Informe de Laboratorio 07
Tema: TRIES

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
07	TRIES	

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

1. Tarea

- Implementa una clase de tries.
- Implemente una programa con parte visual que use tries
- 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/lab07`

4. Actividades : La creacion de Tries y el programa

- Elabore un informe implementando Tries con toda la lista de operaciones necesarias como isEndOfWord.

4.1. Commits Importantes:

Listing 1: Mi primer commit importante es cuando agregue la clase tries y su constructor y antes agregue la clase embebida triesNode .

```
commit 6522fab563913174313a16a19a3dc168a36d46c6
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date:  Sun Jul 23 22:54:57 2023 -0500

   Agregamos la clase tries con su constructor
```

```
18      }
19
20      public TrieNode[] getChildren() {
21          return children;
22      }
23
24      public TrieNode getChild(char ch) {
25          int index = ch -'a';
26          if (index <0 || index >= ALPHABET_SIZE) {
27              return null;
28          }
29          return children[index];
30      }
31
32      public void setChild(char ch, TrieNode node) {
33          int index = ch - 'a';
34          if (index < 0 || index >= ALPHABET_SIZE) {
35              return;
36          }
37          children[index] = node;
38      }
39
40      public boolean isEndOfWord() {
41          return isEndOfWord;
42      }
43
44      public void setEndOfWord(boolean endOfWord) {
45          isEndOfWord = endOfWord;
46      }
47  }
48
49  public class Tries {
50      private TrieNode root;
51      private List<String> InOrder;
52      private Map<String, List<Integer>> Indices;
53
54      public Tries() {
55          root = new TrieNode();
56          InOrder = new ArrayList<>();
57          Indices = new HashMap<>();
58      }
```

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

```
commit e92f71595a30e9008ecfdb26e3905a625297ea56
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date:  Sun Jul 23 23:01:06 2023 -0500

Agregamos el metodo insert
```

```
59
60  public void insert(String word) {
61      TrieNode current = root;
62
63      for (int i = 0; i < word.length(); i++) {
64          char ch = word.charAt(i);
65          TrieNode node = current.getChild(ch);
66
67          if (node == null) {
68              node = new TrieNode();
69              current.setChild(ch, node);
70          }
71          current = node;
72      }
73
74          current.setEndOfWord(true);
75      if (!Indices.containsKey(word)) {
76              InOrder.add(word);
77          if (!Indices.containsKey(word))
78              Indices.put(word, new ArrayList<>());
79
80          Indices.get(word).add(InOrder.size() - 1);
81       }
82  }
```

Listing 3: Mi tercer commit es cuando agregue el metodo search

```
commit c1d632f0ff860185cf2ee1f9eb9412ee06614b59
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date:  Sun Jul 23 23:16:05 2023 -0500

   Agregamos el metodo search
```

‘

```
83  public boolean search(String word) {
84      TrieNode current = root;
85
86      for (int i = 0; i < word.length(); i++) {
87          char ch = word.charAt(i);
88          TrieNode node = current.getChild(ch);
89
90          if (node == null) {
91              return false;
92          }
93
94          current = node;
95      }
96
97      return current.isEndOfWord();
98  }
```

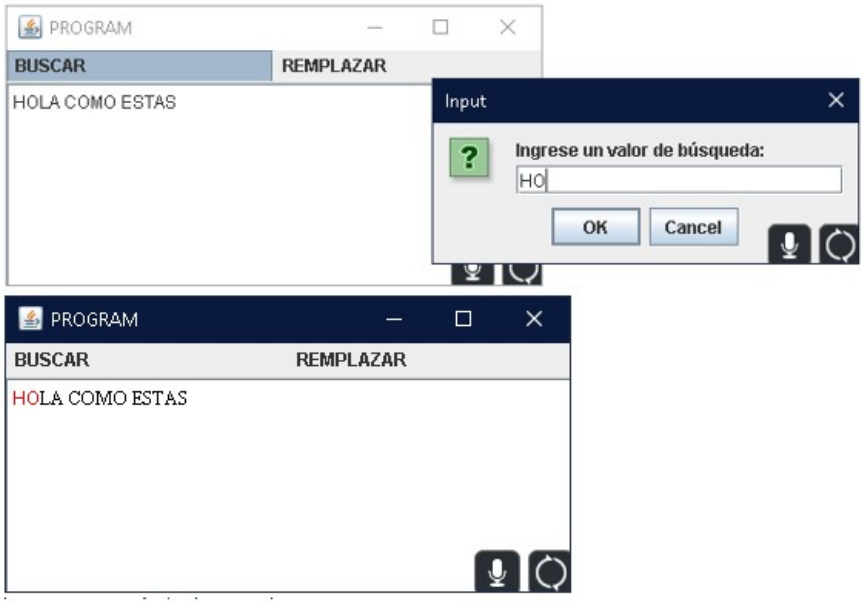
Listing 4: Mi cuarto commit mas importante es cuando agregue la version final del laboratorio

```
commit e527774f7ce2fffd185ced50cc54a6b0b1f81294
Author: Misael Marron <mmarronl@unsa.edu.pe>
Date:  Mon Jul 24 00:05:28 2023 -0500

agregamos forma final del lab07
```

```java
public boolean delete(String word) {
    if (!search(word)) {
        return false;
    }

    TrieNode current = root;
    TrieNode[] ancestors = new TrieNode[word.length()];

    for (int i = 0; i < word.length(); i++) {
        char ch = word.charAt(i);
        TrieNode node = current.getChild(ch);

        ancestors[i] = current;
        current = node;
    }

    current.setEndOfWord(false);

    for (int i = word.length() - 1; i >= 0; i--) {
        TrieNode node = ancestors[i];

        if (current.isEndOfWord() || hasChildren(current)) {
            break;
        }

        char ch = word.charAt(i);
        node.setChild(ch, null);
        current = node;
    }

    List<Integer> indices = Indices.get(word);
    if (indices != null && indices.size() > 0) {
        int indexToRemove = indices.get(indices.size() - 1);
        InOrder.remove(indexToRemove);
        indices.remove(indices.size() - 1);
    }

    return true;
}
public boolean startsWith(String prefix) {
    TrieNode current = root;

    for (int i = 0; i < prefix.length(); i++) {
        char ch = prefix.charAt(i);
        TrieNode node = current.getChild(ch);

        if (node == null) {
            return false;
        }

        current = node;
    }
```

```
153 }
154 private boolean hasChildren(TrieNode node) {
155     for (TrieNode child : node.getChildren()) {
156         if (child != null) {
157             return true;
158         }
159     }
160     return false;
161 }
162 
163 public boolean replace(String word, String newWord) {
164     if (!search(word)) {
165         return false;
166     }
167 
168     List<Integer> indices = Indices.get(word);
169     if (indices != null && indices.size() > 0) {
170         for (int i = indices.size() - 1; i >= 0; i--) {
171             int index = indices.get(i);
172             InOrder.set(index, newWord);
173         }
174         indices.clear();
175     }
176 
177     TrieNode current = root;
178     for (int i = 0; i < word.length(); i++) {
179         char ch = word.charAt(i);
180         TrieNode node = current.getChild(ch);
181         current = node;
182     }
183     current.setEndOfWord(false);
184 
185 
186 
187     return true;
188 }
189 
190 
191 public void printSetence() {
192     for (String word : InOrder) {
193         System.out.print(word + " ");
194     }
195     System.out.print("\n");
196 }
197 
198 public String printSetence2() {
199     String devolver = "";
200     for (String word : InOrder) {
201         devolver += word + " ";
202     }
203     return devolver;
204 }
205 
```

```
            String highlighted = word.substring(startIndex, endIndex);
            String afterHighlight = word.substring(endIndex);

            devolver += beforeHighlight;
            devolver += "\u001B[33m" + highlighted + "\u001B[0m";
            devolver += afterHighlight + " ";
        } else {
            devolver += word + " ";
        }
    }
    return devolver;
}

public void clear() {
    root = new TrieNode();
    InOrder.clear();
    Indices.clear();
}

public void printString2(String word3, JTextPane TextArea1) {
    StyledDocument doc = TextArea1.getStyledDocument();
    SimpleAttributeSet attr = new SimpleAttributeSet();
    StyleConstants.setForeground(attr, Color.BLACK);
    StyleConstants.setFontFamily(attr, "Times New Roman");

    SimpleAttributeSet yellowAttr = new SimpleAttributeSet();
    StyleConstants.setForeground(yellowAttr, Color.RED);

    for (String word : InOrder) {
        int index = word.indexOf(word3);
        int endIndex = index + word3.length();

        int currentPos = 0;
        while (index >= 0) {
            try {
                doc.insertString(doc.getLength(), word.substring(currentPos, index), attr);
                doc.insertString(doc.getLength(), word.substring(index, endIndex), yellowAttr);
            } catch (BadLocationException e) {
                e.printStackTrace();
            }

            currentPos = endIndex;
            index = word.indexOf(word3, endIndex);
            endIndex = index + word3.length();
        }

        try {
            doc.insertString(doc.getLength(), word.substring(currentPos) + " ", attr);
        } catch (BadLocationException e) {
            e.printStackTrace();
        }
    }
}
}
```



4.2. Estructura de laboratorio 07

- 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
    |   |--- codigo4.jpg
    |   |--- codigo4B.jpg
    |   |--- codigo4C.jpg
    |   |--- ejecucion1.jpg
    |--- Lab07-MisaelMarron.pdf
    |--- Lab07-MisaelMarron.tex
```

5. Preguntas:

- Explique. ¿Como se utiliza esta estructura de datos para almacenar prefijos?.
- La principal característica distintiva de un trie es que cada nodo del árbol representa un único carácter en la secuencia. Los nodos en el camino desde la raíz hasta un nodo hoja forman el prefijo

completo de una cadena almacenada en la estructura. Esto hace que los tries sean especialmente útiles para realizar búsquedas y consultas por prefijos.

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