Informe de Laboratorio 06

Tema: Arbol 2-3

Nota

Estudiante	Escuela	Asignatura
Misael Marrón Lope mmarronl@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	EDA Semestre: III Código: 20220575

Laboratorio	Tema	Duración
06	Arbol 2-3	

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	2023	3 julio 2023

1. Tarea

- Implementa una clase de arbol 2-3.
- Implemente una clase Node donde T es un tipo genérico, esta clase debe contener al menos dos propiedades.
- Utilizar Git para evidenciar su trabajo.
- Enviar trabajo al profesor en un repositorio GitHub Privado, dándole permisos como colaborador.

2. Equipos, materiales y temas utilizados

- Sistema Operativo Windows 10 ver. 22H2
- Eclipse IDE, Visual studio
- java 20.0.1
- Git 2.40.1.
- Cuenta en GitHub con el correo institucional.

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/MisaelMarron/eda-lab-b-23a.git`
- URL para el laboratorio 05 en el Repositorio GitHub.
- `https://github.com/MisaelMarron/eda-lab-b-23a/tree/main/lab06`

4. Actividades : La creacion de Arbol Binario de Busqueda

- Elabore un informe implementando Arboles 2-3 con toda la lista de operaciones necesarias como insert() delete().

4.1. Commits Importantes:

Listing 1: Mi primer commit importante es cuando agregue la clase Node .

```
commit 8ef25ae3d0e782be7f34085c7ca3c204653925ae
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date:  Sun Jul 2 18:36:40 2023 -0500

 Agregamos la clase Node
```

```java
package lab06;
import java.util.*;
public class Node<T extends Comparable<T>> {
    private List<T> valores;
    private List<Node<T>> hijos;
    private Node<T> padre;

    public Node(T valor) {
        valores = new ArrayList<>();
        hijos = new ArrayList<>();
        padre = null;
        valores.add(valor);
    }

    public List<T> getValores() {
        return valores;
    }

    public List<Node<T>> getHijos() {
        return hijos;
    }

    public Node<T> getPadre() {
        return padre;
    }

    public void setPadre(Node<T> padre) {
        this.padre = padre;
    }
}
```

Listing 2: Mi segundo commit mas importante es cuando agregue el metodo insert .

```
commit a62ff8a3a32105054e25fd416819c27e6ca7a3f2
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date:  Sun Jul 2 19:17:45 2023 -0500

   Agregamos metodo insert
```

```java

    public void insert(T valor) {
        if (raiz == null) {
            raiz = new Node<>(valor);
        } else {
            Node<T> nodo = buscarNodoParaInsertar(raiz, valor);
            if (nodo == null) {
                return;
            }
            if (nodo.getPadre() == null) {
                dividirNodoRaiz(nodo, valor);
            } else {
                dividirNodoInterior(nodo, valor);
            }
        }
    }

    private Node<T> buscarNodoParaInsertar(Node<T> nodo, T valor) {
        if (nodo.getHijos().isEmpty()) {
            return nodo;
        }

        for (int i = 0; i < nodo.getValores().size(); i++) {
            if (valor.compareTo(nodo.getValores().get(i)) < 0) {
                return buscarNodoParaInsertar(nodo.getHijos().get(i), valor);
            }
        }

        return buscarNodoParaInsertar(nodo.getHijos().get(nodo.getValores().size()), valor);
    }

```

```java
private void dividirNodoRaiz(Node<T> nodo, T valor) {
    if (nodo.getValores().size() == 2) {
        nodo.getValores().add(valor);
        Collections.sort(nodo.getValores());
        Node<T> nuevoNodo = new Node<>(nodo.getValores().get(1));

        nuevoNodo.getHijos().add(nodo);
        nuevoNodo.getHijos().add(new Node<>(nodo.getValores().get(2)));

        nodo.getValores().remove(1);
        nodo.getValores().remove(1);

        nodo.getHijos().get(0).setPadre(nuevoNodo);
        nodo.getHijos().get(1).setPadre(nuevoNodo);

        raiz = nuevoNodo;
    } else {
        nodo.getValores().add(valor);
        Collections.sort(nodo.getValores());
    }
}

private void dividirNodoInterior(Node<T> nodo, T valor) {
    Node<T> padre = nodo.getPadre();

    if (nodo.getValores().size() == 2) {
        padre.getValores().add(nodo.getValores().get(1));
        Collections.sort(padre.getValores());

        int indiceHijo = padre.getHijos().indexOf(nodo);
        padre.getHijos().add(indiceHijo + 1, new Node<>(nodo.getValores().get(2)));

        nodo.getValores().remove(1);
        nodo.getValores().remove(1);

        padre.getHijos().get(indiceHijo).setPadre(padre);
        padre.getHijos().get(indiceHijo + 1).setPadre(padre);

        dividirNodoInterior(padre, valor);
    } else {
        nodo.getValores().add(valor);
        Collections.sort(nodo.getValores());
```

Listing 3: Mi tercer commit es cuando agregue la clase main con todos los metodos terminados y terminamos el lab06

```
commit ba998a2ab903ccfa203238dbccff806f48b88cc0
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date:  Sun Jul 2 19:18:46 2023 -0500

   Agregamos clase main y terminamos el lab
```

‘

```
            int indiceHijo = padre.getHijos().indexOf(nodo);
            padre.getHijos().add(indiceHijo + 1, new Node<>(nodo.getValores().get(2)));

            nodo.getValores().remove(1);
            nodo.getValores().remove(1);

            padre.getHijos().get(indiceHijo).setPadre(padre);
            padre.getHijos().get(indiceHijo + 1).setPadre(padre);

            dividirNodoInterior(padre, valor);
        } else {
            nodo.getValores().add(valor);
            Collections.sort(nodo.getValores());
        }
    }

    public void delete(T valor) {
        if (raiz == null) {
            return;
        }

        Node<T> nodo = buscarNodo(raiz, valor);
        if (nodo == null) {
            return;
        }

        if (nodo.getPadre() == null && nodo.getHijos().isEmpty()) {
            raiz = null;
        } else {
            eliminarValor(nodo, valor);
        }
    }

    private Node<T> buscarNodo(Node<T> nodo, T valor) {
        if (nodo == null) {
            return null;
        }

        for (int i = 0; i < nodo.getValores().size(); i++) {
            if (valor.equals(nodo.getValores().get(i))) {
                return nodo;
            } else if (valor.compareTo(nodo.getValores().get(i)) < 0) {
                return buscarNodo(nodo.getHijos().get(i), valor);
            }
        }

        return buscarNodo(nodo.getHijos().get(nodo.getValores().size()), valor);
    }

    private void eliminarValor(Node<T> nodo, T valor) {
        nodo.getValores().remove(valor);
```

```
119    private void eliminarValor(Node<T> nodo, T valor) {
120        nodo.getValores().remove(valor);
121
122        if (nodo.getValores().isEmpty()) {
123            Node<T> padre = nodo.getPadre();
124
125            if (padre != null) {
126                int indiceHijo = padre.getHijos().indexOf(nodo);
127                if (indiceHijo > 0 && padre.getHijos().get(indiceHijo - 1).getValores().size() > 1) {
128                    Node<T> hermanoIzquierdo = padre.getHijos().get(indiceHijo - 1);
129                    T valorPadre = padre.getValores().remove(indiceHijo - 1);
130                    nodo.getValores().add(0, valorPadre);
131
132                    int indiceValorHermano = hermanoIzquierdo.getValores().size() - 1;
133                    T valorHermano = hermanoIzquierdo.getValores().remove(indiceValorHermano);
134                    padre.getValores().add(indiceHijo - 1, valorHermano);
135                    Collections.sort(padre.getValores());
136
137                    if (!hermanoIzquierdo.getHijos().isEmpty()) {
138                        Node<T> ultimoHijo = hermanoIzquierdo.getHijos().remove(indiceValorHermano + 1);
139                        nodo.getHijos().add(0, ultimoHijo);
140                        ultimoHijo.setPadre(nodo);
141                    }
142                } else if (indiceHijo < padre.getHijos().size() - 1 && padre.getHijos().get(indiceHijo + 1).getValores().size() > 1) {
143                    Node<T> hermanoDerecho = padre.getHijos().get(indiceHijo + 1);
144                    T valorPadre = padre.getValores().remove(indiceHijo);
145                    nodo.getValores().add(valorPadre);
146
147                    T valorHermano = hermanoDerecho.getValores().remove(0);
148                    padre.getValores().add(indiceHijo, valorHermano);
149                    Collections.sort(padre.getValores());
150
151                    if (!hermanoDerecho.getHijos().isEmpty()) {
152                        Node<T> primerHijo = hermanoDerecho.getHijos().remove(0);
153                        nodo.getHijos().add(primerHijo);
154                        primerHijo.setPadre(nodo);
155                    }
156                } else {
157                    if (indiceHijo > 0) {
158                        Node<T> hermanoIzquierdo = padre.getHijos().remove(indiceHijo - 1);
159                        T valorPadre = padre.getValores().remove(indiceHijo - 1);
160                        hermanoIzquierdo.getValores().add(valorPadre);
161                        hermanoIzquierdo.getValores().addAll(nodo.getValores());
162                        hermanoIzquierdo.getHijos().addAll(nodo.getHijos());
163
164                        for (Node<T> hijo : nodo.getHijos()) {
165                            hijo.setPadre(hermanoIzquierdo);
166                        }
167
168                        if (padre.getValores().isEmpty()) {
169                            raiz = hermanoIzquierdo;
170                        }
171
```

```
166                        }
167
168                        if (padre.getValores().isEmpty()) {
169                            raiz = hermanoIzquierdo;
170                        }
171
172                        eliminarValor(padre, valor);
173                    } else {
174                        Node<T> hermanoDerecho = padre.getHijos().remove(indiceHijo + 1);
175                        T valorPadre = padre.getValores().remove(indiceHijo);
176                        nodo.getValores().add(valorPadre);
177                        nodo.getValores().addAll(hermanoDerecho.getValores());
178                        nodo.getHijos().addAll(hermanoDerecho.getHijos());
179
180                        for (Node<T> hijo : hermanoDerecho.getHijos()) {
181                            hijo.setPadre(nodo);
182                        }
183
184                        if (padre.getValores().isEmpty()) {
185                            raiz = nodo;
186                        }
187
188                        eliminarValor(padre, valor);
189                    }
190                }
191            }
192        }
193    }
194
195    public void display() {
196        displayHelper(raiz, "");
197    }
198
199    private void displayHelper(Node<T> nodo, String indent) {
200        if (nodo == null) {
201            return;
202        }
203
204        for (int i = nodo.getValores().size() - 1; i >= 0; i--) {
205            displayHelper(nodo.getHijos().get(i + 1), indent + "    ");
206            System.out.println(indent + nodo.getValores().get(i));
207        }
208
209        displayHelper(nodo.getHijos().get(0), indent + "    ");
210    }
211 }
```

```java
package lab06;
public class Main {
    public static void main(String[] args) {
        arbol23<Integer> arbol = new arbol23<>();

        // Insertar elementos
        arbol.insert(10);
        arbol.insert(5);
        arbol.insert(15);
        arbol.insert(3);
        arbol.insert(7);
        arbol.insert(12);
        arbol.insert(20);
        arbol.insert(9);

        // Mostrar el árbol
        System.out.println("Árbol 2-3:");
        arbol.display();

        // Eliminar un elemento
        arbol.delete(7);

        // Mostrar el árbol después de eliminar
        System.out.println("\nÁrbol 2-3 después de eliminar:");
        arbol.display();
    }
}
```

4.2. Estructura de laboratorio 06

- El contenido que se entrega en este laboratorio es el siguiente:

```
lab05/
|--- codigo
|   |--- Node.java
|   |--- arbol23.java
|   |--- Main.java
|--- latex
    |--- img
    |   |--- logo_abet.png
    |   |--- logo_episunsa.png
    |   |--- logo_unsa.jpg
    |   |--- codigo1.jpg
    |   |--- codigo2.jpg
    |   |--- codigo3.jpg
    |   |--- codigo3b.jpg
    |   |--- codigo3c.jpg
    |   |--- codigo3d.jpg
    |   |--- ejecucion.jpg
    |--- Lab05-MisaelMarron.pdf
    |--- Lab05-MisaelMarron.tex
```

5. Preguntas:

- ¿Explique como es el algoritmo que implemento para obtener el arbol binario de busqueda con la libreria Graph Stream? Recuerde que pueden haber operaciones sobre el BST.
- El algoritmo para mostrar un árbol binario de búsqueda con Graph Stream consta de los siguientes pasos:

  Crear un objeto Graph para representar el grafo. Definir las características visuales del grafo. Agregar nodos y aristas al grafo recursivamente. Mostrar el grafo en una ventana utilizando el visor de Graph Stream.

6. Rúbricas

6.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

6.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumplio con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos lo items.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	3	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	1.5	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	1	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	1.5	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
Total		20		17	

7. Referencias

- https://docs.oracle.com/javase/tutorial/java/generics/types.html
- https://www.eclipse.org/downloads/packages/release/2022-03/r/eclipse-ide-enterprise-java-and-web
- https://www.w3schools.com/java/default.asp