Laboratorio 4
Vuelta atrás (Backtracking)

Este laboratorio debe ser entregado en Eafit Interactiva en la fecha indicada en Eafit Interactiva. No es posible entregar el taller fuera del plazo establecido. Bajo ninguna circunstancia se reciben laboratorios por email o correos de Interactiva. El laboratorio debe ser desarrollado en parejas. El informe de laboratorio debe ser entregado en formato PDF. No se reciben informes de laboratorio en formato docx, el profesor puede no tener Microsoft Office en su computador. Para exportar a PDF utilizando Microsoft Word, haga click en el botón de Microsoft Office, escoja "guardar como", y, a continuación, haga click en PDF. Adicionalmente, el código fuente debe ser entregado en formato ZIP. No se reciben archivos en RAR.

El objetivo de este laboratorio es entender cómo diseñar algoritmos usando la técnica de diseño de algoritmos de *vuelta atrás (backtracking).*

Cómo escribir la documentación HTML de un código usando JavaDoc

Veamos en primer lugar qué se debe incluir al documentar una clase y sus métodos.

1. Nombre de la clase, descripción general, número de versión, nombre de autores.

2. Documentación de cada constructor o método (especialmente los públicos) incluyendo: nombre del constructor o método, tipo de retorno, nombres y tipos de parámetros si los hay, descripción general, descripción de parámetros (si los hay), descripción del valor que devuelve.

En la tabla siguiente mostramos algunas de las palabras reservadas (tags).

TAG	DESCRIPCIÓN	COMPRENDE
@author	Nombre del desarrollador.	Nombre autor o autores
@deprecated	Indica que el método o clase es obsoleto (propio de versiones anteriores) y que no se recomienda su uso.	Descripción
@param	Definición de un parámetro de un	Nombre de parámetro y

	método, es requerido para todos los parámetros del método.	descripción
@return	Informa de lo que devuelve el método, no se aplica en constructores o métodos "void".	Descripción del valor de retorno
@see	Asocia con otro método o clase.	Referencia cruzada referencia (#método(); clase#método(); paquete.clase; paquete.clase#método()).
@version	Versión del método o clase.	Versión

Tomado de http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=646:documentar-proyectos-java-con-javadoc-comentarios-simbolos-tags-deprecated-param-etc-cu00680b&catid=68:curso-aprender-programacion-java-desde-cero&Itemid=188

Criterios de evaluación de la documentación HTML de un código
0% No envió la documentación HTML
70% Envió la documentación HTML pero no documentó los métodos que hizo o lo hizo muy mal
100% Envió la documentación HTML y documentó correctamente cada método que hizo

Cómo calcular el tiempo que toma un código en ejecutarse en Java

```
long startTime = System.currentTimeMillis();
// ... do something ...
long estimatedTime = System.currentTimeMillis() - startTime;
```

Rúbricas de Calificación (9%)
(0.2 %) El formato del documento es PDF y no está en docx ni txt.

Autoevaluación
(1.0 %) Completan el formato de autoevaluación que está en formato xls

UVA
(1 %) Código fuente y el análsis de complejidad dentro del documento PDF.

(0.8 %) Comparte con la cuenta correspondiente los ejercicios (no aplica, se pone la misma nota que en el criterio anterior).

Informe de Lectura
(0.5 %) Mapa conceptual de la lectura
(0.5 %) Tema principal de la lectura
(0.5 %) Ideas principales de la lectura

Informe y código del laboratorio
(1.5 %) Entregan el código desarrollado para el laboratorio en formato .ZIP
(0.5 %) El código que entregan está documentado y entregan la doc en HTML. Si usan C++ documenten con Doxigen. Si usan Python, usen Pydoc.
(2.0 %) Entregan el informe de laboratorio respondiendo las preguntas en PDF

Quiz sobre conocimiento teóricos
(0.5 %) Cada pregunta vale 0.1 %

Código para entregar en ZIP junto con el javadoc en HTML en carpeta doc
En las diapositivas de la clase "Data Structures II: Backtracking" encontrará los algoritmos; en Internet encontrará implementaciones de las N Reinas usando backtracking.

1. Implemente el algoritmo de backtracking para encontrar UNA solución de las N Reinas. Si no logra implementar el algoritmo, busque una implementación en Internet y referénciela.

2. Construya ejemplos usando JUnit para probar su implementación de las N Reinas usando backtracking; por ejemplo, usando los ejemplos que ya conoce para el problemas de las 4 reinas. Puede usar las pruebas del laboratorio pasado

3. Tome los tiempos de ejecución de su algoritmo y complete la siguiente tabla**.** 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(?)

4. Implemente un algoritmo para resolver, usando Fuerza Bruta, el problema de calcular todos los caminos que hay en un grafo. Pruebe su algoritmo con un grafo completo (donde hay n^2 arcos y n vértices) de 4 o 5 nodos.

Preguntas para resolver en un informe en formato PDF

1. Para resolver el problema de las N Reinas, fuera de fuerza bruta y backtracking, ¿qué otras técnicas computacionales existen?

Informe de lectura para entregar en un informe en formato PDF

En Interactiva, encontrará un ejemplo de un informe de lectura, las rúbricas (criterios) de calificación, el PDF con la lectura (de ser posible), la plantilla para el informe y, paulatinamente, las respuestas.

3. Leer y realizar el informe lectura sobre "R.C.T Lee et al., Introducción al análisis y diseño de Algoritmos. Capítulo 5. Páginas 157 – 181."

1.1 Realice un mapa conceptual sobre la lectura. Se recomienda utilizar esta herramienta https://www.mindmup.com/#m:new-a-1437527273469.
1.2 Identifique cuál es el tema principal
1.3 Identifique 5 ideas principales de la lectura

Cálculo de complejidad, para compartir y entregar en informe PDF

En Interactiva, encontrará un ejemplo de una tarea de CodingBat, de forma similar se hacen las de UVA. Las rúbricas (criterios) de calificación, y, paulatinamente, las respuestas.

1. Resolver este problema usando backtracking en lugar de fuerza bruta. Si es posible, resolverlo en la página de UVA, sino se puede hacer en Java en el computador. Calcular la complejidad por favor.

http://acm.uva.es/contest/data/0167/problemset/p0.html

***2.** Si toman la respuesta de alguna parte, citar el URL; de lo contrario es plagio.*
***3.** Explicar con sus propias palabras la estructura de datos que utiliza para resolver el problema y cómo funciona el algoritmo. Si no es posible explicar su implementación, se considera plagio.*

Quiz de concepto teóricos tipo Saber Pro

1. Resolver el Quiz 4 en Interactiva. Lo pueden resolver en parejas, pero cada uno debe responderlo en Interactiva individual, de lo contrario la nota será 0.0 porque se

califica de forma automática e Interactiva no permite calificar en parejas. En las diapositivas tituladas “Data Structures II: Backtracking” encontrará muchas de las respuestas.

Importante
Cada pareja de estudiantes debe dar cuenta completa de *todos* y *cada uno* de los ejercicios que entrega en la tarea. No se debe utilizar código creado por otras personas, con excepción de las librerías de Java y el código que entregue el profesor. Si un estudiante no entiende el código que entregó, se manejará como fraude de acuerdo con el reglamento estudiantil (mirar los artículos 99 y 100 del Reglamento Académico en la siguiente página:

- http://www.eafit.edu.co/institucional/reglamentos/Paginas/reglamento-academico-pregrado.aspx#.UjBMrWTF3Rc
- http://www.eafit.edu.co/institucional/reglamentos/Documents/pregrado/regimen-disciplinario/cap1.pdf

Código de Ética

1. Usted puede conversar con sus compañeros acerca de los enfoques que cada uno está utilizando para la tarea, pero NO se debe mirar el código ni informes de sus compañeros y mucho menos usarlo como parte de su tarea.
2. No debe aceptar que otra persona (compañero, tío, amigo, novia, primo hermano del mocho) “le ayude” escribiendo parte del código o informes de su tarea.