

Estructuras de Datos

Clase 18 – Procesamiento de Texto

(Tries)



Dr. Sergio A. Gómez
<http://cs.uns.edu.ar/~sag>



Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca, Argentina

Tries

- Un trie es una estructura de datos que se usa para implementar conjuntos de strings, y mapeos y diccionarios de string en un tipo E-
- Un trie es un árbol que factoriza prefijos comunes entre las cadenas almacenadas en el mismo.
- Los caminos de la raíz a las hojas representan las palabras del conjunto o las claves del mapeo.

Definición

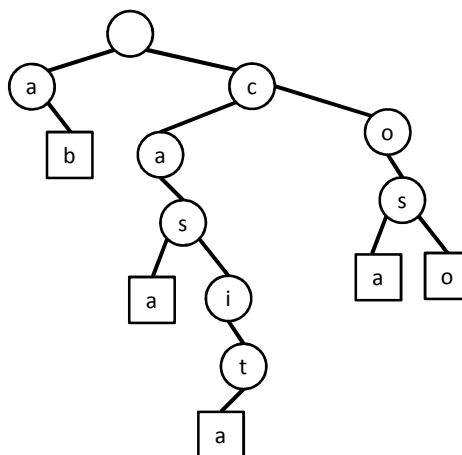
- Sea S un conjunto de s strings sobre un alfabeto Σ .
- Un trie para S es un árbol ordenado T tal que:
 - Cada nodo de T , excepto la raíz, está etiquetado con un carácter de Σ .
 - El orden los hijos de un nodo interno de T está determinado por el orden canónico de Σ .
 - T tiene s nodos externos, cada uno asociado con un string de S , tal que la concatenación de los rótulos de los nodos del camino de la raíz a una hoja v produce el string de S asociado a v .

Estructuras de datos - Dr. Sergio A. Gómez

3

Ejemplo (notación de GT)

Sea $S = \{ \text{"ab"}, \text{"casa"}, \text{"casita"}, \text{"cosa"}, \text{"coso"} \}$



Estructuras de datos - Dr. Sergio A. Gómez

4

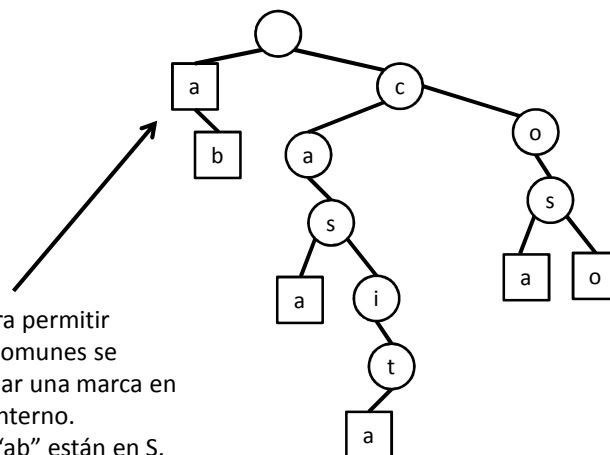
El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 “Estructuras de Datos. Notas de Clase”. Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019

Aclaración

- Nota: Para definir trie, El libro GT, sección 12.3.1, no permite que una palabra del trie sea prefijo de otra palabra del trie.
- Esto se soluciona agregando un carácter especial terminador (párrafo 3 de sec. 12.3.1), que es la política que tomaremos en esta clase.

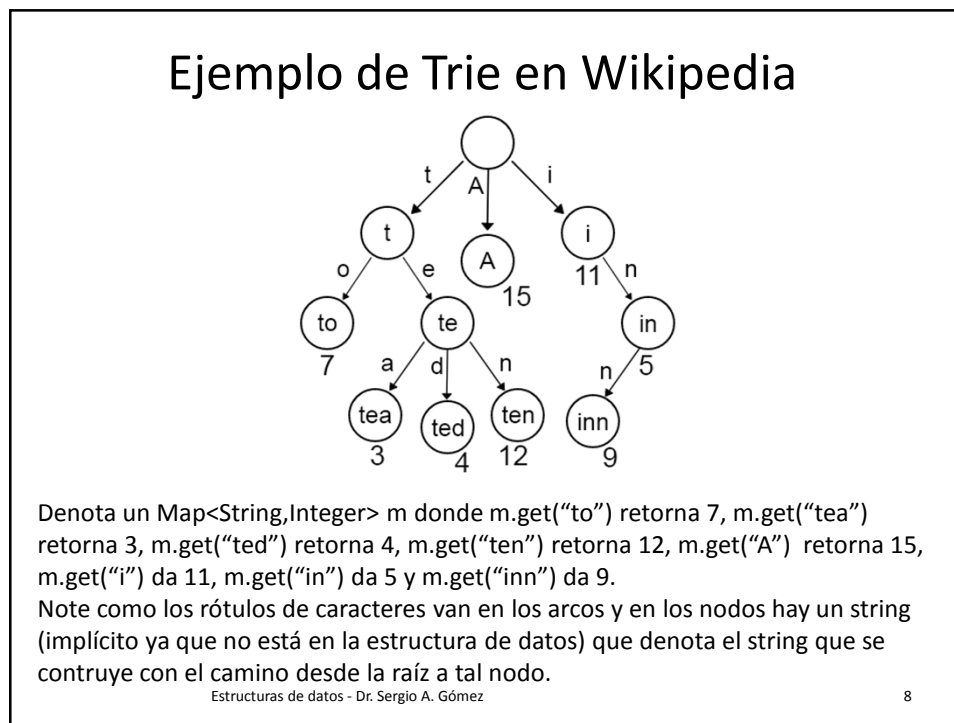
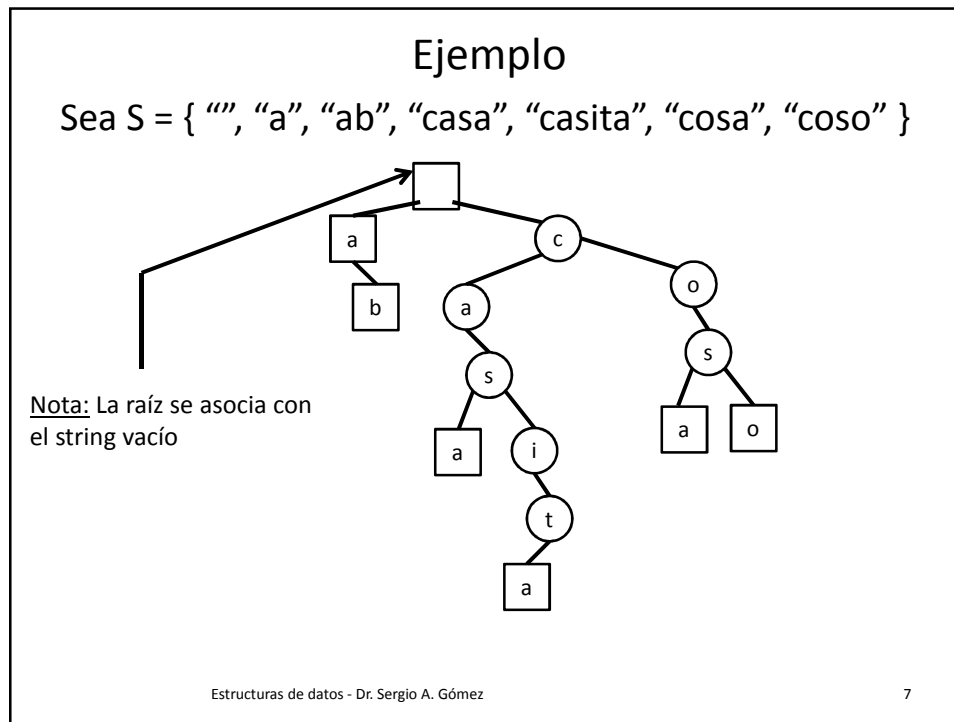
Ejemplo

Sea $S = \{ "a", "ab", "casa", "casita", "cosa", "coso" \}$



Nota: Para permitir prefijos comunes se puede usar una marca en el nodo interno.
Ej: "a" y "ab" están en S.

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
"Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019



El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019

Propiedades

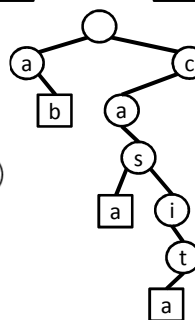
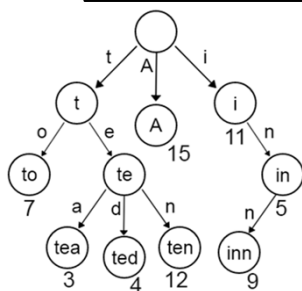
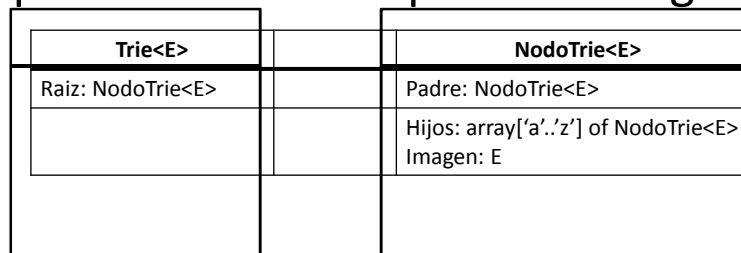
Un trie almacenando una colección S de s strings de longitud total n sobre un alfabeto de tamaño d cumple:

- Cada nodo interno de T tiene a lo sumo d hijos
- T tiene s nodos externos
- La altura de T es igual a la longitud del string más largo de S
- El número de nodos de T es $O(n)$.

Estructuras de datos - Dr. Sergio A. Gómez

9

Implementación: Mapeo de String en E



Los cuadraditos de GT corresponden a Imagen != null. En el trie de wikipedia, los números se almacenan en el campo *Imagen* y los arcos corresponden a los índices del arreglo *Hijos*. Note que nosotros no representamos las mayúsculas.

Estructuras de datos - Dr. Sergio A. Gómez

10

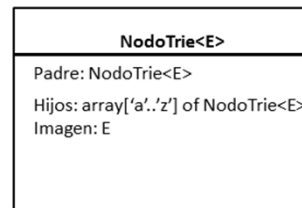
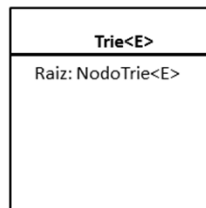
El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente: "Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019

```

public class Trie<E>
{
    protected NodoTrie<E> raiz;

    // Clase anidada estática:
    // Permite usar la clase Trie como paquete:
    public static class ClaveInexistenteException extends Exception
    {
        public ClaveInexistenteException( String msg )
        {
            super( msg );
        }
    }
}

```



Estructuras de datos - Dr. Sergio A. Gómez

11

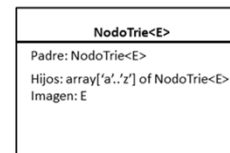
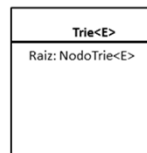
// NodoTrie es parte de la implementación:

```

private class NodoTrie<E> {
    protected E imagen;
    protected NodoTrie<E> [] hijos;
    protected NodoTrie<E> padre;

    public NodoTrie(NodoTrie<E> p) {
        hijos = new NodoTrie[26];
        imagen = null; padre = p; }
    public void setImagen(E imagen) { this.imagen = imagen; }
    public E getImagen() { return imagen; }
    public void setHijo(int i, NodoTrie<E> hijo ) { hijos[i] = hijo; }
    public NodoTrie<E> getHijo(int i) { return hijos[i]; }
    public void setPadre( NodoTrie<E> padre ) { this.padre = padre; }
    public NodoTrie<E> getPadre() { return padre; }
}
// Constructor
public Trie() { raiz = new NodoTrie<E>(null); }

```



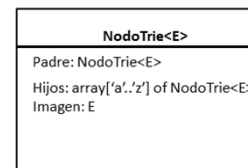
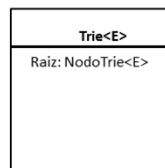
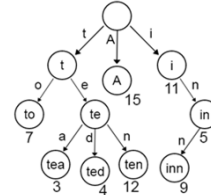
Estructuras de datos - Dr. Sergio A. Gómez

12

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 “Estructuras de Datos. Notas de Clase”. Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019

```
public E put( String clave, E valor ) {
    return putaux( clave, valor, 0, clave.length(), raiz, null );
}
```

```
private E putaux( String clave, E valor, int i, int n,
    NodoTrie<E> raiz, NodoTrie<E> padre ) {
    if( i < n ) {
        int indice = ((int) clave.charAt(i)) - ((int) 'a');
        if( raiz.getHijo(indice) == null )
            raiz.setHijo( indice, new NodoTrie<E>(raiz) );
        return putaux( clave, valor, i+1, n, raiz.getHijo(indice), raiz );
    } else { // i == n
        E imavieja =
            raiz.getImagen();
        raiz.setImagen( valor );
        return imavieja;
    }
}
```

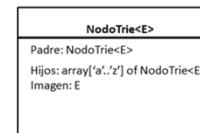
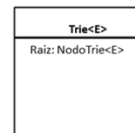
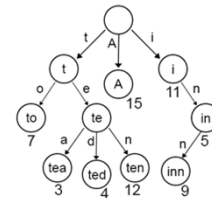


Estructuras de datos - Dr. Sergio A. Gómez

13

```
public E get( String clave )
{
    return getaux( clave, 0, clave.length(), raiz );
}
```

```
private E getaux( String clave, int i, int n, NodoTrie<E> raiz )
{
    if( i == n )
        return raiz.getImagen();
    else {
        int indice = (int) clave.charAt(i) - (int) 'a';
        if( raiz.getHijo(indice) == null ) return null; // La clave no existe en el
        árbol trie.
        return getaux( clave, i+1, n, raiz.getHijo(indice) );
    }
}
```



Estructuras de datos - Dr. Sergio A. Gómez

14

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019

```

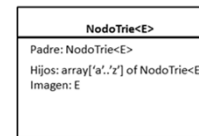
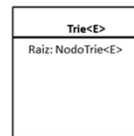
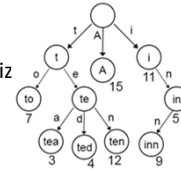
public E remove( String clave ) throws ClaveInexistenteException {
    return removeaux( clave, 0, clave.length(), raiz, 0 ); }

private E removeaux( String clave, int i, int n, NodoTrie<E> raiz, int indiceRaiz
    throws ClaveInexistenteException {

    E toRet = null;
    if( i == n ) {
        if( raiz.getImagen() == null )
            throw new ClaveInexistenteException( "Clave inexistente " );
        E imagen = raiz.getImagen();
        raiz.setImagen( null );
        toRet = imagen;
    } else {
        int indice = (int) clave.charAt(i) - (int) 'a';
        if( raiz.getHijo(indice) == null )
            throw new ClaveInexistenteExcepti

        toRet = removeaux( clave, i+1, n, raiz.getHijo
    }
    // Chequear si el nodo quedó todo nulo: no tiene valor ni hijos.
    if( todoNulo( raiz ) ) { // Desconectar este nodo salvo que sea la raiz
        if( raiz != this.raiz ) { raiz.getPadre().setHijo( indiceRaiz, null );
            raiz.setPadre( null ); }
    } return toRet; }

```

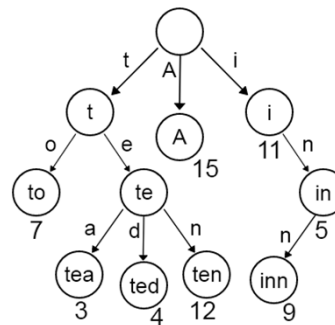


Estructuras de datos - Dr. Sergio A. Gómez

15

Complejidad temporal

- Sea un conjunto S implementado con un trie T sobre un alfabeto Σ .
- Sea s =cardinal de S , d =cardinal de Σ , m =largo de un string a procesar
- $T_{\text{put}}(s,d,m) = O(m)$
- $T_{\text{get}}(s,d,m) = O(m)$
- $T_{\text{remove}}(s,d,m) = O(dm)$



Estructuras de datos - Dr. Sergio A. Gómez

16

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
 "Estructuras de Datos. Notas de Clase". Sergio A. Gómez. Universidad Nacional del Sur. (c) 2013-2019

Aplicaciones: Word matching

- Problema: Dado un documento determinar todas las apariciones de una palabra determinada.
- Ejemplo: Opción Ctrl-F en Google Chrome.
- Solución: Construir un trie donde por cada palabra se almacena la lista de posiciones (una lista de enteros) de las apariciones de la palabra.
- Referencia: Ver Figura 12.7 de GT.

Bibliografía

- Capítulo 12, Sección 3 de M. Goodrich & R. Tamassia, Data Structures and Algorithms in Java. Fourth Edition, John Wiley & Sons, 2006.