Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI2693 – Laboratorio de Algoritmos y Estructuras III
Septiembre–Diciembre 2014

Proyecto I – 10%

Servicio Secreto

Bienvenido, estudiante...

La Agencia Logística de Gestión Operativa Secreta le da la bienvenida a su programa "Inteligencia Implacable e Innovación" (ALGOS III).

Se le ha contactado por medio de los profesores del Laboratorio de Algoritmos III, quienes los han recomendado por sus competencias y destrezas técnicas. La agencia posee una amplia red de espías distribuidos a través del mundo. Sin embargo, por razones de seguridad, no todos se conocen entre ellos. En este momento no contamos con la tecnología para comunicarnos eficientemente con todos nuestros espías y es en este punto que necesitamos su ayuda.

Su misión, si decide aceptarla, es la de construir una infraestructura tecnológica que nos permita representar nuestra red de espías con el objetivo de comunicarnos con ellos de manera más eficiente.

Este mensaje no se autodestruirá...

1 Implementación

Para construir la red de espías necesitará de una estructura de datos que la represente naturalmente. Una posible estructura para esto sería un *Grafo*, aprobado por las normativas de la agencia.

1.1 Grafos

Un grafo $(N, A, \mathcal{F})$ es un conjunto N de nodos, un conjunto A de aristas y una función $\mathcal{F}$ que relaciona aristas con pares de nodos. Es una estructura para representar conceptos (entes) y sus relaciones.

Un grafo tiene diversas implementaciones posibles (ej. Matriz de adyacencias, listas de adyacencias, lista de incidencias, etc.). En esta ocasión debe implementar el grafo haciendo uso de *conjuntos de adyacencia*. En esta representación existe un conjunto donde cada posición contiene un nodo y una lista con todos aquellos nodos con los que está relacionado. La figura 1 muestra esto para un ejemplo particular de grafo.

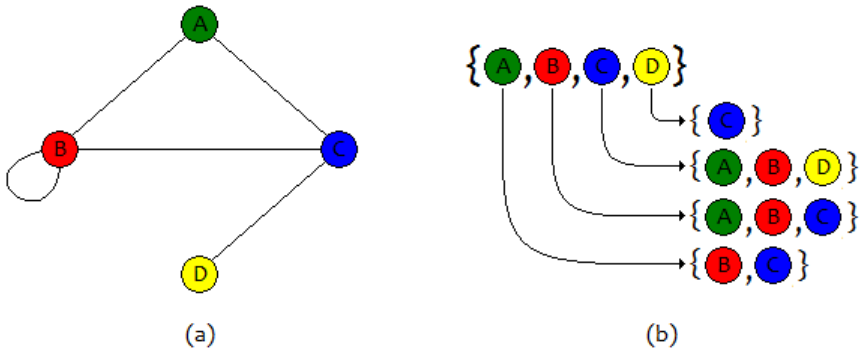

Figure 1: Un grafo (a) y su representación como conjuntos de adyacencias (b).

Nótese que es posible que un nodo esté relacionado consigo mismo (un agente que tiene contacto virtual con sus otras identidades).

1.2 Funcionalidades Básicas (5%)

Se desea que implemente una clase en Java para manejar grafos como conjuntos de adyacencia. Para esto debe primero crear una clase `Nodo`. Esta clase nodo deberá tener un nombre y un conjunto de nodos con el cual está relacionado (nodos adyacentes).

Sugerencia: Puede usar la clase `HashSet` *del paquete* `java.util` *para representar conjuntos.*

Las funcionalidades básicas que se esperan de la clase `Nodo` son las siguientes:

- Un constructor que reciba únicamente el nombre del nodo en cuestión e inicialice el conjunto de nodos adyacentes en vacío.
- Un constructor que reciba el nombre del nodo en cuestión y el conjunto de nodos adyacentes.
- Un método `obtenerNombre` que devuelva el nombre del nodo.
- Un método `obtenerAdyacentes` que devuelva el conjunto de nodos adyacentes.
- Un método `agregarAdyacente` que reciba un nodo y lo agregue al conjunto de adyacentes.
- Un método `agregarVariosAdyacentes` que reciba un conjunto de nodos y agregue todos al conjunto de adyacentes.
- Un método `imprimir` que imprima una representación del nodo con el siguiente formato:

```
[nombre] -> [nombre-ady-0, nombre-ady-1, ..., nombre-ady-n]
```

En el ejemplo de la figura 1, imprimir el nodo etiquetado con `A` resultaría en lo siguiente:

```
A -> [B, C]
```

Una vez implementada la clase `Nodo`, puede pasar a implementar la clase `Grafo` como un conjunto de nodos.

Las funcionalidades básicas que se esperan de la clase `Grafo` son las siguientes:

- Un constructor sin argumentos que inicialice el conjunto en vacío.
- Un método `obtenerNodos` que devuelva el conjunto de nodos del grafo.
- Un método `agregarNodo` que reciba un nodo y lo agrege al conjunto de nodos del grafo.
- Un método `imprimir` que imprima una representación del grafo, donde cada nodo del conjunto es escrito en una línea diferente.

1.3 Conociendo la red (3%)

Nuestra agencia conoce los contactos directos entre agentes, pero desearíamos aplanar nuestra red y conocer también los contactos indirectos. Esto es, si el agente 42 conoce al agente 86 de forma directa y el agente 86 conoce al agente 69 de forma directa. Los agentes 42, 86 y 69 son todos conocidos indirectos.

Se desea que implemente un método `aplanarRed` en `Grafo` que devuelva un grafo nuevo que represente a todos los contactos existentes (directos o indirectos). La figura 2 muestra el resultado de aplicar este método.

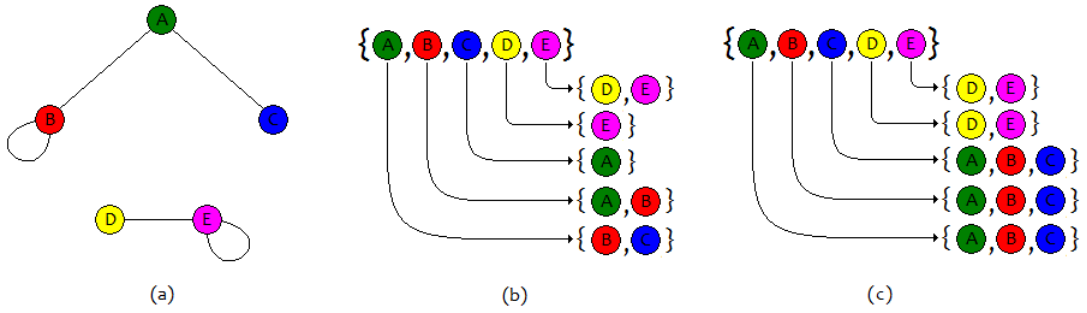

Figure 2: Un grafo (a), sus conjuntos de adyacencias (b) y su versión aplanada (c).

Para esto puede serle útil implementar un método `obtenerAlcanzables` que, dado un nodo, devuela un conjunto con todos los nodos que son alcanzables desde el mismo (incluyéndolo a el mismo). Por ejemplo, en la figura 2, los alcanzables de `B` son `A`, `B`, `C` y los alcanzables de `D` son `D`, `E`.

1.4 Enviando el mensaje (2%)

Ahora que se tiene la posibilidad de conocer la red aplanada, en la agencia queremos mandar mensajes a todos nuestros agentes. Sin embargo, para mantener el mensaje lo más secreto posible, usaremos una comunicación indirecta. Contactaremos a la menor cantidad de agentes posibles y les encargaremos que rieguen la información.

En el ejemplo de la figura 2, para alcanzar a todos los agentes bastaría con enviar el mensaje a dos de ellos:

- Uno de entre `D` o `E`.
- Uno de entre `A`, `B` o `C`.

Se desea que implemente un método `cobertores` en `Grafo` que devuelva la mínima cantidad de nodos necesarios para cubrir todo el grafo. Esto es, seleccionar la menor cantidad posible de nodos, tal que cualquier otro nodo del grafo tenga una relación directa o indirecta con los seleccionados. De esta forma, se tendrá la menor cantidad de agentes necesarios para hacer llegar el mensaje a toda la red de agentes.

2 Entrega

Cada equipo deberá entregar una carpeta comprimida en formato `tar.gz` con nombre `p1gXX` (donde `XX` es el número de su grupo). Esta carpeta debe contener un archivo `Nodo.java` con la implementación de la clase `Nodo`, un archivo `Grafo.java` con la implementación de la clase `Grafo` y un archivo `README` con comentarios que crea relevantes sobre el desarrollo de su proyecto.

El código que usted entregue será usado por un archivo `Main.java`, por lo que los nombres de las clases y de los métodos debe corresponder exactamente a los propuestos en este enunciado. Además su código debe estar bien documentado y seguir buenas prácticas de programación.

Su proyecto deberá der entregado y estar visible en el Aula Virtual a lo sumo el día 2 de Octubre, a las 11:59pm.