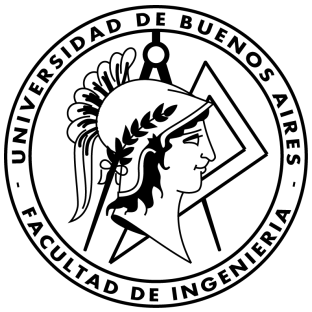

TDA ABB

[7541/9515] Algoritmos y Programación II
Primer cuatrimestre de 2022

Alumno:	de San Vicente, Hernán
Número de padrón:	108800
Email:	hdesanvicente@fi.uba.ar

Índice

1. Introducción

En este trabajo práctico se nos pidió implementar ciertas funciones de un Árbol Binario de Búsqueda y responder ciertas preguntas teóricas.

2. Teoría

Se nos pidió explicar qué es un árbol, un árbol binario y un árbol binario de búsqueda, junto con algunas de sus operaciones y su complejidad computacional. Es importante distinguir estos tres tipos de árboles porque se debe comprender la estructura en sí para luego ir adentrándose en sus diferentes subtipos.

2.1. Árbol

Un árbol es una estructura de datos no lineal que si bien puede ser definido de varias formas, simplemente podemos decir que es una colección de nodos. Cada nodo es un elemento del árbol.

2.2. Árbol Binario

Un árbol binario es un árbol en el cual cada nodo apunta a un máximo de 2 otros nodos, por lo general se representa como nodo izquierda y nodo derecha respectivamente de donde estén ubicados.

2.3. Árbol Binario de Búsqueda

Un árbol binario de búsqueda es un árbol binario que cumple con la condición de estar ordenado, en el caso de la implementación de este TDA, comparando con el nodo raíz, las claves mayores son los nodos que se encuentran a su derecha y las claves menores se encuentran a la izquierda.

2.4. Operaciones más esenciales de un ABB y su complejidad computacional (CC)

1. Recorridos: Existen 3 formas de recorrer: Preorden, inorden y postorden. El recorrido preorden devuelve los valores en el orden en que fueron insertados y se usa para hacer una copia fiel del árbol (haciendo centro, izquierda y luego derecha), el inorden devuelve los valores en orden (haciendo izquierda, centro y derecha) y el postorden nos da el orden correcto de borrado (haciendo izquierda, derecha y centro). CC: O(n) en caso de que el árbol no esté balanceado

2. Insertar: Al insertar un nuevo elemento en el árbol, este siempre queda como una "hoja" (Sin nodos hijos), llegando esta su posición debida recorriendo el árbol y comparando con cada elemento hasta que no hayan más por la rama que se está siguiendo. CC: O(n) en caso de que el árbol no esté balanceado

3. Quitar: Esta es la operación más compleja de todas, ya que se tienen que tener en cuenta varios casos. El caso mas sencillo es si se tiene que eliminar un nodo hoja, en donde simplemente se quita (Diagrama 1). Luego está el caso donde el nodo a eliminar tiene hijos, donde hay que buscar un reemplazo. Existen dos soluciones: buscar el predecesor inorden (agarrando el nodo derecho inmediato y yendo hacia la izquierda hasta que no hayan mas) o el sucesor inorden (agarrando el nodo izquierdo inmediato y yendo hacia la derecha hasta que no hayan mas, como en el diagrama 2) para reemplazar el nodo a eliminar. Si el reemplazo elegido tiene un hijo, simplemente se reemplaza con ese hijo. CC: O(n) en caso de que el árbol no esté balanceado

4. Buscar: En esta operación simplemente se compara el elemento a buscar con cada nodo y se sigue por la respectiva rama hasta que el elemento con el que se compara sea igual (encontrado) o nulo (no existe). CC: O(n) en caso de que el árbol no esté balanceado

3. Detalles de implementación

Todas la funciones que implicaban algún tipo de bucle en este trabajo fueron implementadas de forma recursiva, no solo por la facilidad que implicaba esto si se maneja bien la recursividad, sino porque visualmente queda mucho mejor y mas legible.

3.1. Crear y destruir

Simplemente se reserva memoria para el árbol y luego se destruye aplicando una función recibida por parámetro a cada uno de sus nodos.

3.2. Recorrer

Se tiene un enum que tiene las 3 formas de recorrer un árbol y, dependiendo de la forma que se reciba por parámetro, se guardan todos los valores en un vector.

3.3. Insertar y quitar

Ambas funciones se implementan como se explica en el apartado Operaciones más esenciales de un ABB utilizando el comparador del árbol y tomando en cuenta cada caso particular de cada operación. En el caso de quitar, se libera el nodo quitado y se deja el árbol ordenado.

3.4. Buscar

Acá se va siguiendo el recorrido del árbol comparando cada nodo usando la función comparadora del árbol hasta que se encuentre o se llegue a un nodo nulo.

3.5. Con cada elemento

Para esta función se crearon 3 funciones extras para cada una de las formas de recorrer un ABB. La función recibida por parámetro es aplicada a cada uno de los elementos junto con la variable auxiliar.

3.6. Tamaño y Vacío

Estas funciones son muy simples ya que el tamaño es almacenado en la estructura del árbol, y si ese tamaño es igual a 0, entonces el árbol está vacío.

4. Diagramas

En esta sección van los diagramas que realizas para poder acompañar los detalles de implementación y de funciones que escribiste mas arriba. Trata de que sean lo mas claros posibles, podes hacer mas de un diagrama para una sola función, por ejemplo podes tener un diagrama por cada paso que realiza la función en vez de tener un solo diagrama (que a lo mejor termina siendo poco claro) con toda la información metida junta.

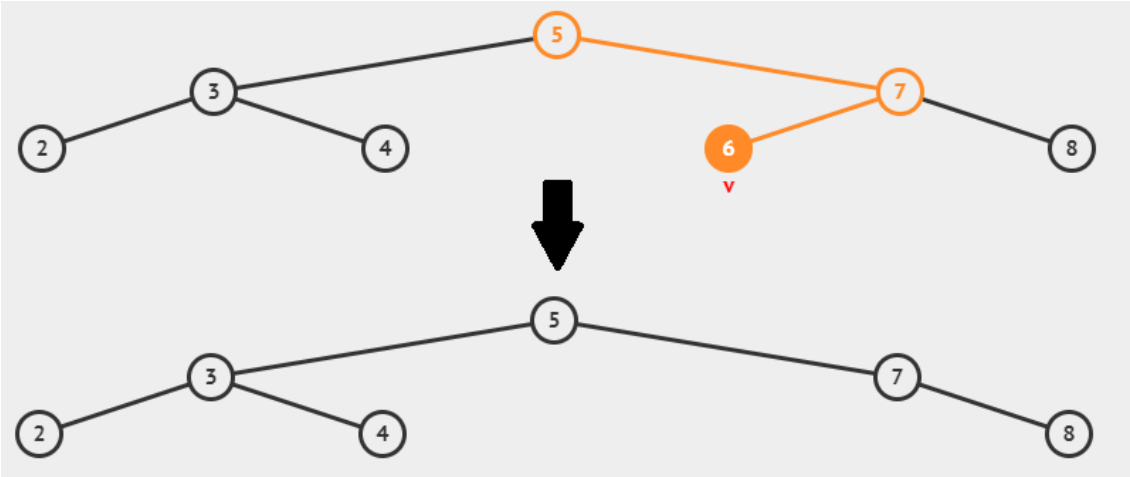

Figura 1: Ejemplo de una hoja siendo borrada.

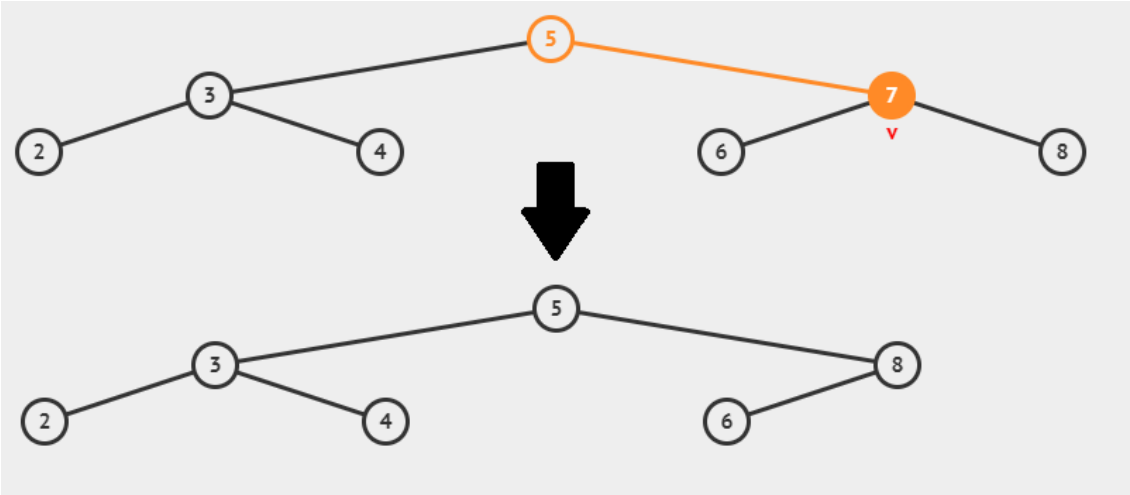

Figura 2: Ejemplo de un nodo padre siendo borrado y reemplazado.