DS2030 Data Structures and Algorithms for Data Science
Week-6 Practice Set

September 23rd, 2025

1 Student Management System using AVL Trees

Problem Statement

You are developing a **Student Management System** for IIT Palakkad, where each student is assigned a unique Student ID (SID). Since the number of students grows rapidly each semester, the system must handle student records efficiently with fast insertion, search, and retrieval.
To ensure optimal performance, you will implement the system using an **AVL Tree**, which is a self-balancing Binary Search Tree. Each node in the AVL Tree will store the following information:

- **Student ID (SID)** – unique integer (acts as the key)
- **Student Name** – string
- **CGPA** – floating-point number
- **Degree Program** – string (e.g., B.Tech CSE, B.Tech ME, etc.)
- **Year** – integer (e.g., 2 for 2nd year)
- **Semester** – integer (e.g., 4 for 4th semester)
- **Subjects Enrolled** – list of strings (e.g., `["Data Structures", "DBMS", "Operating Systems"]`)
- **Height** – used for AVL balancing

Tasks

Implement the AVL-based Student Management System with the following functionalities:

1. **Insert** a new student record `(SID, Name, CGPA, Degree Program, Year, Semester, Subjects)` into the AVL tree.
2. **Search** for a student by `SID`.
3. **Update student details** by `SID`, including:
   - Add/remove subjects from the subject list.
   - Update CGPA (e.g., end of semester).
   - Update Year and Semester (progression).
4. **Delete** a student record by `SID`.
5. **Display all students** in sorted order of `SID` (in-order traversal).
6. **Display top k students** by `CGPA`.
7. **Display top k students** in a given `Degree Program` by `CGPA`.
8. **Count** total students currently in the system.
9. **Count** total students enrolled in a specific `Degree Program`.
10. **Count** total students enrolled in a specific `Subject`.
11. **Check AVL Balance**: Verify that the AVL tree is balanced after every operation.

2 Library Management System using AVL Trees

Problem Statement

You are developing a **Library Management System** for IIT Palakkad. The library contains a large and growing collection of books, so the system must handle operations efficiently. To achieve this, you will implement the library using an **AVL Tree**, which is a self-balancing Binary Search Tree. Each node in the AVL Tree will store the following information:

- **Book ID (BID)** – unique integer (acts as the key)
- **Book Title** – string
- **Edition** – integer
- **Height** – used for maintaining balance

Tasks

Implement the AVL-based Library Management System with the following functionalities:

1. **Insert** a new book record `(BID, Title, Edition)` into the AVL tree.
    - After each insertion, update heights and perform necessary rotations to keep the tree balanced.
2. **Search** for a book by `BID`.
    - If the book is found, display its details.
    - If not found, display `"Book not found."`
3. **Update** book details by `BID`.
    - Example: change the title or edition of a given book.
4. **Delete** a book by `BID`.
    - Ensure that the tree remains balanced after deletion.
5. **Display all books** in sorted order by `BID` using in-order traversal.
    - Output should include both the BID and book details.
6. **Check AVL Balance**: Implement a function `is_balanced(root)` that verifies whether the balance factor of every node is between –1 and +1.
    - Call this function after every insertion, update, and deletion to confirm your AVL implementation is correct.