

# Informe de Laboratorio 04

## Tema: Sort y Listas Enlazadas

INFORMACIÓN BÁSICA					
ASIGNATURA:	Estructura de Datos y Algoritmos				
TÍTULO DEL TRABAJO:	Sort y Listas Enlazadas				
NÚMERO DE TRABAJO:	04	AÑO LECTIVO:	2023-A	NRO. SEMESTRE:	III
FECHA DE PRESENTACIÓN:	11/06/23	HORA DE PRESENTACIÓN:	23:59		
INTEGRANTE (s)				NOTA (0-20)	
Hidalgo Chinchay, Paulo Andre Betanzos Rosas, Taylor Anthony Villafuerte Ccapira Frank Alexis					
DOCENTE(s): Mg. Edith Giovanna Cano Mamani					

Tabla 1: Mi tabla extendida

INTRODUCCIÓN
aqui ira la intro
MARCO CONCEPTUAL
aqui ira la MARCO CONCEPTUAL
SOLUCIONES Y PRUEBAS
<p>Ejercicio 1 //taylor</p> <p>Ejercicio 2 Para lograr ejecutar el algoritmo de ordenamiento por insercion primero fue necesario crear la clase ListaDoble el cual tenia 2 nodos, uno inicial y otro final. Con esto se lograba recorrer la lista de el fin al inicio y en viceversa. El metodo addFinal permitia insertar un nodo al final y el otro metodo addInicio, al inicio, dependeiendo si es que estaba vacia o no la lista.</p> <pre> public void addInicio(E x){     if(!isEmpty()){         inicio=new Node&lt;&gt;(x,inicio,null);         inicio.getNextNode().setPreviousNode(inicio);     }else {         inicio=fin= new Node&lt;&gt;(x);     } }  public void addFinal(E x){     if(!isEmpty()){         fin=new Node&lt;&gt;(x,null,fin);         fin.getPreviousNode().setNextNode(fin);     }else {         inicio=fin= new Node&lt;&gt;(x);     } } </pre>

Asimismo se agrego el metodo isEmpty para saber si estaba vacia y se sobrescribio el metodo toString para mostrar retornar la data del 1er al ultimo nodo. Los getters y setters se omitieron en la imagen.

```
public boolean isEmpty() {
    return inicio == null;
}
//getters y setters de inicio y fin
@Override
public String toString() {
    String text="";
    if (!isEmpty()){
        Node<E> aux = inicio;
        while (aux!=null){
            text += aux.getData()+" ";
            aux = aux.getNextNode();
        }
        return text;
    }
}
```

Se creo la clase Test para implementar el insertionSort y generarPeorCaso. El metodo insertionSort se modifico haciendo que se verificara si es que la lista no estuviera vacia y en caso fuera asi recorria con un for en ves de un while hasta que el nodo siguiente fuera nulo. Dentro de este while habia otro que intercambiaba la data, en este caso Integer, de los nodos haciendo que los menores quedaran al inicio y los mayores al final. Retornaba el tiempo que se demoraba haciendo la operacion.

```
if (!lista.isEmpty()) {
    keyNode = lista.getInicio().getNextNode();
    while (keyNode != null) {
        Integer key = keyNode.getData();
        currentNode = keyNode.getPreviousNode();
        while (currentNode != null && currentNode.getData() > key) {
            currentNode.getNextNode().setData(currentNode.getData());
            currentNode = currentNode.getPreviousNode();
        }
        if (currentNode == null) lista.getInicio().setData(key);
        else currentNode.getNextNode().setData(key);
        keyNode = keyNode.getNextNode();
    }
}
```

Listing 1: Retorno de insertionSort

```
return nano_endTime - nano_startTime;
```

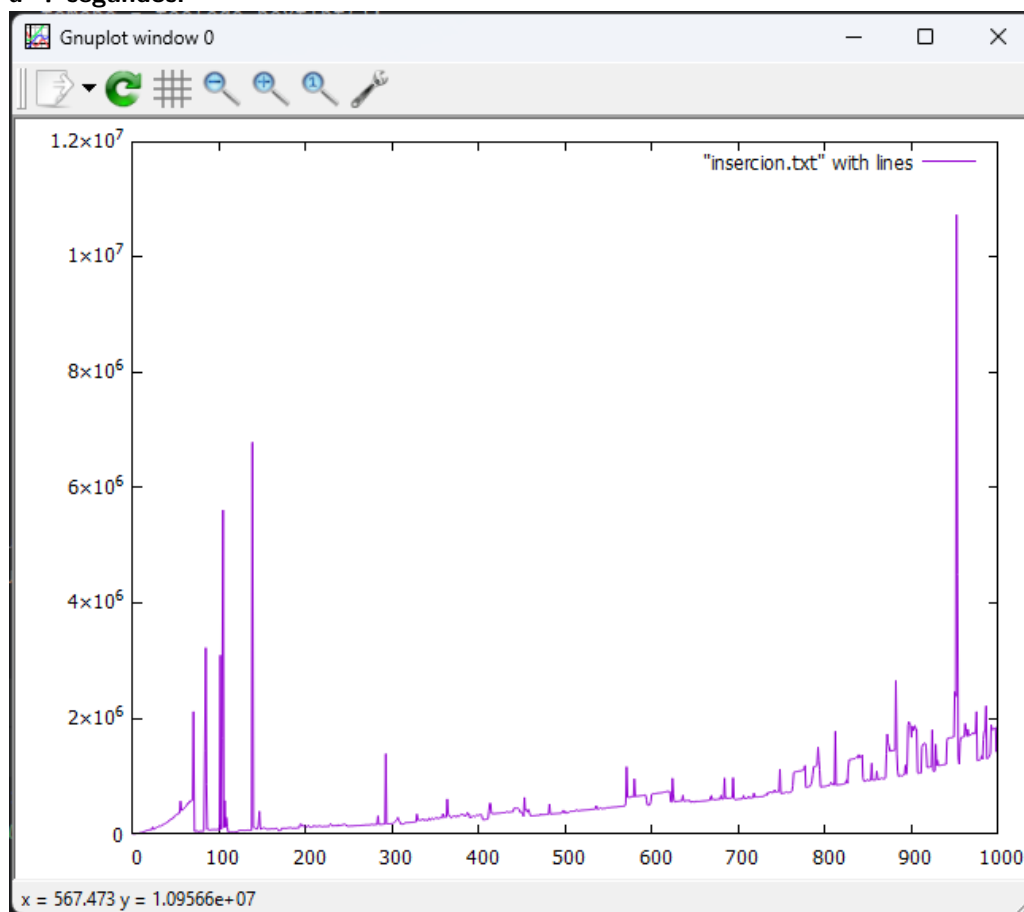
El metodo generarPeorCaso se modifico para que retornara una ListaDoble donde su primer elemento fuera el tamaño-1 y el ultimo 1, donde tamaño era el valor que se recibia por argumento, quedando asi el peor caso para hacer insertionSort.

```
ListaDoble<Integer> listaDoble = new ListaDoble<>();
for (int i = 0; i < tamaño; i++)
    listaDoble.addFinal(tamaño - i);
return listaDoble;
```

Por ultimo se descargo el proyecto java plot de <https://javaplot.yot.is/>. Donde se agrego ejercicio 2 como e2 importando JavaPlot con ayuda de [.] para saber coo usarla y agregarlo a mi ide respectivamente.

```
package e2;
import com.panayotis.gnuplot.JavaPlot;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;
```



Al ejecutarlo con 1000 nodos quedo el siguiente grafico, donde en el eje x es el numero de nodos y en el y el tiempo en nanosegundos que tomo ordenarlo. Nanosegundos = 10 elevado a -7 segundos.



### LECCIONES APRENDIDAS Y CONCLUSIONES

Se aprendió que es el método de ordenamiento por inserción, al igual que como implementarlo para Listas simples y dobles. Por otro lado se a crear los peores casos para este ordenamiento y a como guardar los resultados hasta n casos. Con esto se uso JavaPlot para graficar los resultados, como se muestran en el ejercicio 1 y 2.

### REFERENCIAS Y BIBLIOGRAFÍA

 <https://www.youtube.com/watch?v=35zTmB9HB6g>  
 <https://www.youtube.com/watch?v=7wDeHDASoSw>