# Proyecto de Recuperación

## Estructuras de Datos, Primer Semestre 2020

Prof. Diego Seco

Ayudantes: Catalina Pezo y Alexis Espinoza

## 1. Introducción: Mini-Twitter

Para este proyecto, consideraremos una *Red Social* tipo Twitter. Este tipo de redes sociales pueden ser representadas como un *diGraph*, es decir, un grafo dirigido, en donde las aristas del grafo representan la relación "*seguir*". Es decir, si tenemos un nodo `A` que apunta a un nodo `B`, se interpretará como que el nodo `A` es seguidor o está siguiendo al nodo `B`.

Considerando un grafo como el descrito anteriormente, se pide implementar los siguientes métodos sobre él:

- **Find($u$):** Responde si el usuario de nombre $u$ es o no parte de la red social.
- **Clique():** Encontrar todos los cliques, de tamaño 3 o más, presentes en la red social. Sólo se considerarán cliques maximales (es decir, aquellos que no pueden ser extendidos añadiendo vértices adyacentes). Para ello, considere la siguiente definición de clique: Un clique en un grafo es un subconjunto de vértices, en donde todos los pares de nodos de ese subconjunto son vecinos (hay una arista que los une). Por ejemplo, para el digrafo de la figura 1, los cliques que se pueden encontrar son {`C, D, E`} y {`F, G, H, I, J`}. Para implementar este método se pueden buscar y adaptar soluciones conocidas. Sin embargo, el código del método debe ser escrito por el estudiante y comentado.
- **Compact():** Una de las técnicas utilizadas en el ámbito de las estructuras de datos compactas, en particular en grafos compactos, es introducir *nodos virtuales*. Estos nodos virtuales permiten representar subgrafos completamente conexos (cliques) como un solo nodo. El principal supuesto es que una vez que se llega a uno de los nodos del clique, es posible llegar a cualquier otro nodo del mismo clique. Utilizando lo anterior, el método Compact() deberá entregar una versión compacta del digrafo, reemplazando los cliques por nodos virtuales. Por ejemplo, el digrafo de la figura 1, se puede reducir o compactar al digrafo de la figura 2.
- **Follow($n$):** Sugerir los `n` usuarios más importantes de la red. Para ello, asuma que el grado de importancia de un usuario está dado por la cantidad de usuarios que lo siguen, de manera tal que el usuario más importante es aquel que es seguido por más usuarios. Si existe un empate, se deberán tomar los `n` primeros usuarios más importantes, dejando la definición de "primeros" al alumno.

## 2. Inputs y outputs

A modo de facilitar la revisión y el testing de los métodos, se deberá leer desde un archivo de entrada una secuencia de comandos que deberán ser ejecutados por la clase que se pide. Los formatos de entrada serán los siguientes:

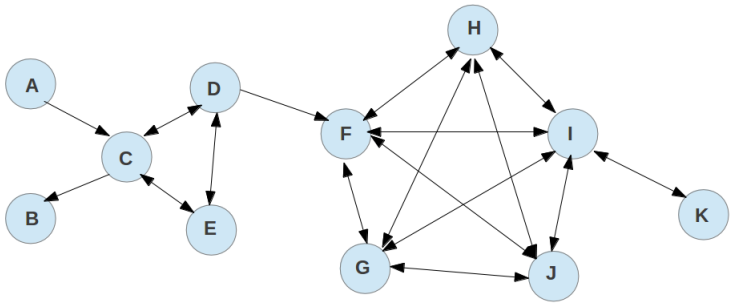


Figura 1: Digrafo con presencia de cliques.

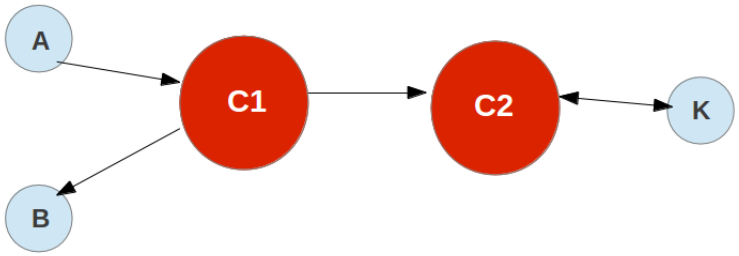


Figura 2: Digrafo compactado.

- `Add u1 u2`: Indica que se debe agregar la relación "El usuario *u1* sigue al usuario *u2*" al digrafo. Si uno de los usuarios (nodos) no existe actualmente en el digrafo, se debe agregar.
- `Find u`: Indica que se debe llamar al método *Find(u)*.
- `Clique`: Indica que se debe llamar al método *Clique*.
- `Compact`: Indica que se debe llamar al método *Compact*.
- `Follow n`: Indica que se debe llamar al método *Follow(n)*.

Los outputs (impresiones por pantalla) de cada uno de los comandos anteriores deberán ser los siguientes:

- Para `Add u1 u2` no se generará un output.
- Para `Find u` el output debe ser *Yes* si el usuario es parte de la red o *No* en caso contrario.
- Para `Clique` se debe entregar una lista de los subconjuntos de vértices que forman un clique (uno por cada línea).
- Para `Compact` se debe imprimir el digrafo resultante de la compactación. Se imprimirá un par *(u,v)* por cada línea, donde *(u,v)* significa que *u* sigue a *v*.
- Para `Follow n` se debe entregar una lista de los `n` usuarios más importantes (uno por cada línea).

Por ejemplo, si el archivo de entrada fuera como la columna `Input` a continuación (representa al grafo de la figura 1), el resultado debería ser como la columna `Output`:

| Input | Output |
|---|---|
| Add A C | |
| Add C B | |
| Add C D | |
| Add C E | |
| Add D C | |
| Add D E | |
| Add E C | |
| Add E D | |
| Add D F | |
| Add F G | |
| Add F H | |
| Add F I | |
| Add F J | |
| Add G F | |
| Add G H | |
| Add G I | |
| Add G J | |
| Add H F | |
| Add H G | |
| Add H I | |
| Add H J | |
| Add I F | |
| Add I G | |
| Add I H | |
| Add I J | |
| Add I K | |
| Add J F | |
| Add J G | |
| Add J H | |
| Add J I | |
| Add K I | |
| Find G | Yes |
| Find M | No |
| Clique | C D E |
| | F G H I J |
| Follow 2 | F |
| | I |
| Compact | (A,C1) |
| | (C1,B) |
| | (C1,C2) |
| | (C2,K) |
| | (K,C2) |

Parte de la evaluación de este proyecto será automática. Es decir, se suministrarán una serie de inputs (al menos 3) a la implementación solicitada, los cuales seguirán el formato descrito anteriormente y se compararán los outputs entregados con el output correcto. Por tanto, es muy importante ajustarse a la especificación del proyecto.

## 3. Evaluación

El proyecto se realizará indiviualmente. Subir en Canvas lo siguiente:

1. Un informe que:
   *a*) Incluya portada y respuestas a todas las preguntas de la tarea. Se incluirá una sección por algoritmo, explicando el funcionamiento del mismo y su pseudocódigo.
   *b*) Sea claro y esté bien escrito. Un informe difícil de entender es un informe que será mal evaluado aunque todo esté bien implementado. La persona que revise el documento debe poder entender su solución sólo mirando el informe.
   *c*) Esté en formato pdf.
2. Un archivo comprimido con todos los ficheros fuente implementados para solucionar la tarea. El informe debe hacer referencia a ellos y explicar en qué consiste cada uno.

**Fecha de Entrega: martes 1 de septiembre 11:59PM**