Royal Education Society's

College of Computer Science and Information Technology, Latur.

Department of Computer Science

Academic Year (2022-23)

Choice Based Credit System (CBCS Revised)

Class/Semester: **B.Sc.(CS) SY SEM-III**

Name of Paper: **Data Structure and Algorithms.**

Prepared by: **Mr. A. M. Saudagar.**

Question Paper

Q.1 Attempt any FIVE of the following (3 Marks each) 15

a) Write linear array.
b) Explain linked list.
c) Write priority queue.
d) Explain binary tree.
e) Write graph theory terminology.
f) Explain stack.
g) Write 2- trees.

Q. 2 Attempt any three of the following (5 Marks each) 15

a) Write an algorithm for Traversing linear array.
b) Write insertion sort with example.
c) Writearray representation of stack.
d) Explain linked representation of binary tree.
e) Write algorithm to search a ITEM from linked list in unsorted linked list.

Q. 3 Attempt any three of the following (5 Marks each) 15

a) Write Basic Terminology, elementary data organization.
b) Write an algorithm for bubble sort method.
c) Write stack operations.
d) Explain types of binary tree.
e) Write algorithm to insert a ITEM at the beginning of the linked linked list.

Q. 4 Attempt any three of the following (5 Marks each) 15

a) Write queue operations.
b) Write algorithm to delete ITEM from linked list.
c) Explain traversing of binary tree.
d) Write representation of linked list in memory.
e) Sequential Representation of graph

Q .5 Short notes on any three of the following (5 Marks each) 15

a) Algorithm complexity.
b) Memory allocation , garbage collection.
c) Recursion.
d) Deque.
e)Header node.

Modal Answer Paper

Q.1 Attempt any FIVE of the following (3 Marks each) 15

a) Write linear array.

Ans: Linear Array :-

- A linear array is a list of finite number 'n' of similar type of data elements such that:
  - The element of an array are referenced by index set consisting of 'n' consecutive numbers.
  - The elements of an array are stored (placed) in successive memory location.
  - A number 'n' is called length or size of the array.
- Suppose the index set consists of the integers 1, 2, 3, ……, n then the length or the number of elements of an array can be obtained from the index set by the formula:

length = UB – LB + 1 OR length = UB (if LB = 1)

- Here 'UB' is the largest index called upper bound and 'LB' is the lower bound i.e. smallest index of an array.

DATA

LB	1	30
	2	15
	3	35
UB	4	40
	5	50

length = 5 – 1 + 1

= 4 + 1

= 5

30	15	35	40	50
0	1	2	3	4

lengt = 4 – 0 + 1

= 4 + 1

= 5

- The element of an array may be denoted by the notation :
  1. Subscript notation as A1, A2, A3, ……, An.
  2. Parenthesis notation as A(1), A(2), A(3), ……., A(n).
  3. By the bracket notation as A[1], A[2], A[3], ……., A[n]

---

b) Explain linked list.

Ans: Linked List :-

- A linked list is a linear collection of data elements, called 'nodes'.
- Here, the linear order is obtained by using pointers.
- A linked as also called as 'one–way–list'.
- Every node of linked list is get divided into two fields.

1. First field is INFO which is used to store information of the element.
2. Second field is LINK or next pointer field which is used to store address of the next node in the list.

- Following figure shows a linked list with four nodes.

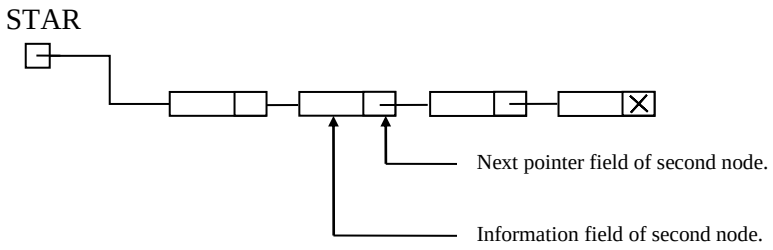


Fig. Linked List with 4 nodes.

- It is clear that, in above fig. each node is pictured with two fields.
- The left field use the actual information, and right field use store the address of the next node.
- Arrow's are used to show the order of the nodes.
- The next pointer or LINK field of the last node contain a invalid address called 'null pointer'.
- The null pointer (X) is used to shows the end of the list.
- The linked list has also a list pointer variable called START which contains address of first node in the list.
- The starting address o the list which is sorted in START is used to traverse the whole linked list.
- Here if the linked list has no any node then such a list is called 'null list' or 'empty list' And it is denoted by the null pointer assigning to START i.e. START = NULL.

c) Write priority queue.

Ans: A priority queue is a collection of element such that each element has been assign a priority and such that the order in which element are deleted and processed comes from the following rules.

1. An element of higher priority is processed before any element of lower priority.
2. Two elements with the same priority are processed according to the order in which they were added to the queue.

- A priority queue is a time showing system.

- Programs of high priority are processed 1st and programs with the same priority from a standard queue.
- There are various ways of maintaining a priority queue in memory.
- We will discuss two of them.
    1. Using one way list.
    2. Using multiple queues.

1. Using one way list :-

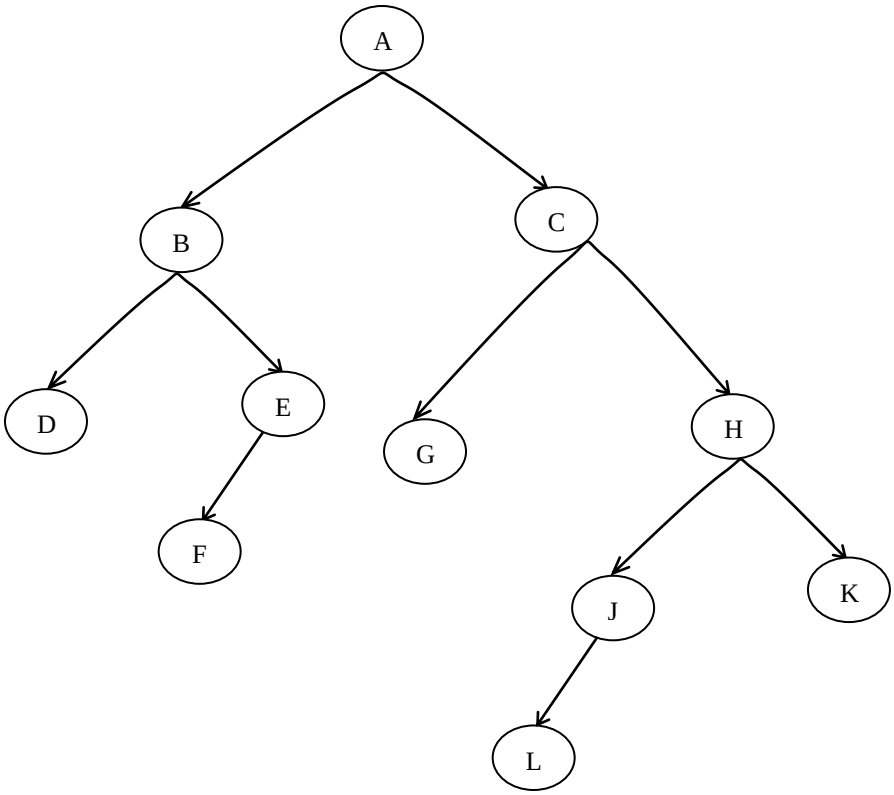
- One way to maintain a priority queue in memory by means of one way list as follow,
    a. Each node in the list with contain three ITEMs of information. An information field INFO, a priority number PRN and a link number LINK.
    b. A node *x* precids node by in the list, when *x* has higher priority than by or when both have the same priority but *x* was added to the list before *y*.
- This means that the order in the one way list corresponds to the order of the priority queue.
- Priority number in operate in the visual way.
- The lower priority number, the highest priority number.

d) Explain binary tree.

Ans: A binary tree T is defined as a finite set of elements, called nodes, such that:

a) T is empty (called null tree or empty tree), or
b) T contains a distinguished node R, called the scoot of T and the remaining nodes of T form an ordered pair of disjoint binary trees T_1 & T_2.

- If binary tree T does contain a root R, then two trees T_1 and T_2 are called left and right sub trees of R respectively.
- If T_1 is non-empty then its root is called left successor of R.
- Similarly, if T2 is non-empty then its root is called the right successor of R.
- Consider following binary tree:



- The binary tree T represents as follows,
    i. T consists of 11 nodes, represented by the letters A to L, excluding I.
    ii. The root node of T is the node A at the top of diagram.
    iii. A left downward slanted line from a node N indicates a left successor of node N, and right downward slanted from N indicated right successor of N.
- In above tree observe that,
    a) B is a left successor and C is a right successor of node A.
    b) The left sub tree of root A consists of the nodes B, D, E & F and right sub tree of A consists of the nodes C, G, H, J, K & L.
- Any node N in binary tree T may have 0, 1 & 2 successors. The nodes with no successor are called Terminal nodes.
- In above tree T, the nodes A, B, C, H has two successors & the node E, J, has one successors and the node D, F, G, K & L has no successor. Hence, these are terminal nodes.

e) Write graph theory terminology.

Ans: A graph is a diagram of points and lines connected to the points. It has at least one line joining a set of two vertices with no vertex connecting itself. The concept of graphs in graph theory stands up on some basic terms such as point, line, vertex, edge, degree of

vertices, properties of graphs, etc. Here, in this chapter, we will cover these fundamentals of graph theory.

Point

A point is a particular position in a one-dimensional, two-dimensional, or three-dimensional space. For better understanding, a point can be denoted by an alphabet. It can be represented with a dot.

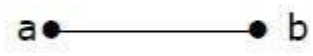
Example

Here, the dot is a point named ‘a’.

Line

A **Line** is a connection between two points. It can be represented with a solid line.

Example



Here, ‘a’ and ‘b’ are the points. The link between these two points is called a line.

Vertex

A vertex is a point where multiple lines meet. It is also called a **node**. Similar to points, a vertex is also denoted by an alphabet.

Example

Here, the vertex is named with an alphabet ‘a’.

Edge

An edge is the mathematical term for a line that connects two vertices. Many edges can be formed from a single vertex. Without a vertex, an edge cannot be formed. There must be a starting vertex and an ending vertex for an edge.

Example

Here, 'a' and 'b' are the two vertices and the link between them is called an edge.

Graph

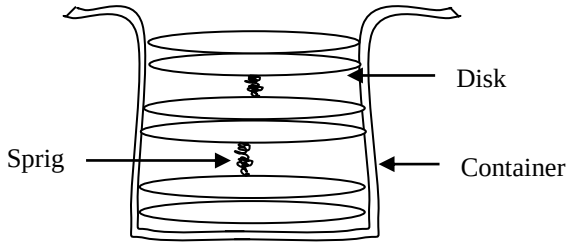
A graph 'G' is defined as G = (V, E) Where V is a set of all vertices and E is a set of all edges in the graph.

Example 1

In the above example, ab, ac, cd, and bd are the edges of the graph. Similarly, a, b, c, and d are the vertices of the graph.

f) Explain stack.

Ans: A stack is a linear list in which insertion and deletion (Push & Pop) of an element can take place only at one end called the Top. Thus structure is called LIFO. The structure is similar in its operation to a stack of dishes on a spring system as shown in following figure.



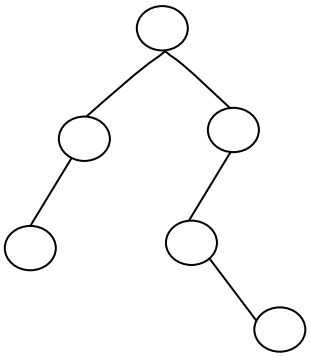
As container is open at top side, new dishes are get interested only at the top of this stack and similarly dishes can be deleted only from the top of this stack.

g) Write 2- trees.

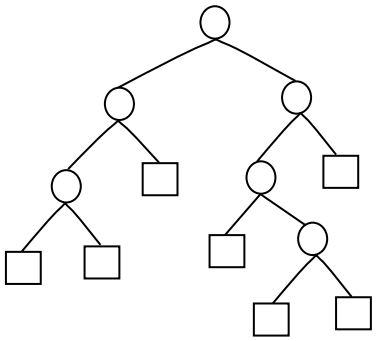
Ans: A binary tree T is said to be an extended binary tree or 2-trees.

- If each node N has either 0 or 2 children.

- The nodes with 2 children are called internal nodes and the nodes with zero children are called external nodes.
- In the diagram, these nodes are differentiated by drawing circles for internal nodes and square for external nodes.
- Ex.: Consider following binary tree.



- The conversion of above binary tree into an extended binary tree or 2-tree as follows.



- This tree is an extended binary tree or 2-tree because the nodes in original tree are here internal nodes, and new nodes are the extended node.

Q. 2 Attempt any three of the following (5 Marks each) 15

a) Write an algorithm for Traversing linear array.

Ans: Following is an algorithm which traverses a linear array LA.

Algorithm 2.1 : TRAVERS LA (LA, LB, UB, k)

Let, LA is linear array with 'n' element. k is a counter, variable LB is lower bound, UB is upper bound.

This algorithm perform traversing linear array LA.

Step 1. : [Initialize counter k]

set k := LB

Step 2. : Repeat step 3 & 4

while (k <= UB)

Step 3. : Apply PROCES to LA[k]

Step 4. : [Increase counter k by 1]

set k := k + 1

[End of step 2 loop]

Step 5. : [Finished] Exit.

b) Write insertion sort with example.

Ans: Suppose an array DATA with N elements DATA[1], DATA[2], ……., DATA[N] is in memory.

- The insertion sort algorithm scan DATA from DATA[1] to DATA[N], inserting each element DATA[PASS] into it's a proper position in the previously sorted sub array DATA[1], DATA[2], …….., DATA[PASS–1], i.e. :

  Pass 1 : Keep DATA[1] as it is which get sorted automatically while the execution.

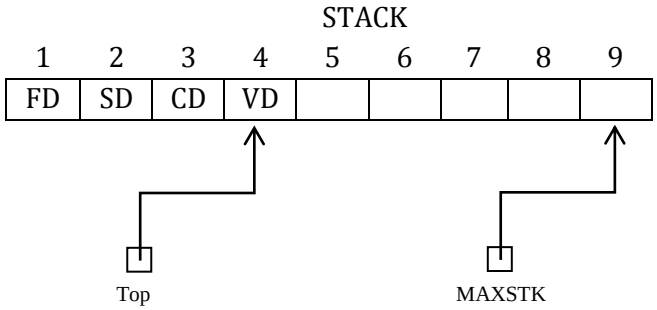
  Pass 2: The insert DATA[2] either before or after DATA[1], so that DATA[1] <= DATA[2].

  Pass N: Insert DATA[1] into its proper place. If DATA[1], DATA[2], …. , DATA[N–1]. So that DATA[N – 1] <= DATA[N] is sorted.
- In this way after making given list become sorted list in ascending order.

c) Write array representation of stack.

Ans: Stack are representation is the computer memory by a one way linked list or linear array.

- By default stack is represented using a linear array say STACK.
- The pointer variable TOP and MAXSTEK used to indicate the TOP element of the stack and the maximum length of stacks.
- The condition TOP = 0 or TOP = NULL will indicate that the stack is empty.
- Ex.: Consider stack of maximum length 9 with 4 elements.



- In above fig. TOP = 4 indicate that stack has four elements and MAXSTK = 9, which gives the maximum length of this stack.
- Here the location 5, 6, 7 indicate blank places for the new ITEM.

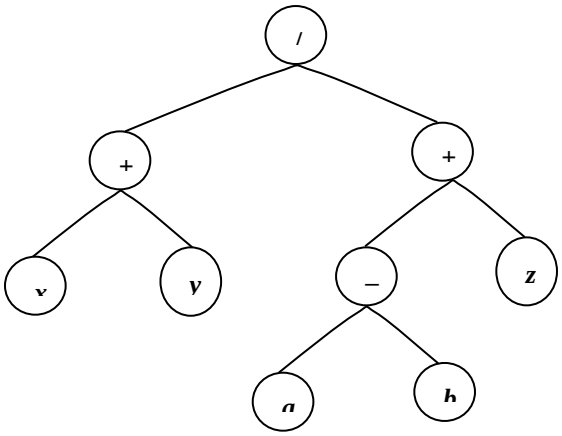
d) Explain linked representation of binary tree.

- Ans: Consider a binary tree T. By default T will be represented in memory using three parallel arrays INFO, LEFT and RIGHT.

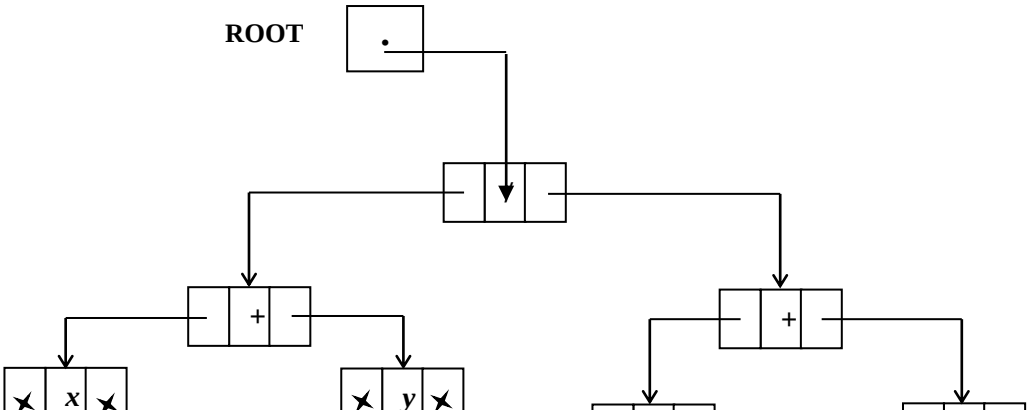
- Here, pointer variable ROOT and AVAIL are used to indicate the beginning locations of the root of tree T and first free location of the AVAIL list.
- If node N present at location L in a given tree T then:
    i) INFO[L] contains the data at the node N.
    ii) LEFT[L] contains the location of the left child of node N.
    iii) RIGHT[L] contains the location of the right child of node N.
- In a tree T, if any sub tree empty, then the corresponding pointer field will contain null values.
- If the given tree itself is empty then ROOT will contain the null values.
- Ex.: Consider following binary tree and its linked representation and memory representation is given as follows,

[(x + y) / ((a – b) + z)]

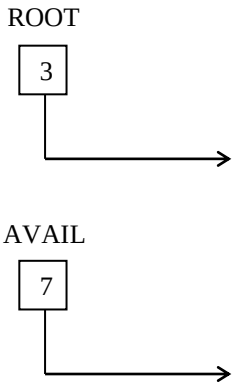
i) Binary Tree:



ii) Linked Representation:



iii) Memory Representation:



	INFO	LEFT	RIGHT
1	+	5	8
2			
3	/	1	10
4			
5	*x*	0	0
6	–	9	11
7			
8	y	0	0
9	a	0	0
10	+	6	12
11	b	0	0
12	z	0	0

e) Write algorithm to search a ITEM from linked list in unsorted linked list.

Ans: Algorithm 4.4 : SEARCHULL (LIST,INFO, LOC, LINK, START, PTR, ITEM)

Let given linked list is unsorted which stored in memory INFO variable, LINK is a linked list and START is a starting pointer variable. This algorithm search a location LOC of the node where ITEM appear or set LOC = NULL. Here PTR is a local pointer variable which indicate the node which is going to process.

Step 1. : [Initialize pointer PTR]

set PTR := START

Step 2. : Repeat step 3

while PTR ≠ NULL

Step 3. : [Is ITEM present at node PTR?]

If ITEM = INFO[PTR], then

Write : “Successful search” and

Set LOC= PTR and Exit.

Else

[Now point to the next node]

Set PTR = LINK[PTR]

[End of if structure]

[End of step 2 loop]

Step 4. : write "Unsuccessful search" and

Set LOC = NULL

Step 5. : [Finished] Exit.

Q. 3 Attempt any three of the following (5 Marks each) 15

a) Write Basic Terminology, elementary data organization.

Ans: Basic Terminology (Elementary data organization) :-

(a) Data.

(b) Data item.

i) Group item.

ii) Elementary data item.

(c) Organization of data.

i) Attribute

ii) Entity

iii) Entity set.

(d) Information.

i) Field

ii) Record

iii) File

(e) Primary key.

(f) Classification of record.

i) Variable length

ii) Fixed length.

(a) Data :- Data are simply value or set of value.

Ex.: "LATUR", 8.

(b) Data item :- Data item is single unit of values.

Ex.: "LTUR".

Further data item are get divided into two parts.

i) Group item

ii) Elementary item.

i) Group item :- Data items that are divided into sub items are called group item.

Ex.: Employee Name : More Ram Hari

Employee name get divided into three items as first name, middle name and last name.

ii) Elementary item :- Data item that are not divided into sub item are called elementary item.

Ex.: 101 – 05 – 2013.

Social security number is treated as a single item.

(c) Organization of data :-

Collection of data are frequently organized into hierarchy of fields, records and files.

- The above concept can be more cleared using following additional terminology i.e. attribute, entity and entity set.

i) Attribute :- A attribute is a specified memory area used to store data.

ii) Entity :- A entity is something that has retain attributes or properties which may be assigned value. Here assigned value may be either numeric and non-numeric.

Ex.:	Attribute	Roll No.	Name	Grade
	Entity Value	01	Ajay	A

iii) Entity Set :- Entity is with similar attribute from an entity set. Each attribute of an entity set as a range of value, the set of all possible values could be assigned to the particular or attribute.

Ex.: All the employee in an organization.

(d) Information :- A meaningful or processed data at particular attribute is called as "information". The way that data are organized into the hierarchy of fields, records, files, shows the relationship between attributes, entities and entity set as:

i) Field :- A field is a single elementary unit of information representing an attribute of an entity.

ii) Record :- A record is a collection of field values of a given entity.

iii) File :- A file is the collection of records of the entities in a given entity set.

(e) Primary key :- Every record in a file may contain many fields item but the value in certain field may determine the record in the file uniquely. Such a field K is called a primary key, and the values K_1, K_2, K_3, …. K_n in such a field are called key values or keys.

(f) Classification of Records :-

Records may also be classified recording to the length.

- A file can have fixed length record or variable length record.
- In fixed length records, all the records contain the same data items with the same amount of space assign to each data item.
- In variable length records file records may contain different length.

  Ex.: Student records usually have variable length, since different student take different number of courses.
- Usually variable length records have a minimum and maximum length.

b) Write an algorithm for bubble sort method.

Ans: Following is an algorithm for Bubbled Sort Method :

Algorithm 2.4 : BUBBLESRT (DATA, N, PASS, PTR, TEMP)

Let DATA is linear array unsorted with N element. PASS, PTR, TEMP are the local variables used to count the PASS, point element location and to store value temporary respectively.

This algorithm perform sort the given list in ascending order.

Step 1: Repeat through step 5

For PASS =1 to N – 1

Step 2 : [Initialize pointer PTR]

Set PTR = 1

Step 3 : Repeat through step 5

While PTR <= N – PASS

Step 4 : [Is element at PTR & PTR + 1 in out of order?]

If DATA[PTR] > DATA[PTR + 1], then

[Interchange (PTR) & DATA(PTR + 1)]

TEMP = DATA[PTR]

DATA[PTR] = [PTR + 1]

DATA[PTR + 1] = TEMP

[End of if structure]

Step 5 : [Increase PTR by 1]

set PTR = PTR + 1

[End of step 3 loop]

[End of step 1 loop]

Step 6 : [Finished] Exit.

c) Write stack operations.

Ans: There are two operations are used to stack.

1. Push.
2. POP.

1. Push :-

- This operation is used to add an ITEM on to a stack.
- To push an ITEM one must check there is any space in the stack for new ITEM.
- Their no available space, then such condition is handle by printing message "OVERFLOW",i.e.If TOP = MAXSTK.
- Following is an algorithm to perform push operation onto a stack.

Algorithm 4.1 : PUSH(STACK, ITEM, TOP, MAXSTK)

Let STACK is a linear array which is used to store a list of elements. Here TOP and MAXSTK indicates the TOP element location and maximum length location of

STACK. This algorithm push an ITEM of information on to a STACK.

Step 1. : [Is stack overflow?]
If TOP := MAXSTK, then
Write "OVERFLOW" and
Exit
[End of if structure]

Step 2. : [Increase TOP by 1]
Set TOP := TOP + 1

Step 3. : [Insert ITEM in new TOP position]
Set STACK[TOP] := ITEM.

Step 4. : [Finished] Exit.

2. POP :-

- This operation is used to POP (remove) an ITEM from the TOP (stack).
- While deleting an ITEM from the stack, one must check there is any element is present in stack or not.
- Here if given stack is empty i.e. TOP = NULL then it handled by printing a message, "UNDERFLOW".
- Following is an algorithm to POP (delete) an ITEM from the TOP of the stack.

Algorithm 4.2 : POP(STACK, ITEM, TOP, MAXSTK)

Let STACK is a linear array which is used to store a list of elements. Here TOP and MAXSTK indicates the TOP element location and maximum length location of STACK. This algorithm perform pop an ITEM of information from a STACK.

Step 1. : [Is stack Underflow?]
If TOP = NULL, then:
Write "UNDERFLOW" and
Exit
[End of if structure]

Step 2. : [Assign TOP element to ITEM]
Set ITEM: = STACK[TOP]

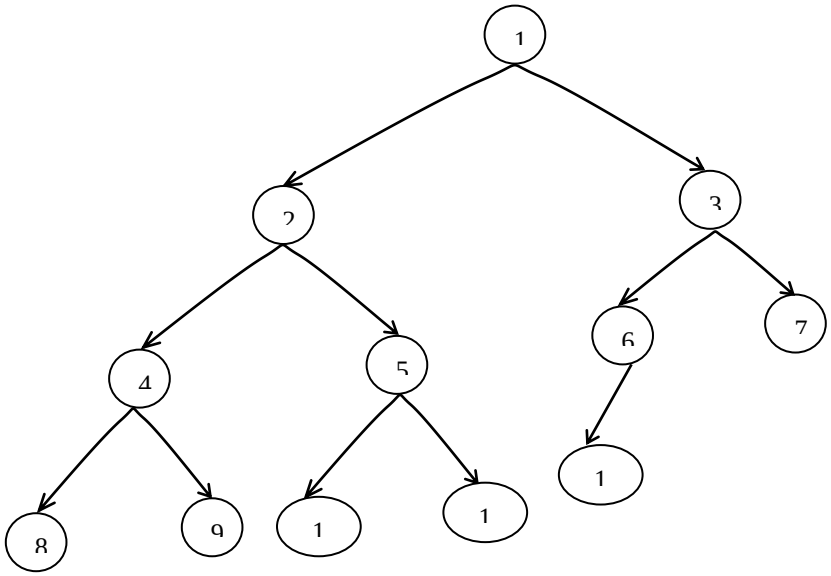
Step 3. : [Decrease TOP by 1]
Set TOP = TOP - 1

Step 4. : [Finished] Exit.

d) Explain types of binary tree.

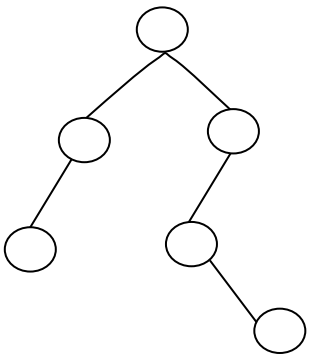
Ans: There are two types of Binary Tree :

1. **Complete Binary Tree :**
   - The tree T is said to be complete binary tree if all its levels, except possibly the last, have the maximum number of possible nodes, and if all the nodes at the last level appear as left.

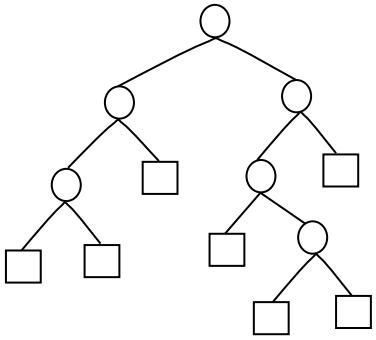


   - Consider above complete binary tree T_{12} with 12 nodes.
   - In complete binary tree, children and parent of any node N can be determined as follows,
     1) A left child of node N is $2 \times N$.
     2) A right child of node N is $2 \times N + 1$.
     3) A parent of node N is integer of N / 2.
   - Ex.: Consider a node 5 then its left, right child and parent may be determined as follows,
     1) A left child of node 5 is $2 \times 5 = 10$.
     2) A right child of node N is $2 \times 5 + 1 = 11$.
     3) A parent of node 5 is integer $[5/2] = 2$.

2. **Extended Binary Tree : 2 Trees :**
   - A binary tree T is said to be an extended binary tree or 2.
   - If each node N has either 0 or 2 children.
   - The nodes with 2 children are called internal nodes and the nodes with zero children are called external nodes.
   - In the diagram, these nodes are differentiated by drawing circles for internal nodes and square for external nodes.
   - Ex.: Consider following binary tree.



- The conversion of above binary tree into an extended binary tree or 2-tree as follows.



- This tree is an extended binary tree or 2-tree because the nodes in original tree are here internal nodes, and new nodes are the extended node.

e) Write algorithm to insert a ITEM at the beginning of the linked linked list.

Ans: Algorithm 4.6 : INSERTBLL (INFO, AVAIL, NEW, LINK, START, ITEM)

Let LINK is a linked list with memory array INFO and LINK. START and AVAIL is a link pointer variable which indicates address of beginning with actual linked list. Here NEW is a local pointer variable used to store address of node. This algorithm perform insert an ITEM of information at the beginning of the given linked list.

Step 1. : [Is Overflow ?]
If AVAIL = NULL, then
Write : "OVERFLOW" and Exit
[End of if structure]

Step 2. : [Remove first node from AVAIL]
Set NEW = AVAIL and
AVAIL = LINK[AVIAL]

Step 3. : [place ITEM into NEW node]
Set INFO[NEW] = ITEM

Step 4. : [link NEW node at the beginning]
Set LINK[NEW] := START

Step 5. : [change START to the NEW node]
Set START = NEW

Step 6. : [Finished] Exit.

Q. 4 Attempt any three of the following (5 Marks each) 15

a) Write queue operations.

Ans: **I] Insert ITEM in QUEUE:**

- The following is an algorithm to insert ITEM at the REAR of QUEUE.

Algorithm 5.7 : QINSERT (QUEUE, N, ITEM, FRONT, REAR)

Let QUEUE is a linear array with N element. FRONT and REAR is a pointer variable indicate FRONT and REAR element. This algorithm perform to insert ITEM of information at the REAR of QUEUE.

Step 1. : [Is QUEUE Overflow?]
If FRONT = REAR = N, then
Write "OVERFLOW" and Exit.
[End of if structure]

Step 2. : [FIND new value for REAR]
If FRONT = NULL, then
Set FRONT = 1 & REAR =1
Else if REAR = N, then:
Set REAR = 1
Else
Set REAR = REAR + 1
[End of if structure]

Step 3. : [Insert new ITEM]
Set QUEUE[REAR] = ITEM

Step 4. : [Finished] Exit.

II] Deleting ITEM from QUEUE :-

Algorithm 5.8 : QDELET (QUEUE, N, ITEM, FRONT, REAR)

Let QUEUE is a linear array with N element. FRONT and REAR is a pointer variable indicate FRONT and REAR element. This algorithm perform to delete an ITEM of information at the FRONT of QUEUE.

Step 1. : [Is QUEUE empty?]
If FRONT = NULL, then
Write "UNDERFLOW" and Exit.
[End of if structure]

Step 2. : [Delete FRONT and store in ITEM]
Set ITEM = QUEUE[FRONT]

Step 3. : [QUEUE has only one element to start]

If FRONT = REAR, then

Set FRONT = NULL and

REAR = NULL

Else if

FRONT = N, then

Set FRONT = 1

Else

FRONT = FRONT + 1

[End of if structure]

Step 4. : [Finished] Exit.

b) Write algorithm to delete ITEM from linked list.

Ans: Algorithm 4.9 : DELNODE (INFO, LINK, AVAIL, LOC, START, LCOP)

Let LIST be a linked list stored in memory with INFO and LINK. LCO is a location of Node X which will delete and LOCP is a location of the node precidy X. When X is the first node then LOCP = NULL. This algorithm delete the node X with location LOC.

Step 1. : If LOCP = NULL, then

Set START = LINK[START]

[Delete first node]

Else

Set LINK[LOCP] = LINK[LOC]

[Delete node X]

[End of if structure]

Step 2. : [Insert deleted node at the AVAIL list]

Set LINK[LOC] = AVAIL and

AVAIL = LOC

Step 3. : [Finished] Exit.

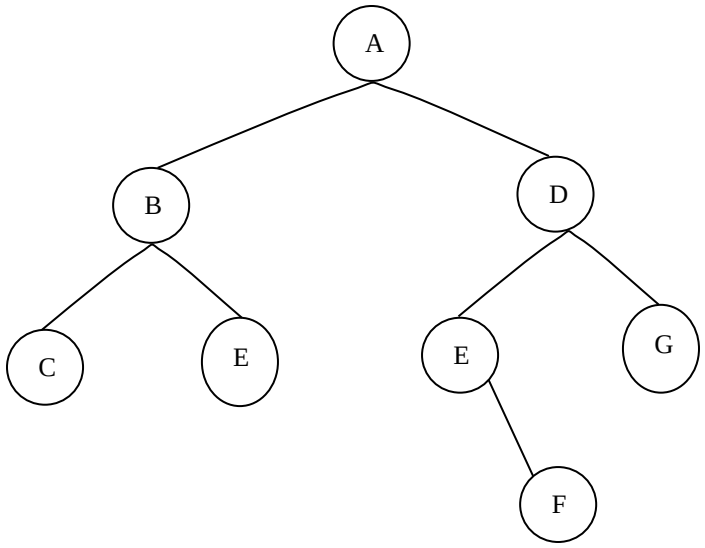
c) Explain traversing of binary tree.

Ans: Traversing is one of the common data structure function which is perform on a tree to process or visit each node of the tree exactly once in a systematic manner.

- There are mainly three standard way used to traverse given binary tree T with root B.
- These three ways are:

1)Preorder Traversing. 2) Inorder Traversing. 3) Postorder Traversing.

- For detail study of above traversal methods, consider a following binary tree.



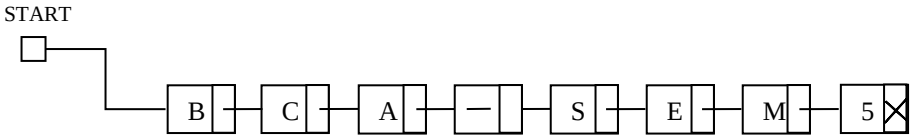
1. Preorder Traversing :-
   - This traversing method works on follows,
     I. Process the Root node R.
     II. Traverse the left sub tree of R in preorder.
     III. Traverse the right sub tree of R in P preorder.
   - Also this traversing method is called node-left-right (N-L-R) traversal.
   - The preorder traversal of given tree T shown in above fig. is, A B C E D E F G.
2. Inorder Traversal :-
   - This traversing method works as follows,
     I. Traverse the left sub tree of R in inorder.
     II. Process the root R.
     III. Traverse the right sub tree of R in inorder.
   - Also this traversing method is called left-node-right (L-N-R) traversal.
   - The inorder traversal of given tree T shown in above figure is, C B E A E F D G.
3. Postorder Traversal :-
   - This traversing method works as follows,
     I. Traverse the left sub tree of R in postorder.
     II. Travers the right sub tree of R in postorder.
     III. Process the root node R.
   - Also this traversing method is also called left-right-node (L-R-N) traversal.
   - The postorder traversal of given tree T shown in above figure is, C E B F E G D A.

d)Write representation of linked list in memory.

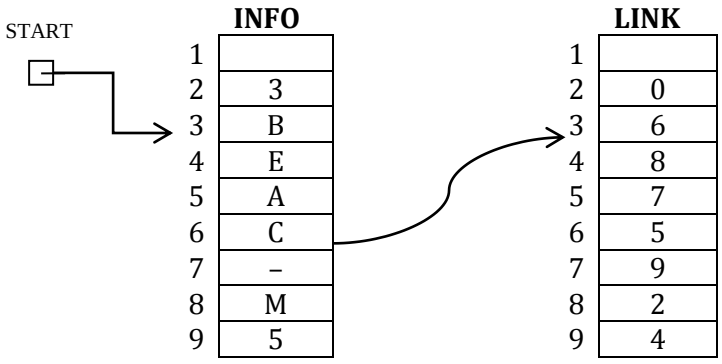
Ans: Suppose list be a linked list for the storage of such a list in memory it requires linear arrays.

- One is a general array set “INFO”, another pointer array set “LINK”.
- An array INFO[K] will contain the information part and an array LINK[k] will contain the address of the next element of the list.

- In linked list, a pointer variable START is used to store starting location of the list. NULL is used to show the end of the list.
- If the subscripts of an array INFO & LINK will be positive integer, then NULL = 0 used to show end of the linked list.
- Here, the nodes of the linked list doesn't need occupy the adjacent dement in the array INFO & LINK.
- Also more than one linked list may be maintain in the same linear array but only by storing the static address of the second list. Another pointer variable set START = 1 & so on.
- Ex.: suppose we want to store the words :- BCA–SEM 3 using a linked list.
- The linked list representation of these words is as follows,



- Internal representation of linked list is as follow,



- The actual list can be obtain as follows,

START = 3	INFO[3] = B
LINK[3] = 6	INFO[6] = C
LINK[6] = 5	INFO[5] = A
LINK[5] = 7	INFO[7] = –
LINK[7] = 9	INFO[9] = S
LINK[9] = 4	INFO[4] = E
LINK[4] = 8	INFO[8] = M
LINK[8] = 2	INFO[2] = 3
LINK[2] = 0	

- The NULL value, so that the list has ended.

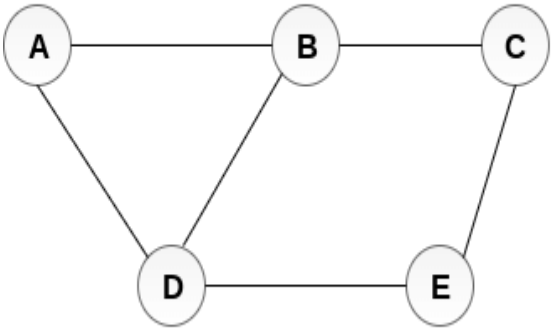
e) Write Sequential Representation of graph.

Ans: Sequential Representation of graph:

In sequential representation, we use adjacency matrix to store the mapping represented by vertices and edges. In adjacency matrix, the rows and columns are represented by the graph vertices. A graph having n vertices, will have a dimension **n x n**.

n entry M_{ij} in the adjacency matrix representation of an undirected graph G will be 1 if there exists an edge between V_i and V_j.

An undirected graph and its adjacency matrix representation is shown in the following figure.



Undirected Graph

	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	1	0
C	0	1	0	0	1
D	1	1	0	0	1
E	0	0	1	1	0

Adjacency Matrix

in the above figure, we can see the mapping among the vertices (A, B, C, D, E) is represented by using the adjacency matrix which is also shown in the figure.

There exists different adjacency matrices for the directed and undirected graph. In directed graph, an entry A_{ij} will be 1 only when there is an edge directed from V_i to V_j.

Q .5 Short notes on any three of the following (5 Marks each) 15

a) Algorithm complexity.

Ans: A well define list of steps for solving a particular problem is called 'Algorithm'.

- The most important use of these steps is to develop effective algorithm for processing our data.
- The time and space are two important measurements which are used to calculate the efficiency of an algorithm.
- "The complexity is the function which gives the running time and memory space used by the inputs".
- There are two case usually investigates to find complexity.
    1. Worst Case.

2. Average Case.

1. Worst Case :-

- The maximum value of F(n) for any possible input.
- C(n) = n is complexity of linear search algorithm.

2. Average Case :-

- The excepted value of F(n).
- Number 'n' is item in DATA, so no. of comparison can be any of the no. 1, 2, 3, ……., n and each no. occurs in probability $P = \frac{1}{n}$ then,

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

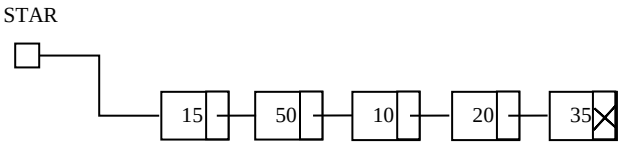
$$= (1 + 2 + \ldots\ldots + n)\ \frac{1}{n}$$

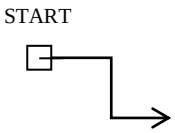
$$= \frac{n(n+1)}{2} = \frac{1}{n} = \boxed{\frac{n+1}{2}}$$

b) Memory allocation , garbage collection.

Ans: 1. Memory Allocation :

- Let a linked list stored in memory using linear array INFO and pointer array LINK.
- The linked representation of a linked list as follows,





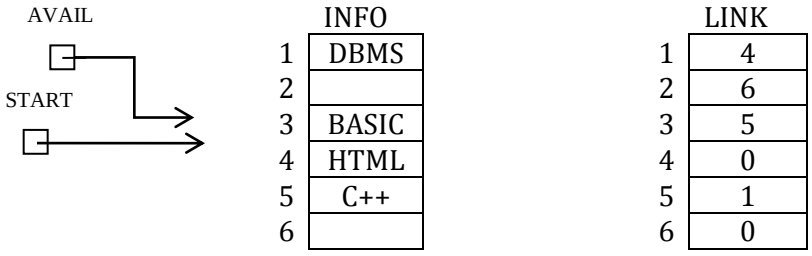
	INFO
1	20
2	15
3	10
4	35
5	50

	LINK
1	4
2	5
3	1
4	0
5	3

- From above representation, it is clear that one can't insert any element (node) in given linked list because there is no any memory space present.
- Now suppose an element 20 is deleted from the list.
- We know in linked list deletion is take place only by changing.
- The link address of previous node.
- Here a previous node of node 20 is 10.
- Hence, in the LINK field of the node 1 10 will contain the address of node 35, if one deleted node 20.

- Now though the element 20 is deleted still it is present physically in the memory cell at location one.
- Hence, one can't insert new element in this linked list.
- To overcome above problem, some mechanism is used which will collects, memory space of deleted node are then available for future use.
- A mechanism which performs such function is known as "Memory Allocation".
- In case of linked list, a memory allocation is perform by describing another special list in the same arrays INFO and LINK.
- Here all the unused memory cells and the cells of deleted nodes will be linked list by this special list.
- A variable AVAIL which is pointer of type of used to store beginning location of the special linked list and invalid address NULL is used to indicate the end of the special list.
- The special list is also known by the list of available space or free storage list.
- A data structure with frequently will be denoted by writing :

  LIST (INFO, LINK, START, AVAIL)
- Ex.: Consider a list of books is stored in linear array book & LINK then available memory space in the linear array book may be linked as :



- In above fig. AVAIL indicates first location i.e. 2 is the available space, next 6 is available space.

2. Garbage Collection :

- It is mechanism or technique which perform the collection of all deleted space on to the free storage list.
- The kernel of operating system such function in computer.
- To make such collection operating system to the following two steps :
  - Computer traverses all lust or memory cell and apply tags (margin) to the memory cell which are currently in use.
  - Then computer check all memory cells and collects all untag memory cells on to the free storage list.

- Generally the garbage collection is take place n two cases:
  - If there is no memory space at all left in the free storage list.
  - When CPU has no any work i.e. CUP in idle state.

c) Recursion.

Ans: Suppose P is a procedure containing either a call statement to itself or call statement to a second procedure that may result in a call statement back to the original procedure P.

- Then P is a recursive procedure.
- The recursion is mainly divided into two parts.
  1. Recursive procedure.
  2. Recursion function.

1. Recursive Procedure :-

- A procedure P is set to be recursive procedure if containing either a call statement to a second procedure that may result in a call statement back to the original procedure P.
- Here main program cannot run independently.
- Following are the two properties :
  1) There must be certain criteria called base criteria for which the procedure does not call itself.
  2) Each time the procedure does not call itself, it must be closer to the base criteria.

2. Recursion Function :

- Similarly a function set to be recursively define.
- If the function definition refers to itself.
- Again in order for the definition not to be circular, it must have the following two properties:
  1) There must be certain arguments, called base value. For which the function does not refer to itself.
  2) Each time the function does refer to itself, the argument of a function must be closer to a base value.

Recursion function with these two properties is also set to be well defined.

d) Deque.

Ans: De-queue is a linear list in which elements can be added or the remove at either end but not in the middle.

- The term de-queue is a contradicting if the name double ended queue.
- There are various way of representing,
- A de-queue is a computer unless it is otherwise stated or implied.
- We will assume our de-queue is maintain by a circular array de-queue with pointer left and right, which point to the two ends of the de-queue.
- We assume that the element extended from the left end to the right end in the array.

- Following are the two de-queue in which in diagram.
- Each with four element maintain in an array with N = 9 memory location.
- The condition LEFT = NULL will indicate de-queue is empty.

DE-QUEUE

			AA	BB	CC	DD		
1	2	3	4	5	6	7	8	9

LEFT = 4 fig. (a) RIGHT = 7

DE-QUEUE

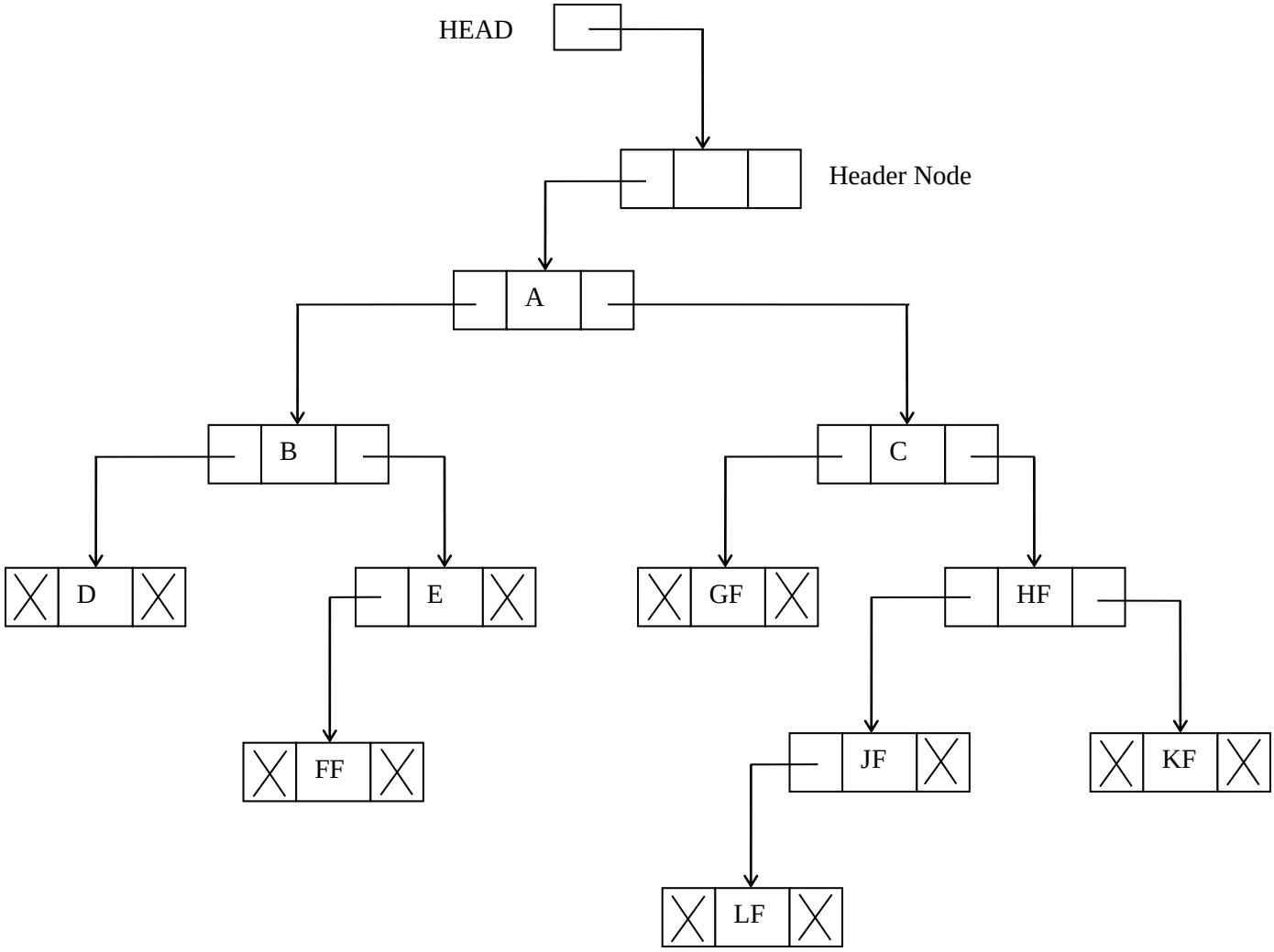
YY	ZZ					WW	XX
1	2	3	4	5	6	7	8

LEFT = 7 fig. (b) RIGHT = 2

- There are two variation of DE-QUEUE.
    1. Input restricted de-queue.
    2. Output restricted de-queue. Which are intermediate between de-queue & a queue.
- An input restricted de-queue is a de-queue which allows deletion at both end of the list.
- An output restricted de-queue is a de-queue which allows deletion at only one end of the list but, allows insertions at both ends of the list.

e)Header node.

- Ans: Suppose a binary tree T is maintained in memory by means of linked representation.
- Sometimes header node, is added to the beginning of T.
- When this extra node is used the tree pointer variable, which we will call HEAD (instead of ROOT), will point to the header node and the left pointer of the header node will point to the ROOT[T].
- Following figure shows a symatic picture of binary tree that uses a linked representation with a header node.



- Suppose, a binary tree T is empty. Then, T will still contain a header node, but the left pointer of the header node will contain the null value. Thus, the condition is,

  LEFT [HEAD] = NULL. It will indicate an empty tree.

- In above representation of binary tree T is used to a header node as a sentinel. That is, if a node has an empty sub tree, then the pointer field for the sub tree will contain the address of the header node instead of the null value.
- Accordingly, no pointer will even contain an invalid address and the condition,

  LEFT [HEAD] = HEAD. It will indicate an empty sub tree.