

Tarea 3

Manejo dinámico de estructuras lineales y arborescentes

Curso 2022

1. Introducción y objetivos

Esta tarea tiene los siguientes objetivos específicos:

- Trabajar sobre el manejo dinámico de memoria, incorporando estructuras arborescentes.
- Trabajar con **árboles binarios de búsqueda**.
- Implementar funciones recursivas.

Se debe recordar que:

- **Como cada tarea, es individual y eliminatoria.**
- **La evaluación se hace con casos de prueba que se publican al terminar el plazo de entrega.**
- **Quien no apruebe en la entrega podrá realizar una re-entrega en un plazo de hasta 24 horas después de publicado el resultado.**
- **La aprobación en la entrega otorga 2 puntos; la aprobación en la re-entrega no otorga puntos.**
- **La tarea debe compilar y funcionar correctamente con la regla make testing en las máquinas de la Facultad.**
- **El archivo a entregar se debe generar mediante la regla make entrega y no se le debe cambiar el nombre en el proceso de entrega.**

2. Materiales

Los archivos para la tarea se extraen de *MaterialesTarea3.tar.gz*. Para conocer la estructura de los directorios ver la Sección **Materiales** de [Estructura y funcionamiento del Laboratorio](#).

En esta tarea los archivos en el directorio `include` son **utils.h**, **info.h**, **palabras.h**, **cadena.h**, **iterador.h**, **abb.h** y **aplicaciones.h**. Estos archivos no se pueden modificar.

En el directorio `src` se incluyen ya implementados **utils.cpp**, **info.cpp** y **palabras.cpp**. Además, se incluyen los archivos **plantilla_cadena.cpp**, **plantilla_iterador.cpp**, **plantilla_abb.cpp** y **plantilla_aplicaciones.cpp**. Estos cuatro archivos contienen las funciones que se deben implementar pero con lo mínimo para que pueda compilar. Se pueden usar para lo cual se les debe cambiar el nombre (esto es remover `plantilla`), poner la cédula (7 dígitos) y completar el cuerpo de las funciones y el struct.

3. Desarrollo

Ver la Sección **Desarrollo** de [Estructura y funcionamiento del Laboratorio](#).

En esta tarea se deben implementar los archivos **cadena.cpp**, **iterador.cpp**, **abb.cpp**, y **aplicaciones.cpp**.

La generación del ejecutable se hace compilando los archivos implementados **cadena.cpp**, **iterador.cpp**, **abb.cpp**, **aplicaciones.cpp**, **info.cpp**, **utils.cpp**, **palabras.cpp** y **principal.cpp** y enlazando los archivos objeto obtenidos.

El archivo **Makefile** (ver el material disponible sobre [Makefile](#)) provee para la compilación la regla **principal**, que es la predeterminada. Por lo tanto el ejecutable se obtiene mediante

```
$ make
```

Se sugiere probar cada uno de los test provistos y al final confirmar mediante la regla **testing**:

```
$ make testing
```

Se debe verificar la compilación y ejecución en las máquinas de Facultad ya que es con esas máquinas con las que se hace la evaluación.

4. Entrega

Ver la Sección **Entregas** de [Reglamento del Laboratorio](#).

4.1. Plazos de entrega

El plazo para la entrega es el **lunes 25 de abril a las 15:00**.

4.2. Archivo a entregar y procedimiento

Se debe entregar el archivo **Entrega3.tar.gz**, que contiene los módulos a implementar **cadena.cpp**, **iterador.cpp**, **abb.cpp** y **aplicaciones.cpp**.

Este archivo se obtiene al ejecutar la regla entrega del archivo *Makefile*:

```
$ make entrega
```

Con esto se empaquetan los módulos implementados y se los comprime.

El archivo a entregar **DEBE** ser generado mediante este procedimiento. Si se lo genera mediante alguna otra herramienta (por ejemplo, usando un entorno gráfico) **la tarea no será corregida**, independientemente de la calidad del contenido. Tampoco será corregida si el nombre del archivo se modifica en el proceso de entrega. Si alguna de estas dos irregularidades ocurre en la entrega la calificación será *Insuficiente* lo que obligará a hacer la reentrega. Si ocurre en la reentrega la calificación será *No Aprobado* lo que implicará la pérdida de la tarea, y por lo tanto del curso.

El archivo que queda en el receptor debe poder descomprimirse con el comando:

```
$ tar zxvf Entrega3.tar.gz
```

4.3. Identificación de los archivos de las entregas

Cada uno de los archivos a entregar debe contener, en la primera línea del archivo, un comentario con el número de cédula del estudiante, sin el guión y sin dígito de verificación.

Ejemplo:

```
/* 1234567 */
```

4.4. Individualidad

Ver la Sección **Individualidad** de [Reglamento del Laboratorio](#).

5. Descripción de los módulos y algunas funciones

En esta sección se describen los módulos que componen la tarea. Estos son *utils*, *info*, *cadena*, *iterador*, *abb*, *aplicaciones* y *principal*.

El módulo *colCadenas* de las tareas anteriores no se incluye en esta tarea, pero se volverá a incluir en las siguientes.

5.1. Módulo *utils*

Igual al de la tarea anterior.

5.2. Módulo *info*

Igual al de la tarea anterior.

5.3. Módulo *cadena*

Igual al de la tarea anterior.

5.4. Módulo *iterador*



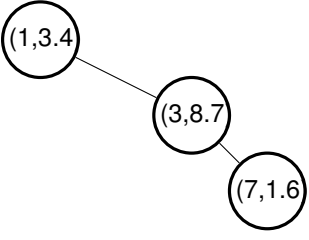
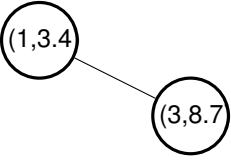
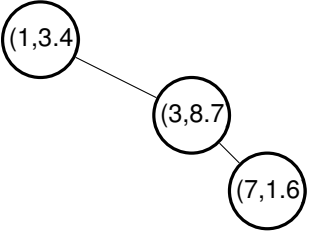
Igual al de la tarea anterior.

5.5. Módulo *abb*

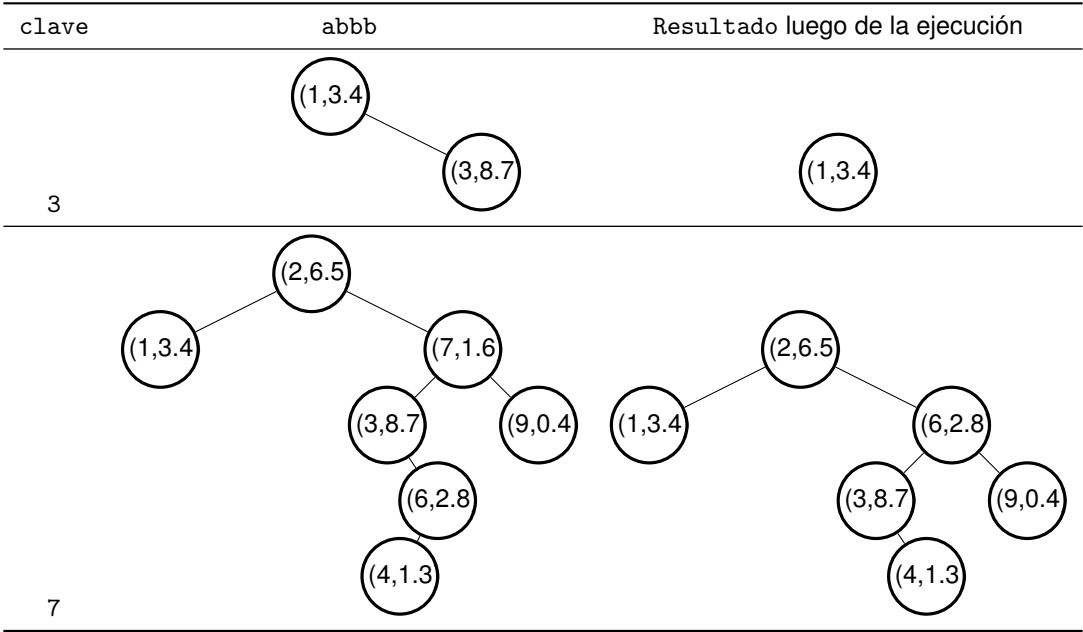
En este módulo se implementará un árbol binario de búsqueda (de tipo *TAbb*) de elementos de tipo *TInfo*. La propiedad de orden de los árboles es definida por el componente natural de sus elementos.

A continuación presentaremos mediante ejemplos algunas funciones representativas del módulo **abb**. El árbol vacío se representa con ●.

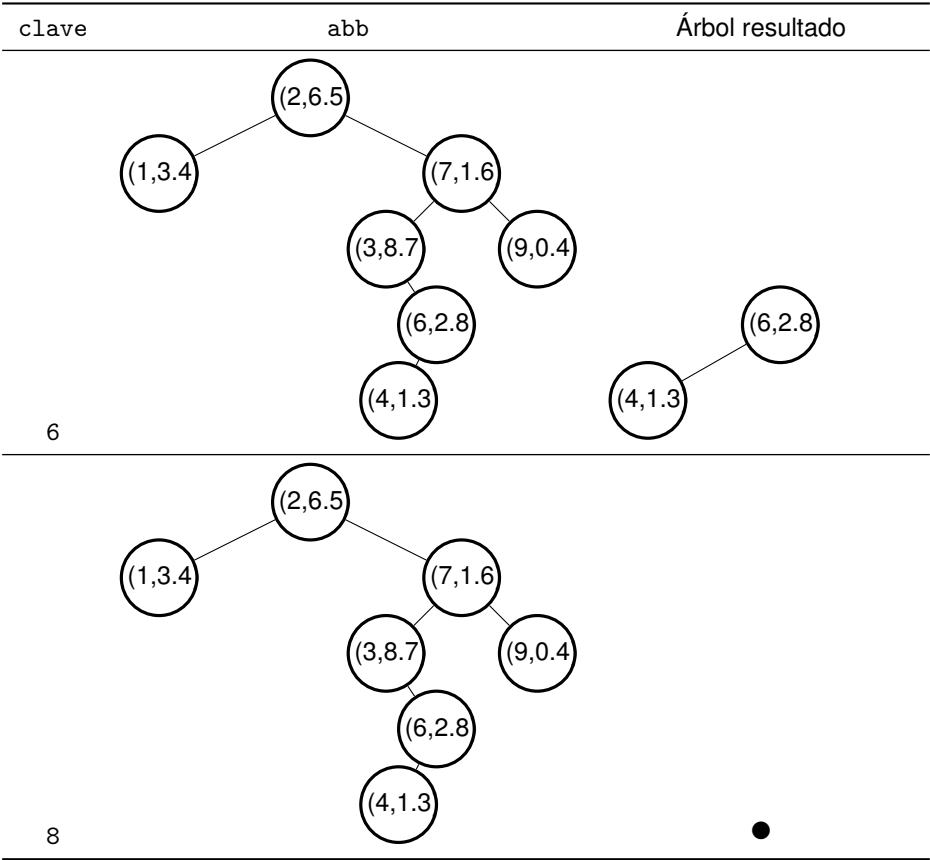
5.5.1. *insertarEnAbb (TInfo dato, TAbb abb)*

dato	abb	Resultado luego de la ejecución
		
(1,3.4)	●	
		
(7,1.6)		

5.5.2. *removerDeAbb (nat clave, TBinario abb)*



5.5.3. *buscarSubarbol (nat clave, TAbb abb)*



5.6. Módulo *palabras*

Este módulo está implementado y no es parte de la entrega.
Mantiene un conjunto de palabras mediante un árbol general cuyo elementos son letras. Un camino que sale de la raíz representa una palabra o un prefijo de palabra. La palabra o prefijo consiste en la concatenación

de las letras contenidas en los nodos del camino. La letra de la raíz del árbol no se toma en cuenta, o sea, no es parte de ninguna palabra. La letra de las hojas, que es $\backslash 0'$, tampoco se considera parte de las palabras, sino que es el terminador de palabra. Esto es, un camino representa una de las palabras del conjunto si su nodo más profundo tiene un hijo cuya letra es $\backslash 0'$.

Un prefijo de palabra es alguna subsecuencia al inicio de la palabra. Una palabra es prefijo de sí misma. Entonces el nodo más profundo de un camino que sale de la raíz, aunque no tenga a $\backslash 0'$ como hijo, corresponde a la última letra de un prefijo, y ese camino representa al prefijo.

Por ejemplo, en el árbol de la Figura 1, se mantienen las palabras *a*, *al*, *cal*, *can*, *la*, *las*, *lo*, *los*, *losas*. Todas ellas son prefijos de sí mismas y además, por ejemplo *la* es prefijo de *las*. En cambio *c*, *l*, *losa* son prefijos, pero no son palabras del árbol.

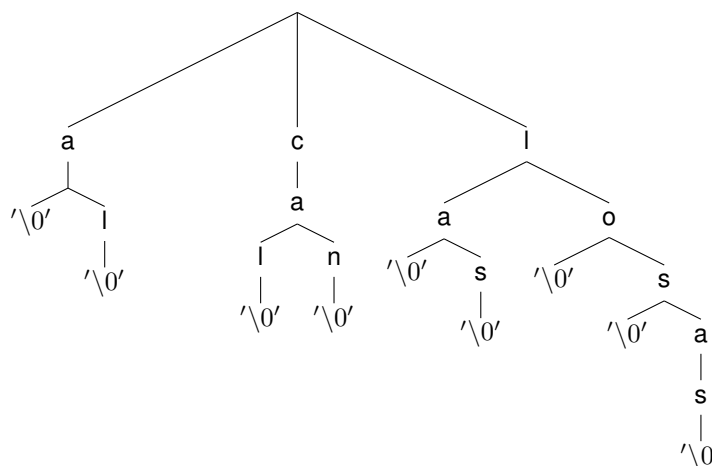


Figura 1: Ejemplo de árbol de palabras.

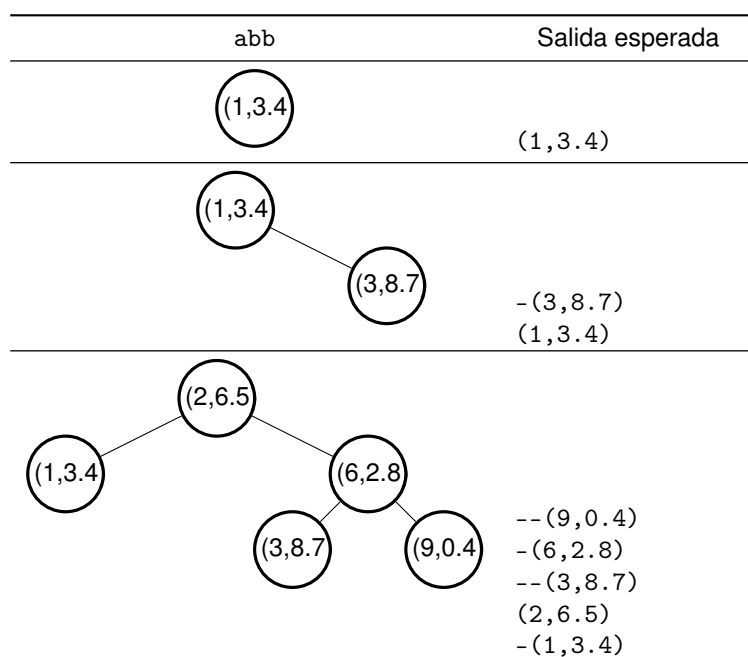
El árbol de palabras se implementa mediante un árbol binario con semántica *primer hijo* — *siguiente hermano*. Los hijos de cada nodo están ordenados de manera creciente. El símbolo $\backslash 0'$ es menor que cualquier otra letra.

En el ejemplo, el segundo hijo de la raíz es un nodo que tiene un campo con la letra *c*, un campo *sigHermano* que apunta al nodo con etiqueta *l*, y un campo *primerHijo* que apunta a uno de los nodos con etiqueta *a*.

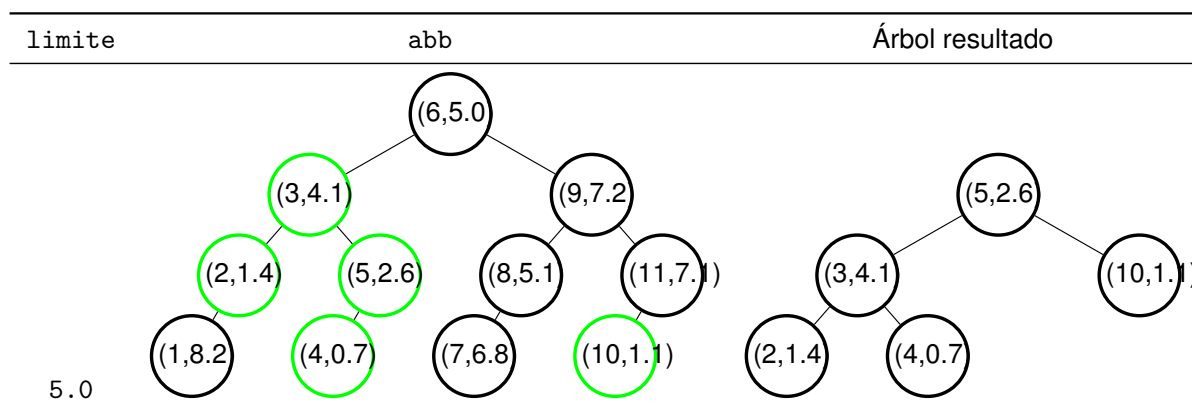
5.7. Módulo aplicaciones

Consiste en funciones que se implementan usando las operaciones de los otros módulos. No se incluyen las operaciones de la tarea anterior, las cuales podrán volver a incluirse en tareas siguientes.

5.7.1. *imprimirAbb (TAbb abb)*



5.7.2. *menores (double limite, TAbb abb)*



Cuadro 1: Ejemplo de llamada a la función menores

En el Cuadro 1 se muestra un ejemplo de llamada a la función `menores`. En dicho ejemplo se destacan en verde los nodos en donde se cumple la condición, cuyas copias serán los nodos del árbol resultado.

El resultado de la operación en el árbol que tenga como raíz uno de los nodos que cumplen con la condición, es un árbol que en cuya raíz está una copia de ese nodo. Los dos subárboles son el resultado de la operación aplicada en los hijos de ese nodo.

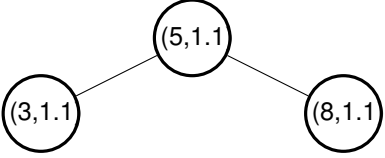
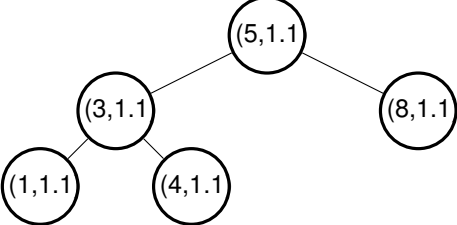
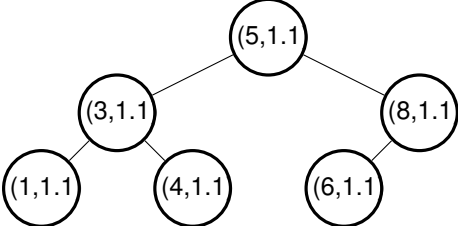
En los nodos cuyos elementos son (1,8.2) y (7,6.8) no se cumple la condición y no tienen descendientes. En (8,5.1) no se cumple y tampoco se cumple en sus descendientes. Por lo tanto el resultado del filtrado en cualquiera de esos nodos es el árbol vacío.

En (11,7.1) no se cumple la condición y no hay nodos en su subárbol derecho. El resultado en este nodo es, entonces, el resultado en su hijo izquierdo, que es el árbol cuyo único nodo es una copia de (10,1.1), por ser el único nodo que cumple con la condición en el subárbol izquierdo. De manera análoga, ni en (9,7.2) ni en su subárbol izquierdo hay nodos que cumplan la condición, por lo que el resultado en (9,7.2) es el resultado en su hijo derecho.

En la raíz del árbol, (6,5,0), no se cumple la condición pero en sus dos subárboles hay nodos que la cumplen. En el resultado la raíz debe ser la copia de algún otro nodo y el árbol devuelto debe mantener la propiedad de orden. Se elige como raíz una copia del nodo mayor (según el orden definido) de los que cumplen la condición en el subárbol izquierdo, que es (5,2,6). Este nodo debe removerse del resultado de la llamada en (3,4,1) para pasar a ser la raíz del árbol resultado.

5.7.3. *esPerfecto (TAbb abb)*

Un árbol binario es perfecto si todas las hojas están en el mismo nivel y los dos subárboles de cada nodo interno son no vacíos.

abb	Resultado esperado
●	true
	true
	false
	false

Se debe notar que:

- El árbol vacío es perfecto.
- En el tercer ejemplo los dos subárboles de cada nodo interno son no vacíos pero no todas las hojas están en el nivel más profundo.
- En el cuarto ejemplo todas las hojas están en el nivel más profundo pero hay un nodo cuyo subárbol derecho es vacío.

5.7.4. caminoAscendente (nat clave, nat k, TAbb abb)

clave	k	abb	Resultado esperado
2	2		[2]
7	0		[]
6	5		[6, 3, 7, 2]
4	3		[4, 6, 3]

5.7.5. imprimirPalabrasCortas (nat k, TPalabras palabras)

Se deben imprimir las palabras mantenidas en palabras cuya longitud es menor o igual a k. La siguiente tabla tiene ejemplos basados en la Figura 1

k	Salida esperada
1	a
3	a al cal can la las lo los
8	a al cal can la las lo los losas

5.7.6. *buscarFinPrefijo (ArregloChars prefijo, TPalabras palabras)*

Se devuelve el nodo de palabras en donde termina el camino que representa a prefijo.
La siguiente tabla tiene ejemplos basados en la Figura 1

k	Salida esperada
c	<pre> c a / \ l n '\0' '\0'</pre>
los	<pre> s / \ '\0' a s '\0'</pre>
lar	NULL

5.8. *Módulo principal*

Similar al de las tareas anteriores al que se agregan comandos para las nuevas operaciones.