Práctica 9
Árboles Rojinegros

Estructuras de datos

Meta

Que el alumno domine el manejo de información almacenada en un *Árbol binario ordenado*, implementando un `Árbol Rojinegro`.

Objetivos

Al finalizar la práctica el alumno será capaz de:

- Visualizar cómo se almacenan los datos en una estructura no lineal.
- Entender el comportamiento de un Árbol Rojinegro.
- Programar dicha estructura en un lenguaje orientado a objetos, reutilizando los algoritmos implementados anteriormente.

Antecedentes

Definición 1

Un **Árbol Rojinegro** es un árbol binario de búsqueda que cumple con las propiedades siguientes:

1. Todo nodo tiene un atributo de color cuyo valor es rojo o negro.
2. La raíz es de color negro.
3. Todas las hojas (NIL) son también de color negro.
4. Un nodo rojo tiene 2 hijos negros.
5. Cualquier camino de un nodo a cualquiera de sus hojas tiene el mismo número de nodos negros (*altura negra*).

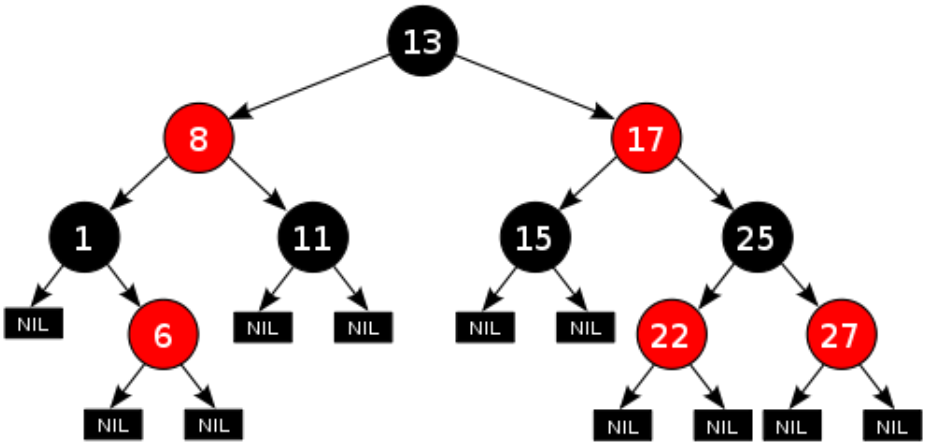

Figura 1 Ejemplo de Árbol Rojinegro

DESARROLLO

Para implementar este tipo de Árboles Binarios se programarán las clases siguientes:

- `NodoRojinegro<C extends Comparable<C>>`.

  Esta clase deberá implementar la interfaz `NodoBinario`. Puedes implementarla directamente o extender `NodoAVL<C>`, pues contiene los mismos métodos, pero asegúrate de sobreescribir los métodos para insertar/borrar/balancear según corresponda y agrega:

  ```
  public Color getColor();
  public void setColor(Color c);
  ```

  Para declarar el color de los nodos se utilizará una enumeración que debes declarar dentro de :

```
public enum Color{
  ROJO,
  NEGRO
}
```

  Así nuestro atributo color será de tipo `Color`: `private Color color;`

- `ÁrbolRojinegro<C extends Comparable<C>>`.

  Esta clase deberá extender `ÁrbolBOLigado<E>`. Se añade el requisito de que al agregar o remover nodos, el árbol debe continuar siendo un árbol rojinegro válido por lo que estos métodos deberán ser sobre-escritos y deben tener complejidad $\mathcal{O}(\log n)$ Cormen y col. 2009.

Para ver los árboles de manera gráfica, se provee un paquete que facilita mostrarlos.

Listing 1: Estracto de `DemoÁrbolesRojinegros.java`

```
package ed.visualización.demos;

import ed.estructuras.nolineales.ÁrbolRojinegro;
import ed.visualización.dibujantes.DibujanteDeÁrbolRojinegro;
```

```java

/**
 *
 * @author veronica
 */
public class DemoÁrbolesRojinegros extends Demo {
    private DibujanteDeÁrbolRojinegro dibujante;

    @DemoMethod(name = "Árbol vacío")
    public String dibujaÁrbol0() {
        ÁrbolRojinegro<Integer> árbol;
        árbol = new ÁrbolRojinegro<>();
        árbol.add("A");
        árbol.add("C");
        return dibujante.drawSVG();
    }

    @DemoMethod(name = "Caso 1 (izquierda)")
    public String dibujaÁrbol4() {
        ÁrbolRojinegro<String> árbol;
        árbol = new ÁrbolRojinegro();
        dibujante = new DibujanteDeÁrbolRojinegro();
        dibujante.setEstructura(árbol);
        árbol.add("C");
        árbol.add("B");
        árbol.add("D");
        árbol.add("A");
        return dibujante.drawSVG();
    }
}
```

- Para esta práctica no se incluyen pruebas unitarias, pero el paquete para visualizar los resultados cumple esta función.

Preguntas

1. ¿Qué ventajas encuentras sobre los árboles AVL? ¿Qué desventajas?
2. ¿Por qué, en teoría, `NodoRojinegro` no extiende `NodoAVL`, aunque reutilice prácticamente todo su código? ¿Qué prefieres: copiar y pegar o heredar a pesar de lo anterior? ¿Por qué?

Forma de entrega

- Asegúrense de borrar todos los archivos generados, incluyendo archivos de respaldo usando `ant clean`.
- Copien el directorio `Practica9` dentro de un directorio llamado `<apellido1_apellido2>` donde `apellido1` es el primer apellido de un miembro del equipo y `apellido2` es el primer apellido del otro miembro. Por ejemplo: `marquez_vazquez`.
- Borren el directorio libs en la copia.
- Agreguen un archivo `reporte.pdf` dentro del directorio `Practica9` de la copia, con el nombre completo de los integrantes del equipo, las repuestas a las preguntas de la sección anterior y cualquier comentario que quieran hacer sobre la práctica.
- Compriman el directorio `<apellido1_apellido2>` creando

REFERENCIAS

`<apellido1_apellido2>.tar.gz`.

Por ejemplo: `marquez_vazquez.tar.gz`.

- Suban este archivo en la sección correspondiente del aula virtual. Basta una entrega por equipo.

Forma de evaluación

Para su calificación final se tomarán en cuenta los aspectos siguientes:

70 %	Calificación generada por las pruebas automáticas. Usaremos nuestros archivos, por lo que si realizan modificaciones a sus copias, éstas no tendrán efecto al momento de calificar.
10 %	Reporte con respuestas a las preguntas.
10 %	Documentación completa y adecuada. Entrega en el formato requerido, sin archivos `.class`, respaldos, bibliotecas de `JUnit` u otros no requeridos.
10 %	Revisión manual del código, para verificar que se cumpla con las especificaciones, que no se haya copiado, etcétera.

Referencias

Cormen, T. H. y col. (2009). *Introduction to Algorithms*. The MIT Press.