UNIVERSIDAD AUTÓNOMA GABRIEL RENE MORENO
FACULTAD DE INGENIERÍA Y CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES

ESTRUCTURAS DE DATOS 2

CONTENIDO: **LAB-1. EDATOS PARA REPRESENTAR GRAFOS**

PORCENTAJE TERMINADO : 100%.

GRUPO:

Grupo	Nombre	Registro
12	Jorge Choque Calle	219074240
14	Cristian Gabriel Gedalge Cayhuara	219022062

Fecha de Presentación: Lunes ,20 de mayo de 2024

COMENTARIO:
Estuvo desafiante el tema de grafos algunos ejercicios estuvieron desafiantes

Class Nodo

```
public class Nodo {
    public String nombre;
    public int cantArcos;
    public Nodo prox;
    public Arco prim;
    public Arco ult;
    public Nodo(String nombre, Nodo prox) {
        this.nombre = nombre;
        this.prox = prox;
        this.prim = this.ult = null;
        this.cantArcos = 0;
    }

    public String toString() {
        String s = "";
        return s;
    }
    public boolean vacia() {
        return cantArcos == 0;
    }
    public void insertArco(int valor, Nodo desti, Arco proxi)
{
        if (vacia()) {
            prim = ult = new Arco(valor, desti, proxi);
        } else {
            ult.prox = ult = new Arco(valor, desti, proxi);
        }
        cantArcos++;
    }
}
```

Class Arco

```
public
class Arco
{
public int
paso;
public Arco
prox;

    public Nodo
dest;


    public Arco(int paso, Nodo dest,Arco prox) {


this.paso =
paso;
this.prox =
prox;


this.dest =
dest;

    }

    public Arco(Arco prox,
Nodo dest) {
this.prox = prox;


this.dest =
dest;

    }

    Public
String
toString(){
String
s="";
return s;

    }

}
```

Class Grafo

```
public class
Grafo {
public Nodo
prim;
public Nodo
ult;

    public int
cantNodos;


    public
Grafo(){
this.prim=this.
ult=null;


this.cantNodos=
0;

    }


    public boolean
vacia(){
return prim==null
&& ult==null;

    }

    public boolean
seEncuentra(String name){
Nodo p=prim;
while(p!=null){
if(p.nombre.equals(name))


return true;


p=p.prox;

        }
```

```
        return
false;
    }
    public Nodo
buscarNodo(String name){
Nodo p=prim;
while(p!=null){
if(p.nombre.equals(name)
)

return p;

p=p.prox;
        }
        return
null;
    }

//metodos de la clase
}
```

1. **G1.insertarNodo(name) :** Método que insertar un nodo en el grafo G1, con rótulo name. *Sugerencia, insertar al último de la Lista de Nodos*

```
    public void insertarNodo(String name){
if(seEncuentra(name))

return;
if(vacia())
            prim=ult=new
Nodo(name,null);         else
            ult.prox=ult=new
Nodo(name,null);         cantNodos++;
    }
```

2. **G1.insertarArco(name1, name2, valor):** Método que inserta un arco en el grafo G1, desde el nodo name1 hasta name2 con un peso del arco igual a valor. *Sugerencia, insertar al último de la Lista de Arcos que sales de name1.*

```
    public void insertarArco(String name1, String
name2,int valor){
        Nodo
origin=buscarNodo(name1);
Nodo
desti=buscarNodo(name2);
if(origin==null
||desti==null)
return;
origin.insertArco(valor,
desti, null);
    }
```

3. **G1.mostrar():** Método que muestra el grafo G. Muestra la lista de nodos. Para cada nodo, muestra la lista de arcos que salen de él, con sus nodos destinos y sus respectivos valores.

```
    public void
mostrar(){
Nodo p = prim;
while(p != null){

System.out.println(p.nombre+":
");            Arco
arco=p.prim;
while(arco!=null){
               System.out.println(" ("+arco.dest.nombre+",
"+arco.paso+") ");                arco=arco.prox;
            }

System.out.println();
p=p.prox;
        }
    }
```

4. **G1.cantidadArcos() :** Método que devuelve la cantidad de arcos que contiene el grafo G1.

```
public int
cantidadArcos(){
int cantArcos=0;
Nodo p=prim;
while(p!=null){

cantArcos+=p.cantArcos;
p=p.prox;
        }
        return cantArcos;
    }
```

5. **G1.cantidadLlegadas(name1) :** Método que devuelve la cantidad de arcos que llegan al nodo name1.

```
    public int
cantidadLlegadas(String name1){
int cantLlegadas=0;          Nodo
p=prim;          while(p != null){
Arco arco=p.prim;
while(arco!=null){
if(arco.dest.nombre.equals(name1))
cantLlegadas++;
arco=arco.prox;
            }
            p=p.prox;
        }
        return cantLlegadas;
    }
```

6. **G1.mostrarNodosBucle():** Método que muestra los nodos que tienen arcos así mismos.

```
public void
mostrarNodosBucle(){
Nodo p=prim;
while(p!=null){
Arco arco=p.prim;
while(arco!=null){
if(arco.dest==p){
                    System.out.println("Nodo con buble:
"+p.nombre+",");
                    //break;
                }
                arco=arco.prox;
            }
         p=p.prox;
        }
    }
```

7 **G1.mostrarNodosIslas() :** Método que muestra los nodos islas. Nodos islas, son aquellos nodos que no tienen arcos que salen de él, ni arcos que llegan a él.

```
    public void
mostrarNodosIslas(){
Nodo p=prim;
while(p!=null){
            if(cantidadLlegadas(p.nombre)==0 &&
cantidadSalidas(p.nombre)==0)
System.out.println("Nodos Isla :"+p.nombre);
p=p.prox;
        }
    }
```

8. **G1.mayorValor() :** Método que devuelve el mayor valor de los arcos del grafo G1.

```
public int
mayorValor(){
int mayor=0;
Nodo p=prim;
while(p!=null){
            Arco
arco=p.prim;
while(arco !=null){
if(arco.paso>mayor)
mayor=arco.paso;
arco=arco.prox;
            }
            p=p.prox;
        }
        return mayor;
    }
```