

Taller en Sala 12 Recorridos Sobre 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) *Introducción* 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



En la vida real, los algoritmos de recorridos de grafos se utilizan para calcular las rutas en software como Google Maps así http://kevanahlquist.com/osm_pathfinding/

Ejercicios a resolver

- 1 En el videojuego *Age of Empires 1: Rise of Rome*, el objetivo del juego es conquistar otras civilizaciones. Para lograrlo, es necesario enviar ejército entre diferentes puntos del mapa.

El mapa se representa internamente como un grafo. Un problema que existe es la existencia de islas porque esto significa que el grafo no es conexo y, por consiguiente, no se puede llegar desde cualquier punto del grafo a cualquier otro punto.



- ▶ Usando el recorrido primero en profundidad (siglas en Inglés de DFS), escriban una implementación que retorne si hay camino o no entre un vértice i y un vértice j en un grafo g que no es necesariamente conexo.

- 2 ▶ Usando el recorrido primero en amplitud (siglas en inglés de BFS), escriban una implementación que retorne si hay camino o no entre un vértice i y un vértice j en un grafo g que no es 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

ESTRUCTURA DE DATOS 2
Código ST0247

- 3** ▶ **[Ejercicio opcional]** Implementen un algoritmo para imprimir los vértices de un grafo, a partir de un vértice utilizando DFS. Supongan que existe una clase `Graph` con métodos `getSuccessors(int i)` y `size()`. El método `g.getSuccessors` retorna los vecinos (o adyacentes) del vértice `i`.
- 4** ▶ **[Ejercicio opcional]** Implementen un algoritmo para imprimir los vértices de un grafo, a partir de un vértice utilizando BFS. Supongan que existe una clase `Graph` con métodos `getSuccessors(int i)` y `size()`. El método `g.getSuccessors` retorna los vecinos (o adyacentes) del vértice `i`.

Ayudas para resolver los Ejercicios

Ejercicio 1	<u>Pág. 5</u>
Ejercicio 2	<u>Pág. 5</u>
Ejercicio 3.....	<u>Pág.6</u>
Ejercicio 4.....	<u>Pág.8</u>



Ejercicio 1



Pista 1: Vean <https://bit.ly/2tFwk0Y>



Error Común 1: No utilizar la respuesta que entrega el llamado recursivo

```

boolean dfsAuxMalo(Graph g, int v, boolean[] vi, int
buscado) {
    vi[v] = true;
    if(v==buscado) return true;
    ArrayList<Integer> vecinos = g.getSuccessors(v);
    for (Integer vecino: vecinos)
        if (!vi[vecino]) // !(vi[vecino] == true)
            dfsAux(g, vecino, vi, buscado);
    return false;
}

```



Ejercicio 2



Pista 1: Vean <https://bit.ly/2tFwk0Y>



Ejemplo 1, Para el grafo de la imagen del punto 1, estas serían las respuestas:

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

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



Pista 2: Ingresen a <https://bit.ly/2IGHNBX>. Dada la representación lógica de varios grafos, infieran el orden en que se visitan los nodos usando búsqueda en amplitud (BFS). Revisen si su respuesta es correcta.



Error Común 1: Solo sabe usar BFS



Ejercicio 3

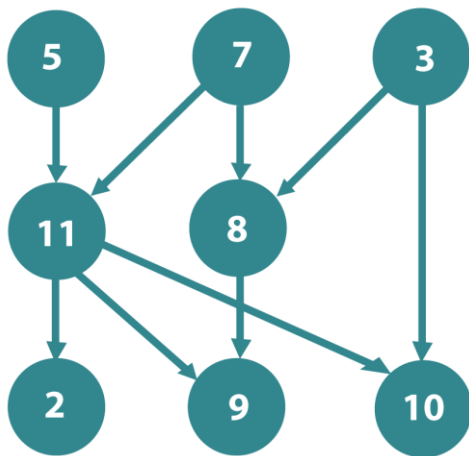


Ejemplo 1, para el grafo de la imagen, esta serían las respuestas:

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

ESTRUCTURA DE DATOS 2
Código ST0247



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



Pista 1: Vean <https://bit.ly/2z3PJxo>



Pista 2: Definan los métodos de esta forma:

```
public void dfs(Graph g, int v) {
    boolean[] visitados = new boolean[g.size()];
    dfsAux(Graph g, int v, visitados);
}

public void dfsAux(Graph g, int v, boolean[] vi){
    ...
}
```



Pista 3: Ingresen a <https://bit.ly/2tlKdvf>. Dada la representación lógica de varios grafos, infieran el orden en que se visitan los nodos usando búsqueda en profundidad (DFS). Revisen si su respuesta es correcta.

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



Error Común 1: No utilizar el arreglo de visitados

```
public void dfsMalo(Graph g, int v) {
    dfsAux(Graph g, int v, visitados);
}
```



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

```
public void dfsAuxMalo(Graph g, int v, boolean[] vi){
    vi[v] = true;
    System.out.println(v);
    ArrayList<Integer> vecinos = g.successors(v);
    for (Integer vecino: vecinos) {
        dfsAux(g, vecino, vi);
    }
}
```



Ejercicio 4



Pista 1: Vean <https://bit.ly/2KuXTn8>

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>