KAZAKH KBTU BRITISH
TECHNICAL
UNIVERSITY

LECTURE 4 – BINARY SEARCH TREE

Prepared by: Izbassar Assylzhan

Course: Algorithms & Data Structures – Fall 2025

LECTURE 4

KAZAKH KBTU BRITISH TECHNICAL UNIVERSITY

In this lecture, we'll cover one of the main data structures - **binary search tree** for efficient *searching*, *insertion*, and *deletion* operations on the data stored in the tree.

TOPICS WE'LL COVER:

- BST Concept
- Node Structure & Insertion
- Search, Min / Max & Traversal
- Deletion in BST
- Practice Problems

GOALS FOR THIS LECTURE:

- Understand the structure and properties of Binary Search Trees
- Implement basic BST operations (insert, search, traversal, delete) in C++
- Analyze BST performance and compare balanced vs unbalanced trees

BASIC BST CONCEPTS

- BST Concept
- Node Structure & Insertion
- Search, Min / Max & Traversal
- Deletion in BST
- Practice Problems

A **Binary Search Tree** is a data structure that helps organize and store data so that the elements remain in sorted order.

BST property:

- Left subtree → **values less than node**;
- Right subtree → **values greater than node**.
- Both the left and right subtrees must individually satisfy the properties of a binary search tree.
- Also, duplicate nodes are not allowed (though some BST variations handle equal values differently).

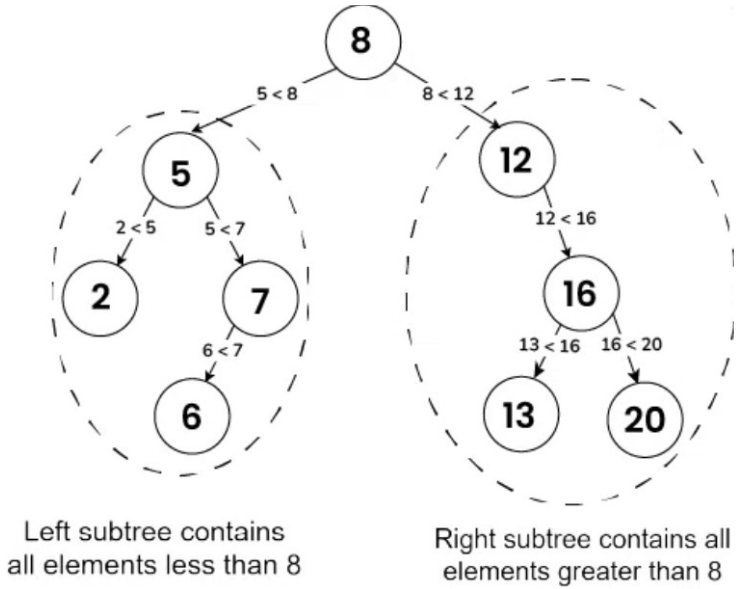

Figure 1 – Structure of BST [1]

[1] https://www.geeksforgeeks.org/dsa/introduction-to-binary-search-tree/

NODE STRUCTURE & INSERTION

KAZAKH BRITISH TECHNICAL UNIVERSITY (KBTU)

A node of BST contains of the *value* of the data, then the links for the *left* and *right* children.

1. Start by setting the current node to the root.
2. Compare the value to be inserted with the value of the current node.
3. If the new key is less than to the current node's value, move to the left child.
4. If the new key is greater than the current node's value, move to the right child.
5. Continue steps 2–4 until you arrive at a node with no child in the needed direction (a leaf).
6. Insert the new key as the left or right child of that leaf, depending on the final comparison.

```
void insert(int value) {
    root = this->insertRecursive(root, value);
}
```

```
Node* insertRecursive(Node* node, int value) {
    if (node == nullptr) {
        return new Node(value);
    }
    if (node->value > value) {
        node->left = insertRecursive(node->left, value);
    } else if (node->value < value) {
        node->right = insertRecursive(node->right, value);
    }
    return node;
}
```

SEARCH, MIN/MAX & TRAVERSAL

KAZAKH BRITISH TECHNICAL UNIVERSITY

- BST Concept
- Node Structure & Insertion
- Search, Min / Max & Traversal
- Deletion in BST
- Practice Problems

In order to search some element from BST, we the definition of the tree as:

- If we have some node in BST, and we found the key → return that node;
- If our key is less than current node → we search in the left part of the tree;
- Otherwise, searching in the right part.

When we travers through the tree, we have 3 options doing that:

- **Inorder**: visit the left subtree first, then the current node, and finally the right subtree.
- **Preorder**: visit the current node first, then the left subtree, and finally the right subtree.
- **Postorder**: Visit the left subtree first, then the right subtree, and finally the current node.

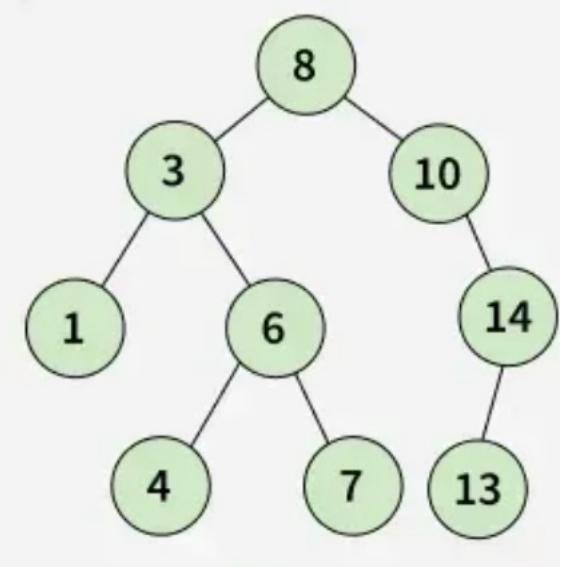

Figure 2 – Example of BST [2]

[2] https://www.geeksforgeeks.org/dsa/binary-search-tree-data-structure/

DELETION IN BST

KAZAKH KBTU BRITISH TECHNICAL UNIVERSITY

When deleting a node from a BST, there are three main cases to consider:

Case 1: Deleting a Leaf Node: if the node has no children, simply remove it from the tree.

Case 2: Deleting a Node with One Child: if the node has only one child, replace the node with its child and then delete the node.

Case 3: Deleting a Node with Two Children:

- Deleting a node with two children requires extra care to maintain BST properties;
- The usual approach is to find the node's inorder successor, copy its value to the node to be deleted, and then delete the inorder successor.

Note: *The inorder successor is only required when the node's right child is not empty. It can be found as the minimum value in the node's right subtree.*

KAZAKH KBTU BRITISH TECHNICAL UNIVERSITY

PRACTICE PROBLEMS

We have two tasks to solve:

- 108. Convert Sorted Array to Binary Search Tree;
- 230. Kth Smallest Element in a BST.

Q & A