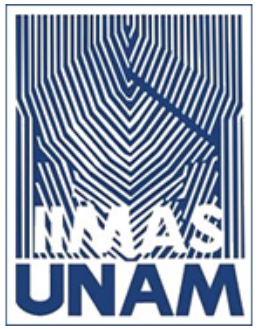


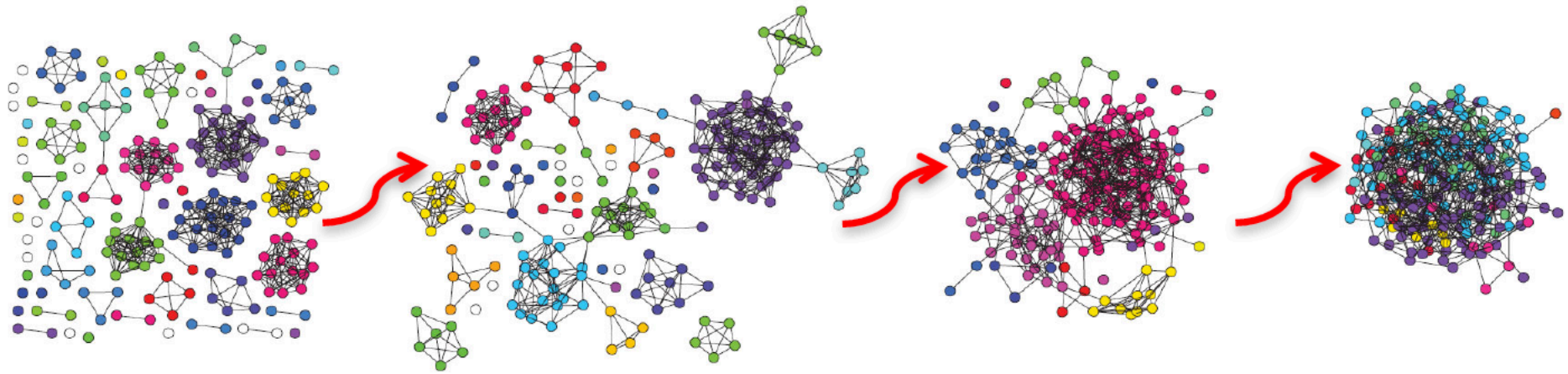
Taller de predicción de links

Dra. Helena Gómez Adorno
helena.gomez@iimas.unam.mx

*Taller basado en:
Curso Applied Machine Learning in Python (Universidad de Michigan)
Graph Analysis and visualization: Discovering Business Opportunity in Linked Data. Richard Brath and David Jonker

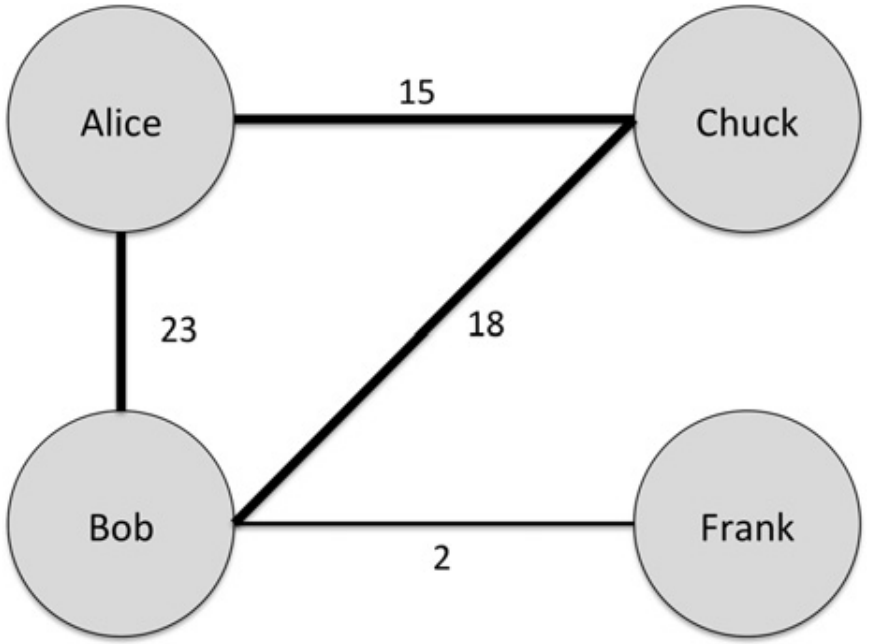
Problema de predicción de links

- Dado un grafo, podemos predecir las aristas que se formarán en el futuro?
- Otras aplicaciones: Limpieza de datos, identificar links perdidos, identificar nodos redundantes.



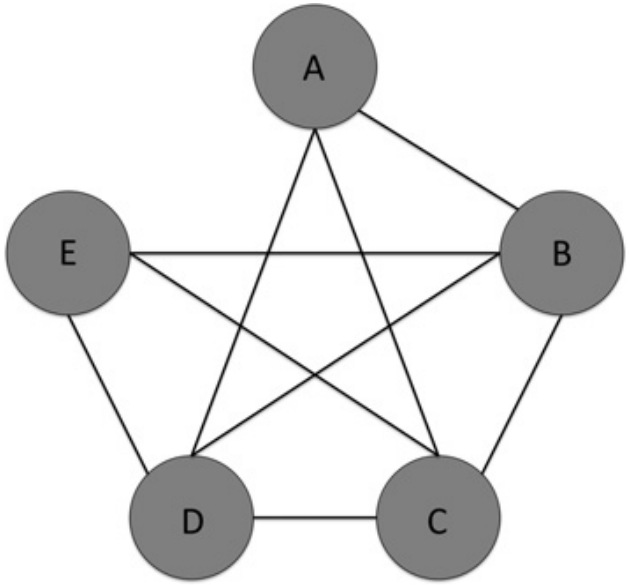
Ejemplo 1

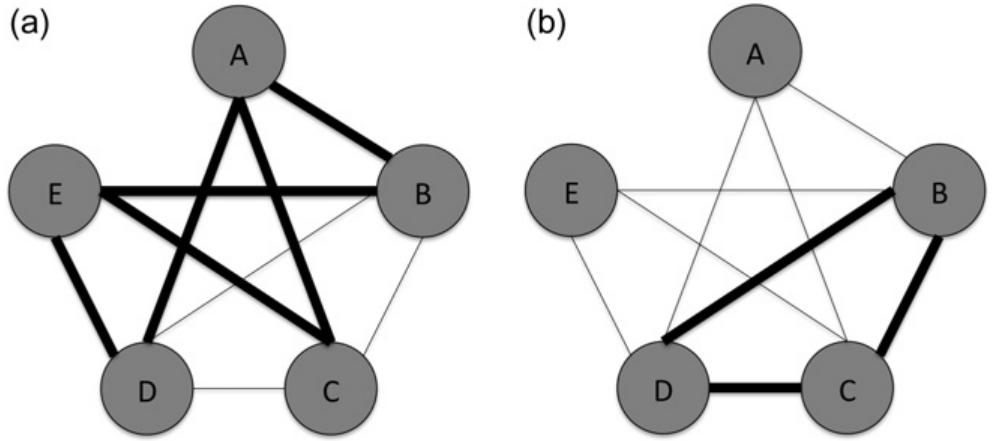
- La figura muestra que Alice, Bob, Chuck y Frank asisten juntos a diversas reuniones.
- Los pesos indican la cantidad de reuniones de un par de nodos.
- Si sabemos que Alice y Bob asistieron juntos a una reunión con una tercera persona, pero no sabemos quien fue.
- **Podemos consultar el grafo para intentar adivinar quien fue esta persona?**



Ejemplo 2

- En la figura de arriba podemos concluir que falta un link.
- Si agregamos pesos (grosor de arista):
  - En la figura (a), los nodos A y E tienen conexiones fuertes con todos los otros nodos del grafo. Es raro que nodos que comparten conexiones fuertes con otros nodos, no estén conectados.
  - En la figura (B), los nodos (A) y (E) solo comparten conexiones débiles, entonces es menos probable que tengan una conexión.





Predicción de links

- Varios métodos para calcular la predicción de links,

- Todos los algoritmos generan una puntuación que indica que tan cercanamente conectados están cada par de nodos.

- **Clausura triadica**: Es la tendencia de las personas que comparten conexiones en una red social a conectarse.

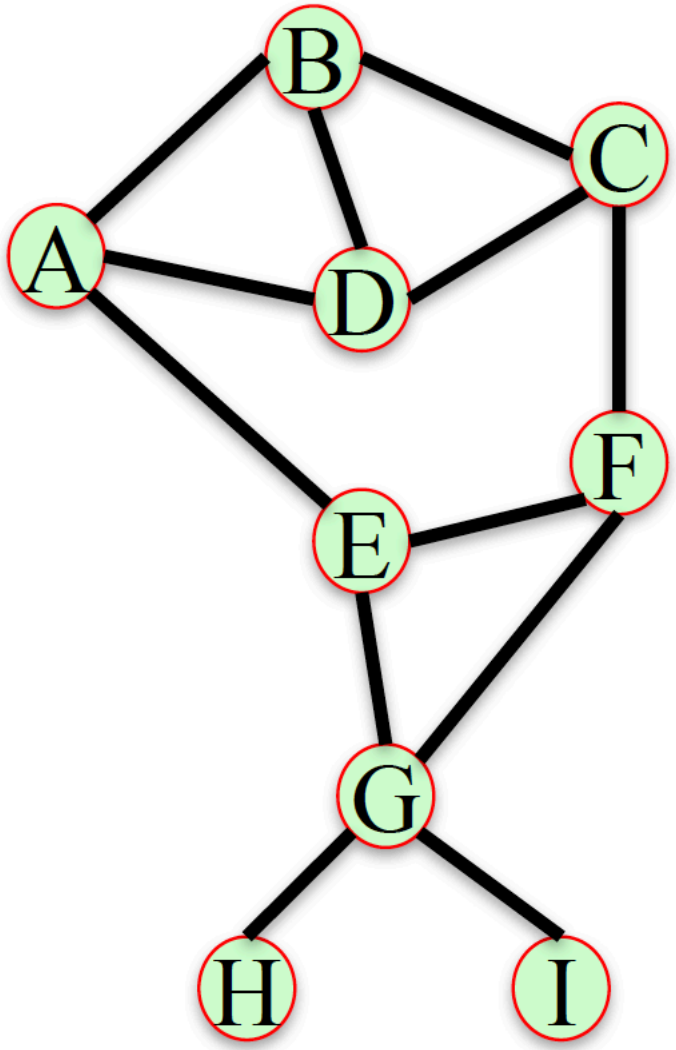
Medida 1: Vecinos comunes

El número de vecinos comunes de dos nodos X, Y es:

$$comm\_neigh(X, Y) = |N(X) \cap N(Y)|$$

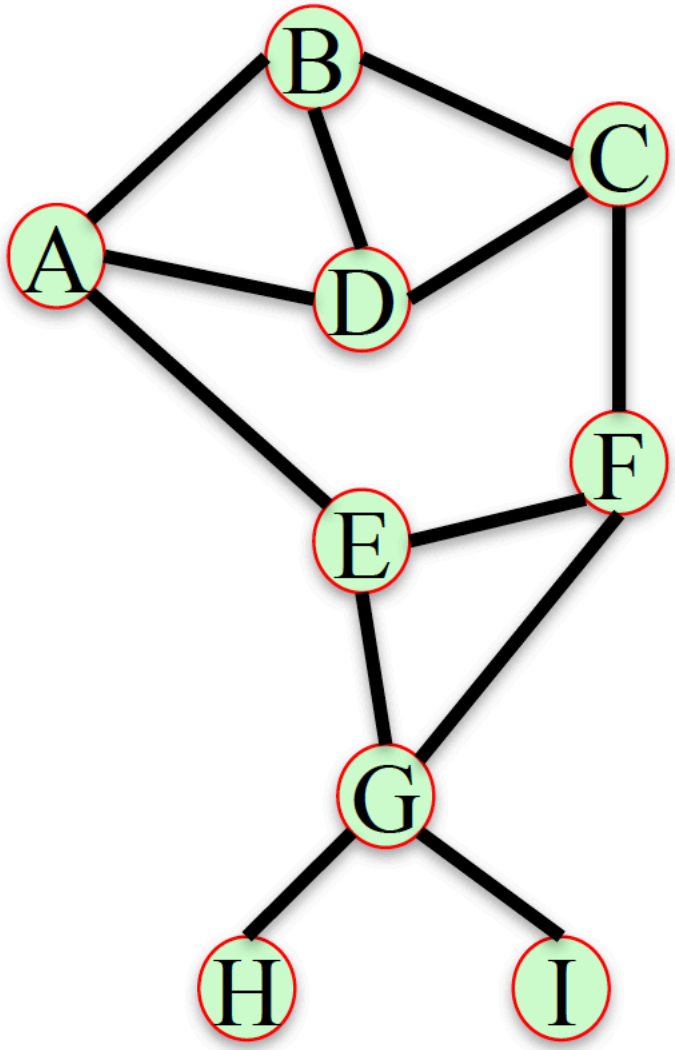
donde $N(X)$ es el conjunto de vecinos del nodo X

$$comm\_neigh(A, C) = |\{B, D\}| = 2$$



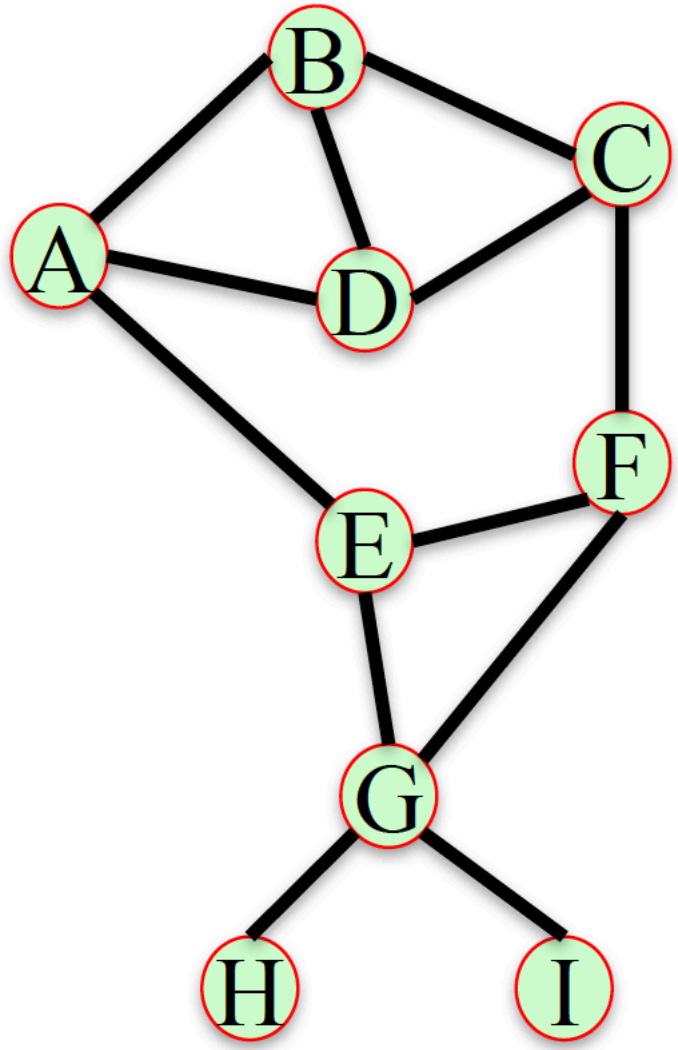
Medida 1: Vecinos comunes

```
>>> import operator

>>> import networkx as nx

>>> G=nx.Graph()

>>> G.add_edges_from([("A","B"),("A","D"), ("A","E"),
("B","D"),("B","C"),("D","C"),("E","F"),("E","G"),("F","G"
,("G","H"),("G","I")])

>>> common_neigh = [(e[0], e[1], len(list(
       nx.common_neighbors (G, e[0], e[1])))) for e in
       nx.non_edges(G)]

>>> sorted(common_neigh,key=operator.itemgetter(2), revers
= True )

>>> print (common_neigh)
```



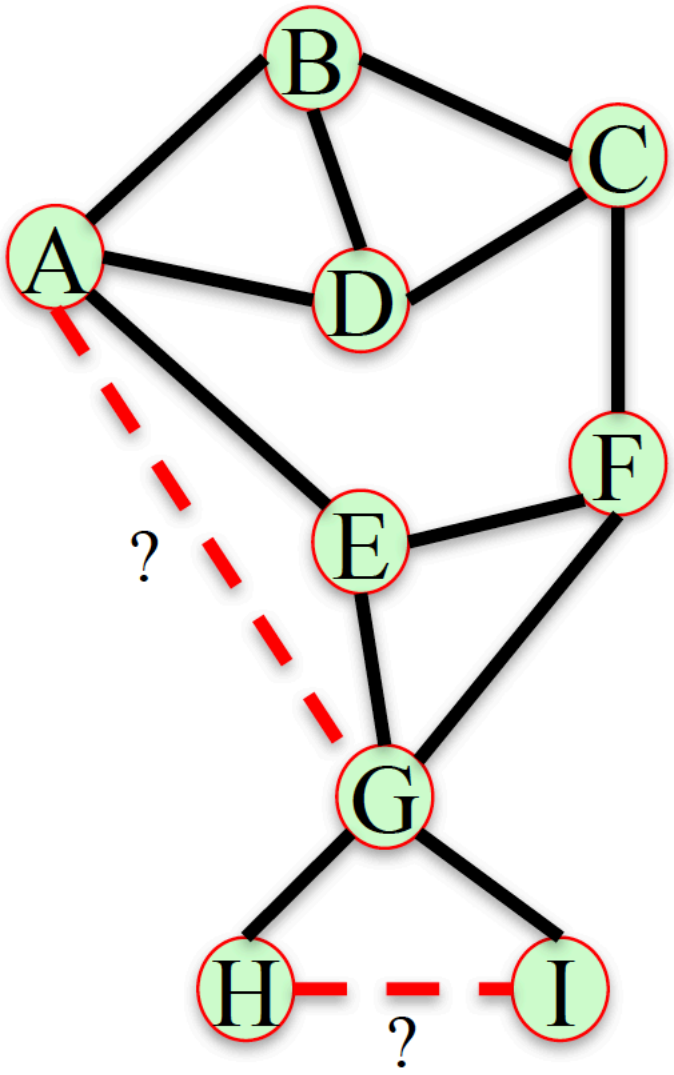
Medida 1: Vecinos comunes

Par	Medida vecinos comunes
A,C	2
A,G	1
A,F	1
C,E	1
C,G	1
B,E	1
B,F	1
E,I	1
E,H	1
E,D	1
D,F	1
F,I	1
I,H	1



Medida 1: Vecinos comunes

Par	Medida vecinos comunes
A,C	2
A,G	**1**
A,F	1
C,E	1
C,G	1
B,E	1
B,F	1
E,I	1
E,H	1
E,D	1
D,F	1
F,I	1
I,H	**1**



Pero el número de conexiones comunes no es todo !!
Celebridades con muchos amigos

Medida 2: Coeficiente de Jaccard

El coeficiente de Jaccard cuenta el número total de conexiones en común y lo divide por el número total de nodos con conexiones de cualquiera de los nodos.

El coeficiente de Jaccard de dos nodos X, Y es:

$$jacc\_coef(X,Y) = \frac{|N(X) \cap N(Y)|}{|N(X) \cup N(Y)|}$$

$$jacc\_coef(A,C) = \frac{|\{B,D\}|}{|\{B,D,E,F\}|} = \frac{2}{4} = \frac{1}{2}$$

El coeficiente de Jaccard cuenta el número total de conexiones en común y lo divide por el número total de nodos con conexiones de cualquiera de los nodos.

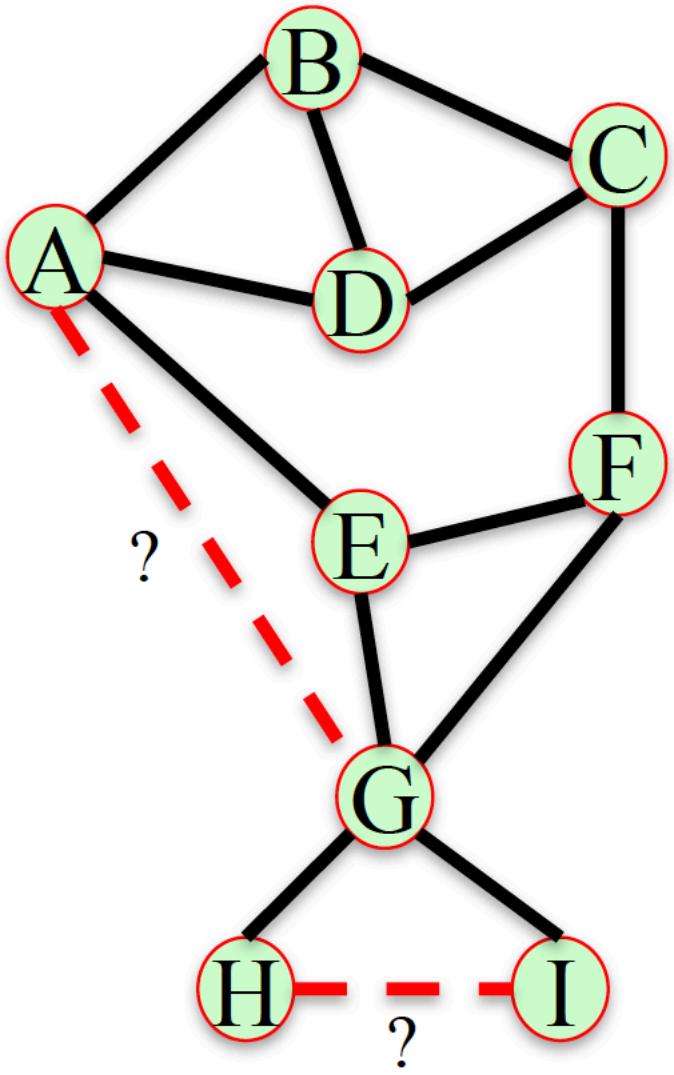
El coeficiente de Jaccard de dos nodos X, Y es:

$$jacc\_coef(X,Y) = \frac{|N(X) \cap N(Y)|}{|N(X) \cup N(Y)|}$$

$$jacc\_coef(A,C) = \frac{|\{B,D\}|}{|\{B,D,E,F\}|} = \frac{2}{4} = \frac{1}{2}$$

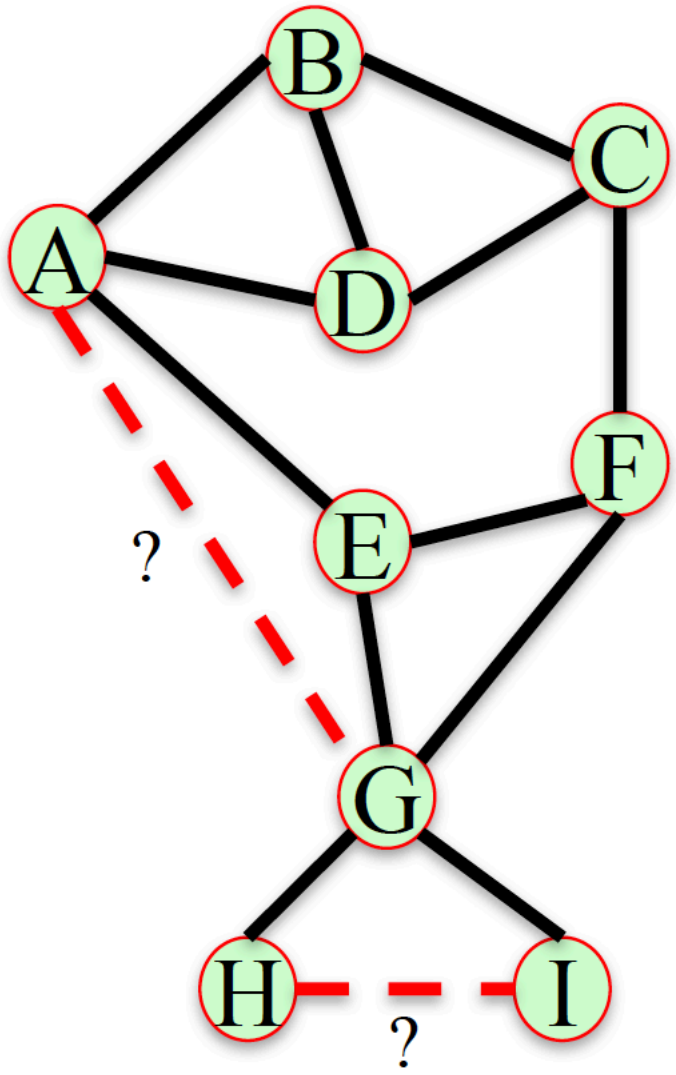
Medida 2: Coeficiente de Jaccard

```
>>> L = list(nx.jaccard_coefficient(G))
>>> L.sort(key=operator.itemgetter(2), reverse = True)
>>> print(L)
```



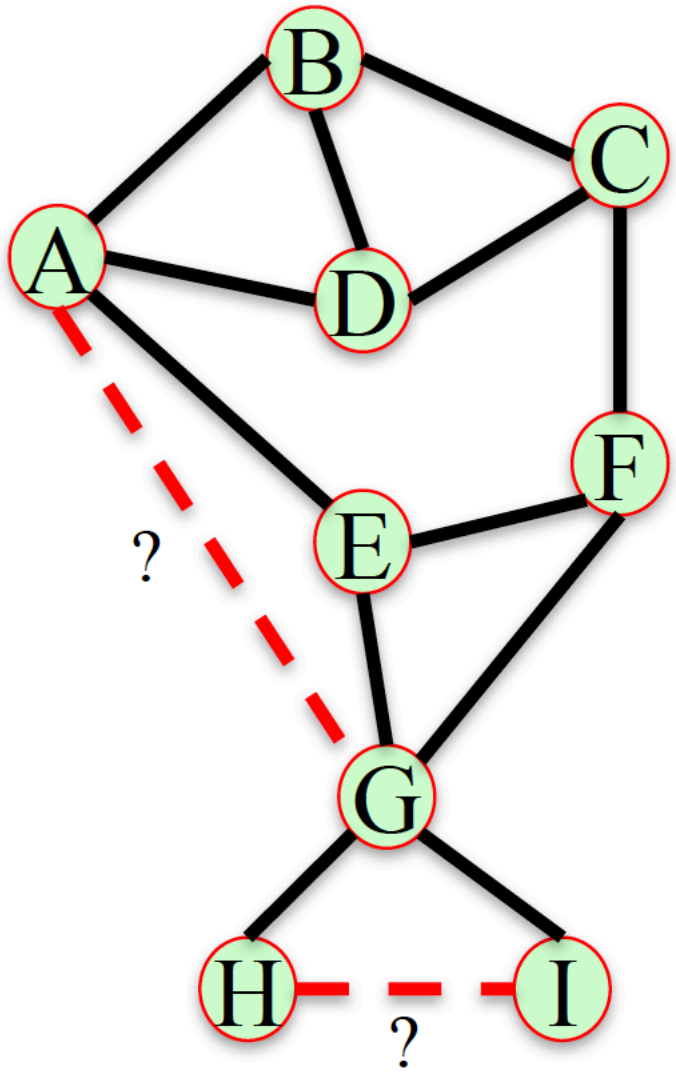
Medida 2: Coeficiente de Jaccard

Par	Medida coeficiente Jaccard
I,H	1
A,C	0.5
E,I	0.333333
E,H	0.333333
F,I	0.333333
F,H	0.333333
A,F	0.2
C,E	0.2
B,E	0.2
B,F	0.2
E,D	0.2
D,F	0.2
A,G	0.166666



Medida 2: Coeficiente de Jaccard

Par	Medida coeficiente Jaccard
I,H	**1**
A,C	0.5
E,I	0.333333
E,H	0.333333
F,I	0.333333
F,H	0.333333
A,F	0.2
C,E	0.2
B,E	0.2
B,F	0.2
E,D	0.2
D,F	0.2
A,G	**0.166666**



Caso de estudio

- Tenemos un grafo con 4 nodos: Alicia, Roberto, Jesús y David.
- Alicia y Roberto son celebridades: 1 millón de amigos cada uno.
- Jesús y David: 100 amigos cada uno.
- Alicia y Roberto: tienen 2,000 amigos en común.
- Jesús y David: tienen 20 amigos en común.
- Calcular el coeficiente de Jaccard para Alicia y Roberto, y para Jesús y David.

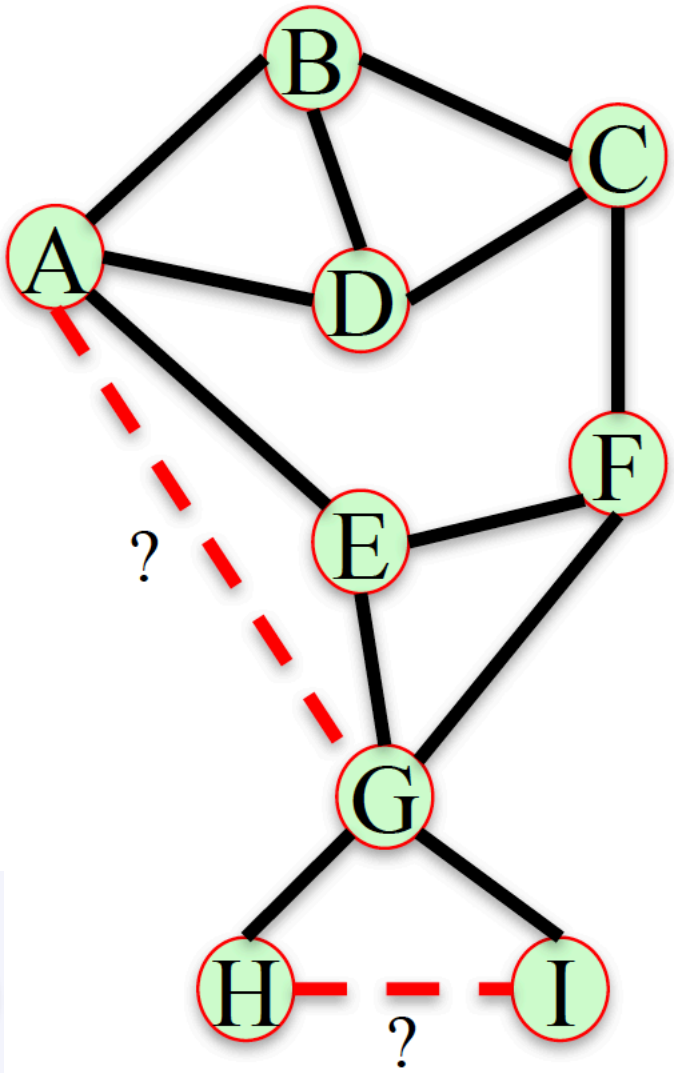
Que pasa si las 20 personas que Jesús y David conocen en común también son celebridades?

Medida 3: Asignación de recursos

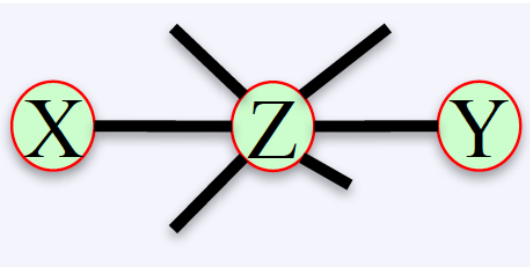
Fracción de un "recursos" que un nodo puede enviar a otro nodo a través de sus vecinos comunes.

El indice de asignación de recursos de dos nodos X, Y es:

$$res\_alloc(X,Y) = \sum_{u \,\epsilon\, N(X) \cap N(Y)} \frac{1}{|N(u)|}$$



Z tiene *n* vecinos
X envía 1 unidad a *Z*, *Z* distribuye la unidad uniformemente entre todos los vecinos
Y recibe *1/n* de la unidad.



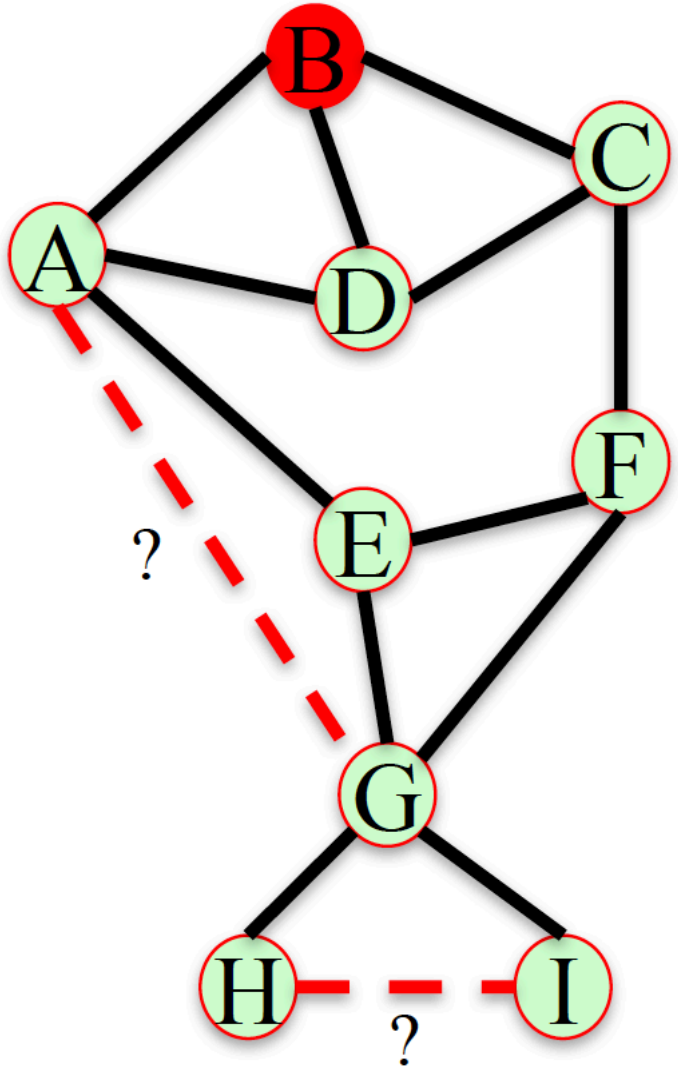
Medida 3: Asignación de recursos

Fracción de un "recursos" que un nodo puede enviar a otro nodo a través de sus vecinos comunes.

El indice de asignación de recursos de dos nodos X, Y es:

$$res\_alloc(X,Y) = \sum_{u \,\epsilon\, N(X) \cap N(Y)} \frac{1}{|N(u)|}$$

$$res_{alloc(A,C)} = \frac{1}{3} +$$



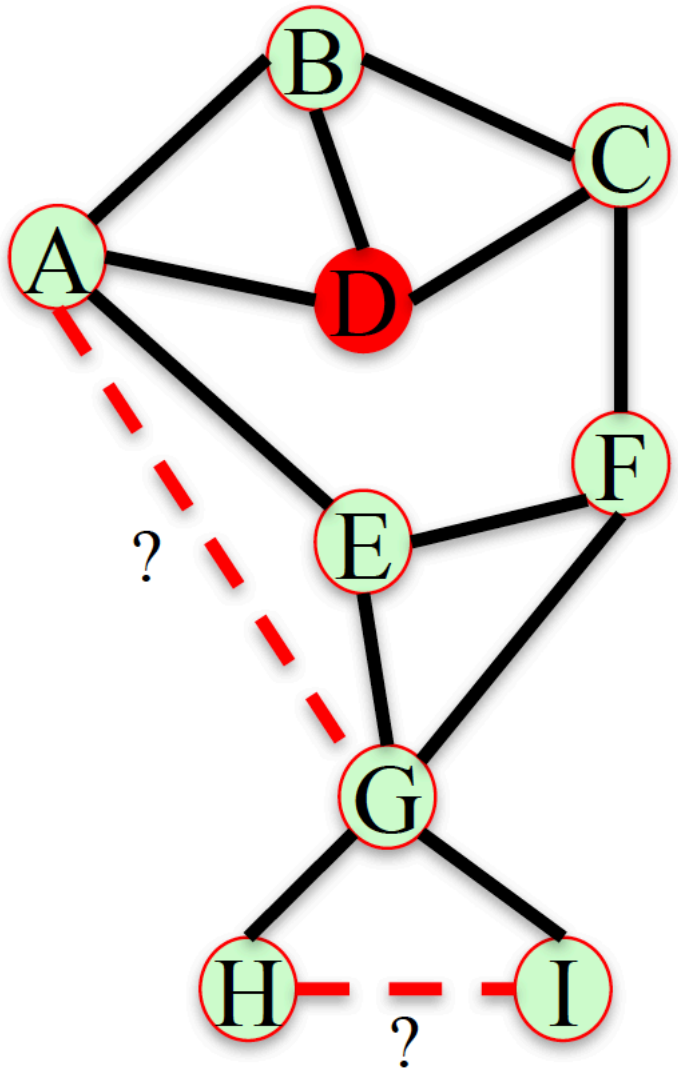
Medida 3: Asignación de recursos

Fracción de un "recursos" que un nodo puede enviar a otro nodo a través de sus vecinos comunes.

El indice de asignación de recursos de dos nodos X, Y es:

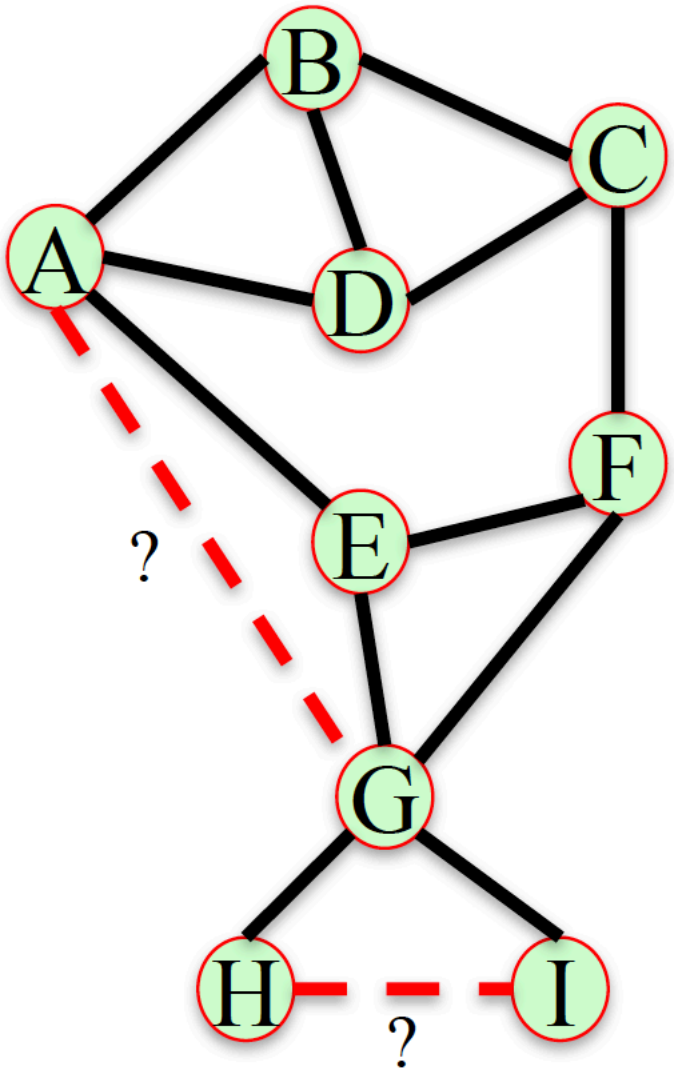
$$res\_alloc(X,Y) = \sum_{u \,\epsilon\, N(X) \cap N(Y)} \frac{1}{|N(u)|}$$

$$res\_alloc = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$$



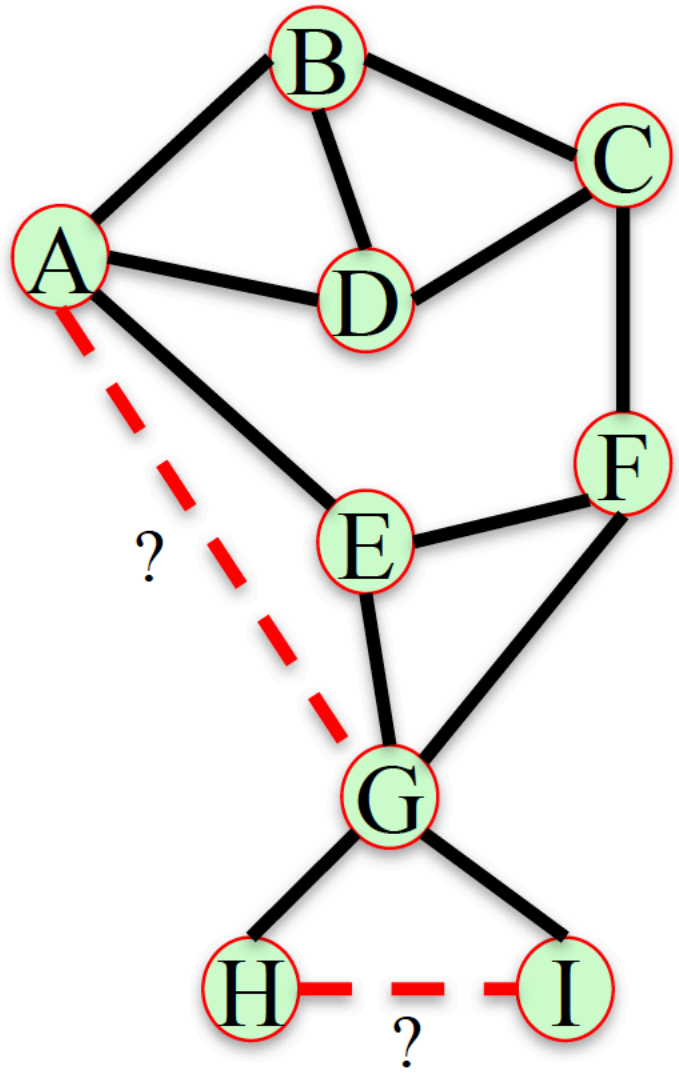
Medida 3: Asignación de recursos

```
>>> L = list(nx.resource_allocation_index(G))
>>> L.sort(key=operator.itemgetter(2), reverse = True)
>>> print(L)
```



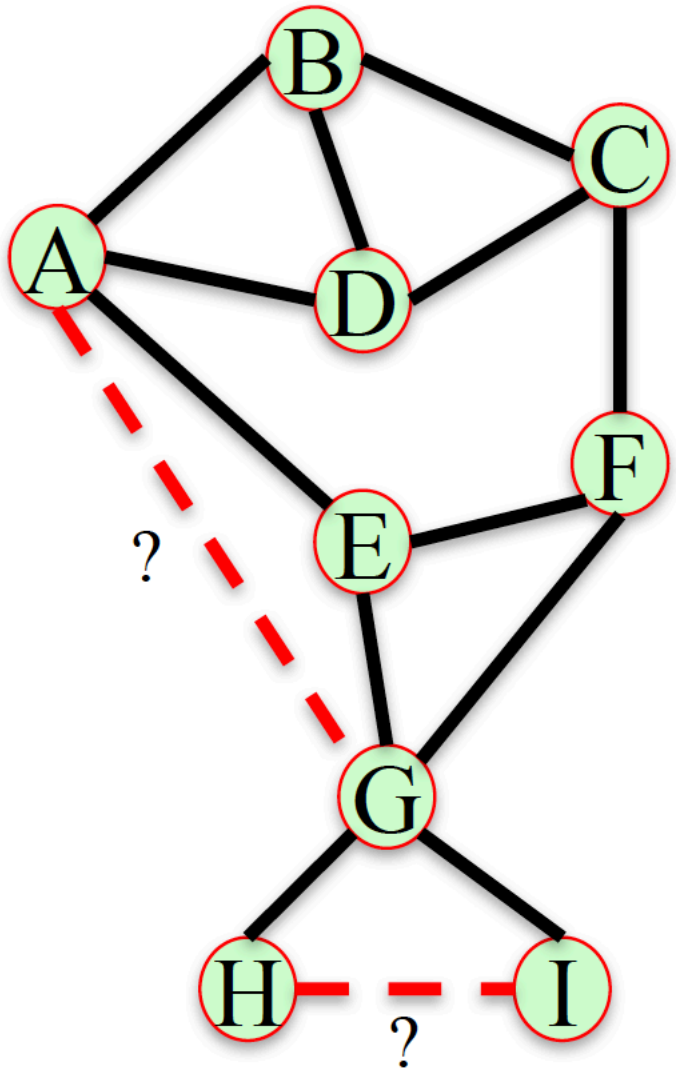
Medida 3: Asignación de recursos

Par	Medida asignación de recursos
A,C	0.666666
A,G	0.333333
A,F	0.333333
C,E	0.333333
C,G	0.333333
B,E	0.333333
B,F	0.333333
E,D	0.333333
D,F	0.333333
E,I	0.25
E,H	0.25
F,I	0.25
I,H	0.25



Medida 3: Asignación de recursos

Par	Medida asignación de recursos
A,C	0.666666
A,G	**0.333333**
A,F	0.333333
C,E	0.333333
C,G	0.333333
B,E	0.333333
B,F	0.333333
E,D	0.333333
D,F	0.333333
E,I	0.25
E,H	0.25
F,I	0.25
I,H	**0.25**



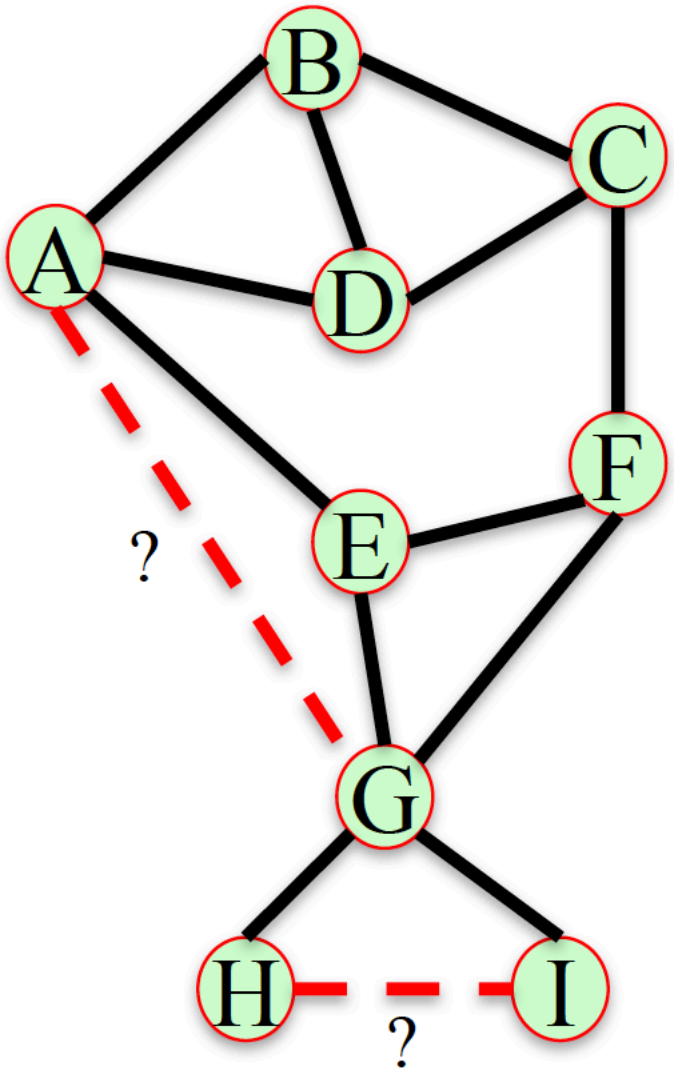
Medida 4: Índice Adamic-Adar

Similar a la asignación de recursos, pero con logaritmo en el denominador

El indice de Adamic-Adar de dos nodos X, Y es:

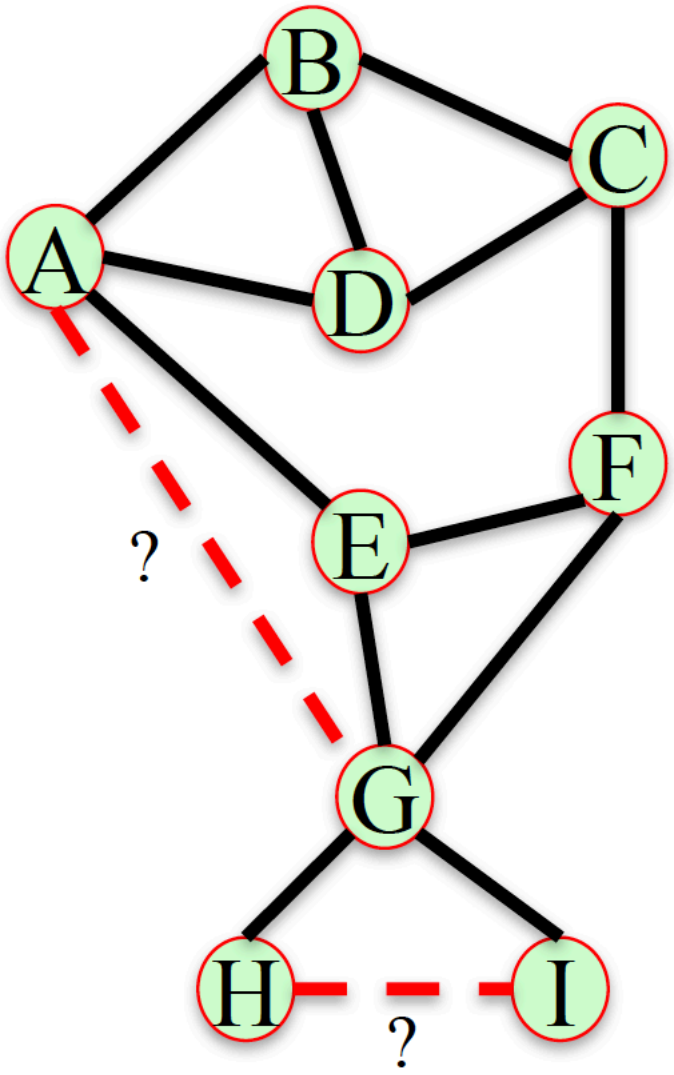
$$adamic\_adar(X,Y) = \sum_{u \,\epsilon\, N(X) \cap N(Y)} \frac{1}{\log(|N(u)|)}$$

$$adamic\_adar = \frac{1}{\log(3)} + \frac{1}{\log(3)} = 1.82$$



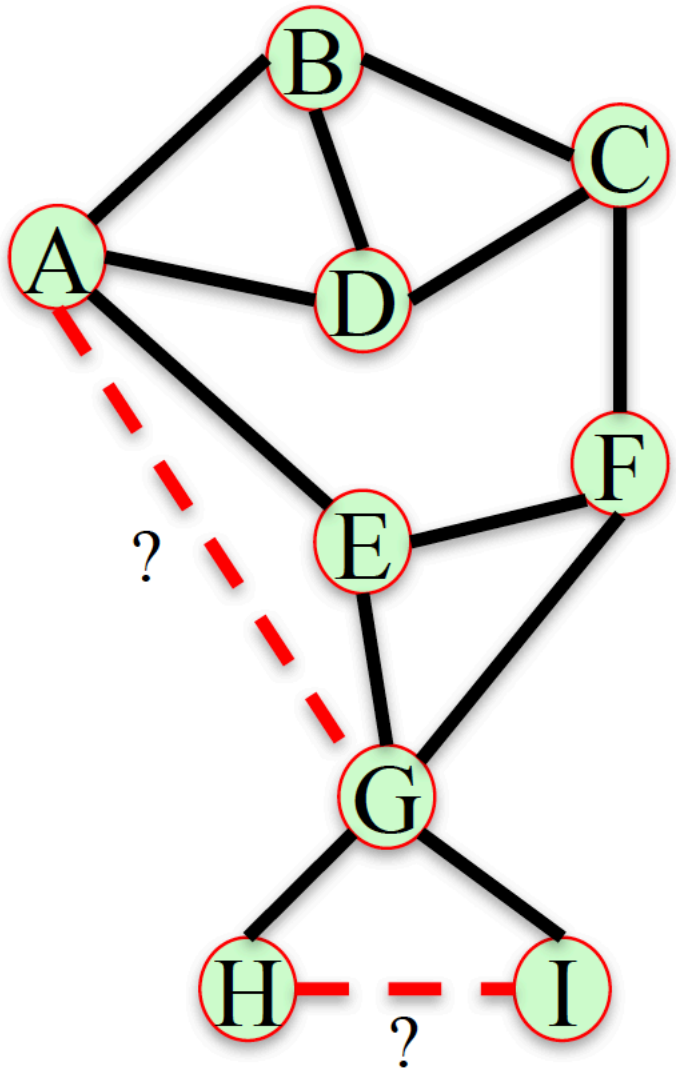
Medida 4: Índice Adamic-Adar

```
>>> L = list(nx.adamic_adar_index(G))
>>> L.sort(key=operator.itemgetter(2), reverse = True)
>>> print(L)
```



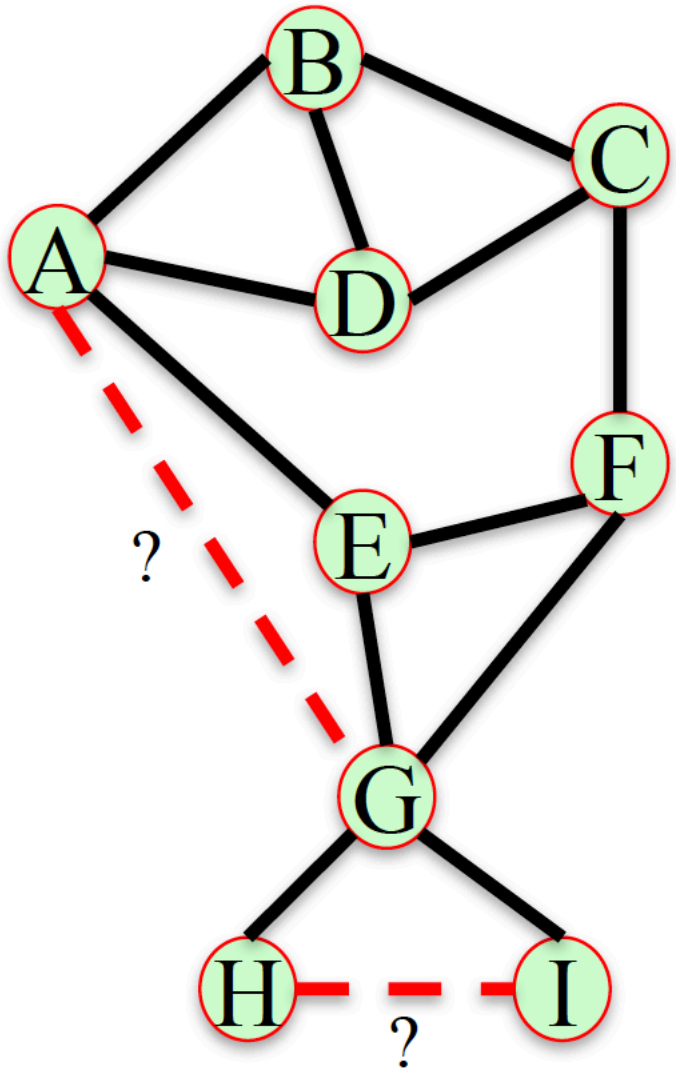
Medida 4: Índice Adamic-Adar

Par	Medida índice Adamic-Adar
A,C	1.8204784532536746
A,G	0.9102392266268373
A,F	0.9102392266268373
C,E	0.9102392266268373
C,G	0.9102392266268373
B,E	0.9102392266268373
B,F	0.9102392266268373
E,D	0.9102392266268373
D,F	0.9102392266268373
E,I	0.7213475204444817
E,H	0.7213475204444817
F,I	0.7213475204444817
I,H	0.7213475204444817



Medida 4: Índice Adamic-Adar

Par	Medida índice Adamic-Adar
A,C	1.8204784532536746
A,G	**0.9102392266268373**
A,F	0.9102392266268373
C,E	0.9102392266268373
C,G	0.9102392266268373
B,E	0.9102392266268373
B,F	0.9102392266268373
E,D	0.9102392266268373
D,F	0.9102392266268373
E,I	0.7213475204444817
E,H	0.7213475204444817
F,I	0.7213475204444817
I,H	**0.7213475204444817**



Medida 5: Unión preferencial

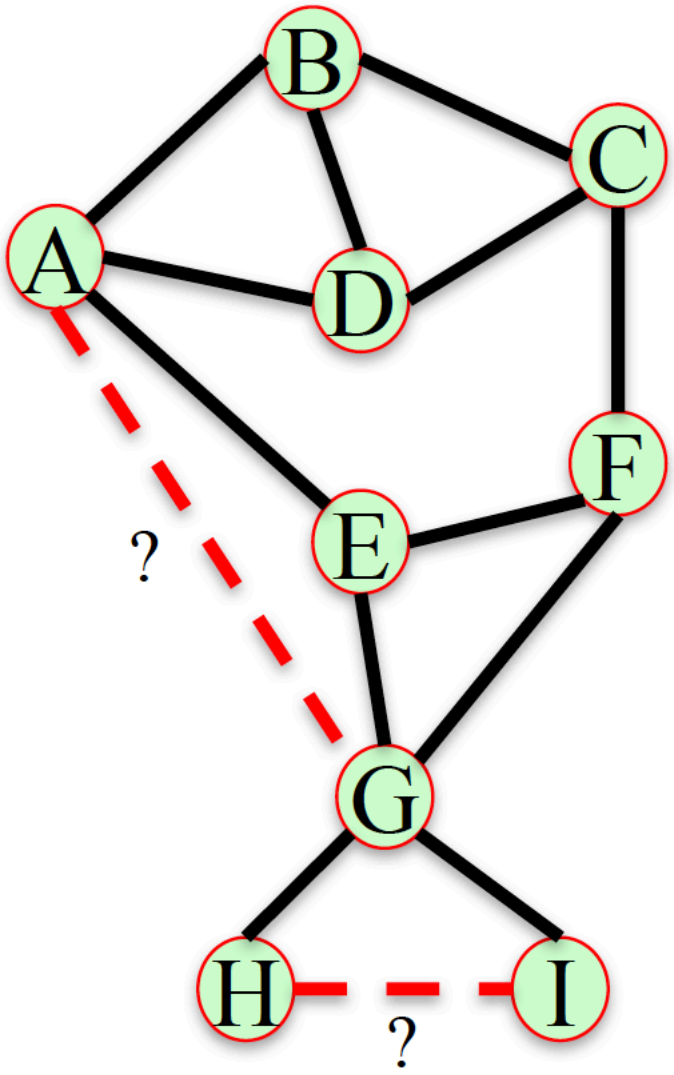
En el modelo de unión preferencial, los nodos con mayor grado obtienen más vecinos.

Producto del grado de nodos.

El puntaje de unión preferencial de dos nodos X, Y es:

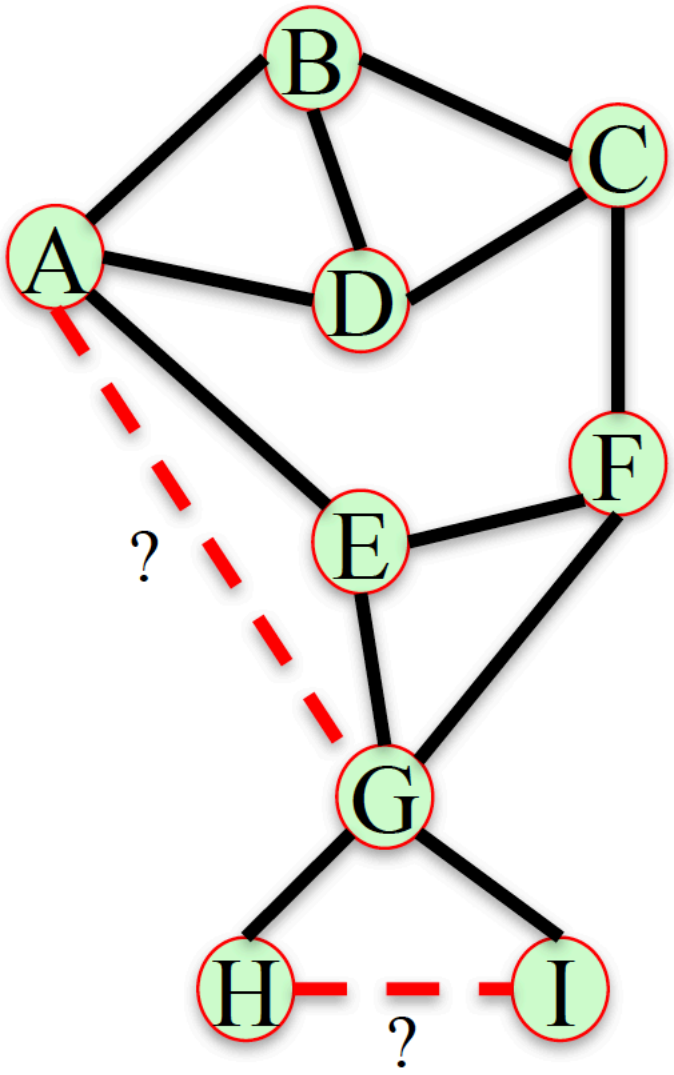
$$pref\_attach(X, Y) = |N(X)||N(Y)|$$

$$pref\_attach(A, C) = 3 * 3 = 9$$



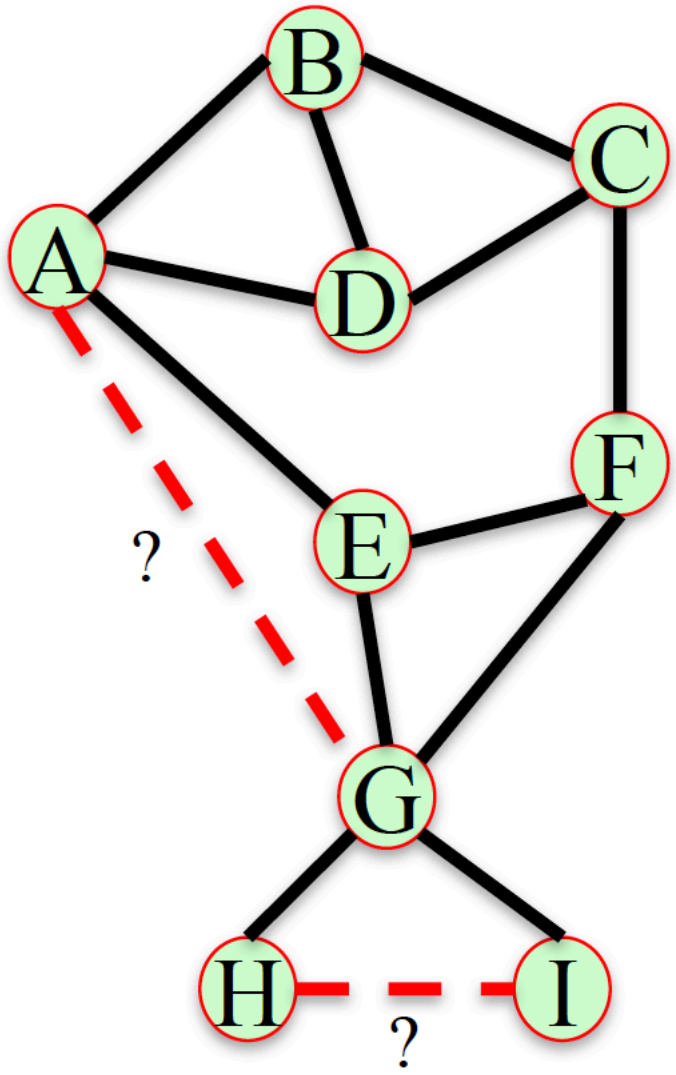
Medida 5: Unión preferencial

```
>>> L = list(nx.preferential_attachment(G))
>>> L.sort(key=operator.itemgetter(2), reverse = True)
>>> print(L)
```



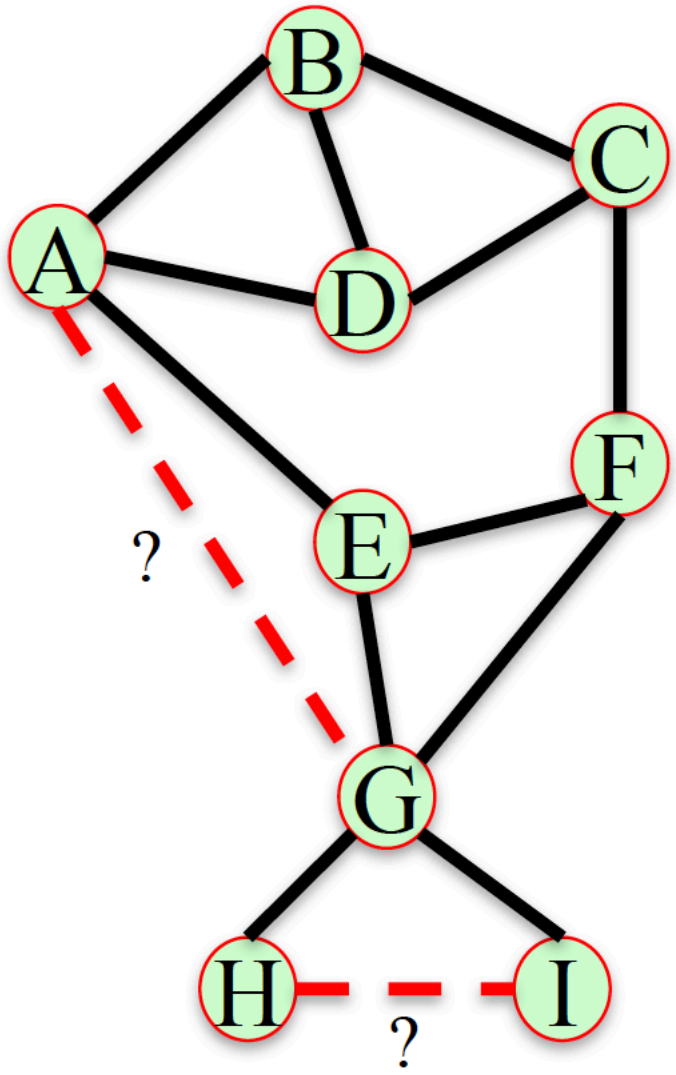
Medida 5: Unión preferecial

Par	Medida unión preferencial
A,G	12
C,G	12
B,G	12
D,G	12
A,C	9
A,F	9
C,E	9
B,E	9
B,F	9
E,D	9
D,F	9
A,I	3
I,H	1



Medida 5: Unión preferecial

Par	Medida unión preferencial
A,G	**12**
C,G	12
B,G	12
D,G	12
A,C	9
A,F	9
C,E	9
B,E	9
B,F	9
E,D	9
D,F	9
A,I	3
I,H	**1**

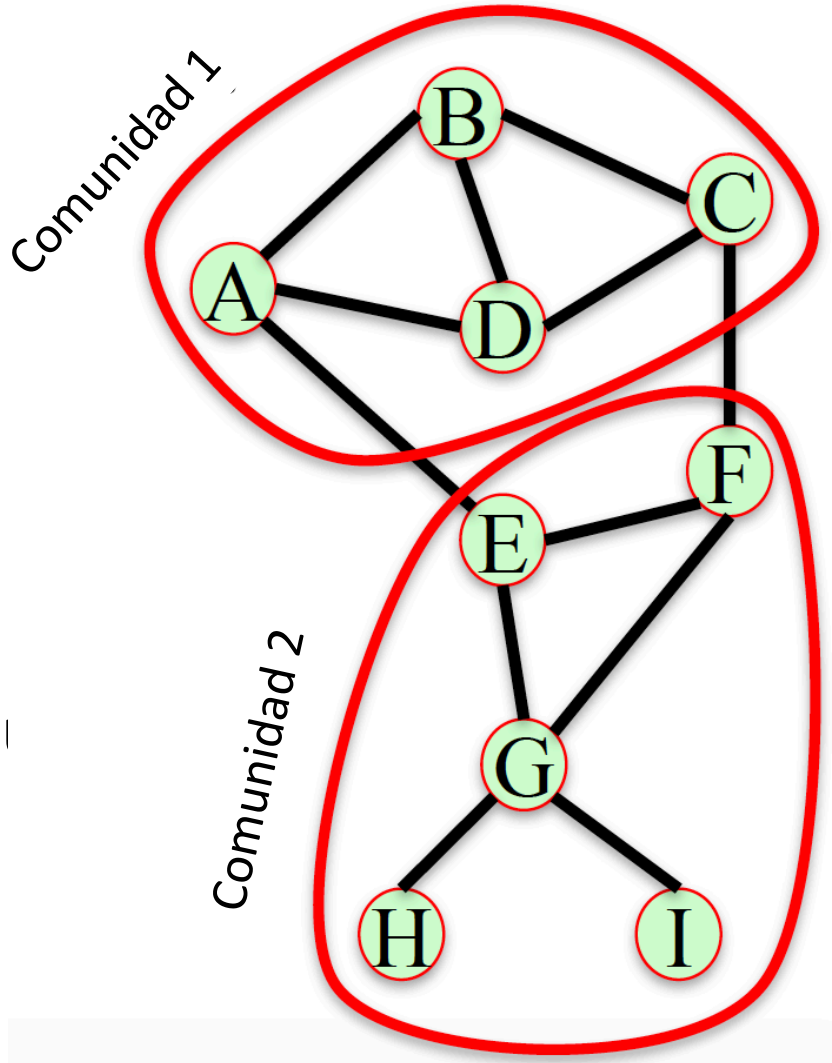


Medida 6: Vecinos comunes de la comunidad

Algunas medidas consideran la estructura de comunidades del grafo para la predicción de links.

Supongamos que los nodos en este grafo pertenecen a diferentes comunidades (conjuntos de nodos).

Los pares de nodos que pertenecen a la misma comunidad y tienen muchos vecinos comunes en s comunidad probablemente formen una conexión.



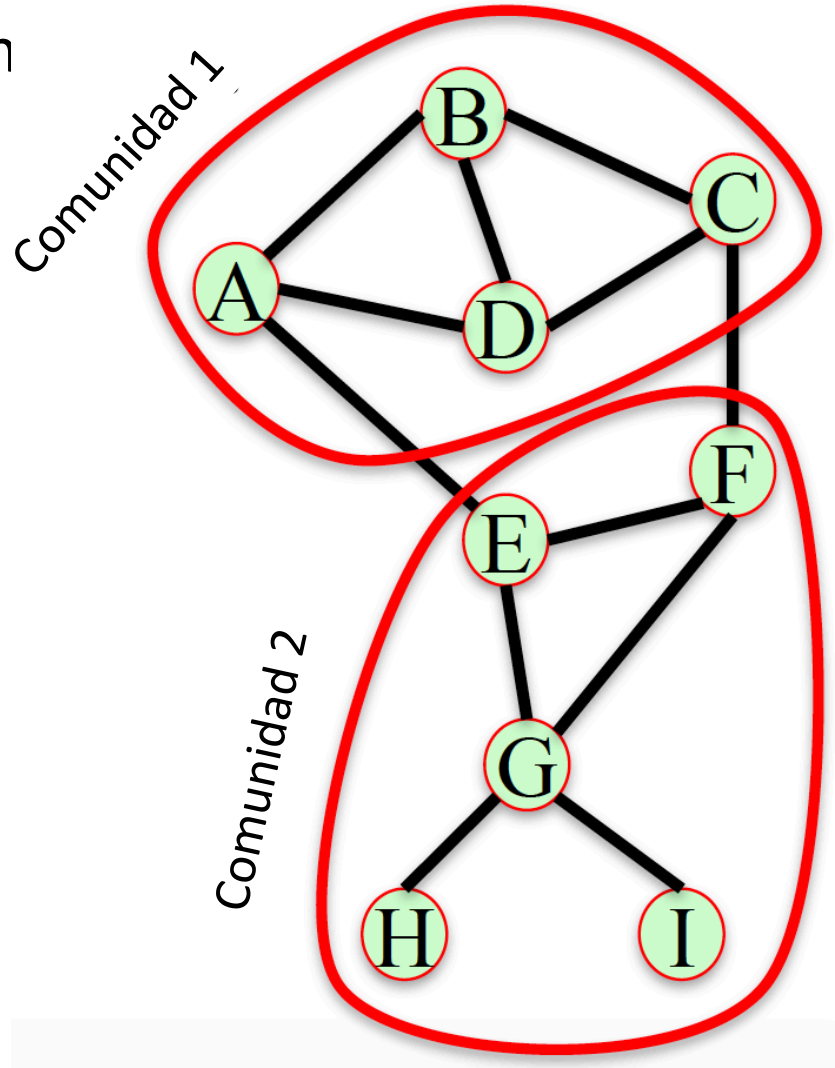
Medida 6: Vecinos comunes de la comunidad

Número de vecinos comunes con puntos por ser vecinos en la misma comunidad.

El score de vecinos comunes Soundarajan-Hopcroft de dos nodos X, Y es:

$$cn\_soundarajan\_hopcroft(X,Y) = |N(X) \cap N(Y)| + \sum_{u \in N(X) \cap N(Y)} f(u)$$

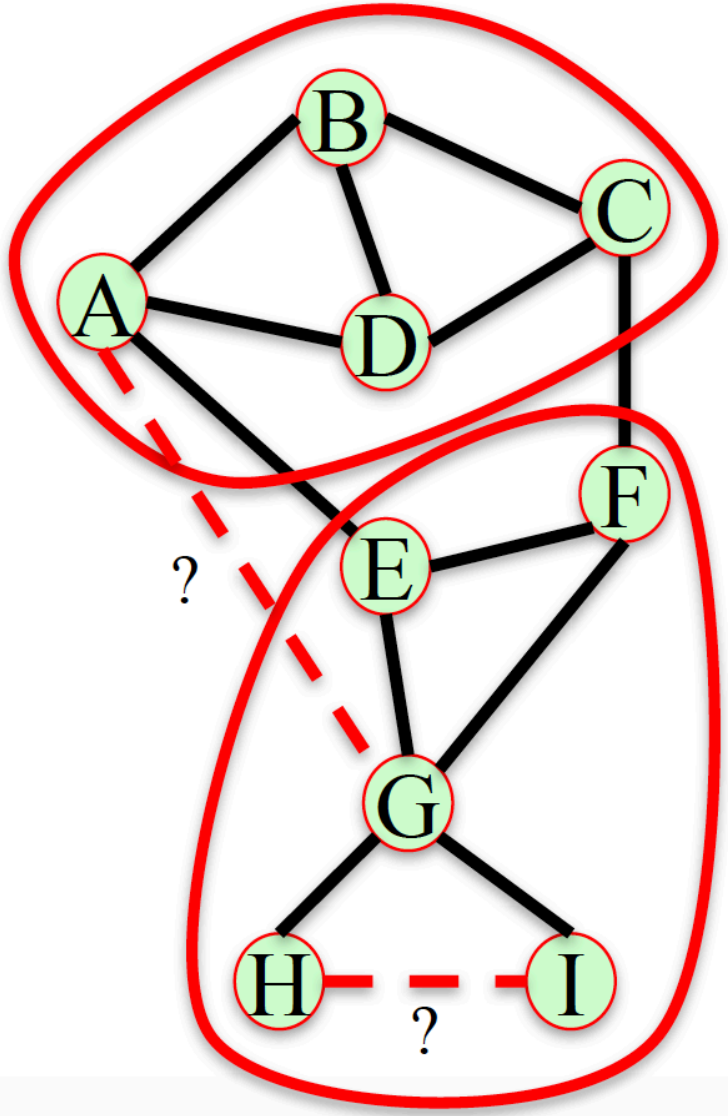
$$donde\ f(u) = \begin{cases} 1, u\ en\ la\ misma\ comunidad\ X,Y \\ 0, en\ diferente\ comunidad \end{cases}$$



Medida 6: Vecinos comunes de la comunidad

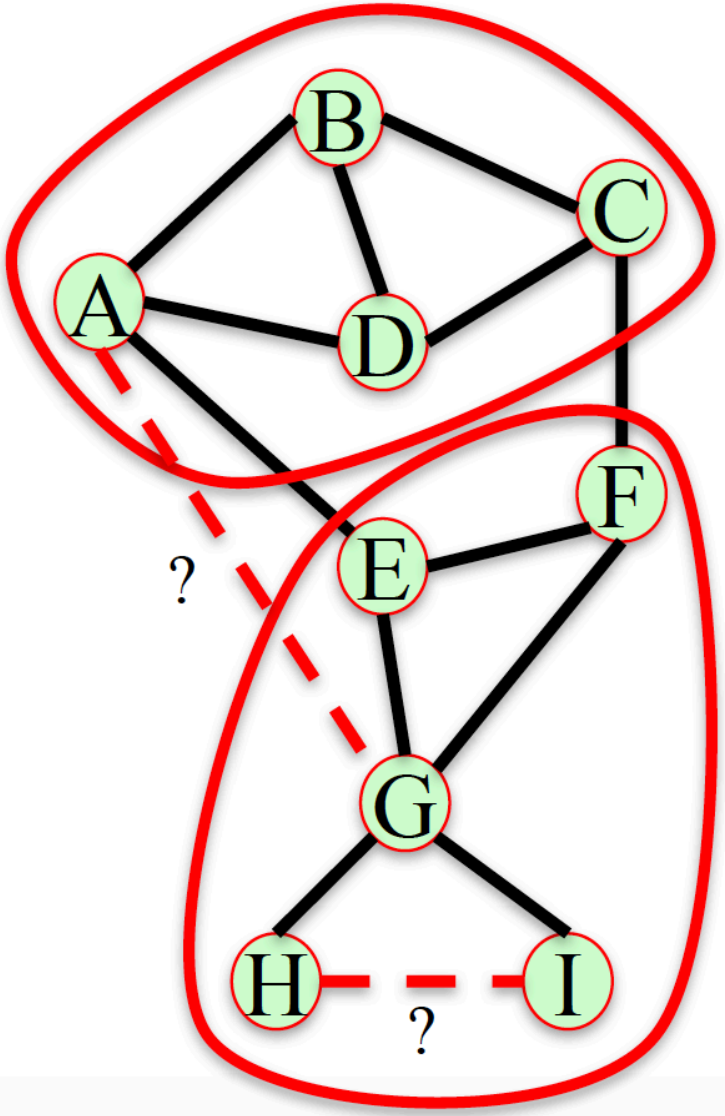
Número de vecinos comunes con puntos por ser vecinos en la misma comunidad.

$$cn\_soundarajan\_hopcroft(A, C) = 2 + 2 = 4$$
$$cn\_soundarajan\_hopcroft(E, I) = 1 + 1 = 2$$
$$cn\_soundarajan\_hopcroft(A, G) = 1 + 0 = 1$$



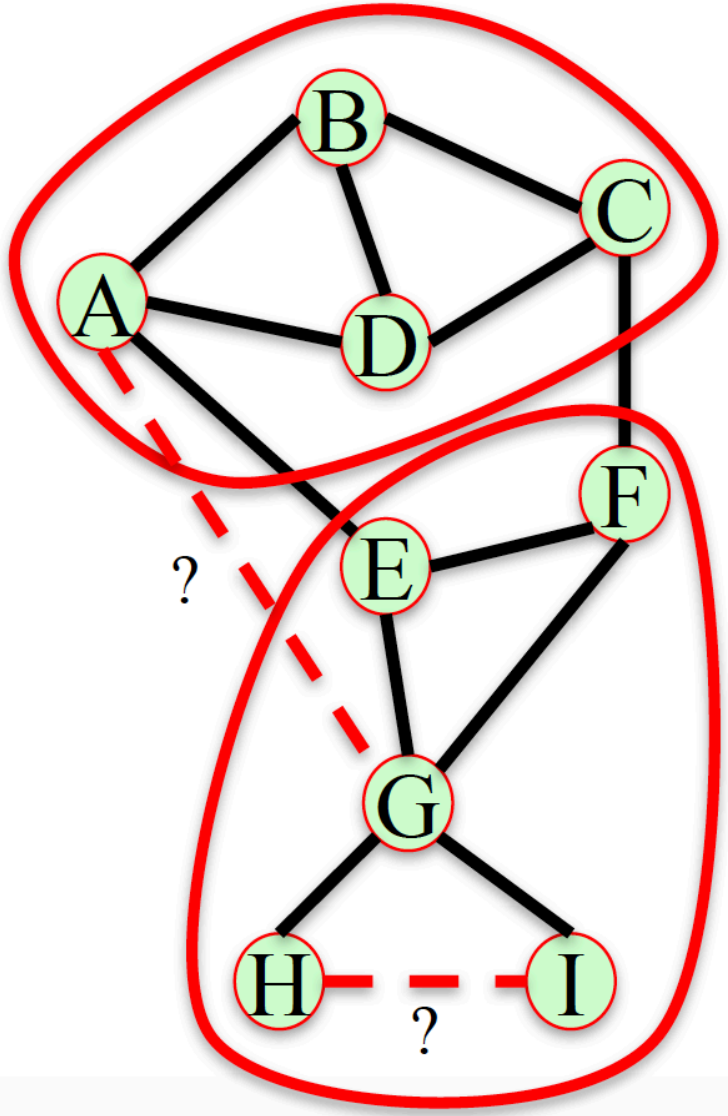
Medida 6: Vecinos comunes de la comunidad

```
#Asignar comunidades a los nodos con atributo “comunidad” de nodo
>>> G.node['A']['community'] = 0
>>> G.node['B']['community'] = 0
>>> G.node['C']['community'] = 0
>>> G.node['D']['community'] = 0
>>> G.node['E']['community'] = 1
>>> G.node['F']['community'] = 1
>>> G.node['G']['community'] = 1
>>> G.node['H']['community'] = 1
>>> G.node['I']['community'] = 1

>>> L = list(nx.cn_soundarajan_hopcroft(G))
>>> L.sort(key=operator.itemgetter(2), reverse = True)
>>> print(L)
```



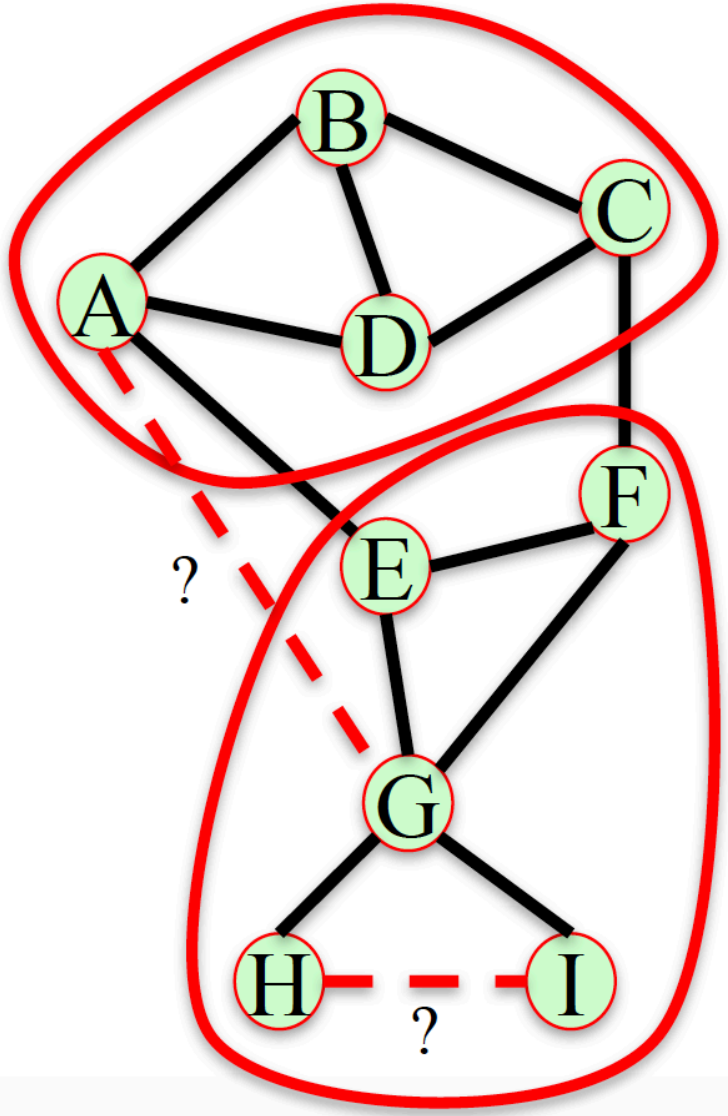
Medida 6: Vecinos comunes de la comunidad

Par	Medida vecinos comunidad
A,C	4
E,I	2
E,H	2
F,I	2
F,H	2
I,H	2
A,G	1
A,F	1
C,E	1
C,G	1
B,E	1
B,F	1
E,D	1



Medida 6: Vecinos comunes de la comunidad

Par	Medida vecinos comunidad
A,C	4
E,I	2
E,H	2
F,I	2
F,H	2
I,H	**2**
A,G	**1**
A,F	1
C,E	1
C,G	1
B,E	1
B,F	1
E,D	1



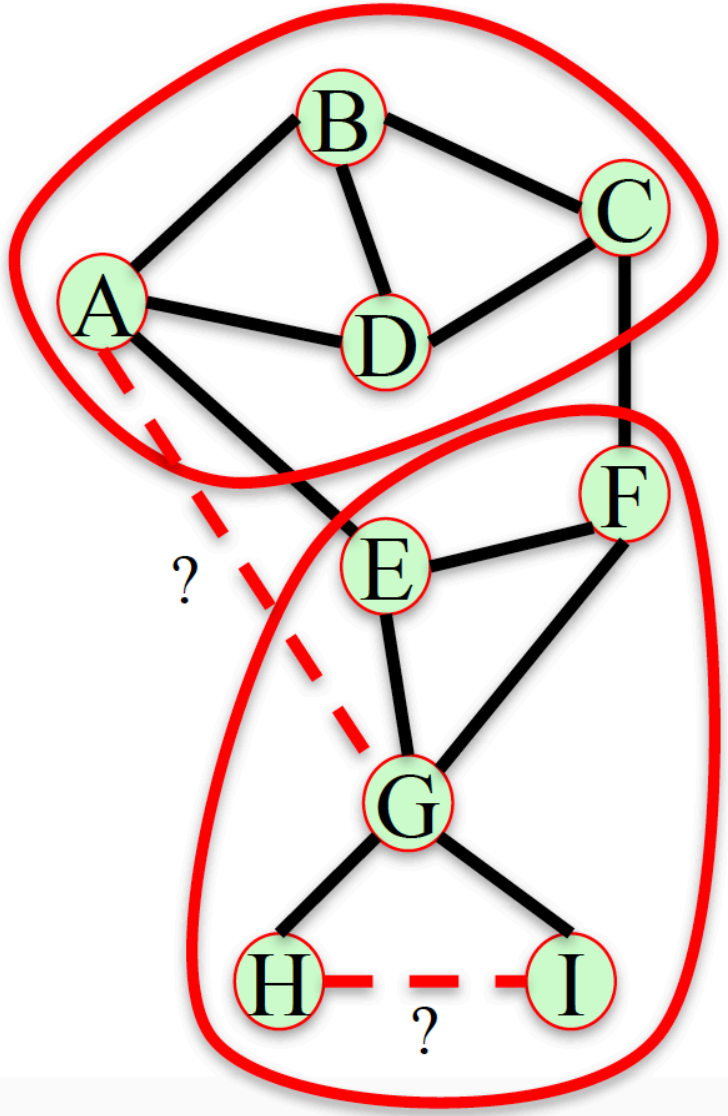
Medida 7: Asignación de recursos en comunidad

Similar al índice de asignación de recursos, pero solo considerando nodos en la misma comunidad.

El score de asignación de recursos Soundarajan-Hopcroft de dos nodos X, Y es:

$$ra\_soundarajan\_hopcroft(X,Y) = \sum_{u \in N(X) \cap N(Y)} \frac{f(u)}{|N(u)|}$$

$$donde\ f(u) = \begin{cases} 1, u\ en\ la\ misma\ comunidad\ X,Y \\ 0, en\ diferente\ comunidad \end{cases}$$



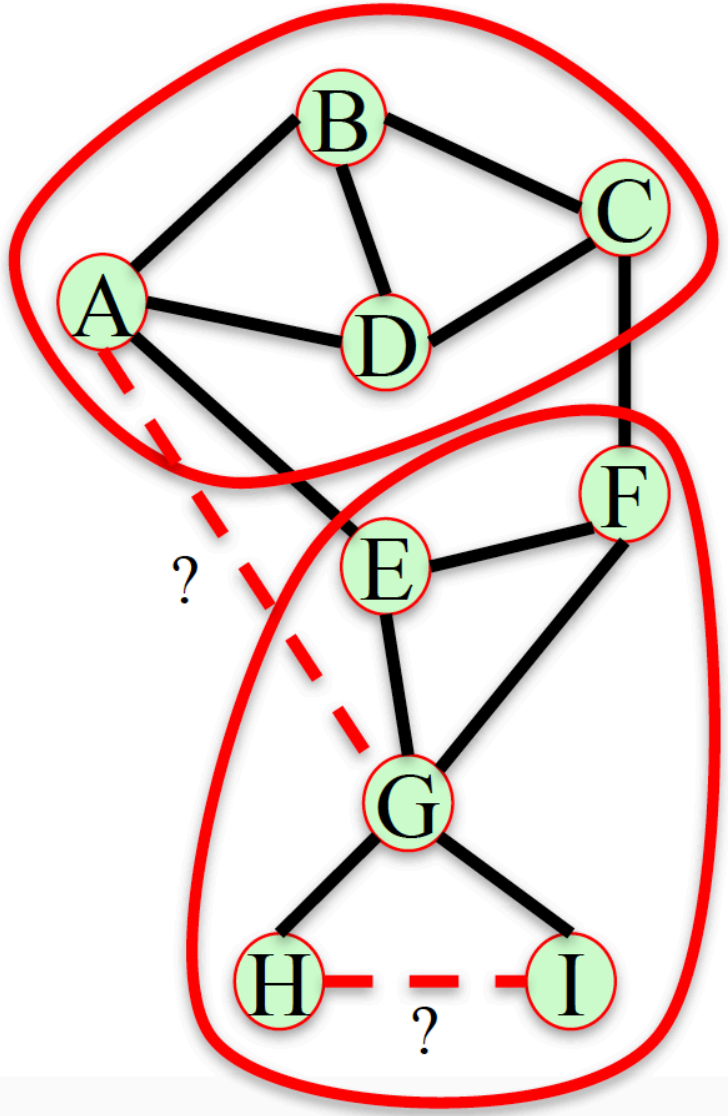
Medida 7: Asignación de recursos en comunidad

Similar al índice de asignación de recursos, pero solo considerando nodos en la misma comunidad.

$$ra\_soundarajan\_hopcroft(A, C) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$$

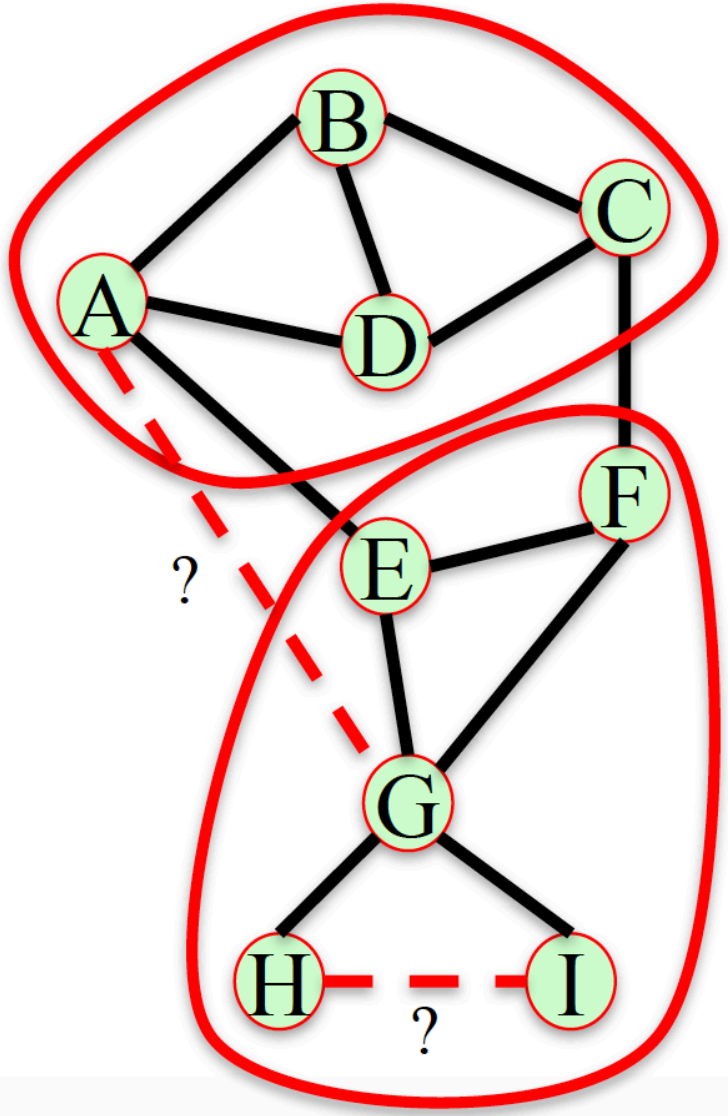
$$ra\_soundarajan\_hopcroft(E, I) = \frac{1}{4}$$

$$ra\_soundarajan\_hopcroft(A, G) = \frac{?}{—}$$



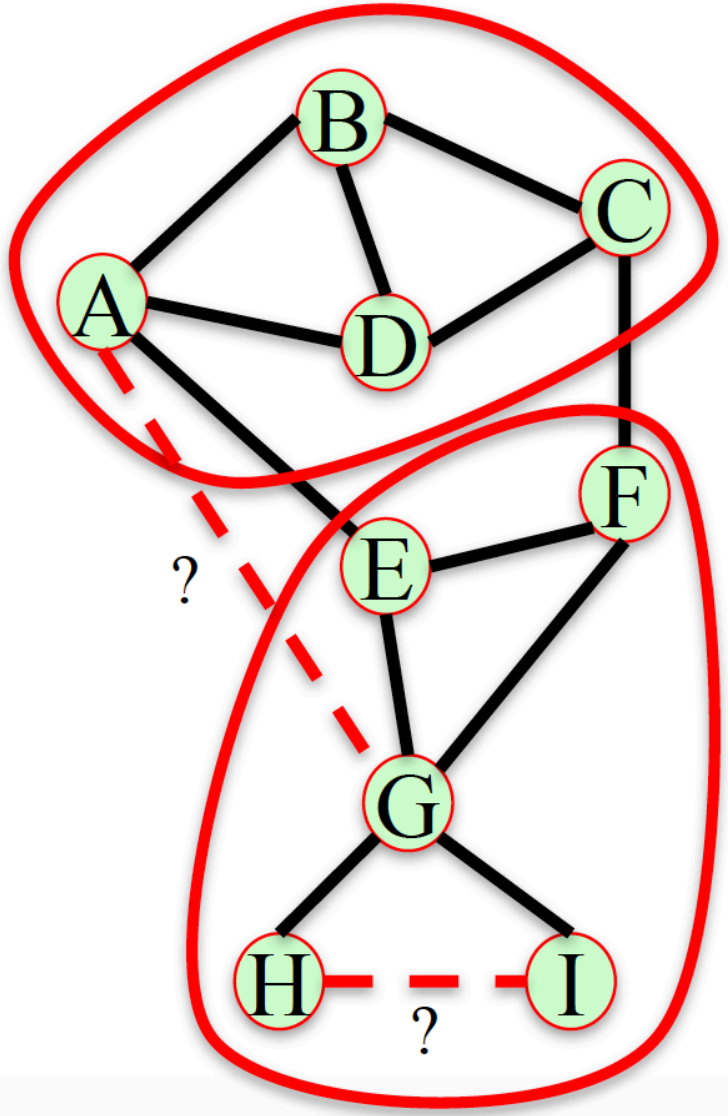
Medida 7: Asignación de recursos en comunidad

```
>>> L = list(nx.ra_index_soundarajan_hopcroft(G))
>>> L.sort(key=operator.itemgetter(2), reverse = True)
>>> print(L)
```



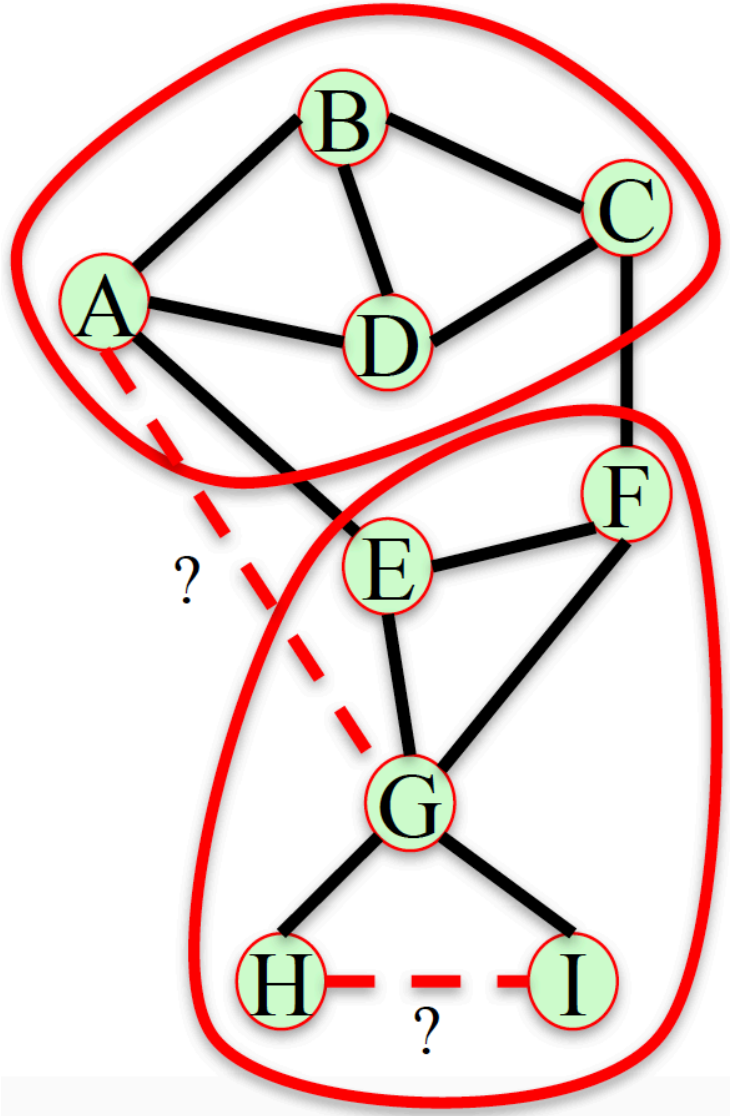
Medida 7: Asignación de recursos en comunidad

Par	Medida asign. recursos com
A,C	0.666666
E,I	0.25
E,H	0.25
F,I	0.25
F,H	0.25
I,H	0.25
A,G	0
A,F	0
C,E	0
C,G	0
B,E	0
B,F	0
E,D	0



Medida 7: Asignación de recursos en comunidad

Par	Medida asign. recursos com
A,C	0.666666
E,I	0.25
E,H	0.25
F,I	0.25
F,H	0.25
I,H	**0.25**
A,G	**0**
A,F	0
C,E	0
C,G	0
B,E	0
B,F	0
E,D	0



Resumen

5 medidas básicas:

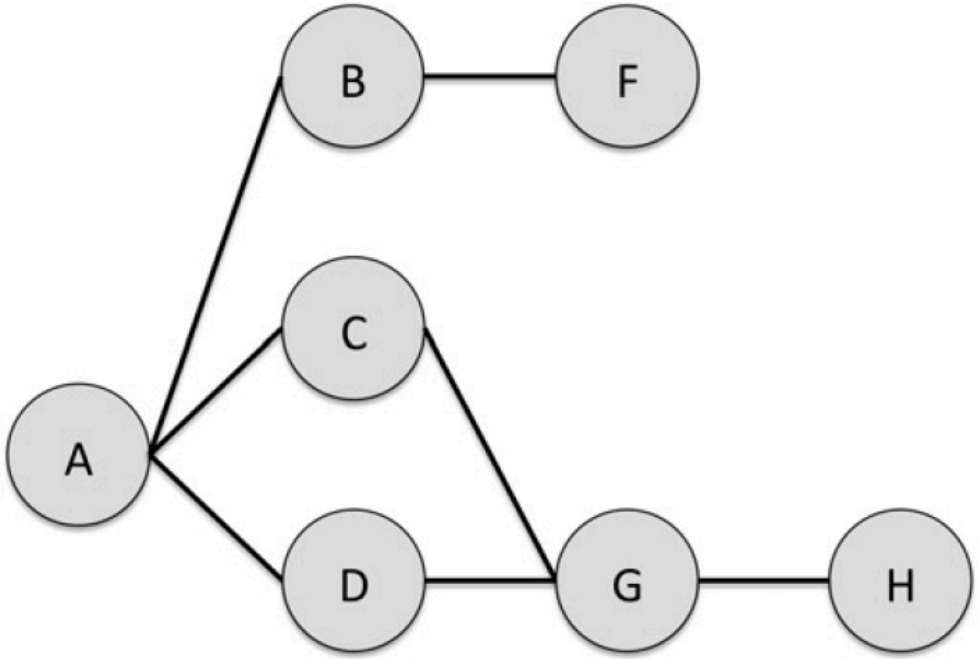
- Número de vecinos comunes
- Coeficiente de Jaccard
- Índice de asignación de recursos
- Índice Adamic-Adar
- Score de unión preferencial

2 medidas que requieren información comunitaria:

- Score de Soundarajan-Hopcroft del vecino común
- Score de asignación de recursos Soundarajan-Hopcroft

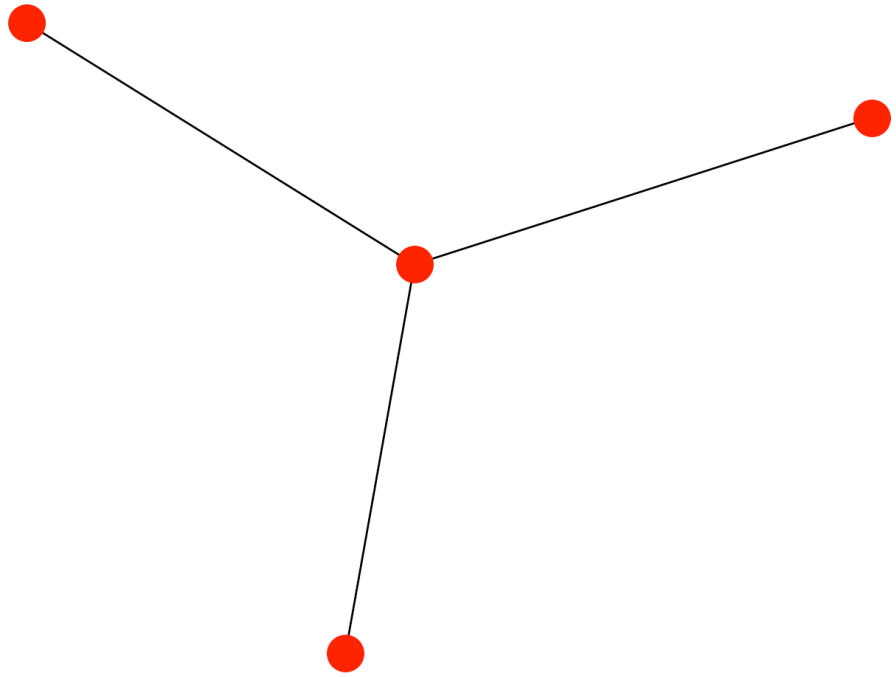
Ejercicio – Calcular predicción de links

- Usar las 5 medidas básicas.



Visualización con Matplotlib

```
>>> import networkx as nx
>>> import matplotlib.pyplot as plt
>>> edges=[['A','B'],['B','C'],['B','D']]
>>> G=nx.Graph()
>>> G.add_edges_from(edges)
>>> nx.draw(G)
>>> plt.show()
```



Visualización: etiquetas en las aristas.

```
import networkx as nx
import matplotlib.pyplot as plt


edges=[['A','B'],['B','C'],['B','D']]
G=nx.Graph()
G.add_edges_from(edges)
pos = nx.random/spring_layout(G)
labels={('A','B'):'AB',('B','C'):'BC',('B','D'):'BD'}
LAS ETIQUETAS SE PASAN EN FORMA DE DICCIONARIO
nx.draw(G,pos,edge_color='black',width=1,linewidths=1,node_size=500,node_color='pink',with_labels=True)
nx.draw_networkx_edge_labels(G,pos,labels)
Y SE DIBUJAN A PARTE
plt.show()
```