Informe de Laboratorio 08
Tema: HUFFMAN TREE

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
08	HUFFMAN TREE	

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	2023	31 julio 2023

1. Tarea

- Implementa una clase de un arbol que use el algoritmo de HuffMan.
- Implemente una programa con parte visual
- 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/lab08`

4. Actividades : La creacion de HuffManTree

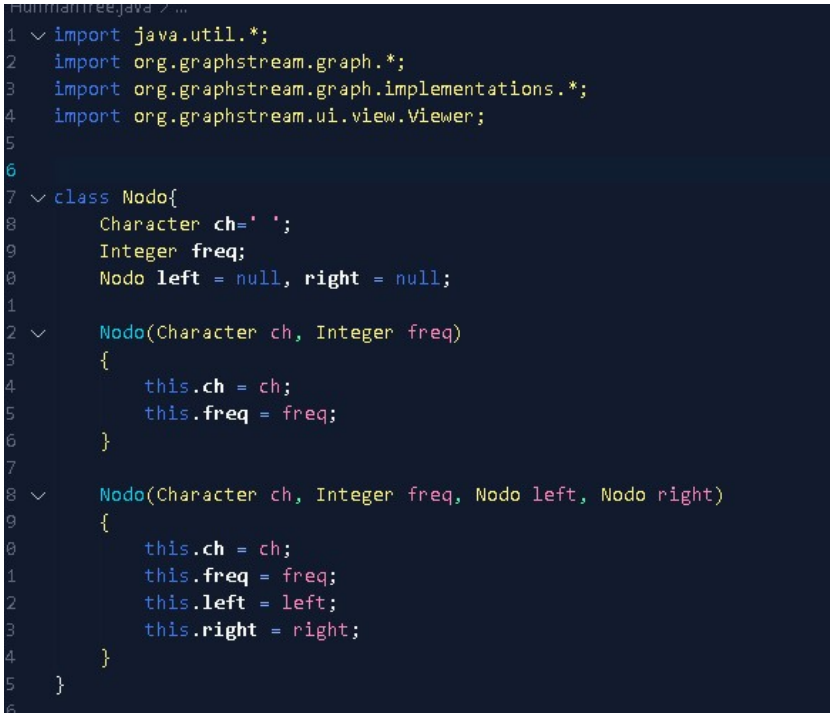
- Elabore un informe implementando el algoritmo de huffman con toda la lista de operaciones necesarias.

4.1. Commits Importantes:

Listing 1: Mi primer commit importante es cuando agregue la clase Nodo para ser posteriormente usada por huffmantree .

```
commit 287ea9bd2896339375562ff0f45e5003096a467b
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date:  Sun Jul 30 19:14:15 2023 -0500

agrego la clase Nodo
```



```java
import java.util.*;
import org.graphstream.graph.*;
import org.graphstream.graph.implementations.*;
import org.graphstream.ui.view.Viewer;


class Nodo{
    Character ch=' ';
    Integer freq;
    Nodo left = null, right = null;

    Nodo(Character ch, Integer freq)
    {
        this.ch = ch;
        this.freq = freq;
    }

    Nodo(Character ch, Integer freq, Nodo left, Nodo right)
    {
        this.ch = ch;
        this.freq = freq;
        this.left = left;
        this.right = right;
    }
}
```

Listing 2: Mi segundo commit mas importante es cuando agregue la clase HuffmanTree por completo , con todos sus metodos y funcionalidades .

```
commit edcf09aabb16bc6cea8f7ea4ddd612f4085f4461
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date:  Sun Jul 30 19:15:47 2023 -0500

Termino la clase HuffmanTree con todos sus metodos
```

```java
class HuffmanTree {
     private static Nodo raiz;
     private static int i=0 , j =0;

    public static void encode(Nodo root, String str,Map<Character, String> huffmanCode){
        if (root == null) {
            return;
        }
        if (isLeaf(root)) {
            huffmanCode.put(root.ch, str.length() > 0 ? str : "1");
        }

        encode(root.left, str + '0', huffmanCode);
        encode(root.right, str + '1', huffmanCode);
    }


    public static int decode(Nodo root, int index, StringBuilder sb){
        if (root == null) {
            return index;
        }

        // Encontré un nodo hoja
        if (isLeaf(root))
        {
            System.out.print(root.ch);
            return index;
        }

        index++;

        root = (sb.charAt(index) == '0') ? root.left : root.right;
        index = decode(root, index, sb);
        return index;
    }


    public static boolean isLeaf(Nodo root) {
        return root.left == null && root.right == null;
    }


    public static void buildHuffmanTree(String text)
    {
        // Caso base: string vacía
        if (text == null || text.length() == 0) {
            return;
        }
```

```
        Map<Character, Integer> freq = new HashMap<>();
        for (char c: text.toCharArray()) {
            freq.put(c, freq.getOrDefault(c, defaultValue:0) + 1);
        }
        PriorityQueue<Nodo> pq;
        pq = new PriorityQueue<>(Comparator.comparingInt(l -> l.freq));

        for (var entry: freq.entrySet()) {
            pq.add(new Nodo(entry.getKey(), entry.getValue()));
        }


        while (pq.size() != 1){

            Nodo left = pq.poll();
            Nodo right = pq.poll();

            int sum = left.freq + right.freq;
            pq.add(new Nodo(ch:null, sum, left, right));
        }

        Nodo root = pq.peek();

        Map<Character, String> huffmanCode = new HashMap<>();
        encode(root, str:"", huffmanCode);

        System.out.println("Codigos de Huffman: " + huffmanCode);
        System.out.println("Texto original: " + text);

        StringBuilder sb = new StringBuilder();
        for (char c: text.toCharArray()) {
            sb.append(huffmanCode.get(c));
        }

        System.out.println("Texto codificado: " + sb);
        System.out.print(s:"Texto decodificado: ");

        if (isLeaf(root))
        {
            while (root.freq-- > 0) {
                System.out.print(root.ch);
            }
        }
        else {
            int index = -1;
            while (index < sb.length() - 1) {
                index = decode(root, index, sb);
            }
```

Listing 3: Mi tercer commit es cuando agregue la parte grafica con todos los metodos que hacen posible el grafo.

```
commit 36fb269b37b0fac48c0c1a0d4a322fbf0dd96419
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date:  Sun Jul 30 19:18:02 2023 -0500

    Agrego las lib para parte grafica y sus metodos
```

‘

```java
public void imprimirArbol() {
    Graph graph = new SingleGraph("Árbol Huffman");
    agregarNodos(graph, raiz, x:0, y:0);
    agregarAristas(graph, raiz);
    System.setProperty(key:"org.graphstream.ui", value:"swing");
    Viewer viewer = graph.display();
    viewer.disableAutoLayout();
    viewer.setCloseFramePolicy(Viewer.CloseFramePolicy.CLOSE_VIEWER);

}

public void cerrarArbol() {
    Graph graph = new SingleGraph("Árbol Huffman");
    agregarNodos(graph, raiz, x:0, y:0);
    agregarAristas(graph, raiz);
    System.setProperty(key:"org.graphstream.ui", value:"swing");
    Viewer viewer = graph.display();
    viewer.disableAutoLayout();
    viewer.setCloseFramePolicy(Viewer.CloseFramePolicy.EXIT);

}

private void agregarNodos(Graph graph, Nodo nodo, double x, double y) {
    if (nodo != null) {

        String ch;
        ArrayList<String> lista = new ArrayList<>();
        char[] nums = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
        String[] letras = { "a", "b", "c", "d", "e", "f", "g", "h", "i", "j" };
        if (nodo.ch == null) {
            nodo.ch = nums[i];
            i++;
        }

        if (nodo.ch != null) {
            if (lista.contains(nodo.ch.toString()) == false)
                lista.add(nodo.ch.toString());
            else {
                lista.add(nodo.ch.toString() + letras[j]);
                j++;
            }

        }

        if (nodo.ch.toString().equals(anObject:" "))
            ;
        ch = "SPACE";
```

```
            ch = nodo.ch.toString();

            Node node = graph.addNode(ch);
            node.setAttribute("ui.label", nodo.ch + "(" + nodo.freq.toString() + ")");
            node.setAttribute("ui.style",
                    "text-size: 20px; text-alignment: under; text-background-mode: rounded-box;text-background-color: #ffffffbb; text-color: blue; ");
            node.setAttribute("xy", x, y);
            agregarNodos(graph, nodo.left, x - 6.0, y - 6.0);
            agregarNodos(graph, nodo.right, x + 6.0, y - 6.0);
        }
    }

    private void agregarAristas(Graph graph, Nodo nodo) {
        if (nodo != null) {
            if (nodo.left != null) {
                Edge edge = graph.addEdge(nodo.ch + "-" + nodo.left.ch, nodo.ch.toString(), nodo.left.ch.toString());
                edge.setAttribute("ui.style", "fill-color: black;");
            }
            if (nodo.right != null) {
                Edge edge = graph.addEdge(nodo.ch + "-" + nodo.right.ch, nodo.ch.toString(), nodo.right.ch.toString());
                edge.setAttribute("ui.style", "fill-color: black;");
            }

            agregarAristas(graph, nodo.left);
            agregarAristas(graph, nodo.right);
        }
    }

}
```

Listing 4: Mi cuarto commit y ultimo es cuando agregue test y el final de mi laboratorio

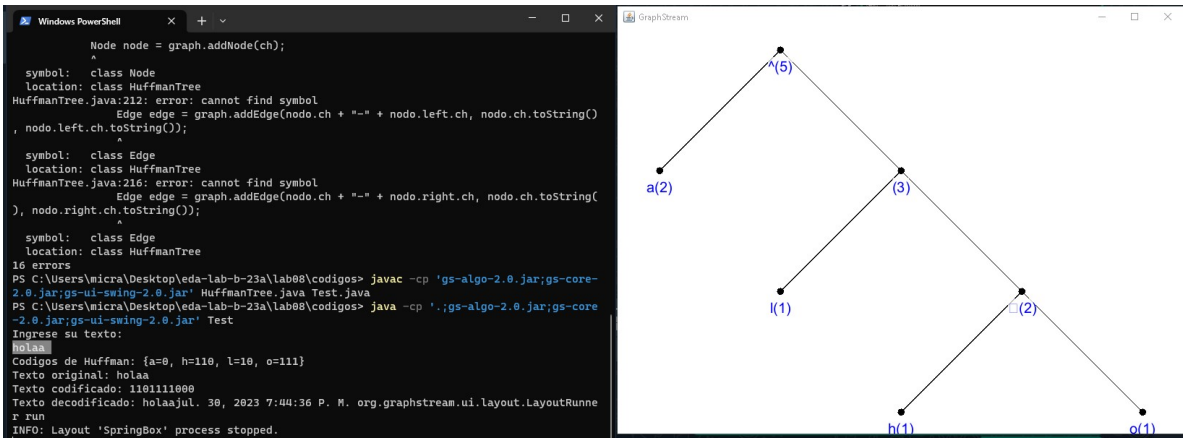
```
commit 0d1de0aa097ddf05c08f8537fbaad40731685ce1
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date:   Sun Jul 30 19:20:20 2023 -0500

Agrego test.java y version final del lab
```

```java
import java.util.*;
public class Test {
    // Ejemplo de uso
    Run | Debug
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        HuffmanTree arbol = new HuffmanTree();

        System.out.println(x:"Ingrese su texto: ");
        String text = scan.nextLine().toLowerCase();
        arbol.buildHuffmanTree(text);
        System.setProperty(key:"org.graphstream.ui", value:"graphstream.ui.swing");
        arbol.imprimirArbol();

    }
}
```



4.2. Estructura de laboratorio 08

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

```
lab08/
|--- codigo
|   |--- HuffmanTree.java
|   |--- Test.java
|--- latex
    |--- img
    |   |--- logo_abet.png
    |   |--- logo_episunsa.png
    |   |--- logo_unsa.jpg
    |   |--- codigo1.jpg
    |   |--- codigo2.jpg
    |   |--- codigo2b.jpg
    |   |--- codigo3.jpg
    |   |--- codigo3b.jpg
    |   |--- codigo4.jpg
    |   |--- ejecucion.jpg
    |--- Lab08-MisaelMarron.pdf
    |--- Lab08-MisaelMarron.tex
```

5. Preguntas:

- No hay preguntas

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		18	

7. Referencias

- https://www.geeksforgeeks.org/trie-insert-and-search/
- https://www.geeksforgeeks.org/introduction-to-trie-data-structure-and-algorithm-tutorials/