UT5_PD3

Property	COMPLETED
Date	@May 26, 2022
BLOCKED	

Ejercicio 1:

1. Hacer una LinkedList<Integer> en el nodo que contenga las paginas

[IMPLEMENTACION EN EL PROYECTO EN JAVA]

Ejercicio 2:

```
// Funcion a implementar

public String buscar2(String s){
        TNodoTrie nodoActual = this;
        int cont = 0;
        for (int c = 0; c < s.length(); c++) {
            int indice = s.charAt(c) - 'a';
            cont++;
            if (nodoActual.hijos[indice] == null) {
                return s + " not found in trie, cant. comparaciones: " + cont;
            }
            nodoActual = nodoActual.hijos[indice];
        }
        if(nodoActual.esPalabra){
            return s + " Esta en las paginas: " + nodoActual.paginas.toString().replace("[","").replace("]","");
        }
        return s + " not found in trie, cant. comparaciones: " + cont;
    }
```

Ejercicio 3:

1.
    a. LISTAS:

       Funcionaria igual salvo el detalle de que es (para la mayoria de los metodos):

       $$T(n) = O(n)$$

    b. COLAS:

       No tendria sentido, pues el TDA cola es poco aplicable a este caso.

    c. ARBOLES:

       Es similar, pero utiliza mucho la recursividad, lo cual lo hace mas dificil de programar.

2. Si fuese unas pocas lineas, lo mas recomendable es usar una lista. Pues un arbol y un trie a nivel de dificultad son similares, ambos deben ser armados cuidadosamente. La desventaja de las listas es que su $T(n) = O(n)$ para casi todos los metodos, por lo cual es menos eficiente que un tire asi como un

arbol. Pero ganamos en sencilles en comparacion a los otros dos, lo cual para pocas lineas, es mas importante que la optimizacion.