ESTRUCTURA DE DATOS 1
Código ST0245

Taller en Sala 7
Listas Enlazadas

Objetivo: 1. Comparar las ventajas y desventajas de las implementaciones dinámicas y estáticas de estructuras de datos. **2.** Escoger la estructura de datos adecuada para resolver un problema dado. **3.** Usar la notación O para encontrar formalmente la complejidad asintótica en tiempo de algoritmos.

Consideraciones: Lean y verifiquen las consideraciones de entrega,

Trabajo en Parejas

Mañana, plazo de entrega

Docente entrega plantilla de código en GitHub

Sí .cpp, .py o .java

No .zip, .txt, html o .doc

Alumnos entregan código sin comprimir GitHub

En la carpeta Github del curso, hay un código iniciado y un código de pruebas (tests) que pueden explorar para solucionar los ejercicios

Estructura del documento: a) Datos de *vida real*, **b)** I*ntroducción* a un problema, **c)** Problema a resolver, **d)** Ayudas. Identifiquen esos elementos así:

a) **b)** **c)** **d)**

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ejercicios a resolver

En la vida real, las listas enlazadas se usan para representar objetos en videojuegos http://bit.ly/2mcGa5w y para modelar pistas en juegos de rol http://bit.ly/2lPyXGC

1 Existe un problema con la trazabilidad de la cadena de suministro agrícola. Los clientes no entienden las etiquetas de sostenibilidad ni el origen de los productos. No disponen de información exhaustiva sobre, por ejemplo, las emisiones que se producen a lo largo de la cadena y la situación de los derechos humanos en los lugares de producción. Esta incertidumbre es un obstáculo para aumentar la compra de productos de origen sostenible para los clientes con un alto poder adquisitivo. Su trazabilidad se refiere a la documentación relacionada con todos los procesos de la cadena de suministro. Una solución para la trazabilidad es la tecnología de cadena de bloques.

La cadena de bloques (*Blockchain*) es un tipo de tecnología de almacenamiento distribuido que guarda la información con el consenso de todas las partes participantes. Una vez que la información es almacenada y verificada por todas las partes, crea un registro indeleble, resistente a la manipulación de cualquier parte individual. A diferencia de una base de datos centralizada, mantenida por una sola entidad, una cadena de bloques continúa funcionando incluso si los participantes individuales se retiran. Una cadena de bloques es una serie de bloques que contienen un número de hash del dato y un número de hash del bloque anterior. Técnicamente, se puede decir que una cadena de bloques es una lista simplemente enlazada. ¡Apliquemos esto a la apicultura!

Implementen una lista simplemente enlazada para almacenar coordenadas de abejas que permita insertar y borrar abejas en cualquier posición. ¿La complejidad del método *agregar abeja* permite que su lista enlazada sea utilizada con millones de abejas? ¿Cuál es la complejidad de agregar n abejas?

Utilicen los conjuntos de datos que se encuentran en la carpeta *datasets*, en Github, para probar su algoritmo.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

2 **Adicionen a la estructura de datos desarrollada en el punto anterior, la funcionalidad de buscar una abeja y decir en qué posición aparece o decir si no aparece.**

3 **[Ejercicio Opcional] En la clase *Taller 7*, definan un método que calcule el máximo valor de los elementos de una lista enlazada de forma recursiva.**

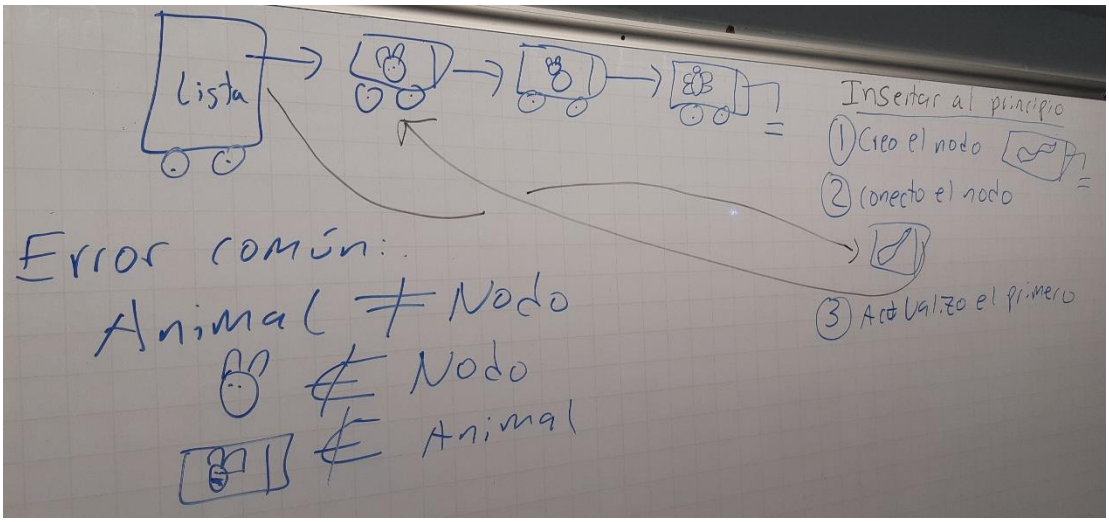
4 **[Ejercicio Opcional] En la clase *Taller 7*, implementen un algoritmo recursivo que permita comparar el contenido de dos listas. Este debe retornar verdadero en caso de que ambas listas sean iguales, o falso en caso contrario.**

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ayudas para resolver los Ejercicios

Ejercicio 1

Error Común 1: Es diferente el dato que está en la lista (el animalito) y el nodo, son dos cosas diferentes. Además, consideren el orden de las acciones que se muestra en la gráfica para insertar al principio de una lista enlazada.



Pista 2: Diferencien *LinkedListMauricio de LinkedList* del API de Java. Un error común es creer que todo se soluciona llamando los métodos existentes en *LinkedList* y, no es así, la idea es implementar una lista con arreglos nosotros mismos. A continuación, un ejemplo del error:

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```java
// Retorna el tamaño actual de la lista
public int size()
{
    return size();
}

// Retorna el elemento en la posición index
public int get(int index)
{
    return get(index);
}

// Inserta un dato en la posición index
public void insert(int data, int index)
{
    if (index <= size())
    {
        insert(data,index);
    }
}

// Borra el dato en la posición index
public void remove(int index)
{
    remove(index);
}

// Verifica si está un dato en la lista
public boolean contains(int data)
{
    return contains(data);
}
```

Pista 3: Tenemos la siguiente implementación de lista:

```java
// Usar esto cuando se salga el índice
import java.lang.IndexOutOfBoundsException;
// Una lista simplemente enlazada
public class LinkedListMauricio {
   // Un nodo para una lista simplemente enlazada
     public class Node {
          public int data;
          public Node next;
          public Node(int data) {
          next = null;
          this.data = data;
          }
      }

private Node first;
private int size;
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
public LinkedListMauricio()
{
	size = 0;
	first = null;
}

	/**
	 * Returns the node at the specified position in this list.
	 * @param index - index of the node to return
	 * @return the node at the specified position in this list
	 * @throws IndexOutOfBoundsException
	 */
	private Node getNode(int index) throws IndexOutOfBoundsException {
		if (index >= 0 && index < size) {
			Node temp = first;
			for (int i = 0; i < index; i++) {
				temp = temp.next;
			}
			return temp;
		} else {
			throw new IndexOutOfBoundsException();
		}
	}

	/**
	 * Returns the element at the specified position in this list.
	 * @param index - index of the element to return
	 * @return the element at the specified position in this list
	 */
	public int get(int index) {
		Node temp = null;
		try {
			temp = getNode(index);
		} catch (IndexOutOfBoundsException e) {
			e.printStackTrace();
			System.exit(0);
		}

		return temp.data;
	}
```

```
// Retorna el tamaño actual de la lista
public int size()
{
	...
}

// Inserta un dato en la posición index
public void insert(int data, int index)
{
	...
}
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
// Borra el dato en la posición index
public void remove(int index)
{
      ...
}

// Verifica si está un dato en la lista
public boolean contains(int data)
{
      ...
}

}
```

Ejemplo 1: Para la lista [1,2,3,4], el método *insert*(10, lista.size() -1) agrega al final el número 10, quedando la lista [1,2,3,4,10]. El *insert*(3, 0) agrega al principio el número 3, quedando la lista [3, 1, 2, 3, 4, 10].

Error Común 1:

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

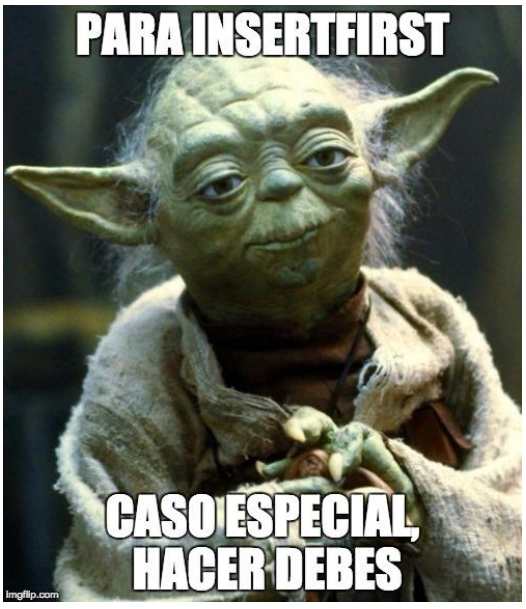
Ejercicio 2

Pista 1: Al código anterior agregue los siguientes métodos *remove* y *contains*

Error Común 1: Al hacer *contains*, NO haga un ciclo que llame n veces al método get porque como *get* es O(n), al llamarlo *n* veces queda el método *contains* $O(n^2)$ y se puede hacer en O(n). **OJO.**

Error Común 2: Mucha atención a estos casos especiales

.



PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

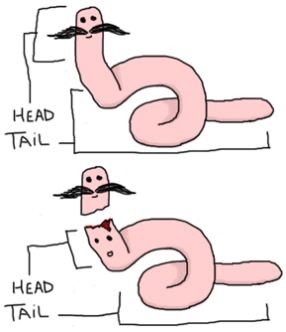
Ejercicio 3

Pista 1: Utilicen su propia implementación de listas, pues en la que trae Java en su API, no es posible acceder directamente a la clase *nodo*, y necesitaría usar iteradores en su lugar.

Pista 2: Paso 1: Condición de parada

Pista 3:

```
private static int maximoAux(Nodo nodo) {
```




```
}
public static int maximo(LinkedListMauricio lista) {
  return maximoAux(lista.first);
}
```

Ejemplo 1: Si tenemos una lista = [1,2,3,4], el método *maximo(lista)* retorna 4.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Error Común 1:

Ejercicio 4

Ejemplo 1: *Comparar*([1,2,3], [1,2,3]) retorna verdadero y *comparar*([1,2,3],[1,2,3,4]) retorna falso.

Error Común 1:

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

¿Alguna inquietud?

CONTACTO

Docente Mauricio Toro Bermúdez
Teléfono: (+57) (4) 261 95 00 Ext. 9473
Correo: mtorobe@eafit.edu.co
Oficina: 19- 627

Agenden una cita dando clic en la pestaña -*Semana*- de http://bit.ly/2gzVg10