Trabajo Práctico AYDA 1

Integrantes:

Jose Maria Goicoechea. joegoico@gmail.com

Bautista Ferraro. bautistaferraro82@gmail.com

Grupo numero: 2

Objetivo

El objetivo de este trabajo práctico es desarrollar un sistema básico de búsqueda para una biblioteca, en cual es posible verificar si un libro existe en la biblioteca, ver todos los libros que tiene la biblioteca cuyo año de lanzamiento pertenezca al rango de búsqueda y por último poder ver los 10 ejemplares más vendidos de un genero en especifico.
Para todo esto debemos implementar las estructuras de datos adecuadas de forma tal que sea lo más eficiente posible.

Implementaciones en TDA Nereus(Parte sintáctica)

CLASS Lista
IMPORTS Nat,Libro
BASIC CONSTRUCTORS inicLista, vincularPorAnio
EFFECTIVE
TYPE Lista
OPERATIONS
inicLista: -> Lista;

imprimirVendidos: Lista->Lista; O(n);
//imprime los 10 más vendidos de un género

vincularListaVendidos: Libro * Lista->Lista; O(n)
//agrega los libros a la lista ordenados de menor a mayor según su año.

masVendidos: Lista * string->Lista; O(n)
//crea la nueva lista de vendidos y llama a otras funciones para agregarlos a la lista

vincularPorAnio: Libro *Lista->Lista ;O(n)
//agrega a una lista los libros de mayor a menor según la cantidad de ejemplares vendidos.

existeLibro: string->Boolean; O(n)
//recorre la lista y devuelve true si encuentra al libro

imprimir: Libro->Libro; O(1)
//imprime el libro que se le pasa por parámetro

listarPorRango: natural* natural*Lista->Lista; O(n)

//lista los libros que se encuentran dentro del rango que ingresa el usuario.

END-CLASS

Describir cómo se combinaron los TDAs identificados para almacenar en memoria la colección de libros y géneros que abarca cada libro:

(Para la organización de todos los libros, decidimos incluir a todos estos en una lista simple ordenada por año de salida del libro, sobre la cual vamos trabajando y realizando las operaciones correspondientes).
CLASS Libro
IMPORTS Nat
BASIC CONSTRUCTORS inicLibro
EFFECTIVE
TYPE Libro
OPERATIONS

inicLibro:->Libro;

//este vendría a ser el constructor, el cual nosotros le pasamos todos los parametros necesarios.

```
Libro(string titulo, string autor, string editorial, int anio, string numeroEdicion, int paginas, int vendidos, string precio, string ISBN,string listaGeneros);
```

getTitulo->string ;
getAutor->string ;
getEditorial->string ;
getAnio->natural ;
getNumeroEdicion->string ;
getPaginas->natural ;
getVendidos->natural ;
getPrecio->string ;
getListaGeneros->string ;
getISBN->string;
/*estos métodos son los getters del libro, cada uno devuelve el atributo del libro que indica su nombre, la complejidad de cada uno de ellos es O(1)*/

END-CLASS

Servicios solicitados:

1-Dado un libro, verificar si existe dentro de la colección de libros del sitio. La búsqueda es por título, se permite usar comodines, por ejemplo “*” para una palabra" y “?” para un carácter. Los acentos deben ser ignorados.

Su complejidad temporal total es $O(n^2)$

Nuestra forma de pensar este servicio fue cargar en una lista simple ordenada por año a la información de todos los libros dados. A partir de ahí se le solicita un título al usuario, y este es buscado entre todos los títulos de los libros de la lista. Si el libro pertenece o no a la lista será informado en pantalla.

En este servicio, la estructura de datos es la mas practica, ya que si se hubiera elegido un árbol ordenado por años, había un problema ya que los años se pueden repetir, lo que no nos daría un árbol balanceado, por lo que el costo de buscar un libro según su año en el árbol o en una lista seria lo mismo.

En el caso de querer ordenar por título, el problema estaría cuando se usa un '*' para reemplazar la primera palabra, en ese caso perdemos el orden del árbol, por lo que habría que recorrerlo de igual manera.

En conclusión, no habría una ganancia utilizando otra estructura de datos en lugar de una lista simple.

2- Dado un rango de años, se deben listar todos los libros lanzados en ese lapso. De cada libro se debe mostrar la información general y los géneros que abarca.
Su complejidad temporal total es $O(n)$
Para la realización de este servicio es que decidimos que la lista de libros sea ordenada por año de salida de este. Para así entonces poder listar de una manera más cómoda y rápida, sobre la misma lista de libros, los libros deseados que pertenezcan a ese lapso.

3- Obtener los 10 libros más vendidos de un género determinado para poder ofrecerlos a los clientes.
Su complejidad temporal total es $O(n^2)$
Nuestra idea aquí fue pedir al usuario el género deseado, y de aquí ir recorriendo la lista de libros. A medida que vamos encontrando libros de ese género los vamos agregando ordenados por cantidad de vendidos a una segunda lista, sobre la cual luego imprimimos los primeros 10 libros de esta.

Conclusión

Estamos muy conformes con el trabajo realizado, con las implementaciones de los TDA en Nereus, tanto la parte sintáctica como semántica, además del funcionamiento de todos los servicios solicitados en la consigna.
Creemos que a partir de este trabajo hemos mejorado mucho en cuanto al manejo de estructuras, como los strings, las listas, y las clases con sus respectivos objetos.