|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación salas A yB

# *Profesor:*Asignatura:

*Grupo:No dePráctica(s):Integrante(s):*

M.I Tista Garcia Edgar

Estructura de Datos y Algoritmos II

5

6

Calzada Martinez Jonathan Omar

# *No.deEquipode*cómputoempleado

*Semestre:Fechadeentrega:*

36

2019-2

12/03/2019

# *Obervaciones:*

CALIFICACIÓN:

**Objetivo:**El estudiante conocera las formas de representar un grafo e identificara las caracteristicas necesarias para entender el algoritmo de busqueda por expansionn.

## Actividades

**Ejercicio 1.**Existen diversas implementaciones de Gra os sue son adaptadas al lenguaje de programaciòn correspondiente, asì como a la orma de representaciònsue se utilizan

**Ejercicio 2.**BFS(Breadth First Search)

## Introduccion :

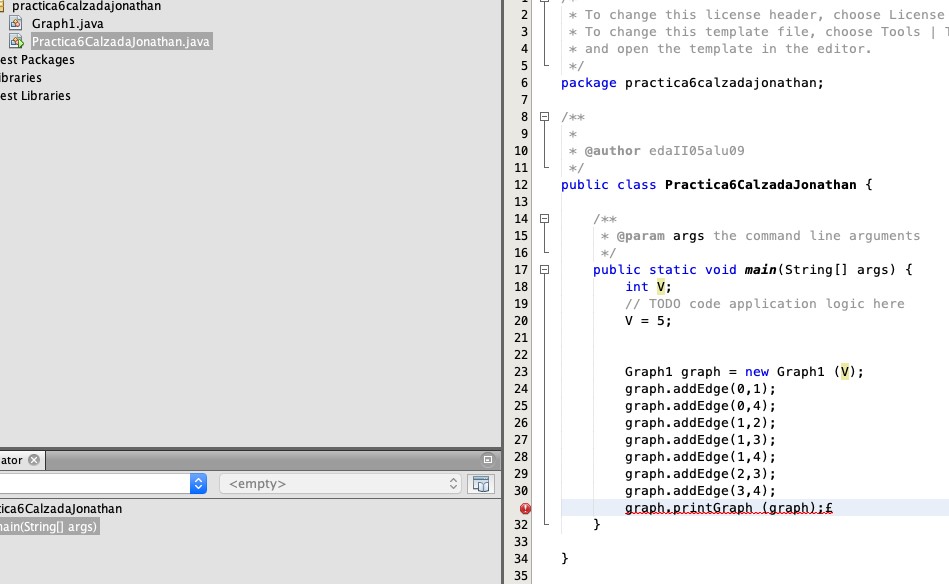
Un Grafo no es mas que un conjunto de nodos o vértices que se encuentran relacionados con unas aristas. Ademas los vértices tienen un valor y en ocasiones las aristas también y se le conoce como el costo.

### Grafos en Java

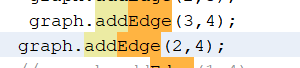
Desde un punto de vista intuitivo un grafo es un conjunto de nodos unidos por un conjunto de arcos. Un ejemplo de grafo que podemos encontrar en la vida real es el de un plano de trenes. El plano de trenes está compuesto por varias estaciones (nodos) y los recorridos entre las estaciones (arcos) constituyen las líneas del trazado.

### Desarrollo: Ejercicio 1)

Se realizò la codificaciòn de las clases el proyecto principal agregando la clase Grahp1 y el metodo en la clase Graph1.

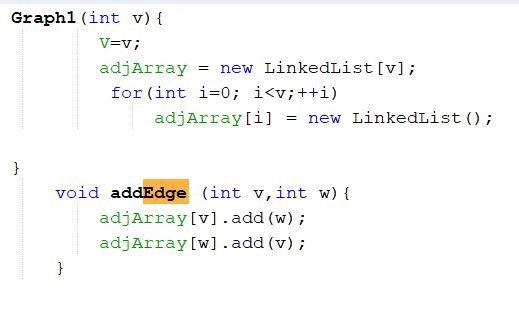


En estas lineas en la clase graph se llama al metodo addEdge y se le agrega el nodo y con cual va a estar conectado



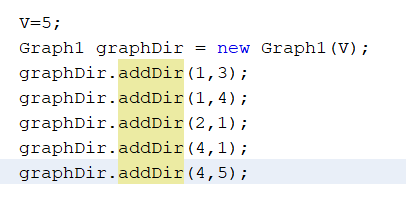
En la funcion Graph se crea la lista adjArray, y se crea un bucle para crear en cada nodo una lista.

En el metodo addEdge se utiliza para poder recibir del main el nodo y el nodo al que va estar conectado o unido.



Se crea un metodo para la impresion de la lista de adyacencia (grafo)

b) IMPLEMENTAR QUE FUNCIONE CON GRAFOS DIRIGIDOS

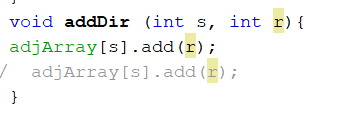


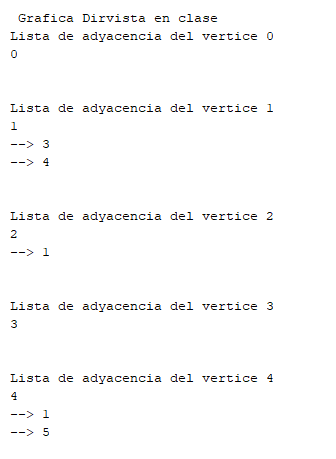
Como se ve en la imagen anterior primero se mando el nodo de origen y el nodo de destino.  
  
Para el nodo 1 se dirigio hacia el 3 y el 4

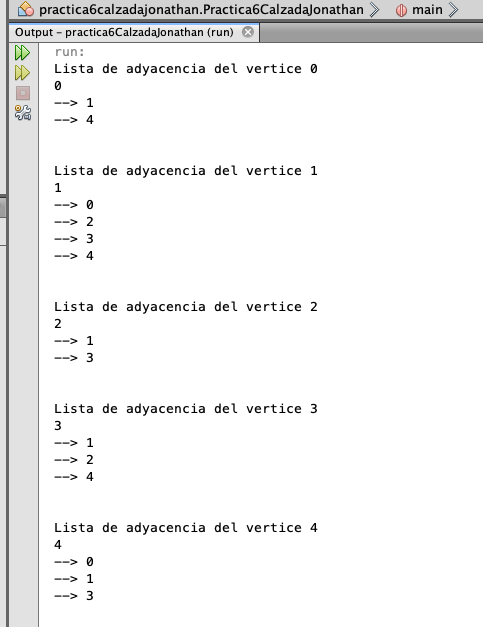
Para el nodo 2 se dirigio hacia el nodo 1

para el nodo 4 se dirigio hacia el nodo 1 y el 5

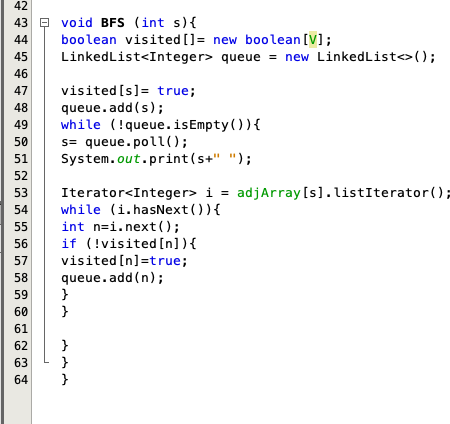
Para esto solo se creo un metodo que

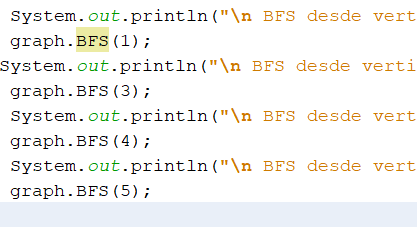




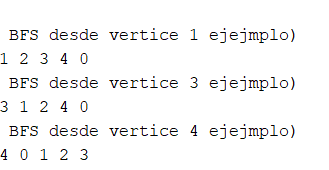
--------------El codigo compilado es el siguiente.

### Ejercicio 2

****

 **En la funcion principal se agregaron las instrucciones necearias para que se ejecute la función:**

**EL grafo va haciendo el recorrido a partir de el nodo dado.**

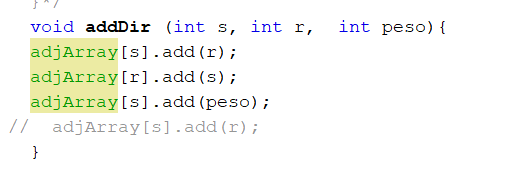
****

**El algoritmo de recorrido BFS trabajadapor bloques, realizando bloque por bloque hasta acabar con el grafo**

**El algoritmo es igual al visto en clase solo cambien pequeñas cosas como por ejemplo el return visitados.**

**ejercicio 3. Grafos ponderados**

**Se realizo la clase con el nombre graph2 y se intento realizar el algoritmo de grafos ponderados, no se logro el objetivo, en la imagen de abajo se ve que se intento mandar 3 datos al llmar al metodo agregar que son los dos vertices y el peso de la arista.**

****

**Conclusiones:**

Los grafos son una gran ayuda en la programacion ya que nos permiten realizar busquedas en algun mapa, por ejemplo las lineas del metro. Si queremos llegar a un lugar en donde nos ayuda un grafo nos ayudaria a encontrar la estacion màs cercana.

El objetivo de la practica no se cumplio del todo ya que el tercer ejercicio no se pudo concluir por una confusion que tuve al hacer el cambio el algoritmo ponderado, en esa parte me atore bastante y ya no pude completarla.