

## Laboratorio Nro. 3

### Vuelta atrás (*Backtracking*)

## Objetivos

Diseñar algoritmos usando la técnica de diseño de vuelta atrás

## Consideraciones iniciales

### Leer la Guía



Antes de comenzar a resolver el presente laboratorio, leer la **“Guía Metodológica para la realización y entrega de laboratorios de Estructura de Datos y Algoritmos”** que les orientará sobre los requisitos de entrega para este y todos los laboratorios, las rúbricas de calificación, el desarrollo de procedimientos, entre otros aspectos importantes.

### Registrar Reclamos



En caso de tener **algún comentario** sobre la nota recibida en este u otro laboratorio, pueden **enviarlo** a través de <http://bit.ly/2g4TTKf>, el cual será atendido en la menor brevedad posible.

### Traducción de Ejercicios



En el GitHub del docente, encontrarán la traducción al español de los enunciados de los Ejercicios en Línea.

**Visualización de  
Calificaciones**

A través de **Eafit Interactiva** encontrarán un **enlace** que les permitirá **ver un registro de las calificaciones** que **emite el docente** para cada taller de laboratorio y según las rubricas expuestas. **Véase sección 3, numeral 3.8.**

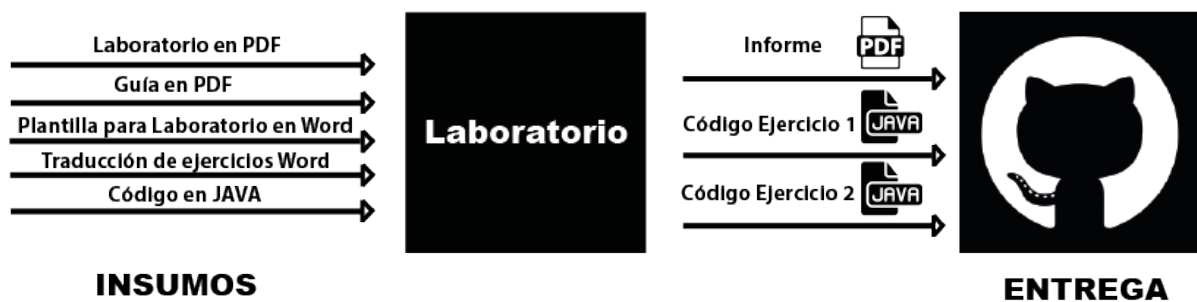
**GitHub**

1. Crear un repositorio en su cuenta de GitHub con el nombre `st0247-suCodigoAqui`. 2. Crear una carpeta dentro de ese repositorio con el nombre `laboratorios`. 3. Dentro de la carpeta `laboratorio`, crear una carpeta con nombre `lab03`. 4. Dentro de la carpeta `lab03`, crear tres carpetas: `informe`, `codigo` y `ejercicioEnLinea`. 5. Subir el informe pdf a la carpeta `infome`, el código del ejercicio 1 a la carpeta `codigo` y el código del ejercicio en línea a la carpeta `ejercicioEnLinea`. Así:

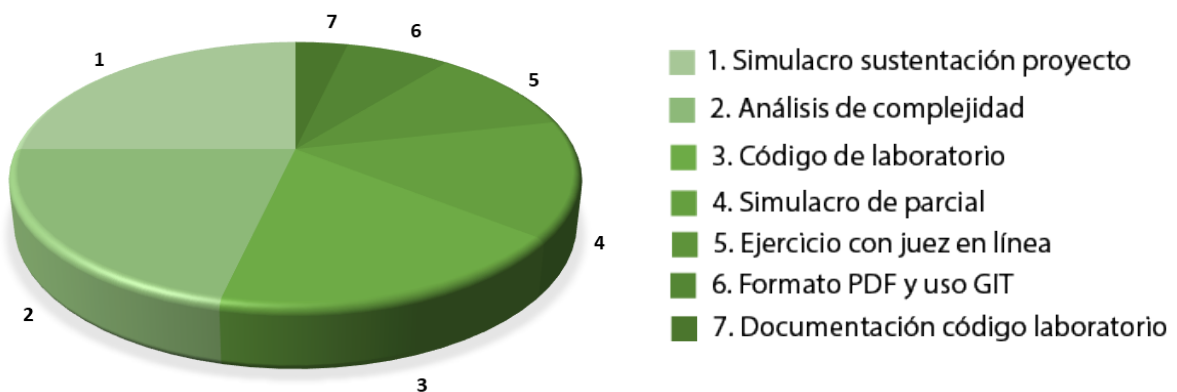
```
st0247-suCodigoAqui
  laboratorios
    lab01
      informe
      codigo
      ejercicioEnLinea
    lab02
      ...
```

## Intercambio de archivos

Los archivos que **ustedes deben entregar** al docente son: **un archivo PDF** con el informe de laboratorio usando la plantilla definida, y **dos códigos**, uno con la solución al numeral 1 y otro al numeral 2 del presente. Todo lo anterior se entrega en **GitHub**.



## Porcentajes y criterios de evaluación para el laboratorio



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

## Resolver Ejercicios

### 1. Códigos para entregar en GitHub:



En la vida real, la documentación del software hace parte de muchos estándares de calidad como CMMI e ISO/IEC 9126



Véase Guía *en Sección 3, numeral 3.4*



Código de laboratorio en *GitHub*. Véase Guía en *Sección 4, numeral 4.24*



Documentación en *HTML*



*No se reciben* archivos en *.RAR* ni en *.ZIP*

**1.1** Implemente el algoritmo de *backtracking* para encontrar UNA solución de las N Reinas.



**NOTA:** Si el algoritmo entrega TODAS las soluciones, quedó malo

**1.2** Construya ejemplos usando JUnit para probar su implementación de las N Reinas usando *backtracking*. Como muestra, use los ejemplos que ya conoce para el problema de las 4 reinas



**NOTA:** Si utilizan Python o C++, utilice una librería equivalente para pruebas unitarias en dichos lenguajes.



**PISTA 1:** Véase Guía, **Sección 4, numeral 4.14** “Cómo hacer pruebas unitarias en BlueJ usando JUnit”



**PISTA 2:** Usen el método *AssertArrayEquals* de *JUnit* que encuentra en <http://junit.sourceforge.net/javadoc/org/junit/Assert.html>



En la vida real, el camino más corto entre dos puntos en un grafo se aplica en sistemas de información de geográfica como *Google Maps* y en enrutadores de red como el *Cisco ISR 4000*

**1.3 Implementen un programa para resolver el problema de calcular el camino más corto entre dos puntos de un grafo usando *backtracking*, es decir, usando *Deep First Search (DFS)***



**PISTA 1:** Solución en pseudocódigo

```
llenar el arreglo de distancias con infinito
marcar la distancia al nodo inicial como 0
llamar dfs con el nodo raiz
dfs {
    Para (cada hijo) {
        si (puedo mejorar distancias hasta este) {
            marque nueva distancia (mejorada)
```

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

```
    llame recursivamente para el hijo  
    }  
}  
}
```



**PISTA 1:** Vean **Problema** y **Solución**



**NOTA:** Esta técnica no es la más eficiente para resolver este problema, pero es la que usaremos en este ejercicio

#### 1.4 Prueben el ejercicio 1.3 con un grafo completo

**1.5** Escriban una explicación entre 3 y 6 líneas de texto del código del ejercicio en línea del numeral 1.3 Digan cómo funciona, cómo está implementado y destaquen las estructuras de datos y algoritmos usados



**NOTA:** Todos los ejercicios del numeral 1 deben ser documentados en formato HTML. Véase *Guía en Sección 4, numeral 4.1 “Cómo escribir la documentación HTML de un código usando JavaDoc”*


## 2) Ejercicios en línea sin documentación HTML en GitHub:







Véase Guía en **Sección 3, numeral 3.3**



**No entregar**  
documentación **HTML**

	UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS	Cód. ST0247
		Estructuras de Datos 2

	Entregar un archivo en <b>.JAVA</b>		<b>No se reciben</b> archivos en <b>.PDF</b>
	<b>Resolver</b> los problemas de <b>CodingBat</b> usando <b>Recursión</b>		Código del ejercicio en línea en <b>GitHub</b> . Véase Guía en <b>Sección 4, numeral 4.24</b>

## 2.1 Resuelvan el siguiente problema usando *backtracking* y *SIN* usar el algoritmo de Dijkstra ni otros algoritmos voraces



**PISTA 1:** Construya un grafo. Utilice el recorrido DFS.



**PISTA 2:** Retorne una pareja que contiene el camino y el peso total



**NOTA:** Esta técnica no es la más eficiente para resolver este problema, pero es la que usaremos en este ejercicio

A usted le entregan un grafo no dirigido con pesos. Los vértices están enumerados del 1 al  $n$ . Su tarea es encontrar la ruta más corta entre el vértice 1 y el vértice  $n$ .

### Entrada

La primera línea contiene 2 enteros  $n$  y  $m$  ( $2 \leq n \leq 105$ ,  $0 \leq m \leq 105$ ), donde  $n$  es el número de vértices y  $m$  es el número de arcos. Después hay  $m$  líneas, donde cada uno contiene un arco de la forma  $a_i, b_i$  and  $w_i$  ( $1 \leq a_i, b_i \leq n$ ,  $1 \leq w_i \leq 106$ ), donde  $a_i, b_i$  son los vértices del arco y  $w_i$  es el peso del arco.

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

Es posible que en el grafo haya ciclos y que haya varios vértices entre el mismo par de vértices.

### Salida

Escribir -1 en caso de que no haya camino. Escribir el camino más corto de lo contrario. Si hay varias soluciones, imprimir cualquiera de ellas.

### Ejemplos

#### Entrada

5 6  
1 2 2  
2 5 5  
2 3 4  
1 4 1  
4 3 3  
3 5 1

#### Salida

1 4 3 5


#### Entrada

5 6  
1 2 2  
2 5 5  
2 3 4  
1 4 1  
4 3 3  
3 5 1

#### Salida





1 4 3 5



	UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS	Cód. ST0247
		Estructuras de Datos 2

2.2 [Ejercicio Opcional]: Resolver el siguiente problema <http://bit.ly/2k8CGSG>

### 3. Simulacro de preguntas de sustentación de Proyectos

	Véase Guía en <b>Sección 3, Numeral 3.5</b>		Entregar informe de laboratorio en <b>PDF</b>
	Usen la <b>plantilla</b> para responder laboratorios		<b>No apliquen Normas Icontec</b> para esto

3.1 Para resolver el problema del camino más corto en un grafo, fuera de fuerza bruta y backtracking, ¿qué otras técnicas computacionales existen?



**PISTA:** Lean <http://bit.ly/2hPomyn>



En la vida real, grandes compañías como Google, valoran más los conocimientos en complejidad computacional que un título de X o Y universidad

Tomado de <http://bit.ly/2hQAZHP>

3.2 Teniendo en cuenta lo anterior, tomen los tiempos de ejecución del programa realizado en el numeral 1.1 y en el laboratorio anterior con la solución de fuerza bruta de las n reinas. Completen la siguiente tabla.

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

Si se demora más de 5 minutos, coloque “se demora más de 5 minutos”, no siga esperando, podría tomar siglos en dar la respuesta, literalmente.

Valor de N	Fuerza bruta	Vuelta Atrás (Backtracking)
4		
8		
16		
32		
N	$O(?)$	$O(?)$



**PISTA:** Véase *Guía en Sección 4, numeral 4.6 “Cómo usar la escala logarítmica en Microsoft Excel 2013”*

**3.3** Expliquen con sus propias palabras la estructura de datos que utiliza para resolver el problema del numeral 2.1 y 2.2 [Ejercicio Opcional] y digan cómo funciona el programa.



**NOTA:** Recuerden que debe explicar su implementación en el informe PDF

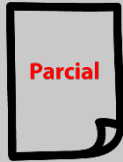
**3.4** Calculen la complejidad de los ejercicios en línea del numeral 2.1 y 2.2 [Ejercicio Opcional] y agréguela al informe PDF



**PISTA:** Véase *Guía en Sección 4, numeral 4.11 “Cómo escribir la complejidad de un ejercicio en línea”*

**3.5** Expliquen con sus palabras las variables (qué es ‘n’, qué es ‘m’, etc.) del cálculo de complejidad del numeral 3.4

#### 4) Simulacro de Parcial en el informe PDF



Para este simulacro, agreguen ***sus respuestas*** en el informe PDF.



***El día del Parcial no tendrán computador, JAVA o acceso a internet..***



**PISTA 1:** Véase ***Guía en Sección 4, Numeral 4.18*** “Respuestas del Quiz”



**PISTA 2:** Lean las diapositivas tituladas “***Data Structures II: Backtracking***”, encontrarán la mayoría de las respuestas

**1.** Wilkenson y Sofronio están aquí de nuevo. En esta vez han traído un juego muy interesante, en el cual Sofronio, en primer lugar, escoge un numero  $n$  ( $1 \leq n \leq 20$ ) y, en segundo lugar, escoge tres números  $a, b$  y  $c$  ( $1 \leq a \leq 9, 1 \leq b \leq 9, 1 \leq c \leq 9$ ).

Después, Sofronio le entrega estos números a Wilkenson y Wilkenson le tiene que decir a Sofronio **la cantidad máxima de números, usando  $a, b$  y  $c$  (se puede tomar un número más de una vez), que al sumarlos den el valor  $n$ .**

Como un ejemplo, si Sofronio escoge  $n=14$  y  $a=3, b=2, c=7$ . ¿Qué posibilidades hay de sumar 14 con  $a, b$  y  $c$ ?

$7+7=14$	cantidad es 2
$7+3+2+2=14$	cantidad es 4
$3+3+3+3+2=14$	cantidad es 5
...	
$2+2+2+2+2+2+2=14$	cantidad es 7

La cantidad máxima de números es 7. Esta sería la respuesta que da Wilkenson a Sofronio.

Como Wilkenson es muy astuto, ha diseñado un algoritmo para determinar la cantidad máxima de números y quiere que le ayudes a terminar su código. Asuma que hay al menos una forma de sumar  $n$  usando los números  $a$ ,  $b$  y  $c$  en diferentes cantidades, incluso si algunos de los números se suman 0 veces como sucede en el ejemplo anterior.

```
1 int solucionar (int n, int a, int b, int c)
2   if (n == 0 )
3       return 0;
4   int res = solucionar(_____) + 1;
5   res = Math.max(_____, _____);
6   res = Math.max(_____, _____);
7   return res;
```

a) Complete el espacio de la línea 04 (10 %)

b) Complete los espacios de la línea 05 (10 %)

c) Complete los espacios de la línea 06 (10 %)

**2. Un camino hamiltoniano** en un grafo dirigido es un camino que visita cada vértice exactamente una vez. Un **ciclo hamiltoniano** es un camino hamiltoniano para el cual existe un arco (en el grafo) que conecta el último vértice del camino hamiltoniano con el primer vértice del camino hamiltoniano.

Su tarea **es determinar si dado un grafo, este grafo contiene un ciclo hamiltoniano o no. Si lo contiene, retorne verdadero; de lo contrario, retorne falso.**

Parte de su tarea ya está hecha. La función `sePuede` verifica si un vértice  $v$  se puede agregar al ciclo hamiltoniano que está almacenado en el arreglo `path` en la posición `pos`, dado un grafo representado con matrices de adyacencia `graph`.

Por simplicidad, sólo se busca si existe un camino que empieza y termina en el primer vértice (es decir, en el vértice 0).

Por esta razón, en el arreglo `path` se entrega con todas sus posiciones en `-1`, excepto la posición `0`, como se muestra en la función `cicloHamil`. También, por esta razón, en el ciclo de la línea 08, `v` inicia se con `1`.

```
boolean cicloHamil(int graph[][]) {
    path = new int[g.length];
    for (int i = 0; i < g.length; i++)
        path[i] = -1;
    path[0] = 0;
    return cicloHamilAux(graph, path, 1);
}

boolean sePuede(int v, int graph[][],
                int path[], int pos) {
    if (graph[path[pos - 1]][v] == 0)
        return false;
    for (int i = 0; i < pos; i++)
        if (path[i] == v)
            return false;
    return true;
}

01 boolean cicloHamilAux(int graph[][],
                        int path[], int pos) {
02     if (pos == _____) {
03         if (graph[path[pos-1]][path[0]] == 1)
04             return true;
05         else
06             return false;
07     }
08     for (int v = 1; v < graph.length; v++) {
09         if (sePuede(_____,_____,_____,_____)) {
10             path[pos] = v;
11             if (cicloHamilAux(_____,_____,_____))
12                 return true;
13             path[pos] = -1;
14         }
15     }
16     return false;
17 }
```

- a) Complete el espacio en línea 02 que corresponde a la condición de parada (10%)  
\_\_\_\_\_
- b) Complete los espacios en línea 09 que corresponden al llamado de la función *sePuede* (10%)  
\_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_
- c) Complete los espacios en la línea 11 que corresponden al llamado recursivo de la función *cicloHamil* (10%)  
\_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_

## 5. [Ejercicio Opcional] Lectura recomendada



"Quienes se preparan para el ejercicio de una profesión requieren la adquisición de competencias que necesariamente se sustentan en procesos comunicativos. Así cuando se entrevista a un ingeniero recién egresado para un empleo, una buena parte de sus posibilidades radica en su capacidad de comunicación; pero se ha observado que esta es una de sus principales debilidades..."

Tomado de <http://bit.ly/2gJKzJD>



Véase Guía en **Sección 3, numeral 3.6 y 4.20** de la Guía Metodológica, "Lectura recomendada" y "Ejemplo para realización de actividades de las Lecturas Recomendadas", respectivamente

Posterior a la lectura del texto “*R.C.T Lee et al., Introducción al análisis y diseño de Algoritmos. Capítulo 5. Páginas 157 – 181.*”, realicen las siguientes actividades que les permitirán sumar puntos adicionales:

- a) Escriban un resumen de la lectura que tenga una longitud de 100 a 150 palabras



**PISTA 1:** En el siguiente enlace, unos consejos de cómo hacer un buen resumen <http://bit.ly/2knU3Pv>



**PISTA 2:** [Aquí](#) le explican cómo contar el número de palabras en Microsoft Word

- b) Hagan un mapa conceptual que destaque los principales elementos teóricos.



**PISTA:** Para que hagan el mapa conceptual se recomiendan herramientas como las que encuentran en <https://cacoo.com/> o <https://www.mindmup.com/#m:new-a-1437527273469>



**NOTA 1:** Si desean una lectura adicional en español, consideren la siguiente “*John Hopcroft et al., Estructuras de Datos y Algoritmos, Sección 10.4. 1983*”, que encuentran en biblioteca



**NOTA 2:** Estas respuestas también deben incluirlas en el informe PDF

## 6. [Ejercicio Opcional] Trabajo en Equipo y Progreso Gradual



El trabajo en equipo es una exigencia actual del mercado. "Mientras algunos medios retratan la programación como un trabajo solitario, la realidad es que requiere de mucha comunicación y trabajo con otros. Si trabajas para una compañía, serás parte de un equipo de desarrollo y esperarán que te comuniques y trabajes bien con otras personas"

Tomado de <http://bit.ly/1B6hUDp>



Véase Guía en **Sección 3, numeral 3.7** y **Sección 4, numerales 4.21, 4.22 y 4.23** de la Guía Metodológica

- a) Entreguen copia de todas las actas de reunión usando el tablero Kanban, con fecha, hora e integrantes que participaron



**PISTA:** Véase **Guía en Sección 4, Numeral 4.21** "Ejemplo de cómo hacer actas de trabajo en equipo usando Tablero Kanban"

- b) Entreguen el reporte de *git*, *svn* o *mercurial* con los cambios en el código y quién hizo cada cambio, con fecha, hora e integrantes que participaron



**PISTA:** Véase Guía en Sección 4, Numeral 4.23 "Cómo generar el historial de cambios en el código de un repositorio que está en svn"

- c) Entreguen el reporte de cambios del informe de laboratorio que se genera *Google docs* o herramientas similares





**PISTA:** Véase Guía en Sección 4, Numeral 4.22 “*Cómo ver el historial de revisión de un archivo en Google Docs*”



**NOTA:** Estas respuestas también deben incluirlas en el informe PDF

## Resumen de ejercicios a resolver

**1.1** Implementen el algoritmo de *backtracking* para encontrar UNA solución de las N Reinas.

**1.2** Construyan ejemplos usando JUnit para probar su implementación de las N Reinas usando *backtracking*.

**1.3** Implementen un programa para resolver el problema de calcular el camino más corto entre dos puntos de un grafo usando *backtracking*, es decir, usando *Deep First Search (DFS)*

**1.4** Prueben el ejercicio 1.3 con un grafo completo

**1.5** Escriban una explicación entre 3 y 6 líneas de texto del código del ejercicio en línea del numeral 1.3 Digan cómo funciona, cómo está implementado y destaquen las estructuras de datos y algoritmos usados


**2.1** Resuelvan el siguiente problema usando *backtracking* y SIN usar el algoritmo de *Dijkstra* ni otros algoritmos voraces: <http://bit.ly/2k8rMN2>

**2.2** Resuelvan el siguiente problema <http://bit.ly/2k8CGSG> [Ejercicio Opcional]

**3.1** Para resolver el problema del camino más corto en un grafo, fuera de fuerza bruta y *backtracking*, ¿qué otras técnicas computacionales existen?

**3.2** Tomen los tiempos de ejecución del programa realizado en el numeral 1.1 y en el laboratorio anterior con la solución de fuerza bruta de las n reinas. Completen la siguiente tabla.

**3.3** Expliquen con sus propias palabras la estructura de datos que utiliza para resolver el problema del numeral 2.1 y 2.2 [Ejercicio Opcional] y digan cómo funciona el programa.

	<p style="text-align: center;">UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS</p>	<p style="text-align: center;">Cód. ST0247</p> <hr/> <p style="text-align: center;">Estructuras de Datos 2</p>
---	---	--

**3.4** Calculen la complejidad de los ejercicios en línea del numeral 2.1 y 2.2 [Ejercicio Opcional] y agréguenla al informe PDF

**3.5** Expliquen con sus palabras las variables (*qué es 'n', qué es 'm', etc.*) del cálculo de complejidad del numeral 3.4

**4.** Simulacro de Parcial

**5.** Lectura recomendada **[Ejercicio Opcional]**

**6.** Trabajo en Equipo y Progreso Gradual **[Ejercicio Opcional]**