	UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS	Código: ST245
		Estructura de Datos 1

### Laboratorio Nro. 3: Listas enlazadas y Listas hechas con arreglos

**Omar Alexis Becerra Sierra**

Universidad Eafit  
Medellín, Colombia  
oabecerras@eafit.edu.co

**Juan José Tamayo**

Universidad Eafit  
Medellín, Colombia  
jjtamayo@eafit.edu.co

**3.1** Teniendo en cuenta lo anterior, calculen la complejidad de cada ejercicio con cada implementación de listas. Es decir, hagan una tabla, en cada fila coloquen el número del ejercicio, en una columna la complejidad de ese ejercicio usando *ArrayList* y en la otra columna la complejidad de ese ejercicio usando *LinkedList*.

	<i>ArrayList</i>	<i>LinkedList</i>
<b>Ejercicio 1.1</b>	$O(n)$	$O(n^2)$
<b>Ejercicio 1.2</b>	$O(n)$	$O(n^2)$
<b>Ejercicio 1.3</b>	$O(n^2)$	$O(n^3)$
<b>Ejercicio 1.4</b>	$O(n^2)$	$O(n^3)$

**3.2** Teniendo en cuenta lo anterior, verifiquen, utilizando *JUnit*, que todos los *test* escritos en el numeral 1.2 pasan. Muestren, en su informe de PDF, que los *test* se pasan correctamente; por ejemplo, incluyendo una imagen de los resultados de los *test*.

**DOCENTE MAURICIO TORO BERMÚDEZ**

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627 Correo:

mtorobe@eafit.edu.co

### Test arraylist

```
import java.util.*;

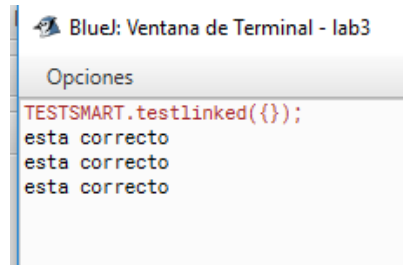
/**
 * Este es el test para la clase de SmartInsert en linkedList
 */
public class TESTSMART{
    public static void testlinked(String [] args){
        SmartInsert x = new SmartInsert();
        LinkedList<Integer> linked = new LinkedList<>();//linked list original
        linked.addAll(Arrays.asList(new Integer[] {1, 3, 5, 7}));
        LinkedList<Integer> c = new LinkedList<>(); //linked de prueba
        c.addAll(Arrays.asList(new Integer[] {1, 3, 5, 7, 9}));
        x.SmartInsert(linked,9);
        boolean d = linked.equals(c);
        if(d){
            System.out.println("esta correcto");
        }else{
            System.out.println("NO esta correcto");
        }

        x.SmartInsert(linked,7);
        d = linked.equals(c);

        if(d){
            System.out.println("esta correcto");
        }else{
            System.out.println("NO esta correcto");
        }

        x.SmartInsert(linked,12);
        d = !linked.equals(c);

        if(d){
            System.out.println("esta correcto");
        }else{
            System.out.println("NO esta correcto");
        }
    }
}
```

**Resultado de los linkedtest**

```
Blue: Ventana de Terminal - lab3
Opciones
TESTSMART.testlinked({});
esta correcto
esta correcto
esta correcto
```

**Test array**


```
/**
 * Este es el test para la clase de SmartInsert en array
 */
public static void testarraylis(String [] args){
    SmartInsert x = new SmartInsert();
    ArrayList<Integer> array = new ArrayList<>(); //arraylist original
    array.addAll(Arrays.asList(new Integer[] {2, 4, 6, 8}));
    ArrayList<Integer> c= new ArrayList<>(); //ararylist prueba
    c.addAll(Arrays.asList(new Integer[] {2, 4, 6, 8, 7 }));
    x.SmartInsert(array,8);
    boolean d = array.equals(c);
    if(!d){
        System.out.println("esta correcto");
    }else{
        System.out.println("NO esta correcto");
    }

    x.SmartInsert(array,7);
    d = array.equals(c);

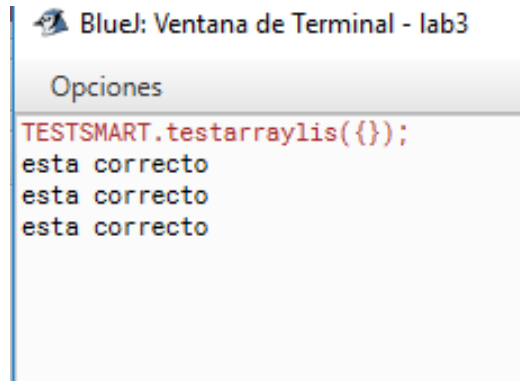
    if(d){
        System.out.println("esta correcto");
    }else{
        System.out.println("NO esta correcto");
    }

    x.SmartInsert(array,2);
    d = array.equals(c);

    if(d){
        System.out.println("esta correcto");
    }else{
        System.out.println("NO esta correcto");
    }
}
```

	<p style="text-align: center;">UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS</p>	<p>Código: ST245</p>
		<p>Estructura de Datos 1</p>

## Resultados del Array test



```
BlueJ: Ventana de Terminal - lab3
Opciones
TESTSMART.testarraylis({});
esta correcto
esta correcto
esta correcto
```

### 3.3 Expliquen con sus propias palabras cómo funciona la implementación del ejercicio 2.1 y [opcionalmente] el 2.2

El problema es que en el texto se encuentra un símbolo “[“ que hace que se mueva al principio, y otro símbolo “]” que hace que se mueva al final entonces.

Primero creamos el scanner para poder leer la entrada y declaramos un String que luego nos servirá para guardar cada línea del texto, hacemos un ciclo mientras la siguiente línea sea diferente de null (también guardamos la línea en el String), en el ciclo creamos una LinkedList, y define una variable “índex” que nos servirá como índice en 0, hacemos otro ciclo for each en el que se transforma el String en un arreglo de chars y toma uno por uno en una variable llamada “c”, en este ciclo for each hay 3 condiciones las cuales son:

- 1) si la c no es un “[“ ni un “]” entonces se adiciona a la lista en la posición index++ esto se hace para adicionarle 1 al índice y así ir avanzando en la lista
- 2) si la c es un “[“, el índice se cambia a 0 de esta manera se empieza a adicionar desde la primera posición
- 3) si es un “]” se cambia el índice al final de la lista de esta manera empieza a adicionar en el final.

Cuando se termina el ciclo for each, pero aun adentro del while, se crea un StringBuilder para hacer la salida, se hace un ciclo for each hasta el final de la lista tomando cada uno de los char que la componen y enviándolos al StringBuilder. Y al final se convierte el StringBuilder en un String y luego imprimiendo el String ya arreglado.

**3.4 Calculen la complejidad del ejercicio realizado en el numeral 2.1 y [opcionalmente] 2.2, y agregarla al informe PDF**

```
* metodo principal encargado de leer el archivo txt y llamar el metodo arreglar
*/
public static void lecturaarreglar(){
    try{
        File f = new File("leer.txt"); //C
        Scanner s = new Scanner(f); //C
        while(s.hasNextLine()){ //N
            //lee linea
            String a = ""; //C * N
            a += s.next(); //C * N
            System.out.println(arreglar(a)); //C * N
        }
    }catch(FileNotFoundException e){
        System.out.println("El archivo no existe..."); //C
    }
}

/**
 * este metodo es el encargado de arreglar el texto este metodo recibe un string y lo
 */
private static String arreglar(String consola){
    LinkedList<Character> lista = new LinkedList<>(); //C
    int index=0; //C
    for (char c: consola.toCharArray()){ // N
        if (c == ' '){ //C * N
            index= lista.size(); //C * N
        }
        else if(c == '['){ //C * N
            index=0; //C * N
        }else {
            lista.add(index++,c); //C * N
        }
    }
    StringBuilder string = new StringBuilder();
    for (char c: lista){ //N
        string.append(c); //C * N
    }
    return string.toString(); //C
}
```

**Complejidad del algoritmo**

$$T(n) = C * n$$

$$T(n) \text{ es } O(c * n)$$

$$T(n) \text{ es } O(n)$$

### 3.5 Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral 3.3

Solo tengo una variable n

#### 4) Simulacro de Parcial

1. c
2. c
3.
  - a. `q.size()>1`
  - b. `<=`
  - c. `q. remove ()`
  - d. `q. remove ()`
4.
  - a. `lista. size () > 0`
  - b. `lista.pop ()`
5.
  - a. `auxiliar1. size () > 0`
  - b.
    - 1) `auxiliar2. size () > 0`
    - 2) `personas. offer(edad)`
6. a.  $O(n^2)$

**5) Resumen de la lectura****a). Título: Chapter Five: Linked Lists****b) Resumen:**

The linked list is a versatile mechanism suitable for use in many kinds of general-purpose. Linked lists aren't the solution to all data storage problems, but they are surprisingly versatile and conceptually simpler than some other popular structures such as trees. In a linked list, each data item is embedded in a link. A link is an object of a class called something like Link. Each link object contains a reference (usually called next) to the next link in the list. A field in the list itself contains a reference to the first link. a Link object doesn't really contain another Link object, although it may look like it does. The next field of type Link is only a reference to another link, not an object. A reference is a number that refers to an object. It's the object's address in the computer's memory, but you don't need to know its value; you just treat it as a magic number that tells you where the object is. In a given computer/operating system, all references, no matter what they refer to, are the same size. In a list the only way to find an element is to follow along the chain of elements. It's more like human relations.

**c) Mapa:**