ESTRUCTURA DE DATOS 2
Código ST0247

Taller en Sala 3
Vuelta Atrás (*Backtracking*)

Objetivos: 1. Usar la técnica de vuelta atrás (*backtracking)* para resolver un problema, como, por ejemplo, salir de un laberinto. **2.** Resolver problemas usando algoritmos de grafos, incluido el problema del camino más corto.

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

En la vida real, las n reinas han servido para diseñar esquemas de almacenamiento de memoria en computación paralela, control de tráfico aéreo, prevención de *deadlocks*, procesamiento de imágenes, entre otros. Tomado de http://bit.ly/2fYhXxs.

Ejercicios a resolver

1. En 1996, el computador *Deep Blue*, desarrollado por IBM, venció al campeón mundial de ajedrez Garry Kasparov.

Posteriormente, en 2016, el programa *AlphaGo*, desarrollado por *Google*, venció a uno de los mejores jugadores de *Go* en el mundo.

Programas que sean capaz de jugar ajedrez o *Go*, son de mucho interés para las grandes multinacionales porque dan los cimientos para sistemas de computación cognitiva como *Watson* de *IBM*. Un primer paso para construir sistemas que jueguen ajedrez, es resolver el acertijo de las *n reinas*

Implementen un algoritmo que resuelva el problema de las *n reinas*, usando *backtracking,* que sea capaz de retornar todas las posibles soluciones.

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 Opcional] En el videojuego *Age of Empires 1: Rise of Rome*, el objetivo es conquistar otras civilizaciones. Para lograrlo, es necesario enviar ejercito 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 un camino entre un vértice *i* y un vértice *j* en un grafo *g* que no es necesariamente conexo. Si no hay camino entre los vértices *i* y *j*, retorne una lista vacía.

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

Ejemplo 1: Estas son las 2 soluciones para el problema de las n reinas con n = 4

#	#	Q	#
Q	#	#	#
#	#	#	Q
#	Q	#	#

#	Q	#	#
#	#	#	Q
Q	#	#	#
#	#	Q	#

Pista 1: Es el mismo problema del taller anterior, sólo que esta vez deben hacerlo con *backtracking* en vez de fuerza bruta, lo cual es más eficiente

Guía para la Implementación

1. Escriban un método para verificar si puede poner una reina en determinada posición. Recuerden que debido a la forma como representamos los tableros es imposible tener dos reinas en una misma fila, por lo que no necesitan verificar esta condición. El parámetro *r* hace referencia a la fila, y el *c* a la columna.

```
private static boolean puedoPonerReina(int r, int c, int[] tablero) {
          // complete...
}
```

2. Implementen el método de *backtracking*. El parámetro *r* representa la fila, el parámetro *n* el número de reinas. Esta función retorna un entero que representa el número de soluciones que existen para las n-reinas. Por ejemplo, para n=4, retorna 2 y para n=8 retorna 92.

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

```
private static int nReinas(int r, int n, int[] tablero) {
// complete...
}
```

Pista: Se itera fila por fila (hay *n* filas) y se van poniendo reinas si es válido hacerlo.

3. Llamen el método creado en el paso **2** desde el *wrapper*.

```
public static int nReinas(int n) {
     // complete...
}
```

Recuerden que tienen a su disposición el método `imprimirTablero(int[] tablero)` para visualizar los tableros.

Ejercicio 2

Pista 1: Consideren el siguiente código:

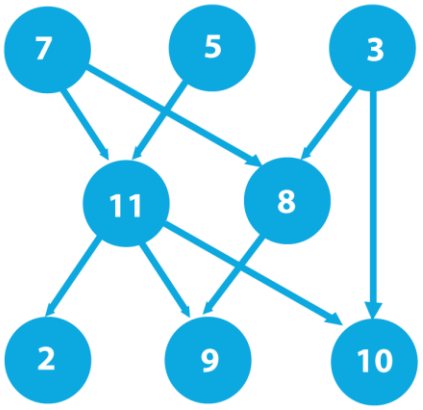
```
public static ArrayList<Integer> camino(Digraph g, int inicio, int
fin) {
     // complete...
}
```

Pista 2: Utilicen DFS (Depth-first search o Búsqueda en profundidad).

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

```
private static boolean dfs(Digraph g, int nodo, int objetivo,
boolean[] visitados, ArrayList<Integer> list) {
     // complete...
}
```

Ejemplo 1: consideren el siguiente grafo:



Algunos de los caminos que hay en este son:

- **7→9:** 7, 11, 9 ó 7, 8, 9 (puede retornar cualquiera)
- **7→10:** 7, 11, 10
- **3→9:** 3, 8, 9
- **3→10:** 3, 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

¿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