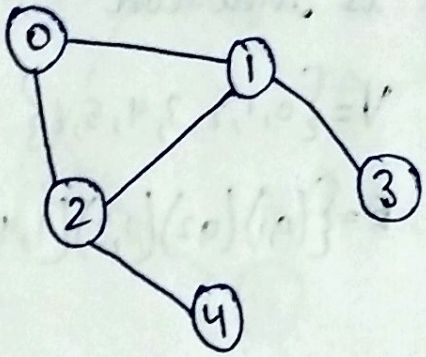
Main Ideas, Questions & Summary:

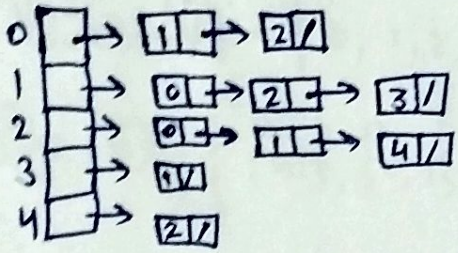
Library / Website Ref.:-

✱ Graph Representation ⇒

① Adjacency Matrix
② Adjacency List

✱ Adjacency List ⇒





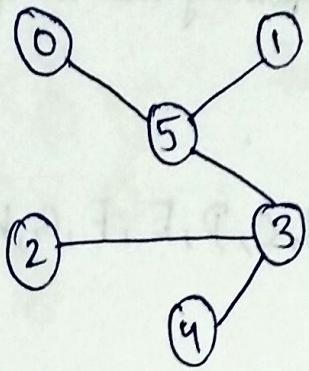
✱ Advantage of Adjacency List ⇒

① Addition of vertex is easy
② Connecting new vertex with existing is easy.
③ Easy to understand

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

★ Adjacency Matrix =>



In cost adjacency matrix, if cost is not given b/w vertex, write '-1'.

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

★ Advantages of Adjacency matrix =>

① Easy to understand, implement

② Adding & removing edges is quick & easy

★ Disadvantage of adjacency matrix =>

① Consume more space

Main Ideas, Questions & Summary:

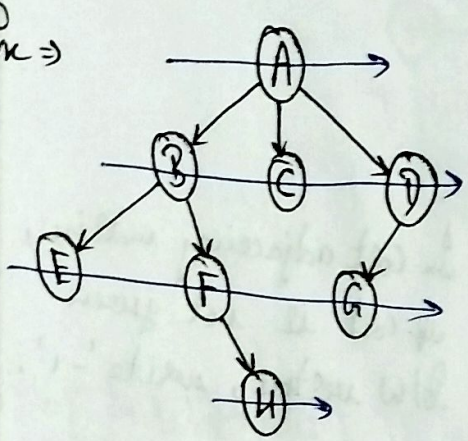
Library / Website Ref.:-

* Graph Traversal ⇒ Visiting all nodes of graph

① Breadth first search (BFS) (Queue DS used)
② Depth first search (DFS) (Stack DS used)

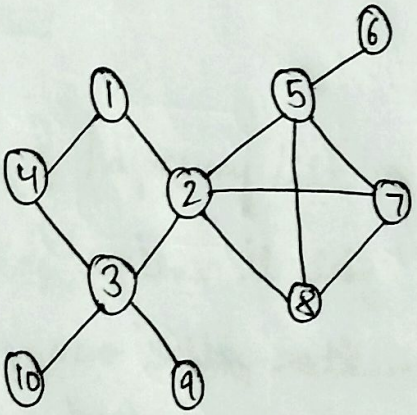
* Breadth First Search (BFS) ⇒ Spanning tree as final result.

Ex ⇒



A, B, C, D, E, F, G, H

Ex ⇒

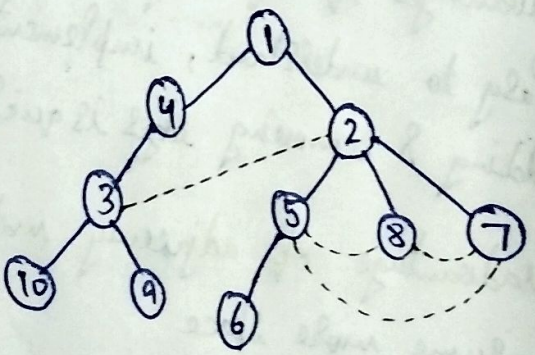


BFS ⇒ 1, 4, 2, 3, 5, 8, 7, 10, 9, 6

(visit) ↑

Q | 1 | 4 | 2 | 3 | 5 | 8 | 7 | 10 | 9 | 6 |

(Explore) ↑



Spanning tree

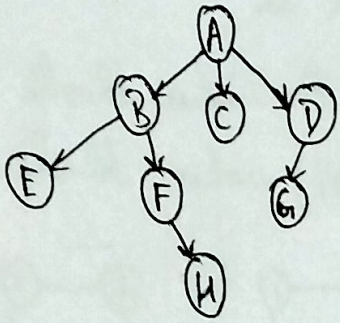
* We can explore any node first
* Any order of visiting

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

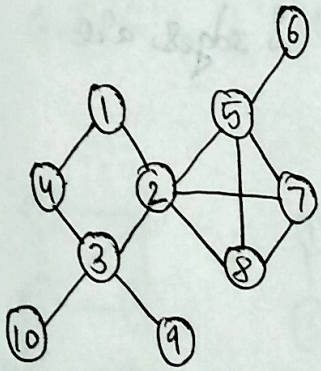
☆ Depth first search (DFS) ⇒

Ex ⇒

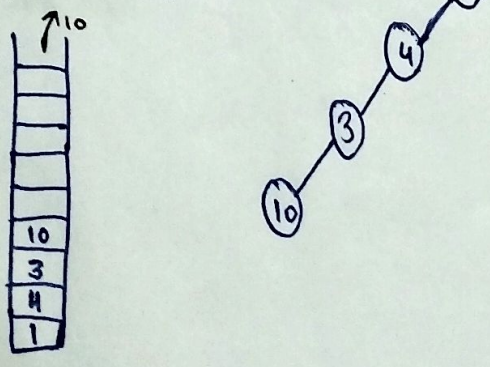


~~Breadth Traversal~~ ⇒ A, B, E, F, H, C, D, G

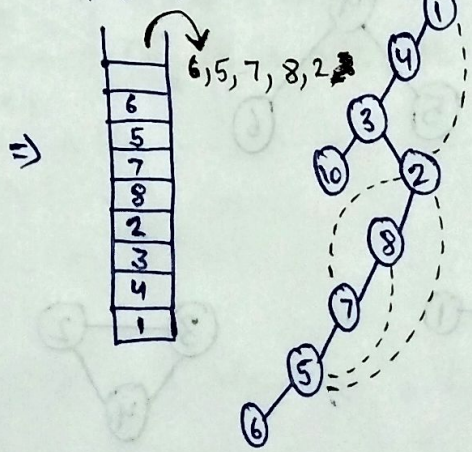
Ex ⇒



⇒ DFS = 1, 4, 3, 10



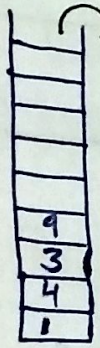
DFS = 1, 4, 3, 10, 2, 8, 7, 5, 6



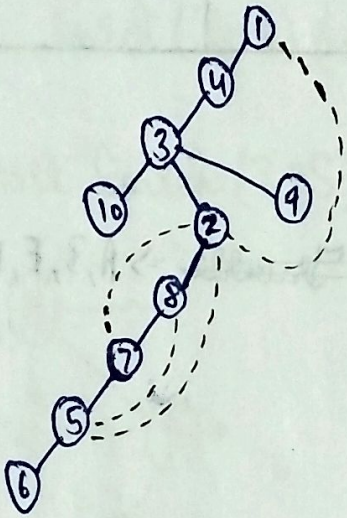
Main Ideas, Questions & Summary:

Library / Website Ref.:-

⇒ DFS = 1, 4, 3, 10, 2, 8, 7, 5, 6, 9

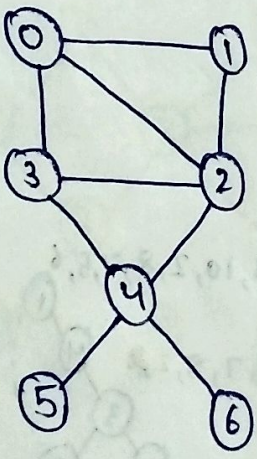


9, 3, 4, 1

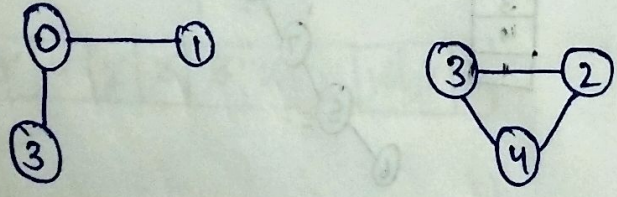


☆ Subgraph ⇒ It is a graph whose vertices & edges are subset of original graph.

Ex ⇒



→ Original graph



Subgraphs

POORNIMA

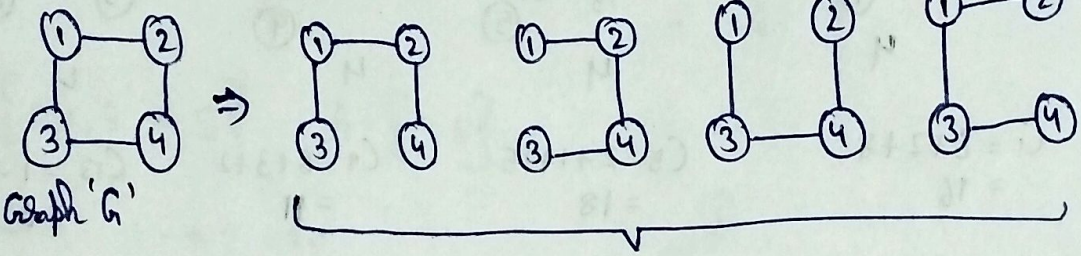
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Spanning Tree ⇒ A connected subgraph 'S' of graph G(V, E) is said to be spanning iff,

(i) 'S' should contain all vertices of 'G'.

(ii) 'S' should contain (|V|-1) edges.

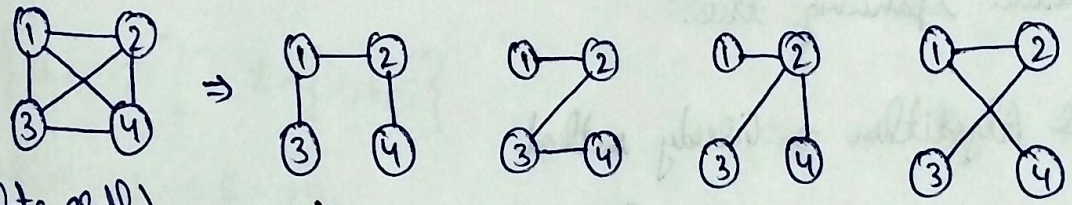
Ex ⇒



Graph 'G'

Subgraph 'S'

Ex ⇒



(Complete graph)

↓

All vertices connected to each other

4 4 4 4 = 16 Subgraph

M^{M-2} M = no. of vertices only for complete graph

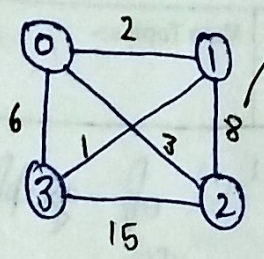
$4^{4-2} \Rightarrow 16$

Main Ideas, Questions & Summary:

Library / Website Ref.:-

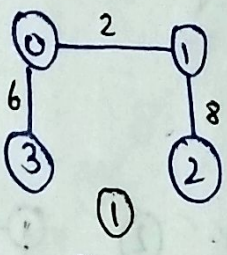
✡ Minimum Spanning tree ⇒

Ex ⇒

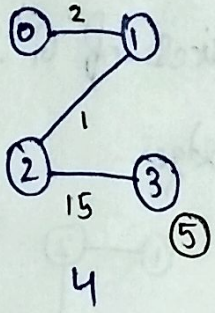


→ Cost (Main graph)

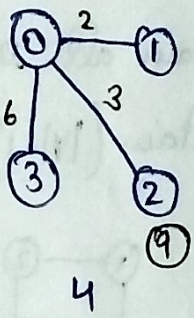
Subgraph ⇒



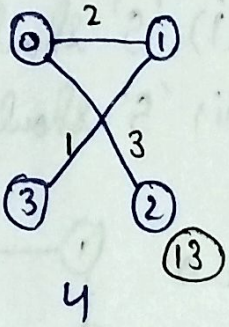
$C_1 = 6+2+8$
$= 16$



$C_5 = 2+1+15$
$= 18$



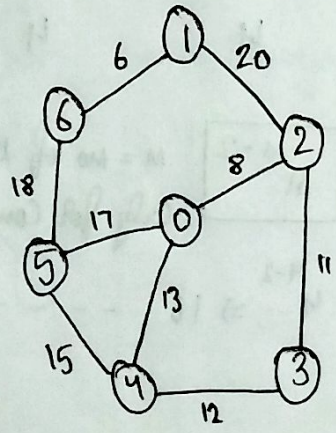
$C_9 = 6+3+2$
$= 11$



$C_{13} = 1+2+3$
$= 6$

- Find cost for others, whichever tree give less cost is minimum spanning tree.

✡ Prims Algorithm ⇒ Greedy method



⇒ Take adjacent vertic with less weight

⇒ Edges are continous in each step

POORNIMA

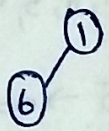
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

⇒ $V = \{0, 2, 3, 4, 5, 6\}$ $A = \{1\}$

↳ Not included in MST ↳ Included in MST

• Options ⇒ $1 \to 6 = 6$ ✓

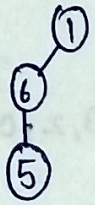
$1 \to 2 = 20$



⇒ $V = \{0, 2, 3, 4, 5\}$ $A = \{1, 6\}$

• Options ⇒ $1 \to 2 = 20$

$6 \to 5 = 18$ ✓

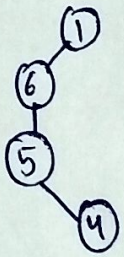


⇒ $V = \{0, 2, 3, 4\}$ $A = \{1, 6, 5\}$

• Options ⇒ $1 \to 2 = 20$

$5 \to 0 = 17$

$5 \to 4 = 15$ ✓

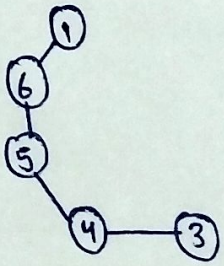


⇒ $V = \{0, 2, 3\}$ $A = \{1, 6, 5, 4\}$

• Options ⇒ $1 \to 2 = 20$

$4 \to 0 = 13$

$4 \to 3 = 12$ ✓



Main Ideas, Questions & Summary:

Library / Website Ref.:-

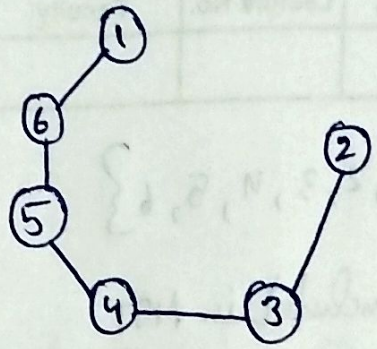
⇒ $V = \{0, 2\}$ $A = \{1, 6, 5, 4, 3\}$

• Options ⇒ $1 \to 2 = 20$

$5 \to 0 = 17$

$4 \to 0 = 13$

$3 \to 2 = 11$ ✓

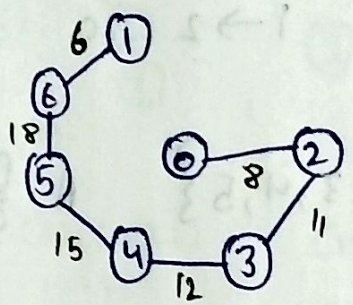


⇒ $V = \{0\}$ $A = \{1, 6, 5, 4, 3, 2\}$

• Options ⇒ ⟹ $5 \to 0 = 17$

$4 \to 0 = 13$

$2 \to 0 = 8$ ✓



⇒ $V = \{\}$ $A = \{1, 6, 5, 4, 3, 2, 0\}$

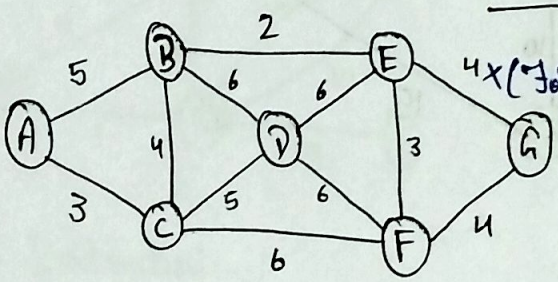
⇒ Cost = 6 + 18 + 15 + 12 + 11 + 8

= 70 (MST)

POORNIMA

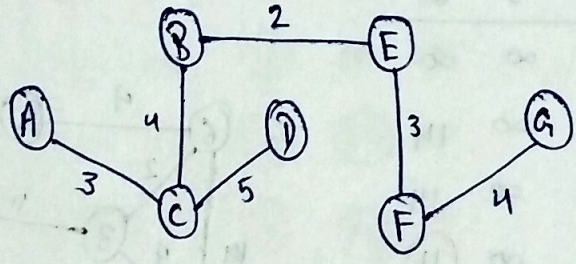
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Kruskal Algorithm ⇒ Greedy Method, Always take less weighted edge & don't form loop / cycle



4 x (Form cycle)

⇒ Edges may be discontinue in starting steps

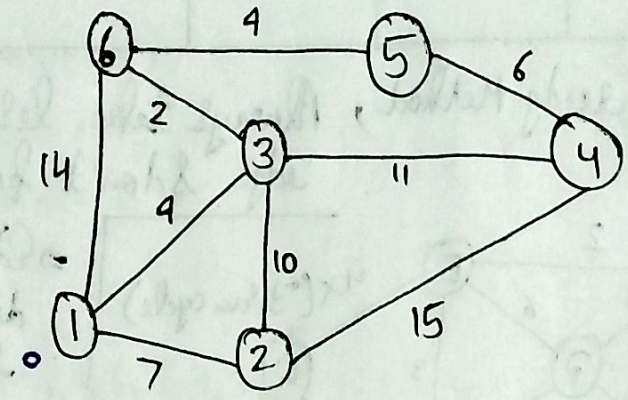


⇒ Cost = 21

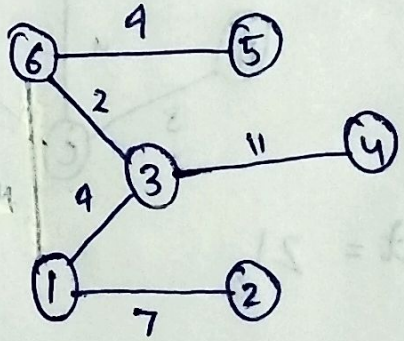
Main Ideas, Questions & Summary:

Library / Website Ref.:-

✡ Dijkstra's Shortest path ⇒



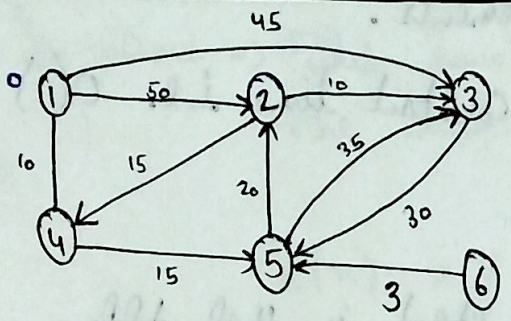
Source 1	Destination 2	3	4	5	6
	∞	∞	∞	∞	∞
1, 2	(7)	9	∞	∞	14
1,2,3	(7)	(9)	22	∞	14
1,2,3,6	(7)	(9)	20	∞	(11)
1,2,3,6,4	(7)	(9)	(20)	20	(11)
1,2,3,6,4,5	(7)	(9)	(20)	(20)	(11)



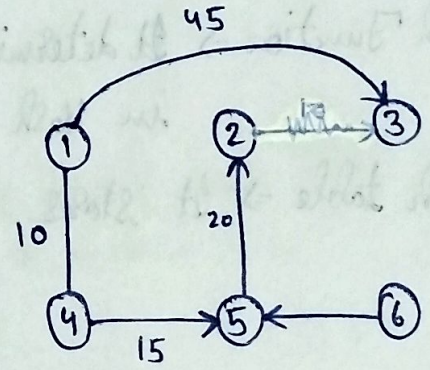
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

Q⇒



Source 1	Destination 2	3	4	5	6
	∞	∞	∞	∞	∞
1, 4	50	45	(10)	∞	∞
1, 4, 5	50	45	(10)	(25)	∞
1, 4, 5, 2	(45)	45	(10)	(25)	∞
1, 4, 5, 2, 3	(45)	(45)	(10)	(25)	∞
~~1, 4, 5, 2,~~					



★ Drawback of Dijkstra shortest path ⇒

① It may be correct or not correct, when weight is in '-Ve'.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

✡ Hashing ⇒ Process of mapping keys & values into the hash table by using a hash function.

⇒ Done for faster access of elements.

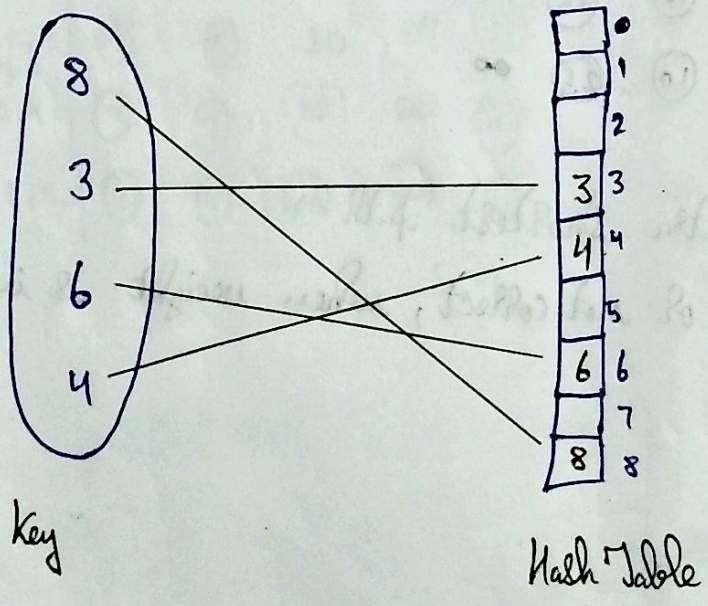
⇒ Search of element is done in constant time i.e. O(1) time

✡ Components of Hashing ⇒

① Key ⇒ Data which is to be stored in Hash table.

② Hash Function ⇒ It determines an index or location of the key in Hash table.

③ Hash table ⇒ It stores the data with unique index.



POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Hash function ⇒ This function is used to give the index to an key in hash table.

$$h(x) = x \,\%\, \text{size of hash table}$$

↳ key value

☆ Types of Hash functions ⇒

① Division Method ⇒ Divide key value by size of hash table & use remainder as a index.

$$h(x) = x \,\%\, \text{size of hash table}$$

② Mid Square method ⇒ To Square of key value & then choose the middle digits as a index.

Ex ⇒ $x = 12$, Size = 10

⇒ $x^2 = 144$, 4 is index

$x = 87431$, Size = 1000

$x^2 = 7644179761$, 417, 479 anyone will be index

③ Folding Method ⇒ Divide key value in no of parts. Each part have same no of digits except last part. Last part have lesser digits.

Ex ⇒ 12345

= 12 + 34 + 5

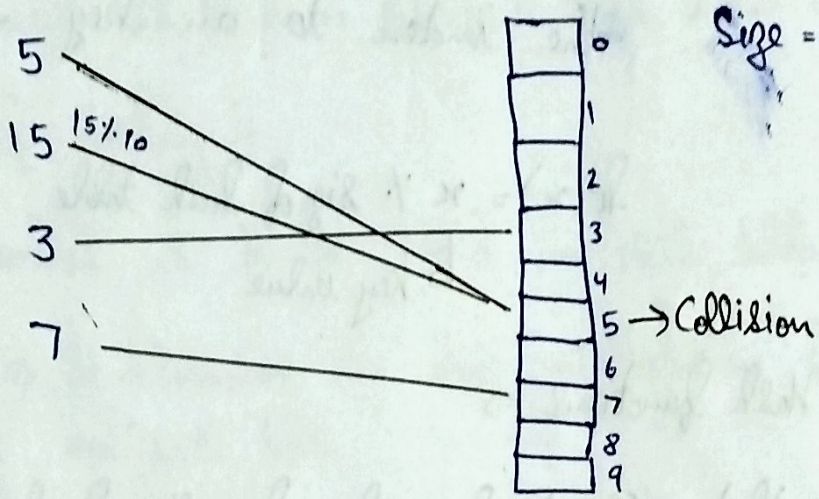
= 51 is index

Main Ideas, Questions & Summary:

Library / Website Ref.:-

✡ Collision in Hashing ⇒ The condition, when different keys get the same index or location in hash table is called collision.

Ex ⇒

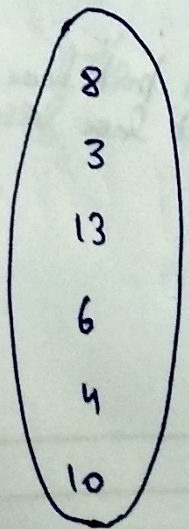


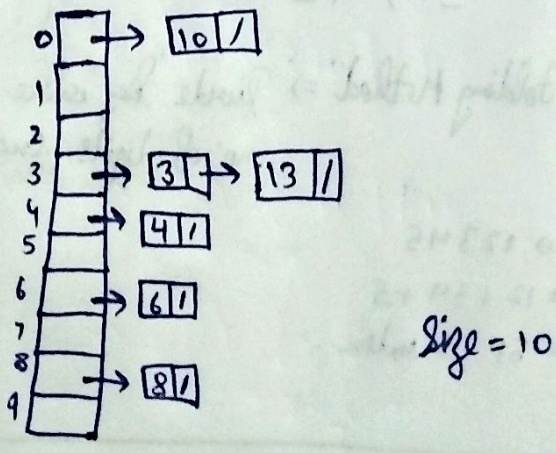
✡ Collision Resolution ⇒

① Open addressing (array method)
   (a) Linear probing
   (b) Quadratic probing

② Separate chaining (Linked list method)

✡ Chaining method ⇒





POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Linear Probing ⇒ If different keys have same index or location, than we apply new hash function.

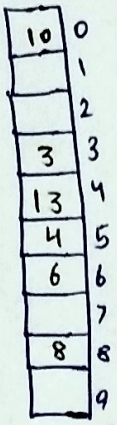
$$h(x) = x \,\%\, \text{size}$$

$$h'(x) = [h(x) + i] \,\%\, \text{size}$$

↳ 0, 1, 2 ---

Ex ⇒

8
3
13
6
4
10



size = 10

- For 13 ⇒ $h(x) = 13 \,\%\, 10$
  $= 3$ [key exist]
  ⇒ $h'(x) = [3+0] \,\%\, 10$
  $= 3$ [key exist]
  ⇒ $h'(x) = [3+1] \,\%\, 10$
  $= 4$ ✓

- For 4 ⇒ $h(x) = 4 \,\%\, 10$
  $= 4$ [key exist]
  ⇒ $h'(x) = [4+0] \,\%\, 10$
  $= 4$ [key exist]
  ⇒ $h'(x) = [4+1] \,\%\, 10$
  $= 5$ ✓

Main Ideas, Questions & Summary:

Library / Website Ref.:-

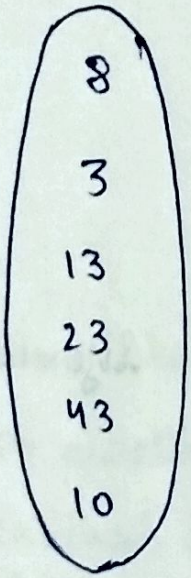
★ Quadratic Probing ⇒ Same as Linear probing but have different hash function

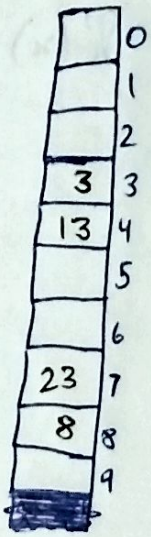
$$h(x) = x \,\%\, \text{size}$$

$$h'(x) = [h(x) + i^2] \,\%\, \text{size}$$

↳ 0, 1, 2 ---

Ex ⇒





• For 13 ⇒ $h(x) = 13 \,\%\, 10$
$= 3$

$\Rightarrow h'(x) = [3 + 0^2] \,\%\, 10$
$= 3$

$\Rightarrow h'(x) = [3 + 1^2] \,\%\, 10$
$= 4$ ✓

• For 23 ⇒ $h(x) = 23 \,\%\, 10$
$= 3$

$\Rightarrow h'(x) = [3 + 0^2] \,\%\, 10$
$= 3$

$\Rightarrow h'(x) = [3 + 1^2] \,\%\, 10$
$= 4$

$\Rightarrow h'(x) = [3 + 2^2] \,\%\, 10$
$= 7$ ✓

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

★ Double Hashing ⇒

$h_1(x) = x \% \text{ size}$

$h_2(x) = 8 - (x \% 8)$

$[h_1(x) + i\, h_2(x)] \% \text{ size}$

Ex ⇒ Keys = 20, 34, 45, 70, 56

Size = 11

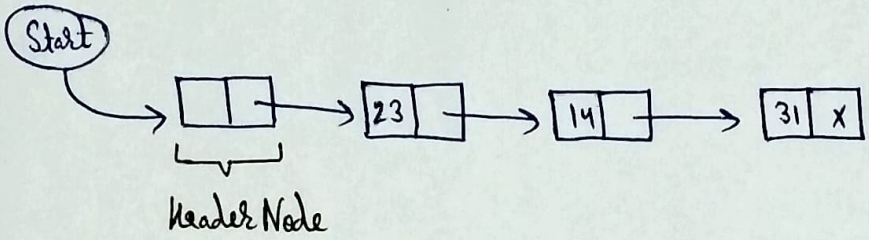
Main Ideas, Questions & Summary:

Library / Website Ref.:-

POORNIMA

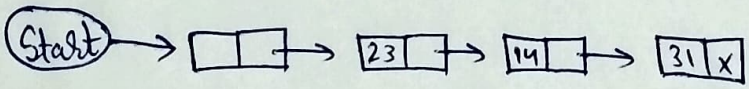
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

* Header Linked List ⇒ It contains a special node called the header node at the very beginning of linked list. It is an extra node kept at the front of a list.

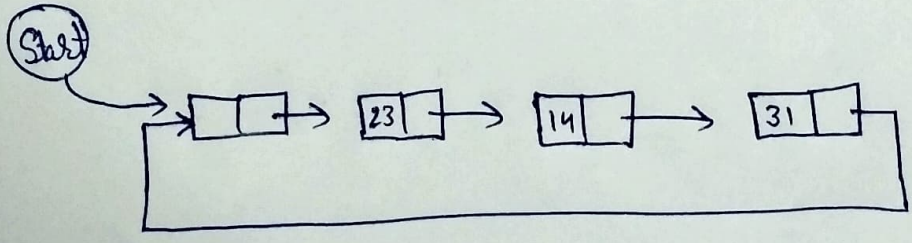


- Does not represent any item
- Store global information like, no. of element in list

① Grounded Header Linked List ⇒ Last node contain NULL pointer.



② Circular Header Linked List ⇒ Last node points back to header node.



Main Ideas, Questions & Summary:

Library / Website Ref.:-