Corrección Dall’acqua Denise - TDA ABB:

Facultad de Ingeniería de la Universidad de Buenos Aires.

Algoritmos y Programación II [7541/9515].

Primer cuatrimestre 2022.

Corrección por Gamberale Luciano Martín.

---

1. General:

1.1. Aspectos positivos

- Buen informe.
- Buenas pruebas.
- Buen código.
- Buena resolución de las funciones.
- Entrega el enunciado del Trabajo Práctico.
- Se tuvieron en cuenta las correcciones de la entrega del TDA Lista.
- La implementación del TDA ABB pasa las pruebas propuestas por el alumno.
- La implementación del TDA ABB pasa las pruebas propuestas por la cátedra.

1.2. Aspectos negativos

- Se incluyó el ejecutable `pruebas` en la entrega.

2. Código:

2.1. Comentarios generales sobre el código:

- Excelente código.
- Se aplicaron las buenas prácticas de programación.
- Muy buena resolución en cada una de las funciones, en especial de `abb_quitar()` y de `abb_recorrer()`.
- Se colocaron correctamente pre y post condiciones en todas las funciones secundarias realizadas.
- En la función `abb_insertar()` no se verifica que se haya insertado correctamente un nodo dentro del árbol.

3. Informe:

- Buen informe.
- Se detalla correctamente la manera en que debe ser compilado el código.
- Se detallan correctamente los conceptos teóricos (Arbol, Arbol Binario y Árbol Binario de Búsqueda).
- Muy buenos diagramas, tanto para la explicación teórica como para la explicación de las funciones implementadas.
- Muy buena explicación de las funciones implementadas.
- En la explicación acerca de los recorridos de un Árbol Binario, se indica que la complejidad algorítmica para recorrer este tipo de árbol es O(log(n)), lo cual es incorrecto. El hecho de recorrer todos los elementos de un árbol conlleva un complejidad algorítmica de O(n) ya que es necesario pasar al menos una vez por los “n” nodos del árbol.
- En la explicación acerca de la búsqueda de elementos en Arbol Binario de Busqueda se podria haber indicado que en el caso de tener un árbol degenerado en lista, la complejidad algorítmica se podría tornar para el peor de los casos en O(n) a causa de tener que recorrer todos los elementos del árbol. Además, se puede observar que en dicha explicación se indica que en el mejor de los casos dicha búsqueda puede lograr una complejidad algorítmica de O(1), sin ejemplificar o explicar cual es el mejor de los casos.

4. Pruebas:

4.1 Creación

- [x] ~~Se prueba que no se pueda crear un abb con una función comparadora nula~~
- [x] ~~Se prueba crear un abb válido y verificar que empieza vacío~~

4.2 Inserción

- [x] ~~Se prueba insertar un elemento que tenga información NULL-0-false~~
- [x] ~~Se prueba insertar un elemento como hoja~~
- [x] ~~Se prueba insertar un elemento como raíz~~
- [ ] Se prueba insertar un elemento repetido
- [x] ~~Se prueba que no se pueda insertar en un árbol nulo~~

4.3 Eliminación

- [ ] Se prueba quitar un elemento que tenga información NULL-0-false
- [x] ~~Se prueba quitar un elemento con y sin hijos~~
- [x] ~~Se prueba quitar la raíz con y sin hijos~~
- [x] ~~Se prueba quitar una hoja~~
- [x] ~~Se prueba que no se pueda quitar de un árbol nulo~~
- [x] ~~Se prueba que no se pueda eliminar un elemento que no se encuentra en el árbol~~

4.4 Búsqueda

- [ ] Se prueba buscar un elemento que tenga información NULL-0-false
- [x] ~~Se prueba buscar un elemento hoja~~

- [x] ~~Se prueba buscar la raíz~~
- [x] ~~Se prueba buscar un elemento intermedio~~
- [x] ~~Se prueba que no se pueda buscar en un árbol nulo~~
- [x] ~~Se prueba buscar un elemento que no está en el árbol~~

4.5 Tamaño

- [x] ~~Se prueba que si al agregar elementos al abb, el tamaño aumenta~~
- [x] ~~Se prueba que al eliminar elementos del abb, el tamaño disminuya~~
- [x] ~~Se prueba que al crear un abb, el mismo tenga tamaño == 0~~
- [x] ~~Se prueba que un abb inválido (nulo) tiene tamaño == 0~~

4.6 Array

- [ ] Se prueba mandar un array con un tamaño menor a la cantidad de elementos
- [ ] Se prueba mandar un array con un tamaño mayor a la cantidad de elementos
- [x] ~~Se prueba mandar un array nulo~~
- [ ] Se prueba mandar un array con tamaño 0
- [x] ~~Se prueba mandar un árbol nulo~~

4.7 Recorridos

- [x] ~~Se prueba recorrer un árbol nulo~~
- [ ] Se prueba recorrer con una función nula
- [ ] Se prueba mandar un contexto nulo
- [x] ~~Se prueba cortar la iteración antes~~
- [x] ~~Se prueba recorrer todo el árbol~~

4.8. Estilo

- Te felicito por las pruebas realizadas.
- Muy buenas descripciones de pruebas.
- Buena modularización en las pruebas.

5. Sugerencias:

- En el archivo `abb.c` se podrían haber dividido mediante títulos, la sección en donde se implementaron las funciones principales y la sección en donde se implementaron las funciones secundarias o auxiliares de la siguiente manera:

```
//FUNCIONES SECUNDARIAS
...
//FUNCIONES PRINCIPALES
...
```

¡Ojalá te sirvan las correcciones y cualquier cosa no dudes en consultarme!

Nota: 2 de un máximo de 2 puntos.

¡Te felicito, excelente trabajo!