UNIDAD TEMÁTICA 2 – ARBOLES GENÉRICOS, TRIES y ARBOLES B

Trabajo de Aplicación 2

EJERCICIO 0 (preguntas de discusión proyectadas en pantalla)

EJERCICIO 1 (referencia Ejercicio Individual Domiciliario #2) (30 minutos)

Se desea construir un índice de palabras de un libro. Para ello, recorriendo el texto del mismo se han de insertar las palabras en una estructura de trie, indicando, para cada palabra, la(s) páginas del libro en que se encuentra la misma.

Utilizando lo desarrollado en el ejercicio domiciliario, el Equipo ha de consolidar una propuesta que tenga:

1. La descripción (**esquema**) de las estructuras de datos necesarias para construir un trie para el alfabeto compuesto por las letras a..z . **Dibujarla** en un **POSTER**.
2. Dibujo del trie correspondiente al siguiente conjunto de datos de prueba **(POSTER)**:

(en cada línea aparece la palabra, seguida por las páginas en que se encuentra en el libro, separadas por comas)

Ala, 1, 3, 88

Alimaña, 11, 22

Alabastro, 4

Perro, 5, 8

Pera, 7,12

Alimento, 9

Casa, 11,13

Casada, 1

Cazar, 33

Programa, 22, 67

Programación, 15

Programar 15,16

3. Un algoritmo para, a partir de un texto, construir el índice del mismo en el trie, indicando las páginas en que aparecen las palabras.
4. Un algoritmo para, dado el trie, buscar una cadena y, si existe, devolver los números de páginas del libro en los que esa palabra se encuentra

El Equipo realizará en conjunto las actividades **1** y **2**. Se ha de separar en sub-equipos para desarrollar las actividades **3** y **4**, cuyos resultados luego serán integrados. Responder las preguntas presentadas en pantalla.

EJERCICIO 2 (30 minutos)

ESCENARIO

Se desea crear un programa que permita indizar todas las palabras almacenadas en múltiples textos digitales. Este programa ha de tener una funcionalidad de búsqueda de las palabras en forma parcial, incremental, similar a las búsquedas de google o al "autocompletar" de muchos editores de código fuente.

El analista del proyecto ha definido que la mejor forma de representar este problema es mediante el uso de un **TRIE**, dada la eficiencia del mismo para la búsqueda de strings y, particularmente, de prefijos, permitiendo una búsqueda de patrones incremental.

De las funcionalidades que debe tener el sistema a desarrollar, el analista ha asignado a su grupo de desarrolladores las siguientes:

- Insertar una nueva palabra en este diccionario (se hará para todas las palabras de los textos contenidos)
- Buscar las ocurrencias de una cierta palabra.

Se requiere: Implementar un método que permita **buscar una clave**, y que como resultado **indique si esta clave está o no en el árbol**, y la **cantidad de comparaciones** realizadas para ello (devolver 0 si la clave no se encuentra, o el número de comparaciones realizadas si efectivamente está en el árbol).

Se proveen las clases Java TArbolTrie.java y TNodoTrie.java.

Mediante las mismas es posible crear un trie e insertar las palabras que se pasarán en un archivo como entrada.

PARA PROBAR LA EJECUCIÓN CORRECTA: SE UTILIZARÁ EL ARCHIVO **"PALABRAS.TXT"** PROVISTO POR LA CÁTEDRA, PARA INSERTAR EN EL **TRIE**. SE REALIZARÁN VARIAS BÚSQUEDAS, CON Y SIN ÉXITO. SE DEBE IMPRIMIR EN CONSOLA EL RESULTADO DE CADA BUSQUEDA.

1. Armar el paquete UT2TA2 y agregar las clases provistas **(5 minutos)**
2. En 2 subequipos, desarrollar 2 versiones del método "buscar" solicitado, luego seleccionar la mejor implementación y ponerla en el repositorio global **(20 minutos)**
3. Sincronizar las computadoras de los integrantes del Equipo con el repositorio **(4 minutos)**

Responder las preguntas presentadas en pantalla

CALIFICACIÓN: se calificará todo lo sincronizado en el repositorio GIT del Equipo hasta la hora 21:15

```
public class TArbolTrie {
	private TNodoTrie raiz;

	public void insertar(String palabra) {
		if (raiz == null) {
			raiz = new TNodoTrie();
		}
		raiz.insertar(palabra);
	}

	public void imprimir() {
```

```
            if (raiz != null) {
                  raiz.imprimir();
            }
      }

// COMPLETAR EL MÉTODO DE BUSCAR, COMO INDICA LA LETRA DEL EJERCICIO.

      public static void main(String[] args){
            // Crear una instancia de un arbol Trie.
            // Leer un archivo palabras.txt
            // Para cada palabra encontrada, insertarla en el Trie
            // Por último, imprimir el trie.


            // Ejemplo de uso del Trie.
            TArbolTrie trie = new TArbolTrie();
            trie.insertar("casa");
            trie.insertar("casamiento");
            trie.insertar("arbol");
            trie.insertar("grito");
            trie.imprimir();

            System.out.println(trie.buscar("casamientos"));
      }
}

public class TNodoTrie {
      private static final int CANT_CHR_ABECEDARIO = 26;
      private TNodoTrie[] hijos;
      private boolean esPalabra;

      public TNodoTrie() {
            hijos = new TNodoTrie[CANT_CHR_ABECEDARIO];
            esPalabra = false;
      }

      public void insertar(String unaPalabra) {
            TNodoTrie nodo = this;
            for (int c = 0; c < unaPalabra.length(); c++) {
                  int indice = unaPalabra.charAt(c) - 'a';
                  if (nodo.hijos[indice] == null) {
                        nodo.hijos[indice] = new TNodoTrie();
                  }
                  nodo = nodo.hijos[indice];
            }
            nodo.esPalabra = true;
      }

      private void imprimir(String s, TNodoTrie nodo){
            if (nodo != null) {
                  if (nodo.esPalabra) {
                        System.out.println(s);
                  }
                  for (int c = 0; c < CANT_CHR_ABECEDARIO; c++) {
                        if (nodo.hijos[c] != null) {
                              imprimir(s + (char) (c + 'a'), nodo.hijos[c]);
                        }
                  }
            }
      }

      public void imprimir() {
            imprimir("", this);
      }
// COMPLETAR EL MÉTODO DE BUSCAR, COMO INDICA LA LETRA DEL EJERCICIO.
}
```

EJERCICIO 3 (30 minutos)

Continuando con el scenario del ejercicio 2, se desea construir un índice, que permita, dada una palabra, indicar en qué páginas se encuentra.
Se provee un archivo “palabras-paginas.txt” que contiene, en cada línea, una palabra y las páginas del libro en que ésta se encuentra.

Se requiere:

1- Modificar las estructuras provistas de TNodoTrie para incluir, en cada palabra, una lista de páginas (se puede utilizar una LinkedList)
2- Instanciar un TArbolTrie para el alfabeto considerado, e insertar las palabras del archivo “palabras-paginas-txt”
3- Imprimir por consola (en preorden) todas las palabras del trie, indicando las páginas en que se encuentran en el libro