	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Jesus Cruz Navarro
Asignatura:	Estructura de Datos y Algoritmos II
Grupo:	Grupo 1
No de Práctica(s):	Práctica 6 – Algoritmos de grafos. Parte 1
Integrante(s):	Edwin Jaret Santiago Díaz
No. de Equipo de cómputo empleado:	22
No. de Lista o Brigada:	22
Semestre:	2022 - 2
Fecha de entrega:	3 abril 2022
Observaciones:	

CALIFICACIÓN: _____

Algoritmos de grafos. Parte 1

Objetivos

1. Implementar un grafo usando el paradigma orientado a objetos.

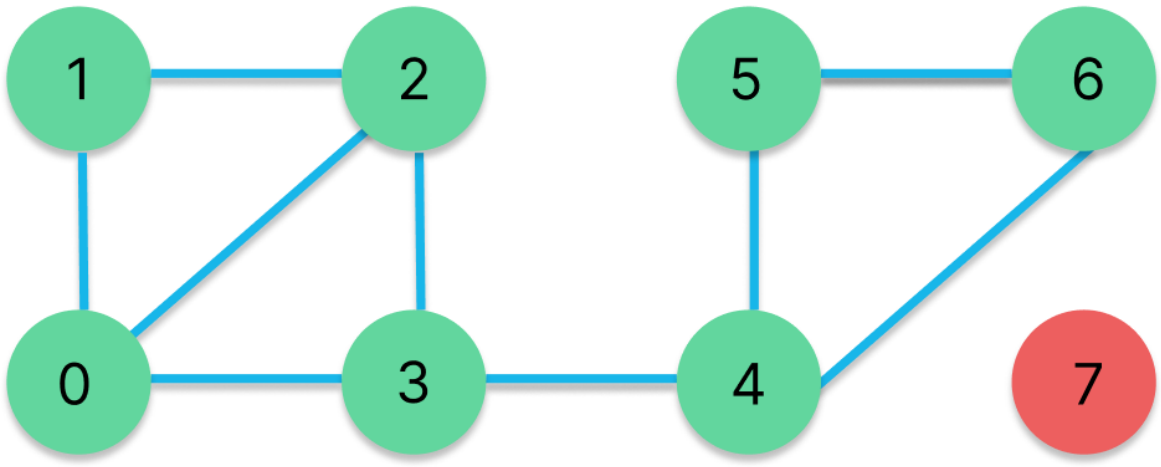
Desarrollo

En el programa "grafos.py" se implementó las clases **Grafo** para crear un grafo con múltiples nodos y **Nodo** para crear nodos. El contenido de las clases es:

- **Nodo**
  - Contiene un nombre y una **lista de vecinos** inicialmente vacía.
  - *AgregarVecino(nodo),* agrega un nodo vecino si y solo si no se ha agregado con anterioridad.

- **Grafo**
  - Contiene un **diccionario de nodos** que guarda el nombre del nodo y el nodo.
  - *AgregarNodo(nombreNodo),* recibe como parámetro el nombre del nodo, el método agrega un nodo con el nombre del parámetro recibido en donde se guarda en el **diccionario vértice**.
  - *AgregarArista(nombre_nodo1, nombre_nodo2),* recibe como parámetro el nombre de dos nodos, si estos nodos fueron agregados previamente a través del método *AgregarNodo(),* se crea una "conexión" (arista) entre estos dos nodos, esto es guardando en la lista de vecinos (que tiene cada nodo) el nodo contrario.
  - *Imprimir(),* imprime los nodos del grafo con sus respectiva lista de vecinos.

Para hacer uso de las clases, en la función *run()* se instancia un objeto de la clase grafo en la cual, se van a agregar 8 nodos, el nombre de los nodos es la numeración del 0 al 7, después se crean aristas entre los nodos 0 – 1, 0 – 2, 0 – 3, 1 – 2, 2 – 3, 3 – 4, 4 – 5, 4 – 6, 5 – 6, como se muestra en la imagen.



Para comprobar esto, se imprime el grafo y esto es lo que devuelve el programa :

Resultado:

```
Grafo:  0
 Vecinos:  1
 Vecinos:  2
 Vecinos:  3

Grafo:  1
 Vecinos:  0
 Vecinos:  2

Grafo:  2
 Vecinos:  0
 Vecinos:  1
 Vecinos:  3

Grafo:  3
 Vecinos:  0
 Vecinos:  2
 Vecinos:  4

Grafo:  4
 Vecinos:  3
 Vecinos:  5
 Vecinos:  6

Grafo:  5
 Vecinos:  4
 Vecinos:  6

Grafo:  6
 Vecinos:  4
 Vecinos:  5

Grafo:  7
```

El programa es a prueba de errores cuando:

- Un vecino ya existe con ese nombre, cuando sucede se le notifica al usuario. En este caso, ya existen los vecinos 0 (para el nodo 1) y 1 (para el nodo 0).

```
Agregar una arista que ya existe (0 - 1)
El nodo vecino 1 ya existe
Por lo tanto no se agrega

El nodo vecino 0 ya existe
Por lo tanto no se agrega
```

- Se repite el nombre de un nodo, cuando sucede se le notifica al usuario. En este caso, se repite el nombre del nodo 0.

```
El nodo 0 ya existe
Por lo tanto no se agrego el nodo
```

- Se desea agregar una arista y uno de los nodos no existe, cuando sucede se le notifica al usuario. En este caso, el nodo 0 si existe, pero el nodo 8 no.

```
No existe uno de los nodos, 0 o 8
Por lo tanto no se agrego la arista
```

Conclusiones

Para trabajar con grafos se necesita implementar una clase **grafo** y una clase **nodo** en donde la clase grafo hace uso de la clase nodo.

Un nodo contiene su información (como en el caso de la práctica se utilizó el nombre de los nodos) y un método para agregar a los nodos vecinos.

g

Un grafo contiene de información un diccionario de nodos en donde guarda el nombre del nodo y el objeto nodo y los métodos para agregar un nodo al grafo (requiere un nombre de un nodo existente) y agregar una arista al grafo (requiere el nombre de dos nodos existentes).

El objetivo de la práctica fue cumplido y la práctica se realizó en su totalidad resolviendo el ejercicio propuesto por el profesor con sus puntos a considerar.