Trabajo Práctico 2

Daniela Alban, Sebastian Cherny, Gabriel Krimker y Agustin Vera.

3 de julio de 2019

Construímos un léxico usando una estructura de árbol. Consideramos las palabras caminos que van desde el nodo raíz hasta los nodos que tienen un True en $finDePalabra$, donde las flechas serán las letras.

Clase Nodo

Le definimos un booleano $finDePalabra$ para saber si termina un camino en ese nodo y un diccionario $referencia$ el cual contiene a las flechas que apuntan a otro nodo.

La estructura de representación del nodo es

$$< finDePalabra : \mathbb{B},\ referencia : diccionario < letra, Nodo >> .$$

El invariante de representación del TAD Nodo está dado por la condición de que cada letra que esté en el diccionario haga referencia a un nodo.

La construcción de un nuevo Nodo:

$n \leftarrow Nodo(x)$

El acceso a los campos del nodo n:

$n.finDePalabra = x$ (donde x es booleano)

$n.referencia = y$ (donde y es un diccionario que relaciona cada letra con un nodo)

Clase Lexico

La estructura de representación es

$$< raiz : ref(Nodo),\ cantidadDePalabras : \mathbb{Z} >$$

El invariante de representación del TAD LEXICO está dado por las siguientes condiciones: en primer lugar $cantidadDePalabras$ coincide con la cantidad de nodos que tienen True en $finDePalabra$; por otro lado, debe tener forma arbórea (es decir, no puede haber ciclos y cada nodo recibe una única flecha) donde cada flecha se relaciona con un nodo; y finalmente, $raiz$ debe apuntar al primer nodo del árbol (es decir a la raíz).

Los algoritmos de las operaciones del TAD Léxico:

Crear un léxico:

El constructor del léxico va a tener una referencia al primer nodo del árbol y otra a la cantidad de palabras del léxico.

NuevoLexico() $\rightarrow$ Lexico:

$L \leftarrow NuevoLexico()$

$L.cantidadDePalabras = k$ (donde k es un número entero)

$L.raiz = n$ (donde n es un nodo)

Como crear un léxico es asignarle un valor a $l.cantidadDePalabras$ y un nodo a $l.raiz$, tiene orden O(1).

Cantidad de palabras:

CantidadDePalabras(L) $\rightarrow \mathbb{Z}$:

$RV \leftarrow L.cantidadDePalabras$

Como Lexico tiene guardada la cantidad de palabras en $L.cantidadDePalabras$, la función $CantidadDePalabras$ devuelve ese valor en O(1).

Existe:

Existe(L,P) $\rightarrow \mathbb{B}$:

$letra \leftarrow L.raiz$ O(1)

$i \leftarrow 0$ O(1)

while ($i < longitud(P) \wedge$ la letra i-ésima de la palabra P esté relacionada con un Nodo en $letra.referencia$) { O(1) while: O(longitud(P)) iteraciones

cambiar $letra$ por el nodo que está relacionado con la i-ésima letra de P en $letra.referencia$ O(1)

$i \leftarrow i + 1$ O(1)

}

$RV \leftarrow (i = longitud(P) \wedge letra.finDePalabra)$ O(1)

Agregar Palabra:

AgregarPalabra (L,P):

If not $(L.Existe(P))\{$ O(longitud(P))

if $(longitud(P) \neq 0)\{$ O(1)

$i \leftarrow 0$ O(1)

$letra \leftarrow L.raiz$ O(1)

while$(i < longitud(P))\{$ O(1) while: O(longitud(P)) iteraciones

if (la letra i-ésima de P no está relacionada con un Nodo en $letra.referencia$)$\{$ O(1)

crear $Nodo(False)$ y relacionar con la i-ésima letra de P en $letra.referencia$ O(1)

}

cambiar $letra$ por el nodo relacionado con la i-ésima letra de P en $letra.referencia$ O(1)

$i \leftarrow i + 1$ O(1)

}

$letra.finDePalabra \leftarrow$ True O(1)

else

$L.raiz.finDePalabra \leftarrow$ True O(1)

$L.cantidadDePalabras \leftarrow L.cantidadDePalabras + 1$ O(1)

}

Eliminar Palabra:

L.EliminarPalabra(P):

If ($Existe(P)$){ O(longitud(p))

$letra \leftarrow L.raiz$ O(1)

$i \leftarrow 0$ O(1)

while ($i < longitud(P)$){ O(1) while: O(longitud(P)) iteraciones

cambiar $letra$ por el nodo relacionado con la i-ésima letra de P en $letra.referencia$ O(1)

$i \leftarrow i + 1$ O(1)

}

letra.finDePalabra $\leftarrow$ False O(1)

$L.cantidadDePalabras \leftarrow L.cantidadDePalabras - 1$ O(1)

}