# UNIDAD TEMÁTICA 5 – ARBOLES BINARIOS AVL y ÁRBOLES ÓPTIMOS

## Trabajo de Aplicación 9

### Contexto

Los árboles binarios de búsqueda óptima son adecuados para aplicaciones de búsqueda estática, es decir, que no se realizan inserciones ni eliminaciones en el árbol (o sea que el árbol no cambia una vez que está construido).

Por ejemplo, si se desea desarrollar un analizador sintáctico para un cierto lenguaje de programación, se pueden almacenar en un árbol binario de búsqueda óptima las palabras reservadas del lenguaje. En tiempo de compilación, el analizador, entre otras cosas, recorrerá el código fuente del programa desarrollado y necesitará acceder a las palabras reservadas, pero también a todos los identificadores encontrados en el código que no sean palabras reservadas, en forma eficiente.

Para calcular las frecuencias de búsqueda, una solución es leer muchos miles -o millones- de líneas de código fuente y contar cada una de las ocurrencias. Una vez calculadas de esta forma las frecuencias exitosas e infructuosas, se puede armar el ABO correspondiente.

### Consigna

Desarrollar una aplicación que cuente con qué frecuencia aparece cada palabra de un conjunto de palabras clave en un cierto texto, y con qué frecuencia las palabras del texto no aparecen en ese conjunto de palabras claves.

Para ello, se debe:

1. Leer un archivo de palabras claves, y con cada una de ellas armar un árbol binario de búsqueda.
2. Leer un texto, y por cada palabra de ese texto, buscarla en ese árbol binario de búsqueda, y:
    a. Si la búsqueda tiene éxito, incrementar un campo de búsqueda exitosa en el elemento
    b. Si la búsqueda no es exitosa por su subárbol izquierdo, incrementar un campo de búsqueda infructuosa izquierda en el elemento
    c. Si la búsqueda no es exitosa por su subárbol derecho, incrementar un campo de búsqueda infructuosa derecha en el elemento
3. Instanciar un vector de frecuencias exitosas, un vector de frecuencias no exitosas y un vector de claves, para ser usados como entradas en la confección del árbol óptimo.
4. Recorrer el árbol para cargar los valores de esos vectores a partir de los valores contenidos en cada elemento.
5. A partir de los vectores obtenidos, construir el árbol binario de búsqueda óptima de las palabras claves

# Trabajo de Aplicación 9

## Ejercicio 1

Se ha de trabajar en sub-equipos.

### *SUB-EQUIPO "A"*

Para el punto 2 de la consigna, desarrollar -en seudocódigo- un algoritmo que cuente las frecuencias, respetando las siguientes firmas:

En **TArbolBB**

> **void cuentaFrec (de tipo clave unArgumento)**: incrementa el campo de frecuencia que corresponda a ese argumento de búsqueda.

En TElementoAB

> **void cuentaFrec (de tipo clave unArgumento)**: incrementa el campo de frecuencia que corresponda a ese argumento de búsqueda.

Para ello, deben agregarse tres atributos al elemento:

> **frecExito**: se incrementa cada vez que la búsqueda es exitosa
>
> **frecNoExIzq**: se incrementa cada vez que la búsqueda no es exitosa por su sub árbol izquierdo
>
> **frecNoExDer**: se incrementa cada vez que la búsqueda no es exitosa por su sub árbol derecho

**NOTA**: los atributos **frecNoExIzq** y **frecNoExDer solamente tienen sentido y utilidad cuando el correspondiente HijoIzquierdo o HijoDerecho ES NULO.**

### *SUB-EQUIPO "B"*

Para el punto 4 de la consigna, desarrollar -en seudocódigo- un algoritmo que recorra el árbol y complete los vectores, respetando las siguientes firmas:

En TArbolBB

> **void completaVectores (de tipo clave[] claves, de tipo entero[] frecExito, de tipo entero[] frecNoExito)**: completa los vectores correspondientes.

En TElementoAB

> **void completaVectores (de tipo clave[] claves, de tipo entero[] frecExito, de tipo entero[] frecNoExito, de tipo entero[] indiceFE, indiceFNE)**: completa los vectores correspondientes.

**MUY IMPORTANTE: COMENZAR CON LA DESCRIPCION EN LENGUAJE NATURAL, PRE Y POST-CONDICIONES!!!!**

**NOTA: tener en cuenta que, para los valores de las frecuencias de búsquedas sin éxito, los campos correspondientes del TElementoAB sólo tienen sentido cuando el correspondiente HijoIzquierdo o HijoDerecho son nulos!!!!**