UT5_PD2

Ejercicio #1 - construirIndice

Lenguaje natural

Recorre cada nodo del Trie y a medida que avanza por el árbol va formando palabras letra por letra. Cuando detecta que llegó al final de una palabra, la imprime junto con las páginas donde aparece.

Precondiciones

1. Debe ser un Trie válido donde
    a. Cada nodo final (palabra) debe tener un atributo “paginas” donde guarda una lista de paginas donde aparece la misma
    b. Las palabras tienen que haberse insertado correctamente
2. La raiz no debe ser nula

Postcondiciones

1. Se imprime una lista de cada palabra en el Trie con las respectivas páginas donde aparece.
2. El Trie no se modifica

Pseudo

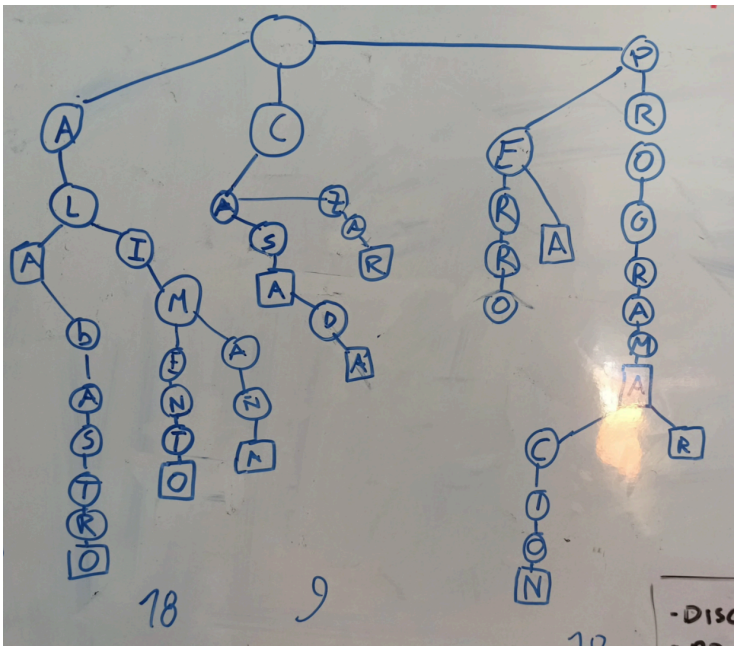
```
clase NodoTrie:
---------------

construirIndice(nodo: nodoTrie, palabraActual: String) → void
COM
   SI nodo.esPalabra ENTONCES
      IMPRIMIR palabraActual + " " + nodo.paginas) --- O(1)
   FIN SI

   PARA CADA x DESDE 0 hasta largoAlfabeto HACER --- O(m)
      SI nodo.hijos[x] <> NULO ENTONCES
         sigLetra ← (char) ('a' + x) --- O(1)
         construirIndice(nodo.hijos[x], palabraActual + sigLetra) --- O(n)
      FIN SI
```

```
    FIN PARA CADA
FIN

clase Trie:
-----------

mostrarIndice() → void
COM
    SI raiz <> NULO ENTONCES
        raiz.construirIndice(raiz, "")
    FIN SI
FIN
```

Análisis de tiempo de ejecución

O(N), donde N es el número total de nodos en el Trie, cada nodo se visita una vez.

Parte 3



1. ¿Cuántas comparaciones realiza el algoritmo de búsqueda si el argumento es “Programa”?
2. ¿Cuántas comparaciones realiza el algoritmo de búsqueda si el argumento es “Proselitismo?•

3. ¿Cuántas comparaciones se realizan para insertar la palabra “cazadores”?
4. ¿cuál es la altura del árbol trie resultante?
5. ¿cuál es su tamaño?

RESPUESTAS

1. 8 comparaciones (1 por letra).
2. 4 comparaciones, termino en **prog.**
3. 9 comparaciones.
4. Dado que la altura de un Trie se define por el largo de su cadena más larga → **ALTURA: 12** porque la cadena más larga es *Programación* con 12 letras.
5. 46 nodos, contando al nodo raíz.

Ejercicio #2

Lenguaje natural

El algoritmo recorre letra por letra del Trie. Si encuentra todos los nodos y el último marca una palabra válida, retorna sus páginas, en caso contrario devuelve nulo.

Precondiciones

1. El Trie está construido correctamente.
2. La raiz no puede ser nula

Postcondiciones

1. Retorna la lista de páginas donde aparece la palabra si está.
2. Si la página no está, devuelve NULO
3. El trie no se modifica

```
clase NodoTrie:
-------------
buscarPaginas(palabra: String) → Lista<int>
COM
   nodoActual ← este

   PARA CADA x desde 0 hasta palabra.length HACER
```

```
        indice ← palabra.charAt(x) - 'a'
        SI nodoActual.hijos[indice] == NULO ENTONCES
            RETORNAR NULO
        FIN SI

        nodoActual ← nodoActual.hijos[indice]
    FIN PARA CADA

    SI nodoActual.esPalabra ENTONCES
        RETORNAR nodoActual.paginas
    SINO
        RETORNAR NULO
    FIN SI
FIN

clase Trie:
-------------
buscarPaginas(palabra: String) → Lista<int>
COM
    SI raiz == NULO ENTONCES
        RETORNAR NULO
    FIN SI

    RETORNAR raiz.buscarPaginas(palabra)
FIN
```

Orden de tiempo de ejecución

Annuntio Vobis Gaudium Magnum: Habemus O(N)! → Donde N es la longitud de la palabra