DATA STRUCTURES – UNIT 5: TREES

COMPLETE EXAM-READY QUESTION BANK WITH ANSWERS

SECTION 1: LONG ANSWER QUESTIONS AND ANSWERS (5 MARKS EACH)

Q1) What is a Tree? Explain the basic definition and components of a tree data structure.

Answer:

A tree is a finite set of one or more nodes where there is a specially designated node called the root, and the remaining nodes are partitioned into disjoint sets, each forming a subtree.

Basic components:

(a) Node: Basic unit that contains data and links to other nodes

(b) Edge: Connection between two nodes representing parent-child relationship

Trees are hierarchical non-linear data structures; unlike arrays or linked lists, trees organize data in parent-child relationships forming multiple levels.

Key characteristics:

(a) Exactly one path exists between any two nodes

(b) No cycles are present in the structure

Adding or removing an edge breaks the tree property; if we add an edge between any two nodes, it creates a cycle and becomes a graph.

Trees grow from root downward (computer science convention); root is at top, leaves at bottom unlike natural trees.

Example: File system hierarchy where folders contain subfolders and files represents tree structure with root directory at top.

Q2) What is a Binary Tree? Explain its definition and properties.

Answer:

Binary Tree is a finite set of nodes that is either empty or consists of a root and two disjoint binary trees called the left subtree and the right subtree.

Every node in a binary tree can have at most two children; children are distinguished as left child and right child.

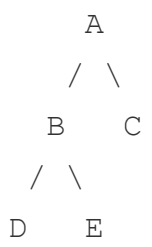
Key properties:

(a) Maximum number of nodes on level i is 2^(i-1), where i >= 1

(b) Maximum number of nodes in binary tree of depth k is 2^k - 1, where k >= 1

Structure representation:

```
Code Example
```



Binary tree is ordered; left and right subtrees are distinguished and cannot be interchanged arbitrarily.

Empty tree is a valid binary tree; both left and right subtrees of any node can be empty (NULL).

Unlike general trees where node can have any number of children, binary tree strictly limits children to maximum two.

Q3) What is a Binary Search Tree (BST)? Explain its properties and characteristics.

Answer:

Binary Search Tree is a binary tree where every node satisfies the ordering property: all keys in left subtree are smaller than node's key, and all keys in right subtree are larger.

BST properties:

(a) Every element has a unique key value

(b) Left subtree keys < root key < right subtree keys

(c) Both left and right subtrees are also binary search trees

BST provides efficient structure for searching, insertion, and deletion operations; average time complexity O(log n) for these operations.

Example BST:

```
Code Example
```

```
         50
       /   \
      30    70
     /  \     \
    20  40    80
```

BST property applied recursively; not only root but every internal node must satisfy the ordering constraint for all descendants.

Empty tree is valid BST; single node tree is also valid BST as it trivially satisfies the ordering property.

Inorder traversal of BST produces sorted sequence; this property useful for retrieving data in sorted order.

Q4) Differentiate between General Tree and Binary Tree.

Answer:

Aspect	General Tree	Binary Tree
Number of children	Node can have any number of children	Node can have at most two children
Children distinction	Children are not distinguished; no left/right concept	Children are distinguished as left child and right child
Ordering	No inherent ordering among children	Ordered tree; left and right subtrees are distinct
Subtrees	Node can have n >= 0 subtrees	Node has exactly two subtrees (left and right), either may be empty

---

Aspect	General Tree	Binary Tree
Degree	Degree of node can be any value >= 0	Maximum degree of any node is 2
Representation	Requires variable-size node structure or conversion	Fixed-size node with two pointers (left and right)
Application	File systems, organization charts	Expression trees, BST, heaps

---

Q5) Explain the important terminologies used in tree data structures.

Answer:

Root: Node without parent; topmost node of tree serving as entry point, exactly one root exists in a tree.

Leaf or Terminal node: Node with degree zero (no children); leaf nodes are at the bottom of tree structure.

Parent and Child: If node A has edge to node B, then A is parent of B and B is child of A.

Siblings: Nodes that share the same parent node; siblings are at the same level in tree.

Level of node: Position of node in tree hierarchy; root is at level 1, its children at level 2, and so on.

Height or Depth of tree: Maximum level of any node in the tree; represents longest path from root to any leaf.

Degree of node: Number of subtrees (children) of a node; degree determines branching at that node.

Degree of tree: Maximum degree among all nodes in the tree; represents maximum branching factor.

Path: Sequence of consecutive edges from one node to another; unique path exists between any two nodes.

Subtree: Tree consisting of a node and all its descendants; any node can be considered root of subtree.

Ancestors of node: All nodes along path from root to that node; root is ancestor of all nodes.

Descendants of node: All nodes in subtrees rooted at that node; children, grandchildren, and so on.

Forest: Collection of disjoint trees; if we remove root from tree, remaining subtrees form a forest.

Q6) Explain the different types of Binary Trees with examples.

Answer:

Full Binary Tree: Binary tree of depth k having exactly 2^k - 1 nodes where k >= 0; every level is completely filled.

Complete Binary Tree:

Binary tree where all levels are completely filled except possibly the last level, and last level has all nodes as left as possible.

Nodes filled from left to right on each level without gaps.

Example:

```
Code Example
```

```
        A
       / \
      B   C
     / \ /
    D  E F
```

Skewed Binary Tree:

(a) Left-skewed: Every node has only left child, no right children exist

(b) Right-skewed: Every node has only right child, no left children exist

Worst case structure; degenerates into linked list with O(n) search time.

Strict Binary Tree: Every non-leaf node has exactly two children; also called proper binary tree or 2-tree.

Difference summary: Full trees are always complete, but complete trees need not be full; skewed trees represent worst-case unbalanced structure.

Q7) What are Binary Tree Traversals? Explain the three depth-first traversal methods.

Answer:

Tree traversal is systematic process of visiting each node in tree exactly once; traversal order determines how nodes are processed.

Inorder Traversal (Left-Root-Right):

Visit left subtree first, then root node, finally right subtree.

For BST, inorder traversal produces sorted sequence of keys in ascending order.

Preorder Traversal (Root-Left-Right):

Visit root node first, then left subtree, finally right subtree.

Used to create copy of tree or to get prefix expression from expression tree.

Postorder Traversal (Left-Right-Root):

Visit left subtree first, then right subtree, finally root node.

Used to delete tree or to get postfix expression from expression tree.

Example tree and traversals:

```
Code Example
```

```
        A
       / \
      B   C
     / \
    D   E

Inorder: D B E A C
Preorder: A B D E C
Postorder: D E B C A
```

Depth-first traversals use stack (implicit recursion stack); breadth-first traversal uses queue for level-order processing.

Q11) Explain the process of converting a General Tree to Binary Tree using left-child right-sibling representation.

Answer:

Any general tree can be transformed into binary tree using left-child right-sibling representation where each node has at most two pointers.

Conversion rules:

(a) Binary tree left child = leftmost child of general tree node

(b) Binary tree right child = right sibling of general tree node

Conversion process:

For each node, identify its leftmost child and make it the left child in binary tree.

Identify the node's immediate right sibling and make it the right child in binary tree.

Apply this rule recursively to all nodes in the tree.

Example:

```
Code Example

General Tree:           Binary Tree:
      A                      A
    / | \                   /
   B  C  D                 B
  /                         \
 E                           C
                              \
                               D
                              /
                             E
```

Resulting binary tree preserves hierarchical relationships; parent-child and sibling relationships maintained through left and right pointers.

This representation allows any general tree to use fixed-size node structure with exactly two pointers.

Advantage: Simplifies tree operations as all trees can be handled using uniform binary tree structure.

Q12) What is a Threaded Binary Tree? Explain its concept and advantages.

Answer:

Threaded Binary Tree replaces NULL pointers with useful "threads" pointing to inorder predecessor or successor, making tree traversal more efficient.

Threading concept:

In normal binary tree, number of NULL links is n+1 where n is number of nodes; these NULL pointers are wasted.

Right NULL pointer replaced by inorder successor; left NULL pointer replaced by inorder predecessor.

Thread identification: Additional boolean fields (leftThread and rightThread) distinguish normal pointers from threads.

Types of threading:

(a) Right-threaded: Only right NULL pointers replaced with inorder successor

(b) Left-threaded: Only left NULL pointers replaced with inorder predecessor

(c) Fully-threaded: Both left and right NULL pointers replaced with threads

Advantages:

Non-recursive inorder traversal becomes simpler and faster without stack overhead.

Can efficiently find predecessor and successor from any node without traversing from root.

Nodes circularly linked through threads allowing movement in both directions.

Disadvantage: Insertion and deletion operations become more complex as threads must be updated to maintain consistency.

Q14) What is a Heap? Explain Max Heap properties and structure.

Answer:

Heap is a complete binary tree that satisfies the heap property; every parent node's value is greater than or equal to (max heap) or less than or equal to (min heap) its children's values.

Max Heap property:

For any node x, key of parent(x) >= key of x; root contains maximum element.

Heap property applies recursively; every subtree is also a heap.

Structure properties:

(a) Heap is always a complete binary tree; all levels filled except possibly last level

(b) Last level filled from left to right without gaps

Array representation: Heap stored in array where parent of node at index i is at index $\lfloor (i-1)/2 \rfloor$, left child at 2i+1, right child at 2i+2.

Example Max Heap:

```
Code Example

         50
        /  \
       30   40
      /  \  /
     10  20 35

Array: [50, 30, 40, 10, 20, 35]
```

Complete binary tree property enables efficient array representation without wasting space; no NULL pointers needed.

Root always contains maximum element in max heap; minimum element in min heap, enabling O(1) access to extreme value.

Q15) Explain the applications of Heap data structure.

Answer:

Priority Queue implementation:

Heap provides efficient priority queue where highest (or lowest) priority element accessed in O(1) time.

Insert and delete operations performed in O(log n) time maintaining heap property.

Heap Sort algorithm:

Heap used to sort array in O(n log n) time with no worst-case degradation.

Build max heap, then repeatedly extract maximum and rebuild heap until sorted.

Selection problem:

Finding kth largest or smallest element efficiently by building heap and deleting k-1 elements.

Selection of kth element performed in O(n + k log n) time.

CPU scheduling and process prioritization:

Operating systems use heaps for scheduling processes based on priority.

I/O scheduling, job scheduling all use priority queues implemented with heaps.

Graph algorithms:

Dijkstra's shortest path algorithm uses min heap to select minimum distance vertex.

Prim's minimum spanning tree algorithm uses heap for efficient edge selection.

Median maintenance: Heaps used to find running median in stream of numbers using two heaps (max heap and min heap).

Heap property ensures no process waits indefinitely (no starvation) in scheduling applications.

Q18) Differentiate between Full Binary Tree and Complete Binary Tree.

Answer:

Aspect	Full Binary Tree	Complete Binary Tree
Definition	All levels completely filled; exactly 2^k - 1 nodes for depth k	All levels filled except possibly last; last level filled left to right
Node count	Must have exactly 2^k - 1 nodes	Can have 2^(k-1) to 2^k - 1 nodes
Last level	Must be completely filled	May be partially filled but nodes pushed to left
Structure	Perfect binary tree; symmetric structure	May be asymmetric on last level
Height	For n nodes, height is exactly $\log_2(n+1)$	Height is $\lceil \log_2(n+1) \rceil$
Example nodes	1, 3, 7, 15, 31, ... (2^k - 1)	Any number between 2^(k-1) and 2^k - 1
Relationship	Every full binary tree is complete	Not every complete binary tree is full
Array representation	No wasted space; all positions used	Minimal wasted space; efficient representation

---

END OF QUESTION BANK