Homework # 02

Lead TAs: Areeba Fatima, Wajid Ali, Sarfraz ,Saif,

Guidelines:

- Attempt all questions by yourself before discussing with peers, this is practice to strengthen your concepts.
- For coding questions, try writing clean, readable code on paper.
- Since this homework is ungraded, focus on learning rather than using LLMs to generate codes and get answers.
- If you get stuck, you are encouraged to:
  - Post your doubts on the course Slack channel.
  - Visit the TAs during office hours for guidance.

Section A: Multiple Choice Questions (MCQs)

Q1)Which traversal of a Binary Search Tree (BST) results in keys in ascending order?

A) Preorder

B) Inorder

C) Postorder

D) Level order

Q2)A B+ tree supports rapid random access and sequential access because:

A) Only random access is efficient

B) Leaves are linked together and tree is balanced

C) Only sequential access is efficient

D) It uses hashing for access

Q3)What is the balance factor in an AVL tree?

A) Height(left subtree) + Height(right subtree)

B) Height(right subtree) − Height(left subtree)

C) |Height(left subtree) − Height(right subtree)| ≤ 1

D) Height(left subtree) − Height(right subtree)

Q4)If you perform a preorder traversal on a BST containing keys {10, 5, 20, 3, 7}, which sequence do you get?

A) 3, 5, 7, 10, 20

B) 10, 5, 3, 7, 20

C) 20, 10, 5, 7, 3

D) 5, 3, 7, 10, 20

Q5)When inserting a new node into an AVL tree, at most how many rotations are required to restore balance?

A) O(n)

B) 2

C) $\log_2(n)$

D) Unlimited

Q6) In a B+ tree, where are the actual data records (or pointers to them) stored?

A) In the internal nodes only

B) In both internal and leaf nodes

C) Only in the leaf nodes

D) Only in the root node

Q7) What **is the primary reason for preferring a B+ tree over a BST for indexing a large database stored on disk?**

A) B+ trees are always perfectly balanced.

B) B+ tree operations have a better

Big-O complexity in terms of CPU time.

C) B+ trees are designed to minimize the number of disk I/O operations.

D) B+ trees can store more distinct keys than a BST of the same height.

Q8) An AVL tree is a self-balancing version of which fundamental data structure?

A) B-Tree
B) Binary Search Tree
C) Red-Black Tree
D) Heap

Q9) If a node in an AVL tree has a balance factor of +2, what does this signify according to the formula

A) The tree is critically left-heavy at that node.
B) The tree is critically right-heavy at that node.
C) The tree is perfectly balanced.
D) The node is a leaf node.

Q10) What is the worst-case time complexity for search, insert, and delete operations in a balanced AVL tree?

A) O(n)

B) O(log n)

C) O(n log n)

D) O(1)

Q11) Which of the following is true for a Binary Search Tree (BST)?
a) The left child is always smaller than the root, and the right child is always larger.
b) The left and right subtrees are always balanced.
c) Inorder traversal of a BST gives elements in descending order.
d) The root must always be the smallest element.

Q12) In a BST, the time complexity of searching for an element in the average case is:
a) O(1)
b) O(log n)
c) O(n)
d) O(n log n)

Q13) When deleting a node with two children in a BST, the node is usually replaced with:
a) Its left child
b) Its right child
c) Its inorder successor or predecessor
d) The root node

Q14) Which traversal of a BST always results in a sorted sequence of elements?
a) Preorder
b) Postorder
c) Inorder
d) Level-order

Q15) Consider the BST formed by inserting the numbers 15, 10, 20, 8, 12. What is the postorder traversal of this tree?

a) 8, 12, 10, 20, 15
b) 12, 8, 10, 20, 15
c) 8, 10, 12, 20, 15
d) 20, 12, 10, 8, 15

Q16) The worst-case height of a BST with n nodes is:

a) O(log n)
b) O(n)
c) O(n log n)
d) O(√n)

Q17) Which of the following sequences cannot represent the inorder traversal of a BST?

a) 10, 20, 30, 40, 50
b) 50, 40, 30, 20, 10
c) 20, 30, 25, 40, 50
d) 5, 10, 15, 20, 25

Q18) A BST is balanced if:

a) Every node has at most two children
b) The left subtree and right subtree of every node differ in height by at most 1
c) The inorder traversal results in a sorted sequence
d) The root is the median of all elements

Q19) If you build a BST by inserting 1, 2, 3, 4, 5 in that order, what structure do you get?

a) A complete binary tree
b) A skewed tree (right-skewed)
c) A skewed tree (left-skewed)
d) A perfectly balanced tree

Q20) If two BSTs contain the same set of elements, then:

a) Their inorder traversals must be the same
b) Their structures must be identical
c) Both a and b
d) None of the above

Q21) If a BST has height h, the minimum number of nodes it can have is:

a) h
b) 2^h
c) h + 1
d) h

Q22)An AVL tree is a type of:

a) Binary Search Tree
b) Heap

c) Graph
d) B-tree

Q23) The balance factor of a node in an AVL tree is defined as:
a) Height of right subtree – Height of left subtree
b) Height of left subtree – Height of right subtree
c) Total number of nodes in the tree
d) Depth of the node

Q24) What are the possible values of the balance factor in a valid AVL tree?
a) -2, -1, 0, 1, 2
b) -1, 0, 1
c) 0 only
d) Any integer

Q25) The height of an AVL tree with *n* nodes is always:
a) O(n)
b) O(log n)
c) O(√n)
d) O(1)

Q26)Which of the following is true for an AVL tree?
a) It is always a complete binary tree
b) It is always a full binary tree
c) It is a height-balanced binary search tree
d) It always has equal number of nodes in left and right subtrees

Q27)Inserting a new node into an AVL tree may require:
a) At most 1 rotation
b) At most 2 rotations
c) At most log n rotations
d) Unlimited rotations

Q28)What is the worst-case height H of an AVL tree with n nodes?
a) h = O(n)
b) h = O(log n)
c) h = O(log log n)
d) h = O(√n)

Q29)The time complexity of search, insertion, and deletion in an AVL tree is:
a) O(1)
b) O(h)
c) O(log n)
d) O(n)

Q30) What is the minimum possible height of an AVL tree with 1000 nodes?
a) ≈ 9
b) ≈ 10
c) ≈ 11
d) ≈ 12

Q31)Which imbalance occurs when a node is inserted into the left subtree of the left child of a node?
a) LL imbalance
b) RR imbalance
c) LR imbalance
d) RL imbalance

Q32)Which rotation is required to fix an LL imbalance?
a) Right rotation
b) Left rotation
c) Left-Right rotation
d) Right-Left rotation

Q33)After an insertion, the balance factor of a node becomes +2 and the balance factor of its left child is +1. What rotation is required?
a) Single left rotation
b) Single right rotation
c) Left-Right rotation
d) Right-Left rotation

Q34)After an insertion, the balance factor of a node becomes +2 and the balance factor of its left child is -1. What rotation is required?
a) Single left rotation
b) Single right rotation
c) Left-Right rotation
d) Right-Left rotation

Q35)Suppose a B+ Tree of order m has n keys. What is the maximum possible number of leaf nodes?
a) n
b) n/m
c) n/(m - 1)
d) log_m(n)

Q36) In a B+ Tree of order m, the minimum number of keys in a non-root internal node is:
a) ⌈m/2⌉ - 1
b) ⌊m/2⌋

c) m/2
d) ⌈m/2⌉

Section B: Short Questions:

Question No.1:
Construct the BST formed by inserting the sequence: 15, 10, 20, 8, 12, 25.

- What is the inorder traversal of the tree?
- What is the height of the BST?

Question No2 :
A node **A** has balance factor **+2**. Its left child has balance factor **–1**.
a)Which case is this?
b)Which rotation will fix it?

Question No3:
A B+ Tree of order 4 is used to store integer keys.

i)What is the minimum and maximum number of keys in a non-root internal node?

ii)Explain why B+ trees are efficient for **range queries** compared to AVL trees.

Question No4:

Write a C++ function that prints all values in a BST that are strictly greater than a given k.

Question No5:

An e-commerce website wants to implement a feature to filter products by price. Users should be able to see all products within a specific price range (e.g., $50 to $200) very quickly. The product database is extremely large and stored on disk.

a) Which data structure (BST, AVL tree, or B+ tree) would be most suitable for indexing the product prices?

b) Explain your choice, focusing on the specific requirements of the scenario.

c) Briefly outline the algorithm to find all products in the price range [minPrice, maxPrice] using your chosen data structure.

Question No6:

You are designing a high-performance in-memory system to check if a username already exists. The system must handle millions of usernames and provide near-instantaneous lookups, insertions (for

new user registrations), and deletions (for account closures). The usernames arrive in a random, unpredictable order.

a) Which data structure (BST, AVL tree, or B+ tree) would you choose to store the usernames in memory?

b) Justify your choice, explaining why it is superior to the others for this specific use case.

c) Write a C++ function bool doesUserExist(Node* root, string username) that performs the lookup.

Question No7:

You are developing a leaderboard for a popular online game. The system must support thousands of updates per second as players' scores change. It needs to perform three operations very quickly:

Update/Insert: Add a new player or update an existing player's score.

Lookup: Find a specific player's rank (which is based on their score).

Top-K Query: Display the top 10 players.

All leaderboard data is held in the server's memory for maximum speed.

a) Which data structure (standard BST, AVL tree, or B+ tree) is most appropriate for this in-memory leaderboard?

b) Justify your choice by explaining how it efficiently handles the required operations compared to the other two options.

c) Briefly describe the algorithm to retrieve the top 10 players using your chosen data structure.

Question No8:

A modern operating system needs to index all files on a hard drive by their "last modified" timestamp. This will power a search feature allowing users to quickly find all files modified within a specific date range (e.g., "all files modified between June 1 and June 15, 2025"). The number of files is in the millions, and the index itself is stored on the disk.

a) Which data structure would you use to build this index?

b) Explain why your choice is superior for this disk-based, range-query-intensive task.

c) A node in the tree becomes imbalanced with a balance factor of +2. Its left child has a balance factor of -1. What is this imbalance case called, and what sequence of rotations is needed to fix it?

Question No9:

Part a) Construct the BST by inserting the following elements in order.

[50, 30, 70, 20, 40, 60, 80, 35, 45, 65, 85]

Part b:

- What is the height of this BST? (height counted in terms of edges).
- How many leaf nodes are there? List them.
- How many internal nodes (non-leaf) are there?
- Is the BST complete? Explain briefly.

Part c:

- List the ancestors of node 65.
- List the ancestors of node 20.
- What are the descendants of node 30?
- What is the parent of node 45?
- What is the inorder successor of node 40?
- What is the inorder predecessor of node 60?

Question No10:

a) A binary tree has a root node, where the left subtree contains 500 nodes and the right subtree contains 200 nodes. For preorder, inorder, and postorder traversals, how many nodes will be visited *before* the root is processed?

b) A binary tree T has the following traversals:

- **Preorder:** P Q R S T U
- **Inorder:** R Q S P U T

Find the **postorder traversal** of T

c) The preorder and postorder traversals of a Binary Search Tree (BST) are given below:

- **Preorder:** 60, 35, 20, 45, 40, 75, 70, 90, 85, 95
- **Postorder:** 20, 40, 45, 35, 70, 85, 95, 90, 75, 60

If the root is at level 0, how many nodes are present at **level 2**?

Question No11:

Construct the follow AVL tree after the following operations

a) Insert([30, 20, 40, 10, 25, 50, 5, 75, 60])
b) Insert(55)
c) Delete(20)
d) Delete (60, 40)

Question No12:

Given a max degree of 5, construct a B+ after the following operations in order:

a)Insert([7, 10, 1, 23, 5, 15, 17, 9, 11, 39, 35, 8, 40, 25])

b) Delete(9)

c)Delete(7)