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)**

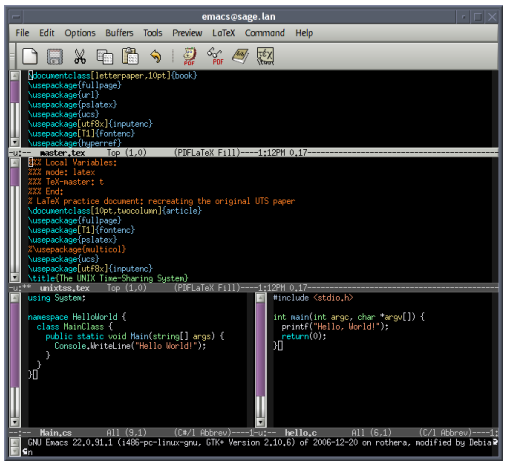
PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

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 Para representar un programa en un editor de texto como Emacs es necesario utilizar una estructura dinámica porque el tamaño del documento crece conforme el usuario escribe nuevos caracteres o líneas.

Adicionalmente, el usuario debe tener la posibilidad de eliminar un caracter, causando que se mueva todo lo que sigue de él hacia la atrás. El usuario también debe poder insertar un carácter, causando que todo que sigue de él se mueva hacia adelante**.**



Implementen una lista enlazada para un editor de texto que permita insertar y borrar caracteres en cualquier posición. ¿La complejidad del método agregar caracter permite que su lista sea utilizada en un editor de texto? ¿Cuál es la complejidad de agregar n caracteres?"

2 **Adicionen a la estructura de datos desarrollada en el punto anterior, la funcionalidad de buscar un caracter 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.**

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 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

Pista 1: Implementen en el código los métodos *size* e *insert* de una lista simplemente 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:

```
// 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);
}
```

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

Pista 3: Tenemos la siguiente implementación de lista:

```
// 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;
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 {
```

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

```
                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)
{
    ...
}

// Borra el dato en la posición index
public void remove(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

```
// 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

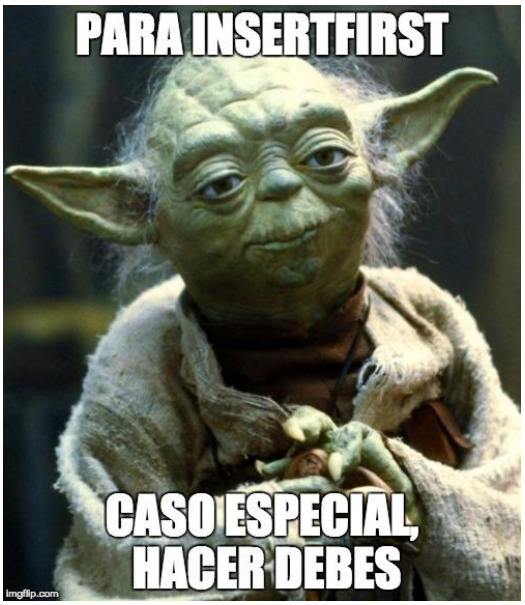
Ejercicio 2

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

Error Común 1: Al hacer *contains*, es hacer 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: Al hacer *contains*, es hacer 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.**

.



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

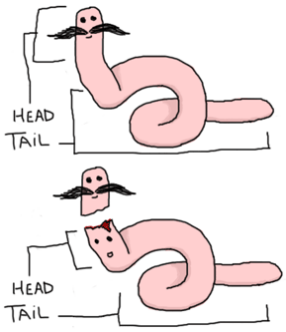
Ejercicio 3

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