Data Structure

Assignment :- 02

Name - Aman Kumar Singh
EN - 20051010159
Batch - C

1.) Explain Memory representation of Binary Tree.

Ans Linked Representation.

Binary trees in Linked representation are stored in the memory as Linked lists. These lists have nodes that aren't stored at adjacent or neighboring memory location and are linked to each other through the parent - child relationship associated with trees.

In this representation, each node has ~~three~~ 3 different parts :-

* pointer that points towards the right node.
* pointer that points towards the left node
* data element.

Sequential Representation

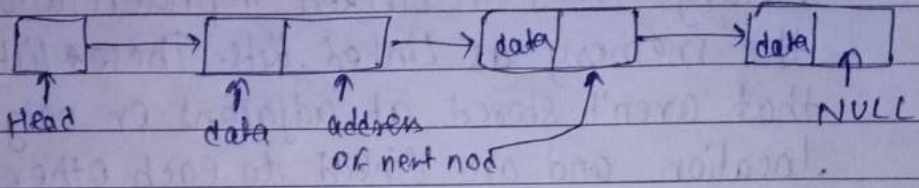
Although it is simplen than linked representation, its in efficiency makes it a less preferred binary tree representation of the two. The inefficiency lies in the amount of space it requires for the storage of different tree elements. The sequential representation uses an array for the storage of tree elements. The number of nodes a binary tree has defines the size of the array being used. The root node of the binary tree lies at the array's first index. The index at which a particular node is stored will defines the indices at which the right and left children of the node will be stored. An empty tree has null or 0 as its first index.

2.) What is linked list ? Write algorithm to copy a linked list (singly).

Ans

Linked list can be defined as collection of objects called nodes that are randomly stored in the memory.

* A Node contains two fields, data and address
* Head Node is connect to the first Node



⇒ Algorithm to copy linked list :-

1. [Empty list ?]
   if FIRST = NULL
   then Return [NULL]
2. [copy first Node]
   IF Avail = NULL
   then Write ['Availability Stack OverFlow']
   Return (0)
   else NEW ← AVAIL
   AVAIL ← LINK (AVAIL)
   FIELD (NEW) ← INFO (FIRST)
   BEGIN ← NEW
3. [Initialise traversal]
   SAVE ← FIRST
4. [MOVE to next node if not at end of list]
   Repeat through Step 6 while LINK [SAVE] = NULL

5. [Update Predecesor and save pointer]
PRED ← NEW
SAVE ← LINK[SAVE]

6. [Copy Node]
If AVAIL = NULL
then Write ['AVAILABILITY STACK UNDERFLOW]
Return (0)
else NEW ← AVAIL
AVAIL ← LINK[AVAIL]
FIELD(NEW) ← INFO(SAVE)
PTR(PRED) ← NEW

7. [Set Link of last node and Return]
PTR(NEW) ← NULL
Return (BEGIN)

3.) What is Doubly Linked List ? Write algorithm to ~~copy~~ insert and delete an element from doubly linked list.

Ans

Doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence.

⇒ Algorithm to insert an element in doubly linked list.

1. [Obtain new node from availability stack]
New ← NODE

2. [Copy information field]
INFO(NEW) ← X

3. [Insertion into an empty list ?]
If R = NULL
then LPTR(NEW) ← RPTR(NEW) ← NULL
L ← R ← NEW
Return

4. [Left-most insertion]
If M = L
then LPTR(NEW) ← NULL
RPTR(NEW) ← M
LPTR(M) ← NEW
L ← NEW
Return.

5. [INSERT in middle]
LPTR(NEW) ← LPTR(M)
RPTR(NEW) ← M
LPTR(M) ← NEW
RPTR(LPTR(NEW)) ← NEW
Return

=> Algorithm to delete an element

1. [Underflow ?]
If R = NULL
then Write ('Underflow')
Return

2 [Delete Node]

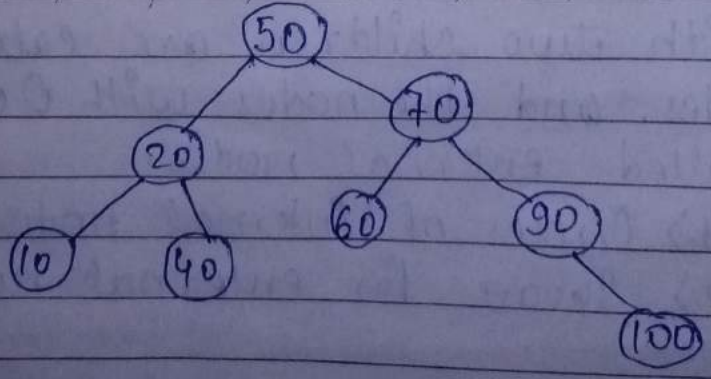
If R = NULL
then write ('Underflow')
Return
If L = R (Single node in list)
then L ← R ← NULL
else if OLD = L (Left-most node being deleted)
then L ← RPTR(L)
LPTR(L) ← NULL
else if OLD = R (Right-most node being deleted)
then R ← RPTR(L)
RPTR(L) ← NULL
~~else if OLD ≠ R (Right-most node being deleted)~~
then R ←
else RPTR(LPTR(OLD)) ← RPTR(OLD)
LPTR(RPTR(OLD)) ← LPTR(OLD)

3. [Return deleted node]

Restore(OLD)
Return

4.) Construct a Binary Search Tree (BST) for the following sequence of numbers (step by step)
50, 70, 60, 20, 90, 10, 40, 100

Ans

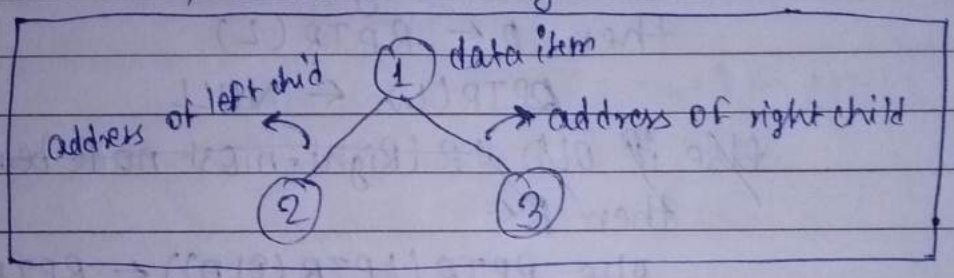


5.) What is Binary Tree ? Explain Complete Binary Tree, Extended Binary Tree with example.

Ans

A Binary Tree is a tree structure in which each parent node can have at most two children. Each node of a binary tree consists of three items :-

1.) data item
2.) address of left child
3.) address of right child



Complete Binary Tree

The tree T said to be complete if all the nodes at the last level appear as far left as possible.

Extended Binary Tree

A Binary tree T is said to be a 2-tree or extended binary tree if each node N has either 0 or 2 children. The nodes with two children are called internal nodes, and the nodes with 0 children are called external nodes.

1.) Circles of internal nodes
2.) Square for external nodes.

—x—y—