NATIONAL UNIVERSITY
of Computer & Emerging Sciences

Project Report

Music Player: A Data Structures Approach

Group Members:

- Anas Ahmed (Roll No: 22k-4154)
- Ibrahim Junaid (Roll No: 22k-4563)
- Zain M. (Roll No: 22k-4502)

Date: 5- December-2023

Project Summary: Music Player

Problem Statement:

The project aims to create a feature-rich music player that allows users to manage their music collection, create playlists, and explore songs efficiently. The challenges include implementing data structures for efficient storage and retrieval of songs, creating playlists, and providing a seamless user experience.

Data Structure Concepts Used:

1. Linked List:

- The main list of songs is implemented as a circular linked list (mainlist class).
- Insertion, deletion, and traversal operations are supported for managing songs.

2. Hashing:

- A hash table (HashMap class) is used to index artists for quick retrieval of songs associated with each artist.
- Hashing enhances search efficiency and provides a streamlined approach for organizing the music library.

3. Binary Search Tree (AVL):

- An AVL tree (PlayList class) is employed to create smart playlists, allowing users to organize songs based on various criteria.
- AVL tree operations maintain a balanced structure for efficient searching and retrieval.

4. Queue:

- A queue is utilized in the display_playlist function to traverse the AVL tree level-wise and display playlists.

Efficiency Analysis:

a. Time Complexity:

- The use of AVL trees in creating playlists ensures logarithmic time complexity for search, insert, and delete operations.
- Hashing accelerates artist-based searches, providing constant time complexity on average.

b. Space Complexity:

- Circular linked lists efficiently use memory for storing songs, providing linear space complexity.
- AVL trees occupy space proportional to the number of songs in a playlist.

Differentiation from Existing Systems:

The project introduces a unique combination of data structures to enhance music management:

- **Efficient Search and Retrieval:**
- The use of hashing optimizes artist-based searches in the music library.
- AVL trees ensure quick access to songs in sorted or shuffled playlists.
- **Playlist Management:**
- Smart playlists created using AVL trees offer versatility in organizing and playing music.
- User-friendly file handling allows seamless creation, storage, and retrieval of playlists.

File Handling:

1. Mainlist:

- Songs are appended to the "allsongs.txt" file upon insertion into the main list.

- During initialization, the program reads "allsongs.txt" to populate the main list.

2. Playlists:

- Playlist names and corresponding songs are stored in separate files.
- Upon user selection, the program reads the playlist file to load songs into the AVL tree for playback.

Conclusion:

The music player project successfully combines various data structures to deliver an efficient and user-friendly application. The incorporation of file handling ensures persistence and seamless management of songs and playlists. The unique blend of data structures distinguishes this project in terms of search efficiency, playlist creation, and overall user experience.