



Informe de Laboratorio 04

Tema: Sort y Listas Enlasadas

Tabla 1: Mi tabla extendida

INTRODUCCIÓN

aqui ira la intro

MARCO CONCEPTUAL

aqui ira la MARCO CONCEPTUAL

SOLUCIONES Y PRUEBAS

Ejercicio 1 //taylor

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 viceverza.

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.

```
public void addInicio(E x){
    if(!isEmpty()){
        inicio=new Node<>(x,inicio,null);
        inicio.getNextNode().setPreviousNode(inicio);
    }else {
        inicio=fin= new Node<>(x);
    }
}

public void addFinal(E x){
    if(!isEmpty()){
        fin=new Node<>(x,null,fin);
        fin.getPreviousNode().setNextNode(fin);
    }else {
        inicio=fin= new Node<>(x);
    }
}
```

Asismimo se agrego el metodo isEmpty para saber si estaba vacia y se sobreescribio 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 tamanio-1 y el ultimo 1, donde tamanio 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 < tamanio; i++)
    listaDoble.addFinal(tamanio - i);
return listaDoble;</pre>
```

LECCIONES APRENDIDAS Y CONCLUSIONES aqui ira la LECCIONES APRENDIDAS Y CONCLUSIONES REFERENCIAS Y BIBLIOGRAFÍA

aqui ira la REFERENCIAS Y BIBLIOGRAFÍA

0.1. USAR COMO GUIA PARA EL INFORME

 Se tuvieron que implementar las funciones negative, join y under. En el codigo se detalla lo que hacen.

Listing 2: Picture version 1

```
import colors
   from colors import *
   class Picture:
       def __init__(self, img):
           self.img = img;
       def __eq__(self, other):
          return self.img == other.img
       def _invColor(self, color):
           if color not in inverter:
              return color
           return inverter[color]
       def negative(self):
           """ Devuelve un negativo de la imagen """
           neg = []
14
           for fila in self.img:
              cadena = ""
              for color in fila:
                  cadena += self._invColor(color)
18
              neg.append(cadena)
19
          return Picture(neg)
20
       def join(self, p):
21
           """ Devuelve una nueva figura poniendo la figura del argumento
              al lado derecho de la figura actual """
           juntos = []
           for i in range(len(self.img)):
              juntos.append(self.img[i] + p.img[i])
26
           return Picture(juntos)
27
       def under(self, p):
28
           """ Devuelve una nueva figura poniendo la figura p sobre la
                  figura actual """
           return Picture(self.img[::] + p.img[::])
```

Para implementar el negativo se cambia el color de cada carácter, a excepción del espacio ya que este no tiene inverso con doble for(1 anidado). Juntando estos caracteres por medio en una cadena y despues juntando esta cadena al nuevo arreglo con el metodo append.



Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Departamento Académico de Ingeniería de Sistemas e Informática Escuela Profesional de Ingeniería de Sistemas Estructura de Datos y Algoritmos



Listing 3: Ejercicio 2 a

```
from interpreter import draw
from chessPictures import *

draw(knight.join(knight.negative()).under(knight.negative().join(knight)))
```

Para poder hacer que se dibuje como se mostrara el la figura de abajo se tuvo de juntar un caballo blanco con uno negro en la primera fila; para ello se utilizo join y negative. Para la segunda fila se necesito saltar a la siguiente fila por lo que se utilizo el metodo under.

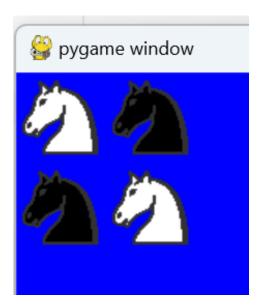


Figura 1: Ejecución exitosa ejercicio 2 a

0.2. Pregunta: Explique: ¿Para qué sirve el directorio pycache?

• Sirve para guardar los compilados de python, así como en java al ejecutarlos se crean los .class en python se crean los .pyc. Esto se hace automaticamente ya que se importan modulos de otras clases como se da en este caso.