Trabajo Práctico: Listas enlazadas y árboles

Estructuras de datos
Comision A (Turno Noche)
Comision B (Turno Mañana)
Primer cuatrimestre de 2020

Docentes	Sergio Gonzalez Ariel Clocchiatti
Email	sergio.gonzalez@unahur.edu.ar aclocchi@gmail.com
Fecha de entrega	10 / 07 / 2020

Índice

1. Introducción **2**
1.1. Presentación del problema 2

2. Objetivos **3**
2.1. Indexado para búsqueda 3

3. Datos de prueba **4**

4. Informe **4**

1. Introducción

En este trabajo práctico vamos a trabajar con la integración de los temas de la segunda parte de la materia, centrándonos en las estructuras de datos Lista enlazada y Árboles binarios de búsqueda. Van a tener que utilizar los conocimientos que fueron adquiriendo hasta ahora, para resolver un problema e implementar esta solución en el lenguaje de programación *Python*.

1.1. Presentación del problema

Los buscadores web logran darnos resultados de nuestras búsquedas muy rápidamente, teniendo en cuenta que deben recorrer toda la información almacenada en Internet para hacerlo (cerca de **4.5 mil millones** de páginas, con cientos de miles de palabras en cada una). Lo que ocurre es que el buscador no recorre toda la web cada vez que realiza una búsqueda, sino que recorre una estructura de datos gigante, en la que tiene almacenadas todas las palabras que estan dando vueltas en las páginas de Internet de todo el mundo. El proceso de creación y llenado de esta estructura se hace periódicamente y se llama ***indexado*** del buscador. Resumiendo, el proceso de indexado, consta de el recorrido de toda la web y el almacenado de la información en un árbol binario de búsqueda. En este trabajo práctico vamos a implementar este árbol binario y distintas operaciones para poder hacer nuestro propio buscador (Vamos por vos Google!!!!!!!!!!!!!!).

Para el modelado de la estructura, se deben tener en cuenta estas condiciones:

- El árbol se ordena usando las palabras que queremos indexar, es decir, las claves de los nodos del árbol son las palabras.
- En cada nodo del árbol, ademas de la palabra, vamos a tener una lista con las direcciones web de las páginas en las que aparece esa palabra.

Figura 1: Algunos de los motores de búsqueda actuales.

2. Objetivos

2.1. Indexado para búsqueda

Definir un TDA que represente al árbol binario de búsqueda para almacenar las palabras que necesita el buscador. Cada vez que necesiten una **Lista** deben utilizar su propio TDA "Lista Enlazada", la implementación que hicieron en la guía de ejercicios (Recursiva o Iterativa, la que quieran), si necesitan alguna operación que no tienen en ese TDA, deben agregarla a la implementación. **No utilizar la implementación de Listas de Python**

El TDA ***ArbolBuscador*** debe incluir las siguientes operaciones:

- **insertarPalabra(palabra, direccionWeb):** Inserta la palabra en el árbol, si la palabra ya existe en el árbol, agrega la dirección web a la lista de páginas en la que está esa palabra (se debe verificar si la dirección web no está en la lista para no duplicar información). Si la palabra no existe en el árbol, agrega un nuevo nodo con la nueva palabra y la dirección web de la página en el lugar correspondiente.
- **insertarPagina(listaDePalabras, direccionWeb):** Recibe una lista de palabras y la direccion web de la página a la que pertenecen y las inserta en el árbol.
- **buscarPalabras(listaPalabras):** Recibe una lista con las palabras buscadas. Retorna una lista con las direcciones web de las páginas en las cuales están **todas** las palabras de la lista de entrada (esas palabras pueden estar en más de una página). Si no hay ninguna página que contenga **todas** las palabras, debe retornar una lista vacía.
- **palabrasDePagina(direccionWeb):** Recibe la dirección web de una página y retorna la lista de todas las palabras en esa página. Si no hay ninguna palabra de la página almacenada en el árbol, retorna una lista vacía.
- **eliminarPalabra(palabra):** Elimina del árbol la palabra que recibe por parámetros.
- **eliminarPagina(direccionWeb):** Elimina del árbol la dirección web que recibe por parámetro. Debe eliminarla de todos los nodos del árbol donde se encuentre.
- **cantidadTotalPalabras(cantidadLetras):** Recibe una cantidad de letras por parámetro y retorna la cantidad **total** de palabras almacenadas en el árbol que tienen esa cantidad de letras o más.
- **estaBalanceado():** Retorna *True* si el árbol esta balanceado y *False* en caso contrario. Un árbol esta balanceado cuando la diferencia de altura entre los subárboles hijos es menor o igual a 1 (Solo para el nodo raíz del árbol).
- **paginasEnNivel(nivel):** Recibe un nivel y retorna una lista con las direcciones web de las páginas de las palabras que estan en ese nivel del árbol. La lista no debe tener direcciones web repetidas. Si en ese nivel no hay nada, retorna una lista vacía.
- **cantidadPalabrasMasUsadas(cantidadPaginas):** Recibe una cantidad de páginas por parámetro y retorna la cantidad de palabras almacenadas en el árbol que se encuentran en esa cantidad de páginas o más.
- **internasMayusculaAlfabetico():** Retorna una lista con las palabras que se encuentran en un nodo interno del árbol (no hojas) y tienen su primer letra en mayúscula. Las palabras deben estar ordenadas en orden alfabético.

3. Datos de prueba

Previo a la entrega del trabajo práctico, deben controlar que su implementación funcione con un lote de datos de prueba que nosotros les vamos a entregar. El lunes 6 de julio, subiremos al webcampus el lote de datos de prueba, junto con un programa en *Python* para ejecutarlo. Por favor, respetar los nombres de las operaciones y de los TDA (***Lista*** y ***ArbolBuscador***).

4. Informe

La entrega del trabajo práctico debe ser con un informe escrito, incluyendo:

- Descripción de cada una de las **estructuras de datos** diseñadas e implementadas. Incluir una descripción escrita de los algoritmos. Pueden incluir diagrama de flujo.
- Descripción de la implementación en *Python*. Explicar **claramente** que hace cada función y procedimiento implementados.
- Código completo de la implementación.
- Luego de la entrega, les vamos a hacer preguntas a las/los integrantes del grupo sobre el trabajo, asi que les aconsejamos no copiar código que encuentren en internet.