

Laboratorio Nro. 1

Implementación de Grafos

Objetivos

- ☒ Entender la implementación de los grafos dirigidos
- ☒ Entender el concepto de sucesor (*successor*) como vecino, es decir, un nodo que es vecino de otro, un nodo que está conectado a otro por un arco.

Instrucciones Generales

Leer la Guía del Laboratorio



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.

Completar Autoevaluación



Como **actividad valorativa**, deben diligenciar el formato de autoevaluación a través del siguiente enlace: <http://bit.ly/2g8T8O6> . Haciendo esta evaluación es la única forma de alcanzar 5.0

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.

Entregar 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. Estos últimos **comprimidos** en un único ZIP.

Traducción de Enunciados



En el ZIP que el docente entrega a los alumnos, 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 Guía en sección 3, numeral 3.8.**

Resolver Ejercicios

1. Códigos para entregar en un ZIP junto la documentación en HTML:



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 **ZIP**



Documentación en **HTML**



No se reciben archivos en **.RAR**



En la vida real, los grafos se utilizan para representar redes sociales como *Facebook*, sistemas de información geográfica como *Google Earth* o enrutadores, como un enrutador ISR 4000 de Cisco

1.1 Teniendo en cuenta lo anterior:

- a) Realicen una implementación de la clase abstracta *Digraph*, llámela *DigraphAM* e implementen grafos con la estructura de datos Matrices de Adyacencia Etiquetadas



PISTA: Un error común es retornar el peso de los arcos en lugar de los identificadores de los vértices en el método *getSuccessors*

- b) Posteriormente, creen la clase *DigraphAL* e implementen grafos con la estructura de datos Listas de Adyacencia. Ambas clases heredan de la clase abstracta *Digraph* (digrafo o grafo dirigido)



PISTA: Un error común es retornar el peso de los arcos en lugar de los identificadores de los vértices en el método *getSuccessors*



PISTA 2: Véase *Guía en Sección 4, numeral 4.8* “Cómo definir una clase Pareja en Java”

1.2 En la clase *GraphAlgorithms*, implementen un método que reciba como parámetro un grafo dirigido y que retorne cuál es el vértice que tiene más sucesores (vecinos). Debe funcionar para ambas implementaciones de grafo.

1.3 Prueben su código con los ejemplos contruidos en los numerales 1.1 y 1.2 para el *Algoritmo de Dijkstra*. Deben obtener la misma respuesta con ambas implementaciones.



PISTA: Véase *Guía en Sección 4, numeral 4.14* “Cómo hacer pruebas unitarias en BlueJ usando JUnit” y *numeral 4.15* “Cómo compilar pruebas unitarias en Eclipse”



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”



En la vida real, una aplicación de los grafos es para describir mapas, como los usados por Google Maps. El archivo *medellin_colombia-grande.txt* que está en el ZIP que el docente les entregó, contiene un grafo que representa todas las calles de Medellín, es un grafo de aproximadamente 300.000 nodos

1.4 Teniendo en cuenta lo anterior, implementen un método que permita crear un grafo a partir de ese archivo del texto *medellin_colombia-grande.txt*



PISTA: Véase *Guía en Sección 4, numeral 4.13* “Cómo usar Scanner o BufferedReader”



PISTA 2: Hay información que sobra, por ejemplo, la latitud y la longitud de cada vértice y el nombre de cada arista.



PISTA 3: Es mejor usar *BufferedReader* porque es más rápido que *Scanner*. La idea es leer en una cadena de caracteres el contenido de cada línea y usando el método *split* de la clase *String* o usando *StringTokenizer*, dividir la cadena en partes cada que hay una coma (,).



PISTA 4: Como los códigos no son secuenciales, es decir, no empiezan en cero y tampoco están todos los números consecutivos, una forma de manejar los vértices es usar un mapa (en Java, *HashMap* o *TreeMap*).

☒ **Error Común:**



2) Ejercicios en línea sin documentación HTML en el ZIP



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



No entregar
documentación **HTML**



Entregar un archivo en
.JAVA



No se reciben archivos
en **.PDF**



Resolver los problemas
de **CodingBat** usando
Recursión



NOTA: Recuerden que, si toman la respuesta de alguna fuente, deben referenciar según el tipo de cita correspondiente. Véase Guía en Sección 4, numerales 4.16 y 4.17

2.1 Resuelvan mínimo 5 ejercicios del nivel **Recursion 2** de la página CodingBat: <http://codingbat.com/java/Recursion-2>



No está permitido el ejercicio **GroupSum**. Pero a continuación encontrarán algunos errores comunes que pueden ser aplicados con los otros ejercicios



PISTA: El algoritmo **GroupSum** falla porque al llamarse recursivamente con el parámetro `start` se queda en una recursión infinita

```
public boolean groupSum(int start, int[] nums, int target)
{
    if (start >= nums.length) return target == 0;
    return groupSum(start+1, nums, target - nums[start])
        || groupSum(start, nums, target );
}
```

}



PISTA 2: El algoritmo *GroupSum* falla porque al llamarse recursivamente con el parámetro `start-1` se sale del arreglo cuando `start = 0`.

```
public boolean groupSum(int start, int[] nums, int target)
{
    if (start >= nums.length) return target == 0;
    return groupSum(start+1, nums, target - nums[start])
        || groupSum(start-1, nums, target );
}
```



PISTA 3: El algoritmo *GroupSum* falla porque al llamarse recursivamente con el parámetro `start` se sale del arreglo cuando `start = 0`.

```
public boolean groupSum(int start, int[] nums, int target) {
    if (start >= nums.length) return target == 0;
    return groupSum(start+1, nums, target - nums[start])
        || groupSum(start+1, nums, target - nums[start - 1]
    );
}
```

2.2[Ejercicio Opcional]: Resuelvan el siguiente problema <http://bit.ly/2gTLZ53>



Pueden **entregar** un
archivo en **JAVA**



O **entregar** un archivo en
.CPP



O **entregar** un
archivo en **.PY**



PISTA: Utilicen *Búsqueda en Profundidad* (Siglas en inglés DFS)



PISTA 2: Véase Guía en **Sección 4, numeral 4.13** “Cómo usar Scanner o BufferedReader”

3) Preguntas para resolver en el informe PDF



Véase Guía en **Sección 3, Numeral 3.5**



Entregar informe de laboratorio en **PDF**



Usen la **plantilla** para responder laboratorios



No apliquen Normas Icontec para esto

3.1 Incluyan una imagen de la respuesta de las pruebas del numeral 1.4

3.2 Escriban una explicación entre 3 y 6 líneas de texto del código del numeral 1.1. Digan cómo funciona, cómo está implementado el grafo con matrices y con listas que hizo, destacando las estructuras de datos y algoritmos usados

3.3 Teniendo en cuenta el Ejercicio en Línea del numeral 2, ¿Qué concluyen sobre la complejidad asintótica en el peor de los casos de los problemas de CodingBat de *Recursion 1* con respecto a los de *Recursion 2*?



PISTA: Resuelvan las **ecuaciones de recurrencia** que obtuvieron del código en la herramienta <https://www.wolframalpha.com/>



PISTA 2: Véase **Guía en Sección 4, numeral 4.12** “Cómo resolver ecuaciones de recurrencia lineales y no lineales”

3.4 ¿En qué grafos es más conveniente utilizar la implementación con matrices de adyacencia y en qué casos es más conveniente listas de adyacencia? ¿Por qué?



Pista: <http://bit.ly/2gzZPLD>



Pista 2: <http://bit.ly/2gSMq1Z>

3.5 Para representar el mapa de la ciudad de Medellín del ejercicio del numeral 1.3, ¿qué es mejor usar, Matrices de Adyacencia o Listas de Adyacencia? ¿Por qué?




En la vida real para una red social como *Facebook*, donde hay al menos 100 millones de usuarios, pero cada usuario tiene en promedio 200 amigos,

3.6 Teniendo en cuenta lo anterior, respondan: ¿Qué es mejor usar, Matrices de Adyacencia o Listas de Adyacencia? ¿Por qué?



En la vida real, los enrutadores tienen una tabla de enrutamiento. Una tabla de enrutamiento guarda la distancia más corta para ir de un dispositivo a otro en la red. Un ejemplo de un enrutador es el ISR 4000 de Cisco. Otro ejemplo, es el que tiene en su casa para el Wifi

3.7 Teniendo en cuenta lo anterior, para representar la tabla de enrutamiento, respondan: ¿Qué es mejor usar, Matrices de Adyacencia o Listas de Adyacencia?

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

3.8 Expliquen con sus propias palabras cómo funciona el ejercicio *GroupSum5*, y opcionalmente, el ejercicio en línea opcional, es decir, el que encuentran en el numeral 2.2



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

3.9 Calculen la complejidad de los ejercicios en línea, numerales 2.1 y 2.2, y agréguelas al informe PDF

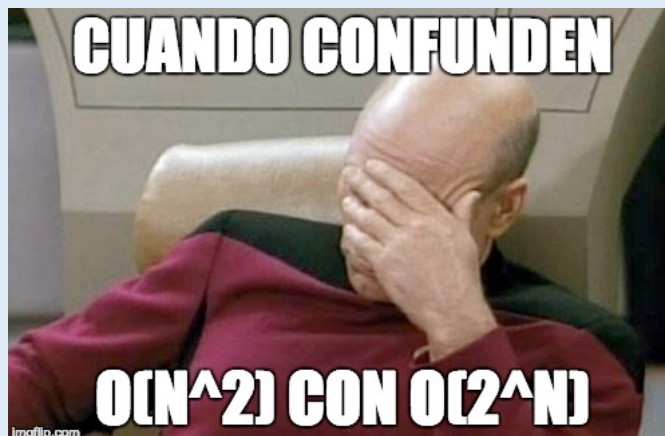
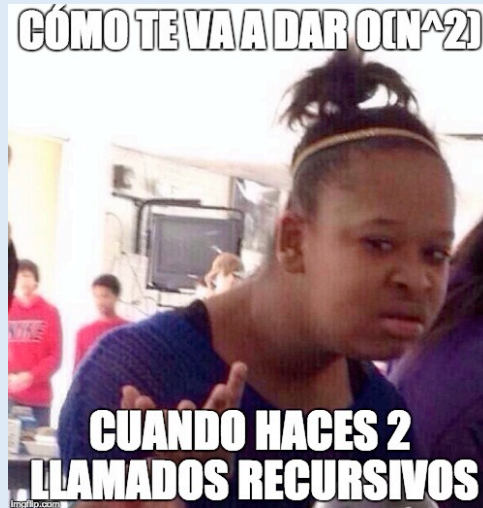


PISTA: Véase *Guía en Sección 4, numeral 4.11* “Cómo escribir la complejidad de un ejercicio en línea”



PISTA 2: Véase *Guía en Sección 4, numeral 4.19* “Ejemplos para calcular la complejidad de un ejercicio de CodingBat”

☒ **Errores Comunes:**



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

3.11 Compartan las respuestas de Recursion 2 con la cuenta estructurasdedatos2-2017-1@hotmail.com Para hacerlo, hagan clic en “prefs” y luego en “share to:”. No se pueden calificar ejercicios no compartidos.

4) Quiz de concepto teóricos tipo Saber Pro para resolver en el informe PDF



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



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

1. Una diferencia entre un árbol y un grafo es:

- a. El árbol no tiene arcos
- b. El grafo no tiene nodos
- c. El árbol no puede contener ciclos
- d. El grafo no puede contener ciclos

2. ¿Cómo se representan mejor los pesos de los arcos de un grafo en la implementación de listas de adyacencia?:

- a. No se puede
- b. Lista de pesos y lista de destinos
- c. Lista de parejas (peso y destino)
- d. Matriz de adyacencia

3. En un grafo que NO sea completo, o sea que hay vértices que no están conectados por un arco, ¿Cuál de las dos implementaciones ocupa más memoria?

- a. Matrices de Adyacencia
- b. Listas de Adyacencia
- c. Ocupan lo mismo
- d. No se puede calcular

4. Una diferencia entre un árbol y un grafo es:

- a. El árbol no tiene arcos
- b. El grafo no tiene nodos
- c. El árbol no puede contener ciclos
- d. El grafo no puede contener ciclos

5. ¿Cómo se representan mejor los pesos de los arcos de un grafo en la implementación de listas de adyacencia?:

- a. No se puede
- b. Lista de pesos y lista de destinos
- c. Lista de parejas (peso y destino)
- d. Matriz de adyacencia

6. En un grafo que NO sea completo, o sea que hay vértices que no están conectados por un arco, ¿Cuál de las dos implementaciones ocupa más memoria?

- a. Matrices de Adyacencia
- b. Listas de Adyacencia
- c. Ocupan lo mismo
- d. No se puede calcular

7. Si se quiere saber si existe un arco entre dos vértices determinados, ¿Qué implementación resolvería este problema en tiempo constante?

- a. Matrices de adyacencia
- b. Listas de adyacencia
- c. Ambas
- d. Ninguna

8. ¿Cuál de las siguientes afirmaciones es verdadera para un grafo no dirigido implementado con matrices de adyacencia?

- a. Ocupa la mitad de espacio que uno dirigido
- b. Ocupa el doble de espacio que uno dirigido
- c. La mitad de los datos son redundantes
- d. No contiene datos redundantes

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 "**Robert Lafore, Data Structures and Algorithms in Java (2nd edition), Chapter 13: Graphs. 2002**" 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: 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: Si desean otra lectura, consideren la siguiente: “*John Hopcroft et al., Estructuras de Datos y Algoritmos, Capítulo 6: Grafos dirigidos. Páginas 267 – 276. 1983*” que pueden encontrarla 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

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



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