Taller en Sala Nro. 3
Backtracking

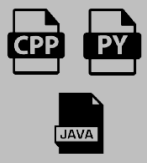
En la vida real, las n reinas han servido para diseñar esquemas de almacenamiento de memoria en computación paralela, control de tráfico aéreo, prevención de *deadlocks*, procesamiento de imágenes, entre otros.

Tomado de http://bit.ly/2fYhXxs

Trabajo en Parejas

Hoy, plazo máximo de entrega

Docente entrega código suelto en GitHub



Sí .cpp, .py o .java

No .zip, .txt, html o .doc

Alumnos entregan código suelto por GitHub

Ejercicio a resolver

1. Escriban una implementación o algoritmo que resuelve el problema de las n reinas usando *backtracking* que retorne todas las posibles soluciones.

Por ejemplo, estas son las 2 soluciones para el problema de las n reinas con n = 4

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

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

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

2. Hallen un camino (cualquiera) entre dos vértices de un grafo dado **utilizando backtracking.** Si no hay camino entre los vértices *inicio* y *fin*, retorne *una lista vacía*.

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

Ayudas para resolver el Ejercicio

Ayudas para resolver el Ejercicio 1

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 como tal.

```
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...
}
```

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

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

Ayudas para resolver el Ejercicio 2

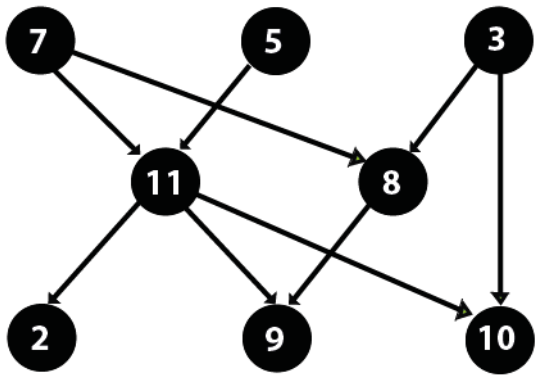
Pista 1: Consideren el siguiente código:

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

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

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

Como un ejemplo, consideren el siguiente grafo



DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

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

¿Alguna inquietud?

CONTACTO

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

Agende una cita con él a través de **http://bit.ly/2gzVg10** , en la pestaña *Semana. Si no da clic en esta pestaña, parecerá que toda la agenda estará ocupada.*