UNIDAD TEMÁTICA 7 – ARBOLES GENÉRICOS Y TRIES

PRACTICOS DOMICILIARIOS INDIVIDUALES - 3

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.

EJERCICIO 1

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.

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).

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.

```java
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.

```java
	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.

```
}
```