Triciclos en un grafo

La digitalización y el registro informátizado de todo tipo de interacciones (navegación en la web, redes sociales, transacciones nancieras, desplazamientos en vehículos, etc.), genera grafos masivos que son el objeto de estudio de numerosas ramas de conocimiento y de multitud de empresas.

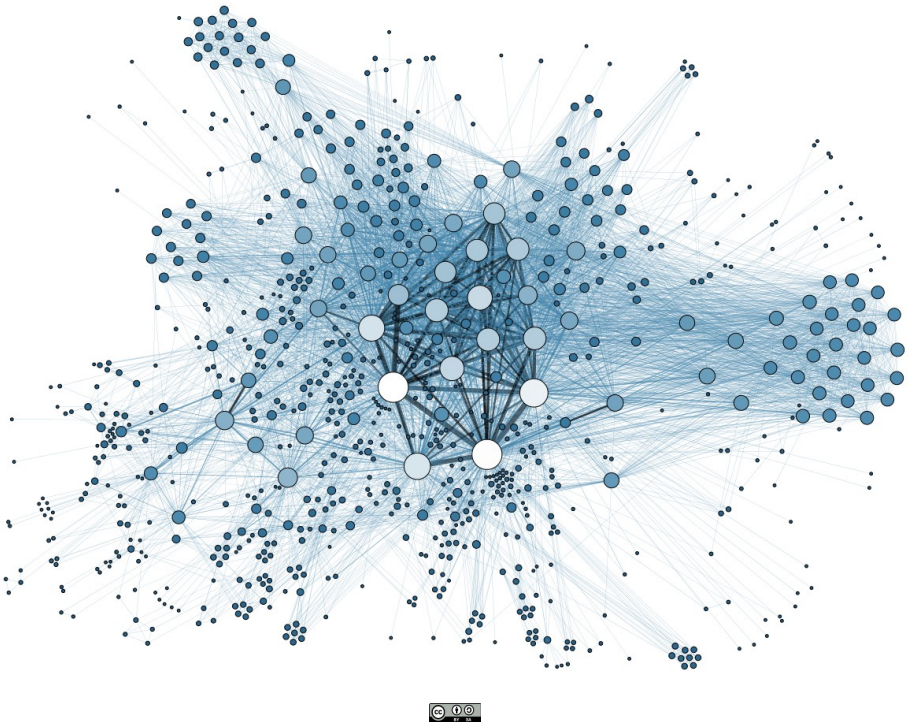


Figura 1: Visualization of social network analysis, Martin Grandjean (2013)

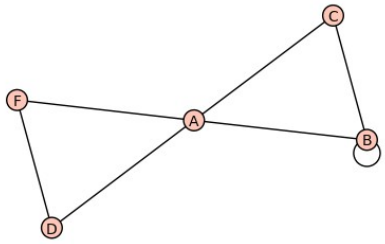
En el resto del ejercicio, consideraremos que los grafos están denidos como listas de aristas.

1 Triciclos (3-ciclos)

Los triciclos, o más formalmente 3-ciclos, son un elemento básico en el análisis de grafos y subgrafos. Los 3-ciclos muestran una relación estrecha entre 3 entidades (vértices): cada uno de los vértices está relacionado (tiene aristas) con los otros dos.

Escribe un programa paralelo que calcule los 3-ciclos de un grafo denido como lista de aristas.

Ejemplo Consideremos el siguiente grafo:



Un posible conjunto de aristas que lo dene es:

$$\{(B, B), (B, A), (C, A), (A, B), (A, D), (B, C), (F, A), (F, D)\}$$

y el conjunto de los 3-ciclos del grafo son:

$$\{(A, B, C), (D, A, F)\}$$

(Para más información consúltese la pista 1.)(Para más información consúltese la pista 2.)

2 Datos en múltiples cheros

Considera que los datos, es decir, la lista de las aristas, no se encuentran en un único chero sino en muchos.

Escribe un programa paralelo que calcule los 3-ciclos de un grafo que se encuentra denido en múltiples cheros de entrada.

3 3-ciclos locales

Supongamos que los datos del grafo se encuentran repartidos en múltiples cheros. Queremos calcular los 3-ciclos, pero sólamente aquellos que sean locales a cada uno de los cheros.

Escribe un programa paralelo que calcule independientemente los 3-ciclos de cada uno de los cheros de entrada.

Pistas

1. Se puede comenzar suponiendo que el grafo de partida, es decir, la lista de aristas que lo representa, está simplicada. Es decir, no tiene repeticiones, ni aristas de un nodo a sí mismo. Una vez resuelto el problema con esta restricción, no es difícil pasar al caso general.
2. Primero construimis la lista de adjacencias considerandos nodos posteriores:

A	[*B, C, D, F*]
B	[*C*]
C	[]
D	[*F*]
F	[]

Supón que tenemos la lista de adyacencia de un nodo. Si nos jamos en la de la arista *A*, su lista de adyacencia es *{B, C, D, F }*. A este nodo le podemos asociar la lista

```
[(( A ,   B ) ,  exists ) ,
(( A ,   C ) ,  exists ) ,
(( A ,   D ) ,  exists ) ,
(( A ,   F ) ,  exists ) ,
(( B ,   C ) ,  ( pending ,    A )) ,
(( B ,   D ) ,  ( pending ,    A )) ,
(( B ,   F ) ,  ( pending ,    A )) ,
(( C ,   D ) ,  ( pending ,    A )) ,
(( C ,   F ) ,  ( pending ,    A )) ,
(( D ,   F ) ,  ( pending ,    A ))]
```

Observa que las aristas están ordenadas lexicográcamente.