

Informe de Laboratorio

Tema: Proyecto Final

Nota

Estudiante	Escuela	Asignatura
Florez Bailon Luis Fernando	Escuela Profesional de Ingeniería de Sistemas	Laboratorio de Estructura de datos y algoritmos

Laboratorio	Tema	Duración
	Proyecto Final	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023- B	Del 07 Septiembre 2023	Al 28 Diciembre 2023

1. Implementación de tries:

- Las clases que necesitaremos para implementar este código, serán el Node, que usaremos para la creación de los nodos del árbol, una clase Trie en la cual se realizarán los métodos necesarios para la creación del árbol trie, además necesitamos una clase en la que estará la interfaz gráfica:

- Detector.java
- InterfazDetector.java
- TestDetector.java
- Trie.java
- test.java

1.1. Clase node.java:

En esta clase se inicializarán los nodos que conformarán la clase Trie, los atributos que tendrá cada nodo son:

- hijo: un arreglo de 26 elementos que representarán los caracteres del alfabeto inglés, estos servirán como punteros apuntando al siguiente nodo en el Trie para un carácter.
- fin: booleano que indicará si el nodo actual marca el final de una cadena.
- posi: array que almacena las posiciones en las que se encontró una palabra.

Listing 1: código de node.java

```
1 class Node{
2     Node[] hijos;
3     boolean end;
4     public ArrayList<Integer> pos;
5     Node(){
6         hijos=new Node[26];
7         for(int i=0;i< hijos.length;i++) {
8             hijos[i]=null;
9         }
10        end=false;
11        pos= new ArrayList<>();
12    }
13    public void setPosi(int a) {
14        pos.add(a);
15    }
16    public ArrayList<Integer>getPos(){
17        return this.pos;
18    }
19 }
```

1.2. Clase Trie.java:

Esta clase tendrá un nodo root que se inicializará null, se crearán los métodos insertar(), buscar(), insertarTexto (), con este último también será necesario crear el método eliminar():

- Insertar(String word): método para insertar palabras al árbol, lo que hará es con la palabra recibida del parámetro primero pondrá en minúsculas todo lo que contenga esta palabra con el método toLowerCase, luego en un bucle for crearemos un entero el cual nos ayudará a saber el número decimal del carácter que estamos pidiendo con el método charAt, luego mediante condicionales verificará si ya existe un nodo hijo con ese carácter en el nodo actual, si no existe, crea un nuevo nodo hijo con ese carácter, luego avanzará al nodo hijo correspondiente, y así hasta llegar al final del bucle.

Listing 2: código de Insertar.java

```
1 public void Insertar(String word,int a) {
2     word=word.toLowerCase();
3     Node actual=root;
4     for(int i=0;i<word.length();i++) {
5         int indx=word.charAt(i)-'a';
6         indx = (indx % 26 + 26) % 26;
7         if(actual.hijos[indx]==null) {
8             actual.hijos[indx]=new Node();
9         }
10        if(i==word.length()-1) {
11            actual.hijos[indx].end =true;
12        }
13    }
14 }
```

```
12         actual.hijos[indx].setPosi(a);
13     }
14     actual=actual.hijos[indx];
15 }
16 }
17 }
```

- **Buscar(String word):** método para buscar una palabra del árbol, lo que hará con la palabra recibida del parámetro al igual que en el método insertar(), usará un bucle for y creará un entero que almacenará el decimal en ascii de un carácter, entonces en cada pasada que realice el bucle usará condicionales para verificar si el nodo hijo es vacío, si es así significa que no está el carácter, además en otro condicional se verificará si el índice del bucle es igual a la longitud de la palabra, y si el end del nodo hijo con índice del bucle es verdadero, ya que si es así significa que es fin de la palabra y que esa palabra se encuentra en el trie.

Listing 3: código de Buscar.java

```
1 public boolean buscar(String word) {
2     word=word.toLowerCase();
3     Node actual=root;
4     for(int i=0;i<word.length();i++) {
5         int indi=word.charAt(i)-'a';
6         indi = (indi % 26 + 26) % 26;
7         if(actual.hijos[indi]==null) {
8             return false;
9         }
10        if(i==word.length()-1 && actual.hijos[indi].end==false) {
11            return false;
12        }
13        actual=actual.hijos[indi];
14    }
15    return true;
16 }
```

- **Insertartexto(String text):** método para insertar texto.

Listing 4: código de insertarTexto.java

```
1 public void InsertarTexto(String text) {
2     text=text.toLowerCase();
3     text=text+" ";
4     String word="";
5     int j=0;
6     char [] lista=text.toCharArray();
7     for(int i=0;i<lista.length;i++) {
8         if(Character.compare(' ', lista[i])==0) {
9
10            Insertar(word,i-j);
11            word="";
12            j=0;
13        }
14        else {
15            word+=lista[i];
16
17            j++;
18        }
19    }
20 }
```

```
18     }  
19     }  
20  
21  
22 }
```

- `getPosiciones(String text)`: método para obtener las posiciones de las cadenas guardadas en el trie.

Listing 5: código de `getpo.java`

```
1 public ArrayList<Integer> GetPosiciones(String word) {  
2     ArrayList<Integer> pos= new ArrayList<>();  
3     if(buscar(word)) {  
4  
5         Node actual=root;  
6         for(int i=0;i<word.length();i++) {  
7             int indi=word.charAt(i)-'a';  
8             indi = (indi % 26 + 26) % 26;  
9             if(actual.hijos[indi]==null) {  
10                actual.hijos[indi]=new Node();  
11            }  
12            if(i==word.length()-1 && actual.hijos[indi].end==true) {  
13                pos=actual.hijos[indi].getPos();  
14            }  
15            actual=actual.hijos[indi];  
16        }  
17    }  
18 }  
19  
20 return pos;  
21 }
```

1.3. Clase `Detector.java`:

Esta clase se encargará de hacer la detección del plagio:

- `ResultadoConsola()`: En este método se realizará dar el resultado pero en consola, mediante las condicionales se verificará si hubo plagio parcial, total o si no existió el plagio.

Listing 6: código de `resultado.java`

```
1 public void ResultadoConsola() {  
2     if(NoPlagio>PlagioTotal+PlagioParcial) {  
3         System.out.println("NO HAY PLAGIO EN LOS DOCUMENTOS");  
4     }  
5     else if(NoPlagio+PlagioTotal<PlagioParcial) {  
6         System.out.println(" HAY PLAGIO PARCIAL EN LOS DOCUMENTOS");  
7     }  
8     else {  
9         System.out.println("HAY PLAGIO TOTAL EN LOS DOCUMENTOS");  
10    }  
11 }
```

- **Resultado():** En este método se dará el resultado, mediante las condicionales se verificará si hubo plagio parcial, total o si no existió el plagio y esto lo retornada, este método es importante para devolver el resultado en la interfaz.

Listing 7: código

```
1 public String Resultado() {
2     if(NoPlagio>PlagioTotal+PlagioParcial) {
3         //System.out.println("NO HAY PLAGIO EN LOS DOCUMENTOS");
4         return "NO HAY PLAGIO EN LOS DOCUMENTOS";
5     }
6     else if(NoPlagio+PlagioTotal<PlagioParcial) {
7         //System.out.println(" HAY PLAGIO PARCIAL EN LOS DOCUMENTOS");
8         return "HAY PLAGIO PARCIAL EN LOS DOCUMENTOS";
9     }
10    else {
11        //System.out.println("HAY PLAGIO TOTAL EN LOS DOCUMENTOS");
12        return "HAY PLAGIO TOTAL EN LOS DOCUMENTOS";
13    }
14    //return "Ocurrio un problema";
15 }
```

- **Comparar():** En este método se realizará la comparación para indicar si existe plagio o no.

Listing 8: código de compara.java

```
1 public void Comparar() {
2     ArrayList<Integer> palabraUno,palabraDos;
3     String[] words=this.tex2.split(" ");
4     for(int i=0;i<words.length;i++) {
5         if(uno.buscar(words[i])) {
6             palabraUno=uno.GetPosiciones(words[i]);
7             palabraDos=dos.GetPosiciones(words[i]);
8
9             if(palabraUno.size()<palabraDos.size()) {
10                for(int j=0;j<palabraUno.size();j++) {
11                    if(palabraUno.get(j)==palabraDos.get(j)) {
12                        PlagioTotal++;
13                    }else {
14                        PlagioParcial++;
15                    }
16                }
17            }else {
18                for(int k=0;k<palabraDos.size();k++) {
19                    if(palabraUno.get(k)==palabraDos.get(k)) {
20                        PlagioTotal++;
21                    }else {
22                        PlagioParcial++;
23                    }
24                }
25            }
26        }
27        else{
28            NoPlagio++;
29        }
30    }
31 }
```

1.4. Clase `interfazDetector.java`:

Esta clase se encargará de crear el GUI, donde en dos `TextArea` se buscara que el usuario ingrese cadenas de texto para realizar la verificación de si existe plagio o no.

- `interfaz()`: este método es el constructor de la clase, aquí se le indica las dimensiones del `JFrame` y se crean los paneles que usaremos además de crear los `TextField` y las funcionalidades de nuestro sistema.

Listing 9: código de eventos de los botones del método `buscar` de `interfaz.java`

```
1
2 import java.awt.EventQueue;
3 import javax.swing.JFrame;
4 import javax.swing.JPanel;
5 import javax.swing.border.EmptyBorder;
6 import javax.swing.JTextArea;
7 import javax.swing.JButton;
8 import javax.swing.JLabel;
9 import java.awt.Font;
10 import java.awt.event.ActionListener;
11 import java.awt.event.ActionEvent;
12 import java.awt.Choice;
13
14 public class InterfazDetector extends JFrame {
15     private Detector prueba;
16     private JPanel contentPane;
17
18     /**
19      * Launch the application.
20      */
21     public static void main(String[] args) {
22         EventQueue.invokeLater(new Runnable() {
23             public void run() {
24                 try {
25                     InterfazDetector frame = new InterfazDetector();
26                     frame.setVisible(true);
27                 } catch (Exception e) {
28                     e.printStackTrace();
29                 }
30             }
31         });
32     }
33
34     /**
35      * Create the frame.
36      */
37     public InterfazDetector() {
38         prueba=new Detector();
39         setTitle("Detector de Plagio");
40         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
41         setBounds(100, 100, 934, 461);
42         contentPane = new JPanel();
43         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
```

```
44
45     setContentPane(contentPane);
46     contentPane.setLayout(null);
47
48     JTextArea txtUno = new JTextArea();
49     txtUno.setBounds(59, 86, 322, 148);
50     contentPane.add(txtUno);
51
52     JTextArea txtDos = new JTextArea();
53     txtDos.setBounds(499, 86, 335, 148);
54     contentPane.add(txtDos);
55     JLabel lblRespuesta = new JLabel("");
56     lblRespuesta.setBounds(401, 320, 422, 78);
57     contentPane.add(lblRespuesta);
58
59     JButton btnDetectar = new JButton("Comparar");
60     btnDetectar.addActionListener(new ActionListener() {
61         public void actionPerformed(ActionEvent e) {
62             String pri=txtUno.getText();
63             String dos=txtDos.getText();
64             prueba = new Detector(pri,dos);
65             lblRespuesta.setText(prueba.Resultado());
66         }
67     });
68     btnDetectar.setBounds(145, 360, 116, 38);
69     contentPane.add(btnDetectar);
70
71
72     JLabel lblNewLabel = new JLabel("Ingrese textos encada area");
73     lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 17));
74     lblNewLabel.setBounds(67, 34, 254, 30);
75     contentPane.add(lblNewLabel);
76
77     Choice choice = new Choice();
78     choice.setBounds(334, 320, 200, 50);
79     contentPane.add(choice);
80
81
82 }
83 private static class __Tmp {
84     private static void __tmp() {
85         javax.swing.JPanel __wbp_panel = new javax.swing.JPanel();
86     }
87 }
88 }
```

1.5. Clase testDetector.java :

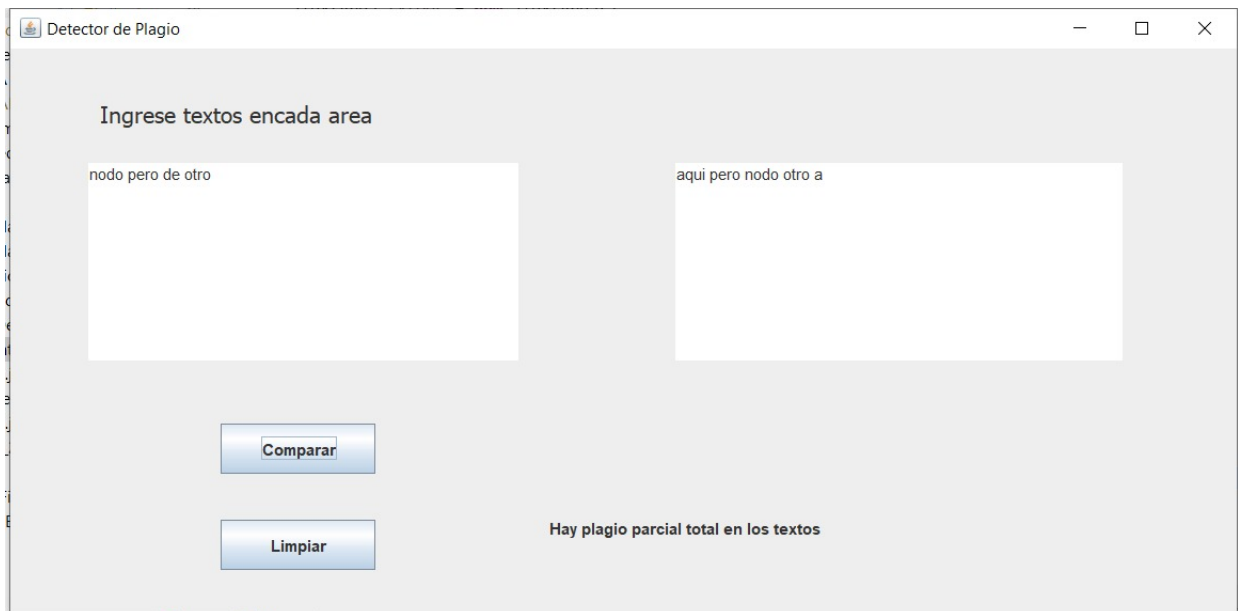
Debido a que no existio una funcionalidad total de nuestra interfaz se implemento esta clase, para mostrarlo en consola


Listing 10: código de eventos de los botones del método buscar de interfaz.java

```
1 public class TestDetector {
2
3     public static void main(String[] args) {
```

```
4      // TODO Auto-generated method stub
5      String tex1="Hola texto uno para comparar";
6      String tex2="Hola texto para compartir";
7      String tex3="fdksajfdasklfjdaskfj daskfjldaskjflkdsa fkjdasklfj sdkjfklds fjk
           dskjfsakl fjsaldek fjklskjdkjfs lkjfas Hola texto uno para comparar";
8      Detector test= new Detector(tex1,tex3);
9      test.Comparar();
10     test.ResultadoConsola();
11
12 }
13
14 }
```

2. Ejecución de código:



 Detector de Plagio

— □ ×

Ingrese textos encada area

Hola mundo de gatos

Hola mundo de perros

Comparar

Limpiar

Hay plagio total en los textos

4. Referencias:

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