Gestión de Lista de Reproducción de Canciones Usando Estructuras de Datos

Garambel Marin Fernando, Luque Condori Luis Guillermo, Quispe Arratea Alexandra Raquel[1],
[˙]*Universidad Nacional de San Agustın de Arequipa, Peru´,*
fgarambel@unsa.edu.pe, aquispearr@unsa.edu.pe , lluquecon@unsa.edu.pe

Abstract–

Keywords--Listas de reproducción, estructuras de datos, B+ tree, Trie.

I. Introducción

II. Keyword 1 (Marco Teórico)

A. Árbol B

Un árbol B + es una estructura de datos de tipo árbol auto-balanceado que mantiene los datos ordenados y permite búsquedas, inserciones y eliminaciones eficientes. Es una mejora del árbol B, donde todas las claves están almacenadas en las hojas del árbol y las demás nodos contienen sólo índices.

B. Trie

Un Trie es una estructura de datos especializada en la recuperación de datos, especialmente útil para manejar cadenas. Cada nodo del Trie representa un carácter de una cadena, y es eficiente para búsquedas de prefijos y coincidencias exactas.

C. Lista Enlazada

La lista enlazada es una estructura de datos lineal donde cada elemento es un objeto separado que contiene un puntero al siguiente elemento en la lista. Es adecuada para inserciones y eliminaciones rápidas.

III. Descripción del Proyecto

A. Características Clave del Programa

Agregar canciones a la lista de reproducción: Permite añadir nuevas canciones a la lista.
Eliminar canciones de la lista de reproducción: Permite eliminar canciones específicas de la lista.
Cambiar el orden de las canciones en la lista: Permite reorganizar las canciones moviéndolas de una posición a otra.
Reproducción aleatoria de canciones (opcional): Permite reproducir las canciones en un orden aleatorio.
Ajuste dinámico del tamaño de la lista de reproducción en memoria: La lista de reproducción se ajusta dinámicamente en tamaño conforme se agregan o eliminan canciones.
Interfaz gráfica (opcional): Proporciona una interfaz gráfica para una mejor experiencia del usuario.

B. Requisitos del Proyecto

Estructuras de Datos: El programa utilizará una o varias estructuras de datos implementadas durante el curso para almacenar las canciones. No se permite el uso de estructuras provenientes de las clases y/o colecciones de Java, a excepción de aquellas utilizadas exclusivamente para la interfaz gráfica si es necesario.

C. Funciones Principales

`agregar_cancion(cancion)`: Añade una canción a la lista de reproducción.
`eliminar_cancion(cancion)`: Elimina una canción de la lista de reproducción.
`cambiar_orden(posicion_actual, nueva_posicion)`: Cambia la canción de la posición actual a una nueva posición.
`reproduccion_aleatoria()`: Reproduce las canciones en orden aleatorio.
Manejo Dinámico de Memoria: La lista debe ajustarse dinámicamente en tamaño al agregar o eliminar canciones.
Diferentes vistas de las canciones: Permite ordenar las canciones por popularidad, año, o duración, en orden ascendente o descendente según la selección del usuario.

D. Conjunto de Datos de Prueba

E. *Estructura del Repositorio*

El repositorio de nuestro proyecto, EDA2024_5, está organizado de la siguiente manera:

```
EDA2024_5/
|
├── main.py
|
```

```
├── img/
│
├── model/
│   ├── __init__.py
│   └── song.py
│
├── test/
│   ├── __init__.py
│   ├── test_bplustree_manager.py
│   ├── test_file_manager.py
│   ├── test_hashmap_manager.py
│   ├── test_memory_manager.py
│   └── test_trie.py
│
└── util/
    ├── __init__.py
    ├── bplustree_manager.py
    ├── file_manager.py
    ├── gui_manager.py
    ├── hashmap_manager.py
    ├── memory_manager.py
    ├── playlist_manager.py
    └── trie.py
```

main.py*: Archivo principal del programa.*
carpeta img*: Contiene imágenes utilizadas en el proyecto.*
carpeta model*: Contiene archivos relacionados con el modelo de datos*
carpeta test*: Contiene archivos de pruebas*
carpeta util*: Contiene utilidades del proyecto.*

V.. Metodología

A. *Semejanzas*
B. *Diferencias*
C. *Discusión*
D. *Propuesta*

Conclusiones

Referencias

[1] Manuscript Templates for Conference Proceedings, IEEE. http://www.ieee.org/conferences_events/conferences/publishing/templates.html
[2] M. King, B. Zhu, and S. Tang, “Optimal path planning,” *Mobile Robots*,

Note caption is centered below figures, but above tables.

ANEXOS