UNIT–3 NOTES

TREES IN DATA STRUCTURE

---

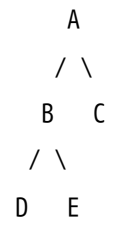
1. Introduction to Tree

Tree ek **non-linear data structure** hai.

Tree nodes aur edges se milkar banta hai.

Tree hierarchical relationship show karta hai.

Example of Tree



Here:

- A = Root node
- B,C = Child nodes
- D,E = Leaf nodes

Applications of Tree

- File system
- Database indexing
- Searching
- Expression evaluation
- Compiler design

Important Terminologies of Tree

---

1. Root Node

Sabse top node.

Example:

```
A
```

is root node.

---

2. Parent Node

Jo node child ko connect kare.

Example:

```
A is parent of B and C
```

---

3. Child Node

Jo node parent se connected ho.

Example:

```
B and C are child nodes
```

---

4. Leaf Node

Jinke koi child nahi hote.

Example:

```
D and E
```

---

5. Degree of Node

Node ke children ki number.

Example:

```
Degree of B = 2
```

---

6. Degree of Tree

Maximum degree of any node.

---

7. Level of Node

Root ka level = 0

Next level = 1 and so on.

---

8. Height of Tree

Root se leaf tak maximum path length.

---

9. Depth of Node

Root se node tak distance.

---

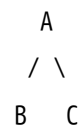
10. Edge

Nodes ko connect karne wali line.

Binary Tree

Aisa tree jisme har node ke maximum 2 children ho sakte hain.

Example



Types of Binary Tree

---

1. Full Binary Tree

Har node ke either:

- 0 child

  OR
- 2 children

---

2. Complete Binary Tree

Sab levels filled except last.

---

3. Perfect Binary Tree

All internal nodes have 2 children and all leaves same level par.

---

4. Skewed Binary Tree

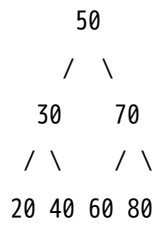
All nodes ek side par.

Binary Search Tree (BST)

BST ek binary tree hai jisme:

```
Left subtree < Root < Right subtree
```

Example



Operations on BST

---

1. Insertion

New node correct position par insert hoti hai.

Algorithm

```
If value < root:
    insert left

Else:
    insert right
```

---

2. Searching

Target value ko compare karke search karte hain.

Example

Search 60:

```
60 > 50 → right
60 < 70 → left
Found
```

---

3. Deletion

Cases:

- Leaf node
- One child
- Two children

Traversal in Tree

Traversal means tree nodes ko visit karna.

Types of Traversal

---

1. Preorder Traversal

```
Root → Left → Right
```

Example

```
A B D E C
```

---

2. Inorder Traversal

```
Left → Root → Right
```

Example

```
D B E A C
```

---

3. Postorder Traversal

```
Left → Right → Root
```

Example

```
D E B C A
```

AVL Tree

AVL tree ek self-balancing BST hai.

Har node ka balance factor maintain hota hai.

Balance Factor

```
Height(left subtree)
-
Height(right subtree)
```

AVL Condition

```
-1 ≤ Balance Factor ≤ 1
```

Rotations in AVL Tree

---

1. Left Rotation

---

2. Right Rotation

---

3. Left-Right Rotation

---

4. Right-Left Rotation

Advantages of AVL Tree

- Faster searching
- Balanced structure
- Reduced height

Heap Tree

Heap ek complete binary tree hai.

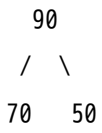
Types of Heap

---

1. Max Heap

Parent > children

Example



---

2. Min Heap

Parent < children

Applications of Heap

- Priority queue
- Heap sort
- Scheduling

Comparison of Trees

Tree	Feature
BST	Fast searching
AVL	Self balanced
Heap	Priority handling
B Tree	Disk storage
Red Black Tree	Balanced BST

Forest

Forest = collection of trees.

Example

```
Tree1 + Tree2 + Tree3
```

Multi-way Tree

Aisa tree jisme node ke 2 se zyada children ho sakte hain.

B Tree

B Tree ek balanced multi-way tree hai.

Databases aur file systems me use hota hai.

Features

- Multiple keys
- Multiple children
- Balanced height

Advantages

- Fast disk access
- Efficient searching

B+ Tree

B+ tree me:

- data leaf nodes me store hota hai
- internal nodes indexing ke liye use hote hain

Advantages

- Faster sequential access
- Better database performance

B* Tree

B* tree B+ tree ka improved version hai.

Better space utilization provide karta hai.

Red-Black Tree

Red-black tree ek self-balancing BST hai.

Each node:

- Red

  OR
- Black

Properties

1. Root always black
2. Red node ka child black hoga
3. Every path same black nodes contain karega

Advantages

- Balanced searching
- Faster insertion/deletion

Applications of Trees

- Database indexing
- File system
- Compiler design
- Routing tables
- Expression trees

- AI decision trees

Advantages of Tree

- Fast searching
- Hierarchical structure
- Dynamic memory allocation

Disadvantages

- Complex implementation
- Extra memory usage