ESTRUCTURAS DE DATOS

2024

TRABAJO PRACTICO N°6
Tries

Profesores:
HECTOR REINAGA
FERNANDA DANIELA OYARZO
MIRTHA FABIANA MIRANDA

Alumno:
GONZALO ALEJANDRO ULLOA

UNPA
Universidad Nacional de la Patagonia Austral

Índice

Desarrollo

2 y 3. Desarrollé ambos puntos en un mismo main

Clase Trie (Matriz)

```java
package tp6;
//Matriz
class Trie {
    private TrieNode nodoRaiz;
    private int cantidad;
    private int begin = 0;
    private int end = 26;
    public Trie(){
        this.nodoRaiz = null;
    }
    public void insertWord(String palabra) {
        palabra = palabra.toLowerCase();

        TrieNode aux;
        int i = 0;
        int pos;

        if (nodoRaiz == null){
            nodoRaiz = new TrieNode(end);
        }

        aux = nodoRaiz;

        while (i < palabra.length()){

            pos = getPosition(palabra.charAt(i));

            if (pos == -1 || pos > end) {
                break;
            } else {
                if (aux.getValueAt(pos) == null){
                    aux.setValueAt(pos, new TrieNode(end));
                }

                aux = aux.getValueAt(pos);
```

```
                i++;
            }
        }

        aux.setValueAt(begin, aux);
    }

    public boolean searchWord(String palabra) {
        palabra = palabra.toLowerCase();

        TrieNode aux = this.nodoRaiz;
        int i = 0;

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

        int pos;

        while (i < palabra.length()) {
            pos = getPosition(palabra.charAt(i));

            if (pos == -1 || pos > end){
                break;
            } else {
                if (aux.getValueAt(pos) != null) {
                    aux = aux.getValueAt(pos);

                    i++;
                } else {
                    return false;
                }
            }
        }

        if (aux.getValueAt(begin) == aux) {
            return true;
        } else {
            return false;
        }
    }
```

```
    public void mostrarPalabras(){
        this.mostrarPalabras(this.nodoRaiz, "");
    }

    private void mostrarPalabras(TrieNode nodo, String imp){
        for (int i = 0; i <= end; i++) {
            if (nodo.getValueAt(i)!=null){
                if (i==0) {
                    System.out.println(imp);
                } else {
                    this.mostrarPalabras(nodo.getValueAt(i),
imp+(recuperarCaracter(i)));
                }
            }
        }
    }

    public int contarPalabras(){
        this.cantidad = 0;
        contarPalabras(nodoRaiz);

        return this.cantidad;
    }

    private void contarPalabras(TrieNode nodo){
        for (int i = 0; i <= end; i++){

            if (nodo.getValueAt(i) != null){

                if (i == 0){
                    cantidad++;
                } else {
                    contarPalabras(nodo.getValueAt(i));
                }
            }
        }
    }
```

```
    public int contarPrefijos(){
        this.cantidad = 0;
        this.contarPrefijos(nodoRaiz);

        return this.cantidad;
    }

    private void contarPrefijos(TrieNode nodo){
        boolean control = false;

        for (int i = 0; i <= end; i++){
            if (nodo.getValueAt(i) != null) {
                if (i != 0) {
                    this.cantidad++;
                    this.contarPrefijos(nodo.getValueAt(i));

                    if (control) {
                        this.cantidad++;
                    }
                } else {
                    this.cantidad--;
                    control = true;
                }
            }
        }
    }

    public void buscarPrefijos(String pref){
        this.buscarPrefijos(nodoRaiz, "", pref);
    }

    private void buscarPrefijos(TrieNode nodo, String imp ,
String pref){
        boolean ok = false;

        for (int i = 0; i <= end; i++) {
            if (nodo.getValueAt(i) != null) {
                if (i == 0) {
                    if (!ok && (imp.length() > pref.length()) &&
(imp.substring(0, pref.length()).equals(pref))) {
                        ok=true;
```

```
                        System.out.println(imp);
                    }
                } else {
                    this.buscarPrefijos(nodo.getValueAt(i),
imp+(recuperarCaracter(i)),pref);
                }
            }
        }
    }

    public static int getPosition(char c){
        switch(c) {
            case '@': return 0;
            case 'a': return 1;
            case 'b': return 2;
            case 'c': return 3;
            case 'd': return 4;
            case 'e': return 5;
            case 'f': return 6;
            case 'g': return 7;
            case 'h': return 8;
            case 'i': return 9;
            case 'j': return 10;
            case 'k': return 11;
            case 'l': return 12;
            case 'm': return 13;
            case 'n': return 14;
            case 'o': return 15;
            case 'p': return 16;
            case 'q': return 17;
            case 'r': return 18;
            case 's': return 19;
            case 't': return 20;
            case 'u': return 21;
            case 'v': return 22;
            case 'w': return 23;
            case 'x': return 24;
            case 'y': return 25;
            case 'z': return 26;
        }
```

```
        return -1;
    }

    public static char recuperarCaracter(int val){
        char c='a';

        for (int i = 1; i < val; i++) {
            c++;
        }

        return c;
    }
}
```

Clase TrieNode (Matriz)

```
class TrieNode {
    TrieNode[] chars;
    TrieNode(int end) {
        chars = new TrieNode[end + 1];
    }
    TrieNode getValueAt(int pos) {
        if(pos>=chars.length || pos<0){
            return null;
        }
        return chars[pos];
    }
    void setValueAt(int pos, TrieNode newNode) {
        if(pos<chars.length && pos>=0){
            chars[pos] = newNode;
        } else{
            System.err.println("error: el valor no pudo ser
establecido");
        }
    }
}
```

Clase Trie (Arreglos)

```
package tp6;
class TrieArr {
    private TrieNodeArr root;
    private int end = 26;
```

```
TrieArr() {
    root = null;
}

void insertWord(String word) {
    TrieNodeArr aux;
    int pos, i=0;
    word = word.toLowerCase();
    if (root == null) {
        root = new TrieNodeArr(end+1);
    }
    aux = root;
    while (i < word.length()) {
        pos = getPosition(word.charAt(i));
        if(pos==-1||pos>end){
            break;
        } else{
            TrieNodeArr next=aux.getValueAt(pos);
            if (next == null) {
                next=new TrieNodeArr(end+1);
                aux.setValueAt(pos, next);
            }
            aux=next;
            i++;
        }
    }
    aux.setEnd(true);
}

boolean searchWord(String word) {
    word = word.toLowerCase();
    TrieNodeArr aux = root;
    int i = 0;
    int pos;
    if (aux == null) {
        return false;
    }
    while (i < word.length()) {
        pos = getPosition(word.charAt(i));
        if(pos==-1 || pos>end){
```

```
                break;
            } else{
                TrieNodeArr next= aux.getValueAt(pos);
                if (next!=null){
                    aux=next;
                    i++;
                }else{
                    return false;
                }
            }
        }
        return (aux != null && aux.isEnd());
    }

    static int getPosition(char c) {
        return c-'a';
    }
    static char getLetra(int c) {
        return (char) ('a'+c);
    }

    int cantidadPalabras(){
        TrieNodeArr aux = root;
        return cantidadPalabras(aux, 1);
    }
    private int cantidadPalabras(TrieNodeArr n, int indice){
        if(n==null || indice>= 28){
            return 0;
        }
        int cantidad = (n.getValueAt(0)!=null)?1:0;
        cantidad+=cantidadPalabras(n.getValueAt(indice),
indice+1);
        return cantidad;
    }
    void mostrarPalabra(){
        TrieNodeArr aux=root;
        mostrarPalabra(aux,"");
    }
    void mostrarPalabra(TrieNodeArr nodo, String palabra){
        if(nodo==null){
            return;
```

```
        }
        for (int i = 0; i < end; i++) {
            mostrarPalabra(nodo.getValueAt(i),
palabra+recuperarCaracter(i));
        }
        if(nodo.getValueAt(0)!=null){
            System.out.println(palabra);
        }
    }
    void mostrarPrefijo(String pref){
        TrieNodeArr aux=root;
        int i=0,pos=-1;
        while (i<pref.length()){
            pos=getPosition(pref.charAt(i));
            aux=aux.getValueAt(pos);
            i++;
        }
        mostrarPalabra(aux, pref);
    }
    char recuperarCaracter(int val){
        char c='a';

        for (int i = 1; i < val; i++) {
            c++;
        }

        return c;
    }
}
```

Clase TrieNode (Arreglos)

```
package tp6;
class TrieNodeArr {

    private TrieNodeArr[] children;
    private boolean end;

    TrieNodeArr(int size) {
        children = new TrieNodeArr[size];
        end=false;
    }
```

```
    TrieNodeArr getValueAt(int pos) {
        if(pos>=children.length && pos<0){
            return null;
        }
        return children[pos];
    }
    void setValueAt(int pos, TrieNodeArr newNode) {
        if(pos<children.length && pos>=0){
            children[pos] = newNode;
        } else{
            System.err.println("error: el valor no pudo ser
establecido");
        }
    }
    public boolean isEnd() {
        return this.end;
    }

    public boolean getEnd() {
        return this.end;
    }

    public void setEnd(boolean end) {
        this.end = end;
    }
}
```

Clase Trie (Listas)

```
package tp6;

public class TrieL {
    private TrieNodeL root;
    TrieL(){
        this.root=null;
    }
    void insertWord(String word){
        if(word==null||word.isEmpty()){
            return;
        }
        word.toLowerCase();
        root=insertWord(root,word,0);
```

```
    }
    private TrieNodeL insertWord(TrieNodeL nodo, String palabra,
int indice){
        if(indice==palabra.length()){
            return null;
        }
        char actual= palabra.charAt(indice);
        if(nodo==null){
            nodo=new TrieNodeL(actual);
        }
        if (nodo.getC()==actual) {

nodo.setDown(insertWord(nodo.getDown(),palabra,indice+1));
        }else{

nodo.setRight(insertWord(nodo.getDown(),palabra,indice+1));
        }
        return nodo;
    }
    public boolean searchWord(String palabra) {
        if (palabra == null || palabra.isEmpty()){
            return false;
        }
        return searchWord(root, palabra.toLowerCase(), 0);
    }

    private boolean searchWord(TrieNodeL nodo, String palabra, int
indice) {
        if (nodo == null){
            return false;
        }
        if (indice == palabra.length()){
            return true;
        }
        char actual = palabra.charAt(indice);
        if (nodo.getC() == actual) {
            return searchWord(nodo.getDown(), palabra, indice+ 1);
        } else {
            return searchWord(nodo.getRight(), palabra, indice);
        }
    }
```

```
    void mostrarPalabras(){
        TrieNodeL aux=root;
        mostrarPalabras(aux,"");
    }
    void mostrarPalabras(TrieNodeL nodo,String palabra){
        if(nodo==null){
            return;
        }
        String actual=palabra+nodo.getC();
        if(nodo.getDown()==null){
            System.out.println(actual);
        }
        mostrarPalabras(nodo.getDown(),actual);
        mostrarPalabras(nodo.getRight(),actual);
    }

    int cantidadPalabras() {
        TrieNodeL aux =root;
        return cantidadPalabras(aux);
    }

    private int cantidadPalabras(TrieNodeL nodo) {
        if (nodo == null) {
            return 0;
        }
        int cantidad = (nodo.getDown() == null) ? 1 : 0;
        cantidad += cantidadPalabras(nodo.getDown());
        cantidad += cantidadPalabras(nodo.getRight());
        return cantidad;
    }
    void mostrarPrefijo(String prefijo) {
        TrieNodeL nodo = buscarNodo(root, prefijo, 0);
        if (nodo != null) {
            mostrarPalabras(nodo, prefijo);
        }
    }
    private TrieNodeL buscarNodo(TrieNodeL nodo, String prefijo,
int indice) {
        if (nodo == null || indice == prefijo.length()) {
            return nodo;
        }
```

```
        char actual = prefijo.charAt(indice);
        if (nodo.getC() == actual) {
            return buscarNodo(nodo.getDown(), prefijo, indice +
1);
        } else {
            return buscarNodo(nodo.getRight(), prefijo, indice);
        }
    }
}
```

Clase TrieNode (Listas)

```
package tp6;
public class TrieNodeL {
    private char c;
    private TrieNodeL right;
    private TrieNodeL down;
    TrieNodeL(char car){
        this.c=car;
        this.right=null;
        this.down=null;
    }
    public char getC() {
        return this.c;
    }
    public void setC(char c) {
        this.c = c;
    }
    public TrieNodeL getRight() {
        return this.right;
    }
    public void setRight(TrieNodeL right) {
        this.right = right;
    }
    public TrieNodeL getDown() {
        return this.down;
    }
    public void setDown(TrieNodeL down) {
        this.down = down;
    }
}
```

Main

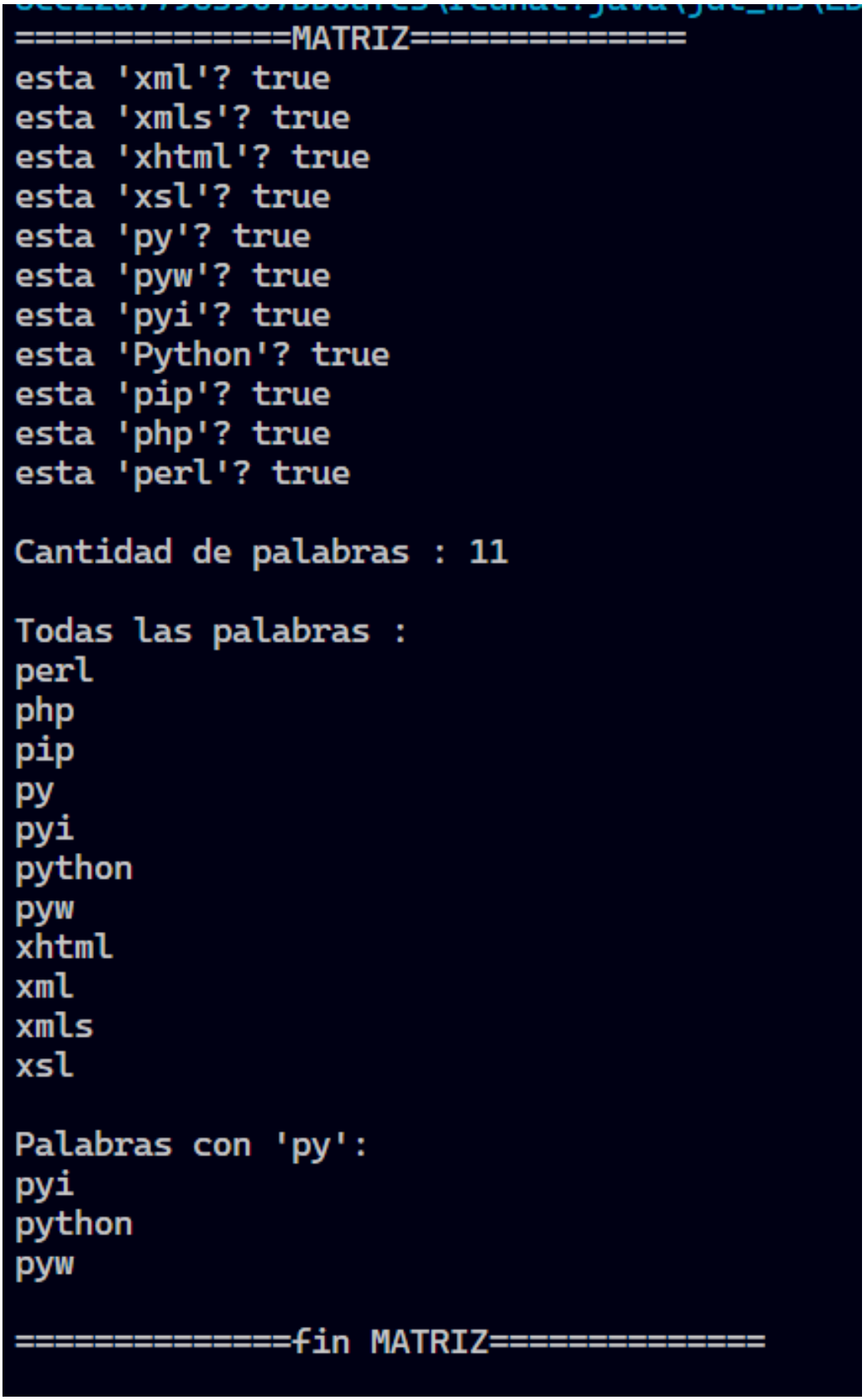
```
package tp6;
public class Main {
    public static void main(String[] args) {
        String[] words = {"xml", "xmls", "xhtml", "xsl", "py",
"pyw", "pyi", "Python", "pip", "php", "perl"};
        //--------MATRIZ--------
        System.out.println("==============MATRIZ==============");
        Trie trie = new Trie();
        // insertar palabras
        for (String word : words) {
            trie.insertWord(word);
        }
        // buscar palabras
        for (String word : words) {
            System.out.println("esta '" + word + "'? " +
trie.searchWord(word));
        }

        // cantidad de palabras
        System.out.println("\nCantidad de palabras : " +
trie.contarPalabras());

        // mostrar todas las palabras
        System.out.println("\nTodas las palabras : ");
        trie.mostrarPalabras();

        // mostrar palabras con un prefijo
        String pref = "py";
        System.out.println("\nPalabras con '" + pref + "':");
        trie.buscarPrefijos(pref);
        System.out.println("\n==============fin
MATRIZ==============");

        //--------ARREGLO ENLAZADO--------

System.out.println("\n==============ARREGLOS==============");
        TrieArr trieArr = new TrieArr();
        // insertar palabras
```

```
        for (String word : words) {
            trieArr.insertWord(word);
        }
        // buscar palabras
        for (String word : words) {
            System.out.println("esta '" + word + "'? " +
trieArr.searchWord(word));
        }
        // mostrar todas las palabras
        System.out.println("\nTodas las palabras del arreglo : ");
        trie.mostrarPalabras();

        // mostrar palabras con un prefijo
        String pref2 = "ph";
        System.out.println("\nPalabras con '" + pref2 + "':");
        trie.buscarPrefijos(pref2);

        System.out.println("\n==============FIN
ARREGLOS==============");

        //--------LISTAS ENLAZADAS--------

System.out.println("\n==============LISTAS==============");
        TrieL trieL = new TrieL();
        // insertar palabras
        for (String word : words) {
            trieArr.insertWord(word);
        }
        // buscar palabras
        for (String word : words) {
            System.out.println("esta '" + word + "'? " +
trieL.searchWord(word));
        }
        // mostrar todas las palabras
        System.out.println("\nTodas las palabras del arreglo : ");
        trie.mostrarPalabras();

        // mostrar palabras con un prefijo
        String pref3 = "xm";
        System.out.println("\nPalabras con '" + pref3 + "':");
        trie.buscarPrefijos(pref3);
```

```
        System.out.println("\n==============FIN
LISTAS==============");
    }
}
```

👇Resultado👇

Resultado consola



```
==============ARREGLOS==============
esta 'xml'? true
esta 'xmls'? true
esta 'xhtml'? true
esta 'xsl'? true
esta 'py'? true
esta 'pyw'? true
esta 'pyi'? true
esta 'Python'? true
esta 'pip'? true
esta 'php'? true
esta 'perl'? true

Todas las palabras del arreglo :
perl
php
pip
py
pyi
python
pyw
xhtml
xml
xmls
xsl

Palabras con 'ph':
php

==============FIN ARREGLOS==============
```

```
==============LISTAS==============
esta 'xml'? false
esta 'xmls'? false
esta 'xhtml'? false
esta 'xsl'? false
esta 'py'? false
esta 'pyw'? false
esta 'pyi'? false
esta 'Python'? false
esta 'pip'? false
esta 'php'? false
esta 'perl'? false

Todas las palabras del arreglo :
perl
php
pip
py
pyi
python
pyw
xhtml
xml
xmls
xsl

Palabras con 'xm':
xml
xmls

==============FIN LISTAS==============
PS C:\Users\GonzaloUlloa\Desktop\gon\EDA>
```

Referencias

[1] Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press. ISBN: 978-0262033848. Recuperado de: https://drive.google.com/file/d/1mhixZxHUC1qU7bofbRgk6yeQLNQvfMEz/view
[2] The-Algorithms. Implementación de *Trie en C#* [Repositorio GitHub]. Recuperado de: https://github.com/TheAlgorithms/C-Sharp/blob/master/DataStructures/Tries/Trie.cs