ESTRUCTURA DE DATOS 2
Código ST0247

Taller en Sala 11
Implementación de Grafos

Objetivo: Resolver problemas fundamentales de grafos, incluyendo la búsqueda DFS y BFS

Consideraciones: Lean y verifiquen las consideraciones de entrega,

Trabajo en Parejas

Mañana, plazo de entrega

Docente entrega plantilla de código en GitHub

Sí .cpp, .py o .java

No .zip, .txt, html o .doc

Alumnos entregan código sin comprimir GitHub

En la carpeta Github del curso, hay un código iniciado y un código de pruebas (tests) que pueden explorar para solucionar los ejercicios

Estructura del documento: **a)** Datos de *vida real*, **b)** I*ntroducción* a un problema, **c)** Problema a resolver, **d)** Ayudas. Identifiquen esos elementos así:

a) **b)** **c)** **d)**

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ejercicios a resolver

1. En videojuegos como *World of Warcraft*, los grafos se utilizan para calcular las rutas que toman los personajes.

- **Implementen grafos con la estructura de datos Matrices de Adyacencia Etiquetadas.**
- **Implementen grafos con la estructura de datos Listas de Adyacencia Etiquetadas.**

Utilicen el conjunto de datos *Medellin-Colombia-grande.zip* que se encuentran en la carpeta *datasets*, en Github, para probar su algoritmo.

2. **[Ejercicio Opcional]** Generen una representación usando *Graphviz* de los grafos anteriores.

3. **[Ejercicio opcional]** Implementen un algoritmo para decir si hay camino entre dos vértices de un grafo dirigido no necesariamente conexo.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ayudas para resolver los Ejercicios

Ejercicio 1 Matrices de Adyacencia…… **Pág. 4**

Ejercicio 1 Listas de Adyacencia…….. **Pág. 5**

Ejercicio 2…… **Pág.6**

Ejercicio 1_Matrices de Adyacencia

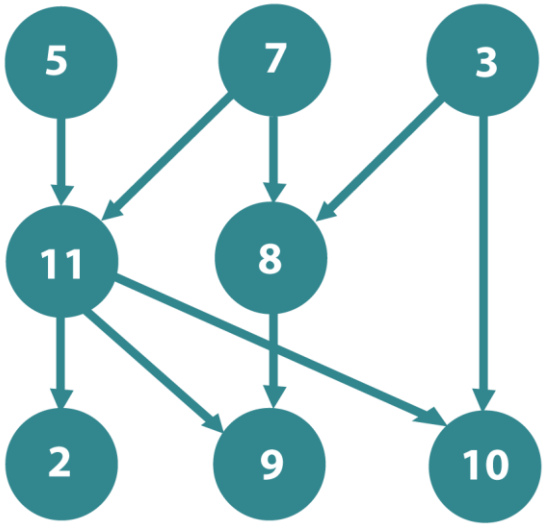
Pista 1: Vean en Guía de Laboratorios, numeral 4.10, *"Cómo hacer clases abstractas"*

Error Común 1: El método *getSuccessors* debe retornar los sucesores los identificadores de los vértices, no los pesos de los arcos.

Error Común 2: El método *getSucessors* retorna los vecinos de un nodo, es decir, los nodos adyacentes. No es un recorrido en profundidad.

Ejemplo 1, para el grafo de la imagen, esta debe ser la matriz:

Grafo



Matriz

	2	3	5	7	8	9	10	11
2	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	1	0
5	0	0	0	0	0	0	0	1
7	0	0	0	0	1	0	0	1
8	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	1	0	0	0	0	1	1	0

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ejercicio 1_Listas de Adyacencia

Pista 1: Vean en Guía de Laboratorios, numeral 4.8, *"Cómo definir una clase pareja en Java"*

Error Común 1: El método *getSuccessors* debe retornar los identificadores de los vértices, no los pesos de los arcos.

Error Común 2: El método *getSucessors* retorna los vecinos de un nodo, es decir, los nodos adyacentes. No es un recorrido en profundidad.

Error Común 3: Un error común es intentar acceder a una lista de listas con la instrucción `listaDeListas.get(source).get(destination)` porque el destino no se encuentra necesariamente en esa posición de la lista. No es una matriz.

Error Común 4: Una lista de listas de parejas se define en Java como `ArrayList<LinkedList<Pair<Integer,Integer>>> listaDeListas = new …`

Ejemplo 1, para el grafo de la imagen del punto 1, esta debe ser la representación con listas:

2 →
3 → 8,10
5 → 11
7 → 8, 11
8 → 9
9 →
10 →
11 → 2, 9, 11

La implementación de los puntos 3-6 se hacen en la clase "*Recorridos.java*".

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ejercicio 2

Pista 1: Para visualizar el grafo, utilicen **http://www.webgraphviz.com/**

Ejemplo 1, para grafo no dirigido:

```
graph {
  a -- b;
  a -- c;
  a -- e;
  b -- c;
  c -- d;
  c -- e;
}
```

Ejemplo 2, para grafo dirigido:

```
digraph G {

  "Welcome" -> "To" [label = "a"]
  "To" -> "Web"
  "To" -> "GraphViz!"

}
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

¿Alguna inquietud?

CONTACTO

Docente Mauricio Toro Bermúdez
Teléfono: (+57) (4) 261 95 00 Ext. 9473
Correo: mtorobe@eafit.edu.co
Oficina: 19- 627

Agenden una cita dando clic en la pestaña -*Semana*- de http://bit.ly/2gzVg10