BINARY SEARCH TREE (BST) — COMPLETE NOTES

1. Definition

A **Binary Search Tree (BST)** is a **binary tree** where every node follows this property:

- **Left child < Parent node**
- **Right child > Parent node**
- No duplicate values (in standard BST).

Formally:
For any node **N**,
All values in **left subtree** of N are **< N.data**
All values in **right subtree** of N are **> N.data**

2. Properties

1. Each node has **at most 2 children**.
2. It is **ordered** based on the BST rule.
3. Inorder traversal of BST gives **sorted order**.
4. Time complexity depends on height:
    - **Best/Avg case height = O(log n)**
    - **Worst case (skewed) = O(n)**

3. Basic Operations in BST

Operation	Time (Avg)	Time (Worst)
Search	O(log n)	O(n)
Insert	O(log n)	O(n)
Delete	O(log n)	O(n)
Traversal	O(n)	O(n)

4. BST Traversals

✔ Inorder:

Left → Root → Right
Always gives sorted order

✔ Preorder:

Root → Left → Right

✔ Postorder:

Left → Right → Root

✔ Level Order (BFS):

Uses Queue

5. Searching in BST – Algorithm

```
SearchBST(root, key):
    if root == NULL or root.data == key:
        return root
    if key < root.data:
        return SearchBST(root.left, key)
    else:
        return SearchBST(root.right, key)
```

6. Insertion in BST – Algorithm

```
Insert(root, key):
    if root == NULL:
        return newNode(key)

    if key < root.data:
        root.left = Insert(root.left, key)
    else:
        root.right = Insert(root.right, key)

    return root
```

7. Deletion in BST (3 Cases)

Case 1: Node is a leaf

→ Simply delete it

Case 2: Node has 1 child

→ Replace node with its child

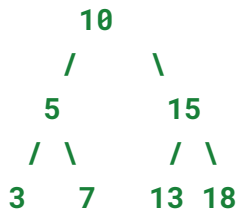
Case 3: Node has 2 children

→ Replace with **inorder successor** (minimum in right subtree)
or **inorder predecessor**

8. Example of BST Creation

Insert: **10, 5, 15, 3, 7, 13, 18**

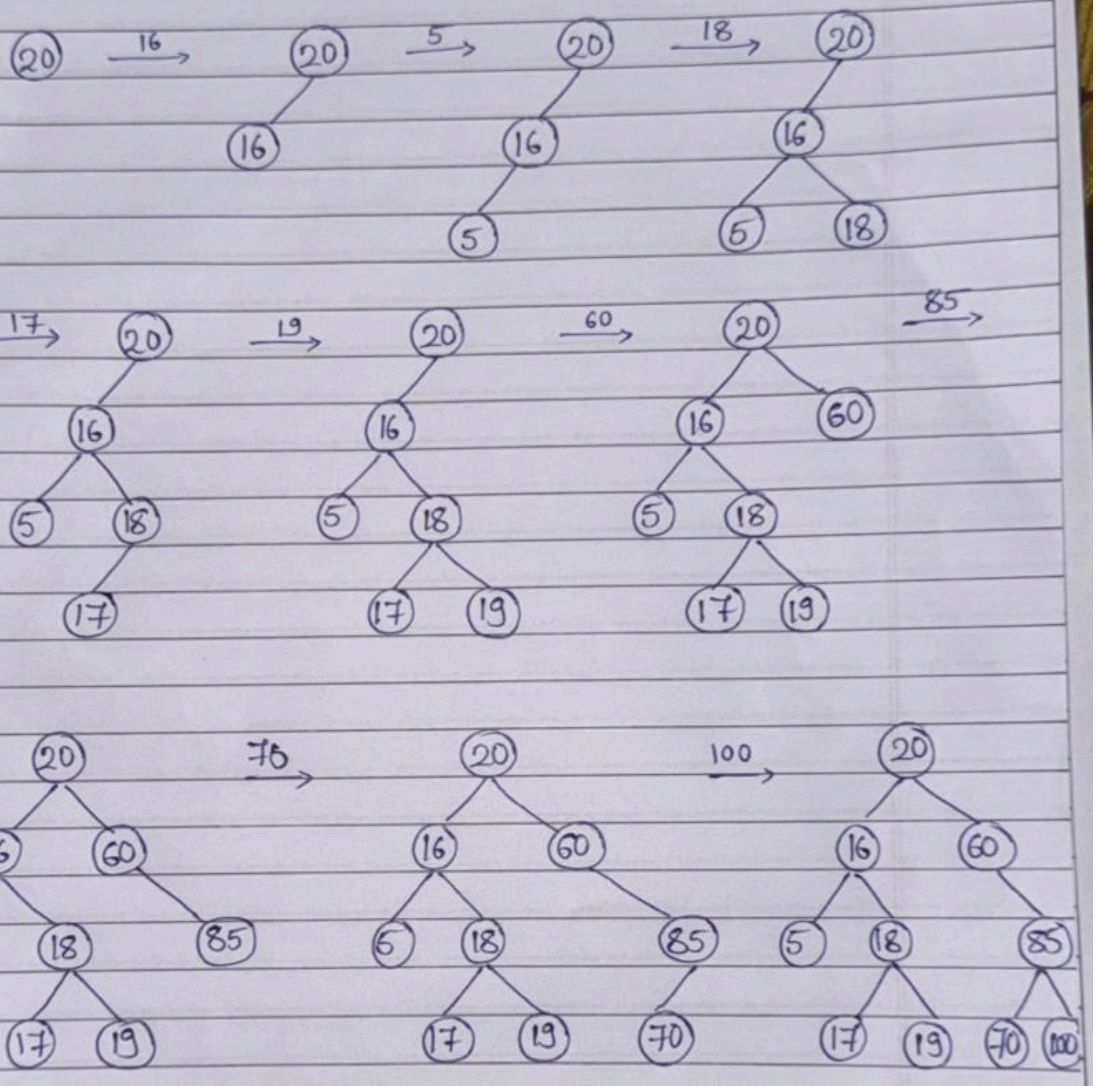
Tree becomes:



Inorder traversal output:
3 5 7 10 13 15 18

Binary Search Tree

Ex: 20, 16, 5, 18, 17, 19, 60, 85, 70, 100



Inorder :- 5, 16, 17, 18, 19, 20, 60, 70, 85, 100.
Preorder :- 20, 16, 5, 18, 17, 19, 60, 85, 70, 100
Postorder :- 5, 17, 19, 18, 16, 70, 100, 85, 60, 20.

9. Types of BST

✔ 1. Strict BST

Follows exact < and > rule

✔ 2. Balanced BST

Height = O(log n)
Examples:

- AVL Tree
- Red-Black Tree
- Splay Tree
- Treap
- B-Tree / B+Tree

✔ 3. Degenerate BST (Skewed Tree)

All nodes on one side – behaves like linked list
Worst case: O(n)

✔ 4. Threaded BST

Uses null pointers to store inorder successor pointers.

10. Height of a BST

Minimum height = $\mathbf{\log_2(n)}$ → Balanced
Maximum height = **n** → Skewed
Height affects all operations.

11. Advantages of BST

✔ 1. Fast search, insertion, deletion

O(log n) in a balanced BST

✔ 2. Sorted data traversal

In order give a sorted order.

✔ 3. Dynamic data structure

Grows and shrinks as needed.

12. Disadvantages of BST

1. **Unbalanced BST becomes slow** → O(n)
2. Performance depends on input order.
3. Extra memory for pointers.
4. No random access (unlike arrays).

13. Applications of BST

✔ 1. Searching & Sorting

- Faster lookup than arrays when balanced
- In-order traversal givesa sorted output

✔ 2. Symbol Tables in Compilers

Identifiers stored in BST

✔ 3. Database Indexing

(Balanced BST variants like B-Trees)

✔ 4. Implementing Sets & Maps

C++ STL set, map → Red-Black Tree

✔ 5. Directory Structure in File Systems

Hierarchical order

✔ 6. Priority Searching

(priority search tree uses BST properties)

✔ 7. Auto-complete Search

Words stored in BST for fast matching

✔ 8. Network Routing Algorithms

Maintain sorted routing tables

14. Real-Life Analogy

BST is like:

📚 *A library bookshelf arranged alphabetically (A-Z)*
Searching a book becomes faster due to ordering.

15. Interview Questions

1. What is BST?
2. Time complexities of search/insert/delete?
3. Why inorder traversal gives sorted output?
4. Cases in deletion of BST?
5. Difference between BST and binary tree?
6. What happens when BST becomes unbalanced?
7. How to balance a BST?
   → Convert to AVL / Red-Black Tree
8. Applications of BST?
9. Difference between BST and Heap?
10. Can duplicates be inserted in BST?