

Medidas estadísticas y conectividad de las redes

M.C. Jorge Carlos Reyes Magaña
Posgrado en Ciencia e Ingeniería de la Computación

*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

Recursos

Networkx

<https://networkx.github.io/documentation/stable/index.html>

Documentación networkx

<https://networkx.github.io/documentation/stable/reference/index.html>

Materiales del curso

<https://github.com/helenpy/TallerRedes>

Estadísticas de grafos

- Proporcionan una gran cantidad de información.
- Algunas estadísticas de alto nivel responderán preguntas sobre el tamaño, la densidad y la cantidad de grafos por separado (componentes).
- ¿Qué estadísticas de los grafos son relevantes?
 - Depende, en parte, del objetivo del estudio.
 - Algunas de las estadísticas de grafos más simples pueden ser la densidad, el grado y la centralidad.

Tamaño

- Número de nodos y número de aristas.
- Se utilizan para establecer la metodología de trabajo.
 - Los grafos con cientos o cientos de miles de aristas pueden procesarse en una computadora local, mientras que los grafos con millones de aristas requieren el uso de múltiples computadoras.
 - Los grafos con miles de aristas se pueden visualizar directamente, mientras que los grafos más grandes requieren una estrategia para seleccionar y filtrar a subconjuntos más pequeños del grafo.
- Proporcionan una validación rápida sobre la importación correcta de los datos del grafo en el software de visualización y análisis.

Tamaño

```
import networkx as nx

G = nx.Graph()
G.add_edge('a', 'b', weight=2)
G.add_edge('b', 'c', weight=4)
G.size() #número de lados
G.size(weight='weight') #suma de los pesos
G.number_of_nodes() #número de nodos
```

Densidad

- La densidad del grafo es la relación entre las aristas reales y las aristas máximas posibles. Un grafo **totalmente conectado** (es decir, un grafo denso) tendrá todos los nodos conectados a todos los demás nodos; para n nodos, el grafo máximo se acercará a n^2 aristas para un grafo dirigido (o $1/2 n^2$ nodos para un grafo no dirigido).
 - Función `nx.density(G)`
 - Grafos densos
 - Grafos dispersos

Grados

- El **grado del nodo** es el número de aristas que se conectan a un nodo determinado.

G.degree()

G.degree['a']

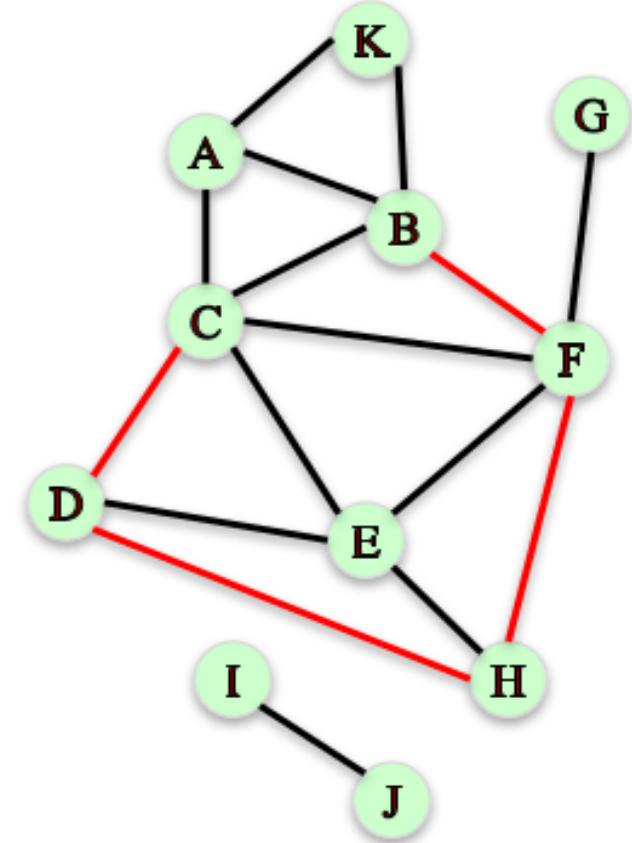
G.degree(['a','b'])

- Un nodo con grado cero es un nodo sin aristas, es un nodo aislado.
- Un nodo con grado uno tiene solo una sola arista se denomina nodo hoja.
- El **grado promedio** es simplemente un promedio del grado de todos los nodos. Cuando el grado promedio es alto, cada nodo tiene una conectividad alta, el grafo está conectado densamente con un gran número de aristas.
- El nodo con el **grado máximo** es el que tiene más nodos conectados.

Conectividad de las redes

Clausura triádica (triadic closure)

- Es la tendencia de las personas que comparten conexiones en una red social a conectarse.
- ¿Cómo podemos medir la prevalencia de clausura triádica en una red?

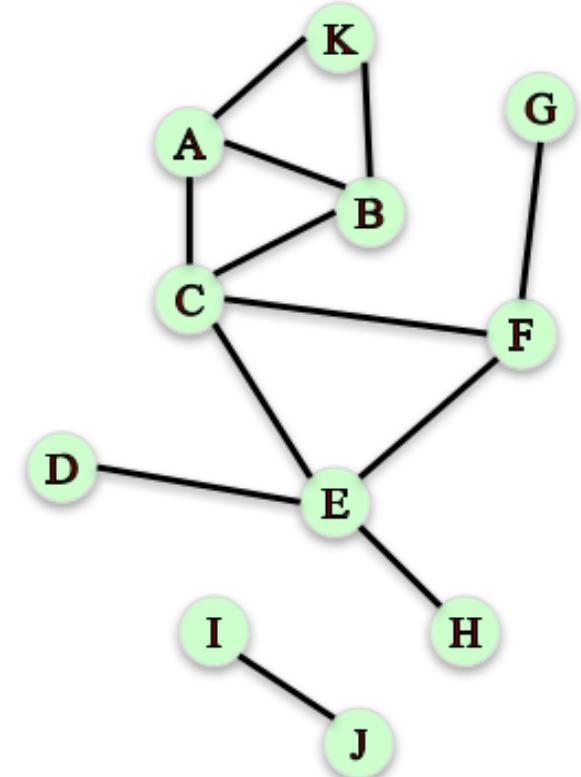


Coeficiente de Agrupamiento local

- Cuantifica qué tan cerca están sus vecinos de ser un clique.
- Es la fracción de pares de nodos adyacentes al nodo que son adyacentes entre sí.

Calcular el coeficiente de agrupamiento local del nodo C:

$$\frac{\text{\# de pares de nodos adyacentes a C que son adyacentes}}{\text{\# de pares adyacentes a C}}$$

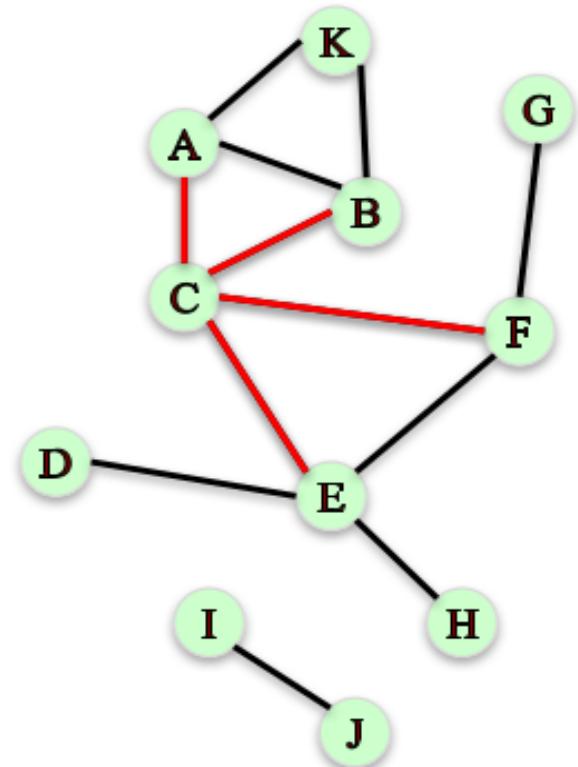


Coeficiente de Agrupamiento local

de pares de nodos adyacentes a C que son adyacentes

de pares adyacentes a C

de nodos adyacentes a C = $d_c = 4$ (el “grado” de C)



Coeficiente de Agrupamiento local

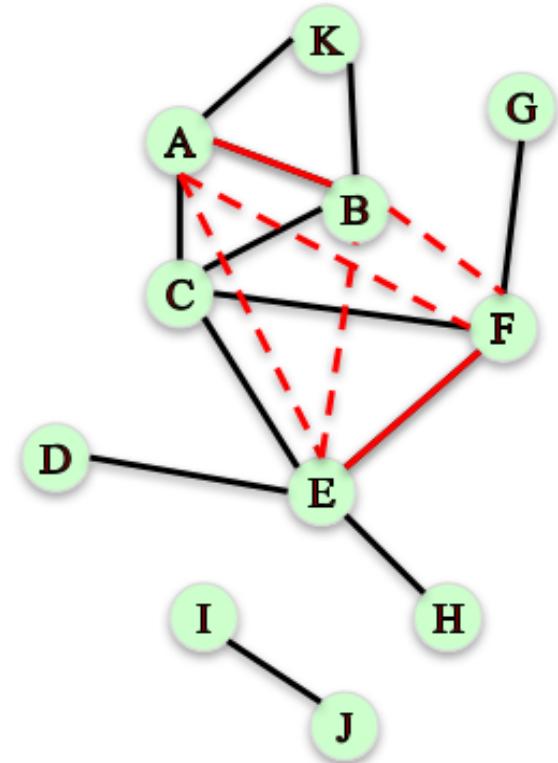
de pares de nodos adyacentes a C que son adyacentes

de pares adyacentes a C

de nodos adyacentes a C = $d_c = 4$ (el “grado” de C)

numero de pares de nodos adyacentes a C =

$$d_c (d_c - 1) / 2$$



Coeficiente de Agrupamiento local

de pares de nodos adyacentes a C que son adyacentes

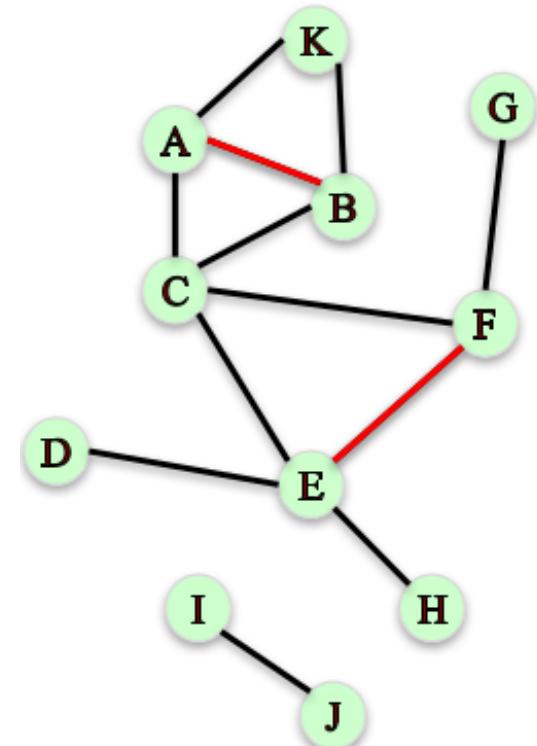
de pares adyacentes a C

de nodos adyacentes a C = $d_c = 4$ (el “grado” de C)

numero de pares de nodos adyacentes a C =

$$d_c (d_c - 1) / 2 = 12 / 2 = 6$$

numero de pares de nodos adyacentes a C que son adyacentes



Coeficiente de Agrupamiento local

de pares de nodos adyacentes a C que son adyacentes

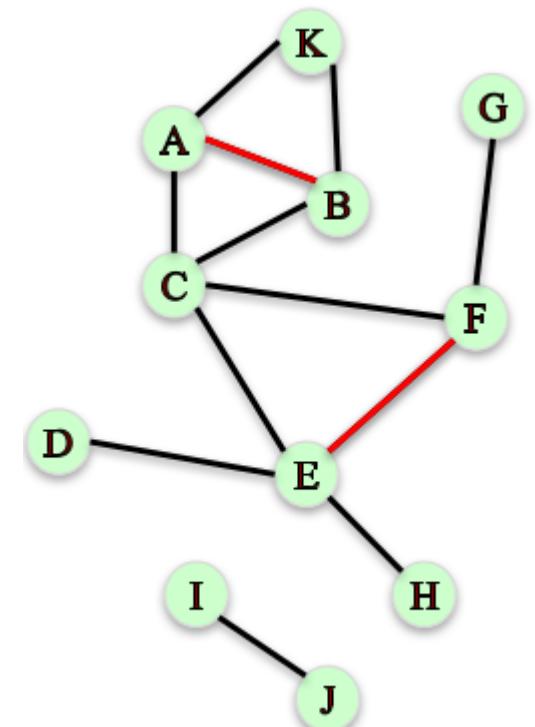
de pares adyacentes a C

de nodos adyacentes a C = $d_c = 4$ (el “grado” de C)

numero de pares de nodos adyacentes a C =

$$d_c (d_c - 1) / 2 = 12 / 2 = 6$$

numero de pares de nodos adyacentes a C que son adyacentes = 2



Coeficiente de Agrupamiento local

$$\frac{\text{# de pares de nodos adyacentes a } C \text{ que son adyacentes}}{\text{# de pares adyacentes a } C}$$

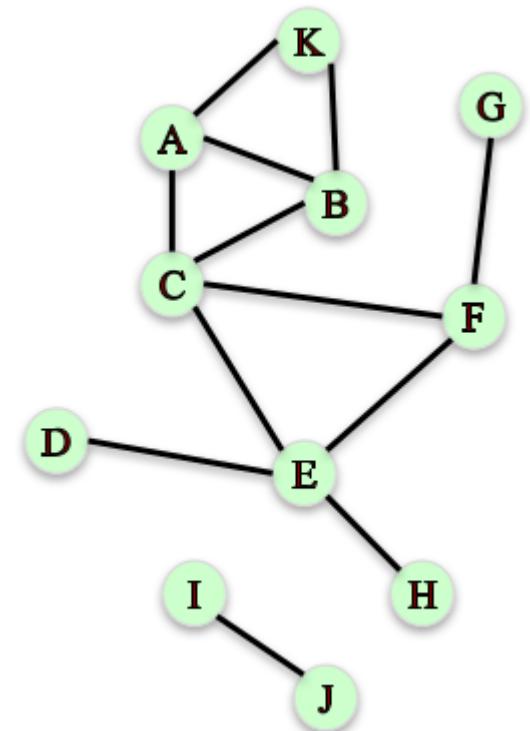
de nodos adyacentes a $C = d_c = 4$ (el “grado” de C)

numero de pares de nodos adyacentes a $C =$

$$d_c (d_c - 1) / 2 = 12 / 2 = 6$$

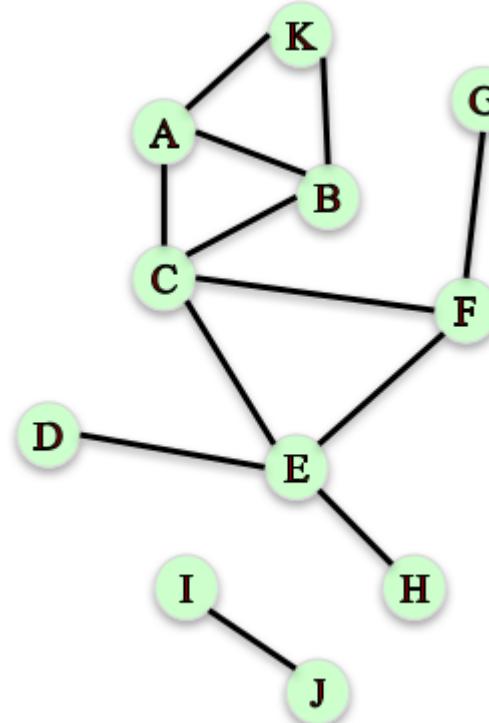
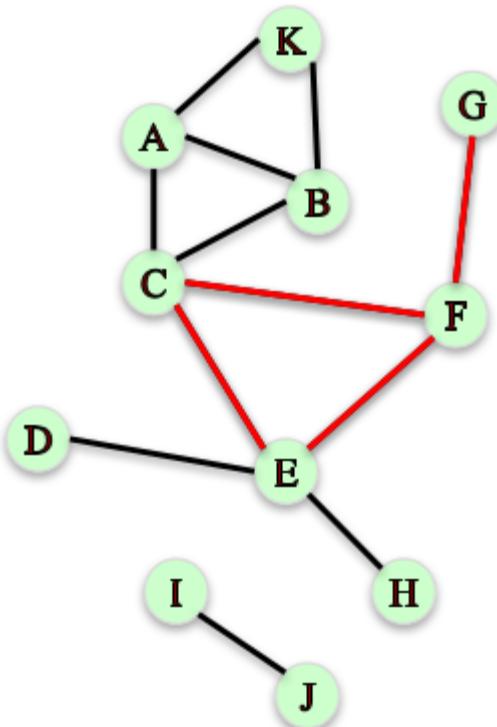
numero de pares de nodos adyacentes a C que son adyacentes = 2

Coeficiente de agrupamiento local de $C = 2/6 = 1/3$



Coeficiente de Agrupamiento local

- Calcular el coeficiente de agrupamiento local del nodo F.
- Calcular el coeficiente de agrupamiento local del nodo J.



Coeficiente de Agrupamiento local

de pares de nodos adyacentes a F que son adyacentes

de pares adyacentes a F

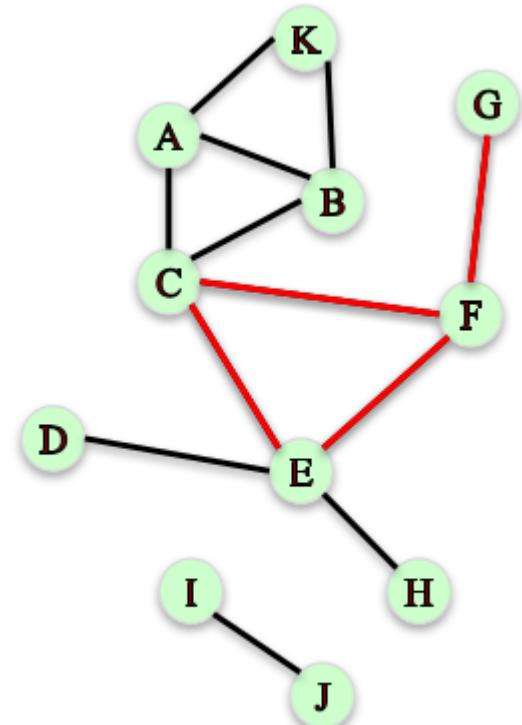
de nodos adyacentes a F = $d_f = 3$ (el “grado” de F)

numero de pares de nodos adyacentes a F =

$$d_f (d_f - 1) / 2 = 6 / 2 = 3$$

numero de pares de nodos adyacentes a F que son adyacentes = 1

Coeficiente de agrupamiento local de F = 1/3



Coeficiente de Agrupamiento local

de pares de nodos adyacentes a J que son adyacentes

de pares adyacentes a J

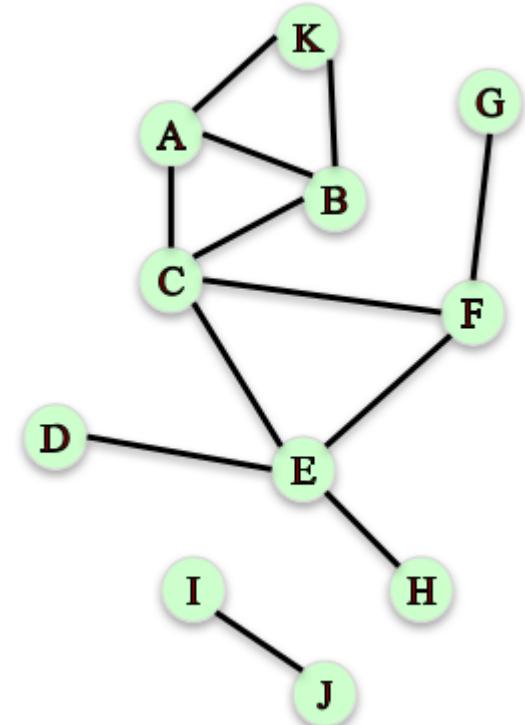
de nodos adyacentes a J = $d_j = 1$ (el “grado” de J)

numero de pares de nodos adyacentes a J =

$$d_j (d_j - 1) / 2 = 0$$

No se puede dividir entre 0.

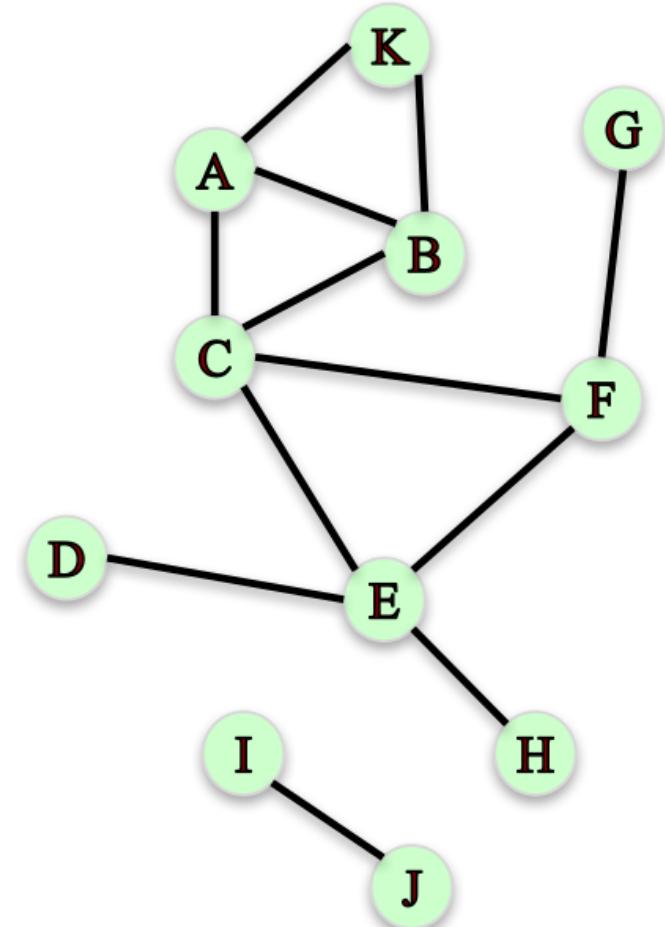
Supondremos que el coeficiente de agrupamiento local de un nodo de grado inferior a 2 sea 0.



Coeficiente de Agrupamiento local

```
G = nx.Graph()  
G.add_edges_from([('A', 'K'), ('A', 'B'), ('A', 'C'),  
('B', 'C'), ('B', 'K'), ('C', 'E'), ('C', 'F'), ('D', 'E'), ('E',  
'F'), ('E', 'H'), ('F', 'G'), ('I', 'J')])
```

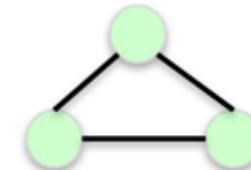
```
nx.clustering(G, 'F')  
nx.clustering(G, 'A')  
nx.clustering(G, 'J')
```



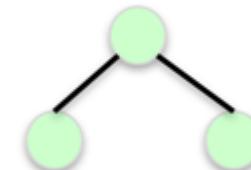
Coeficiente de Agrupamiento global

- Se mide el agrupamiento en el grafo completo.
- **Enfoque 1:** Promedio del coeficiente de agrupamiento local en todos los nodos del grafo. `nx.average_clustering(G)`
- **Enfoque 2:** Porcentaje de tripletas que son triángulos en el grafo. `nx.transitivity(G)`

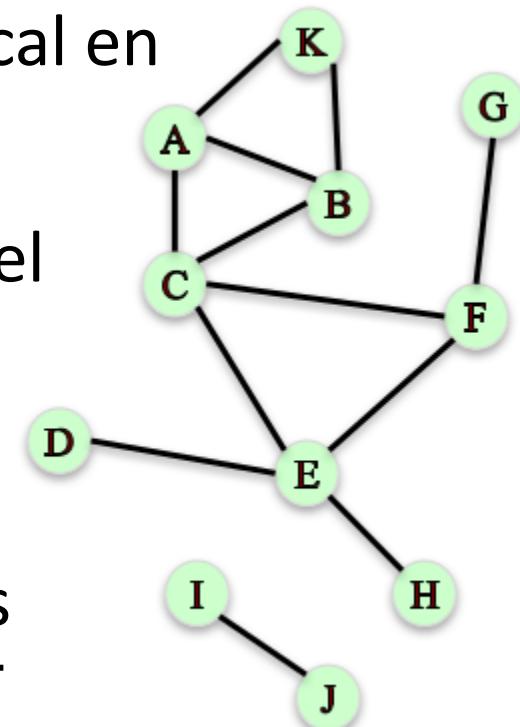
Triángulos



Tripletas

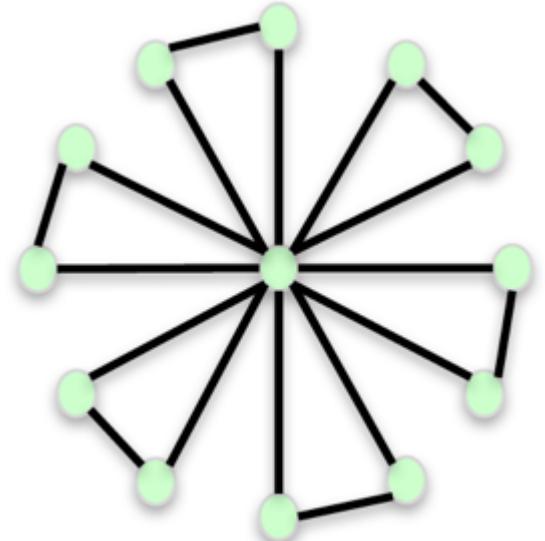


$$\text{Transitividad: } \frac{3 * \# \text{ de triángulos}}{\# \text{ de tripletas}}$$



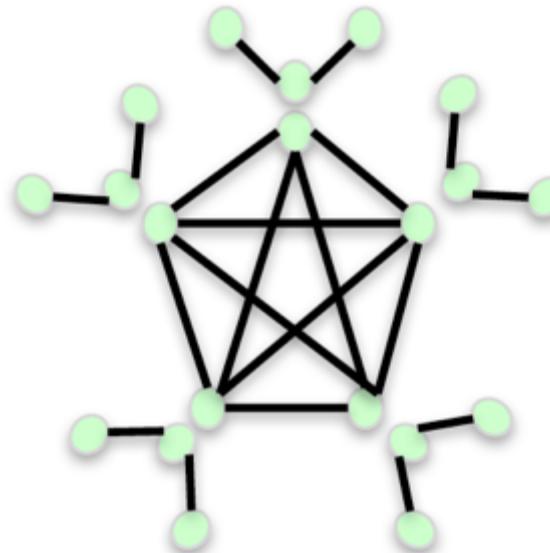
Transitividad vs. Promedio de coeficiente de agrupamiento

- Ambos miden la tendencia de las aristas a formar triángulos.
- La transitividad pesa los nodos con grados mayores más alto.



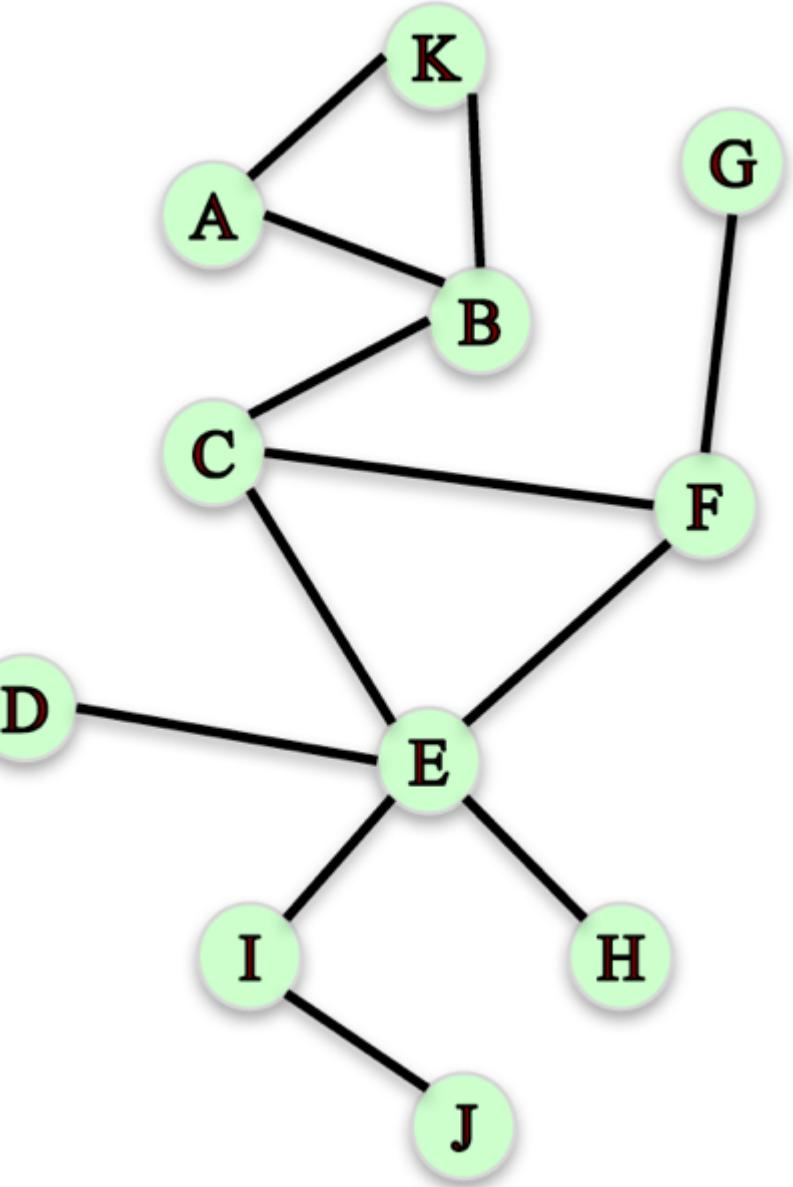
Prom. Coef. Agrup. = 0.93
Transitividad = 0.23

La mayoría de los nodos tienen LCC alto
Los nodos de alto grado tienen bajo LCC



Prom. Coef. Agrup. = 0.25
Transitividad = 0.86

La mayoría de los nodos tienen bajo LCC
Los nodo de alto grado tiene alto LCC



Distancia

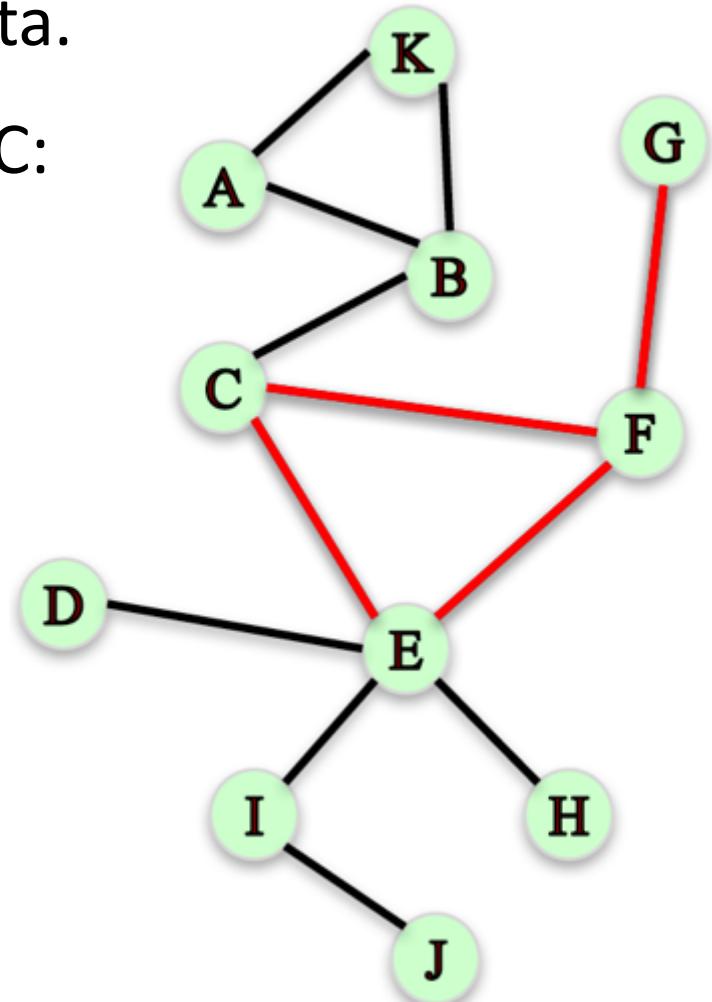
- ¿Qué tan lejos está el nodo A del nodo H?
- ¿Están los nodos lejos o cerca unos de otros en este grafo?
- ¿Qué nodos son "más cercanos" y "más lejanos" a otros nodos?
- Necesitamos un sentido de distancia entre los nodos para responder estas preguntas.

Caminos (Paths)

- Una secuencia de nodos conectados por una arista.
- Encuentre dos caminos desde el nodo G al nodo C:

G – F – C

G – F – E – C



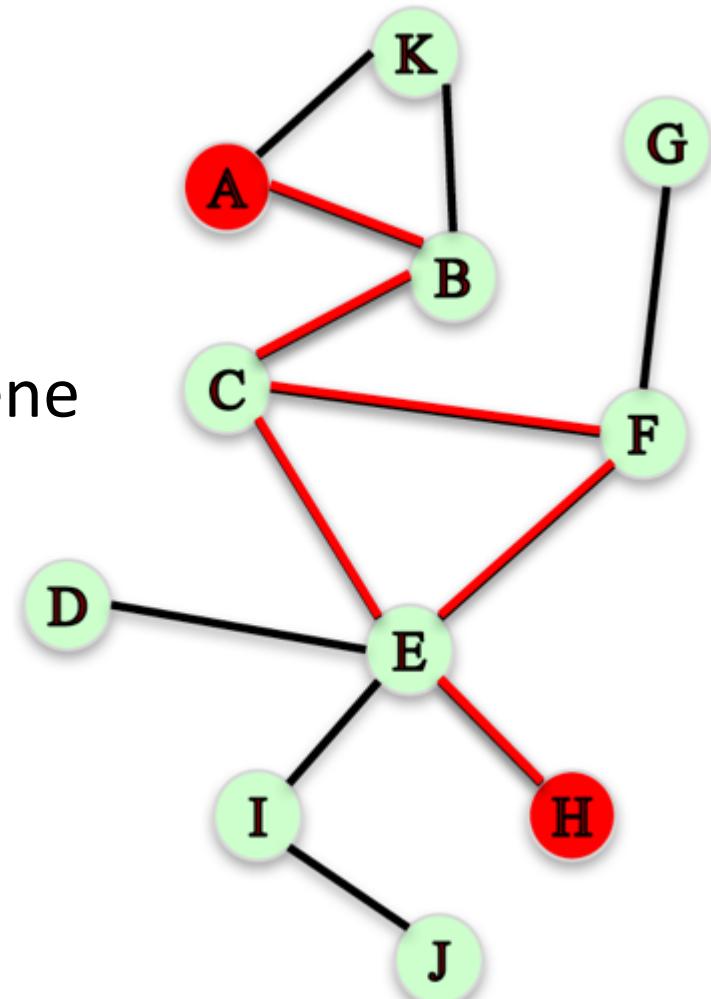
Distancia

- ¿Qué tan lejos está el nodo A del nodo H?

Camino 1: A - B - C - E - H (4 “saltos”)

Camino 2: A - B - C - F - E - H (5 “saltos”)

- Longitud del camino: Número de pasos que contiene de principio a fin.
 - El camino 1 tiene longitud 4
 - El camino 2 tiene longitud 5

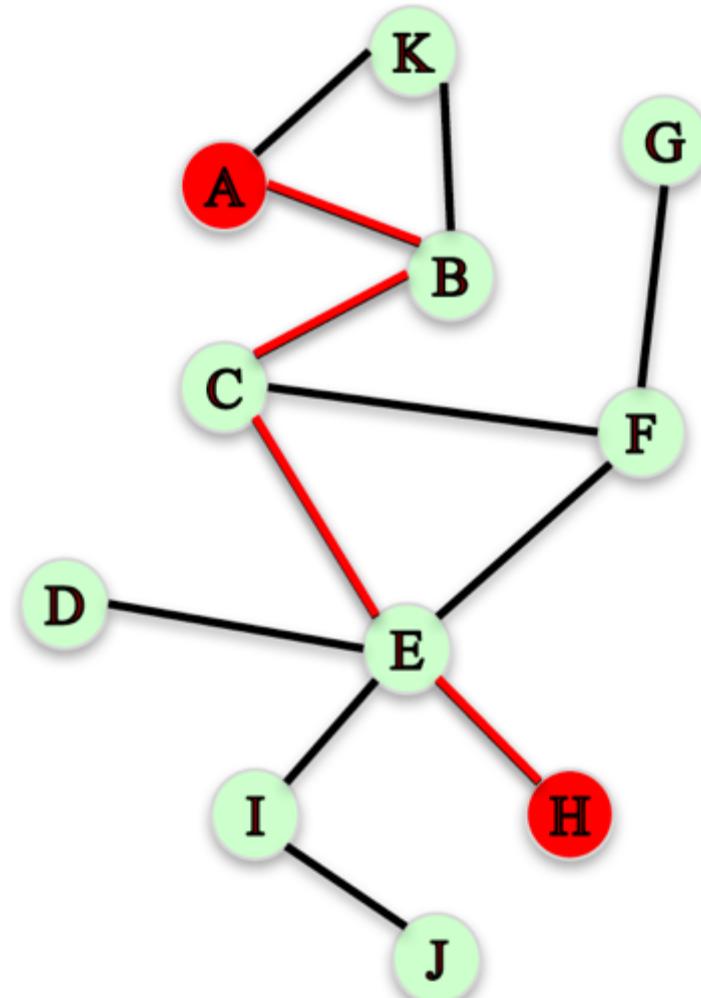


Distancia

- **Distancia entre dos nodos:** la longitud del camino más corto entre ellos.
- La distancia entre los nodos A y H es 4.

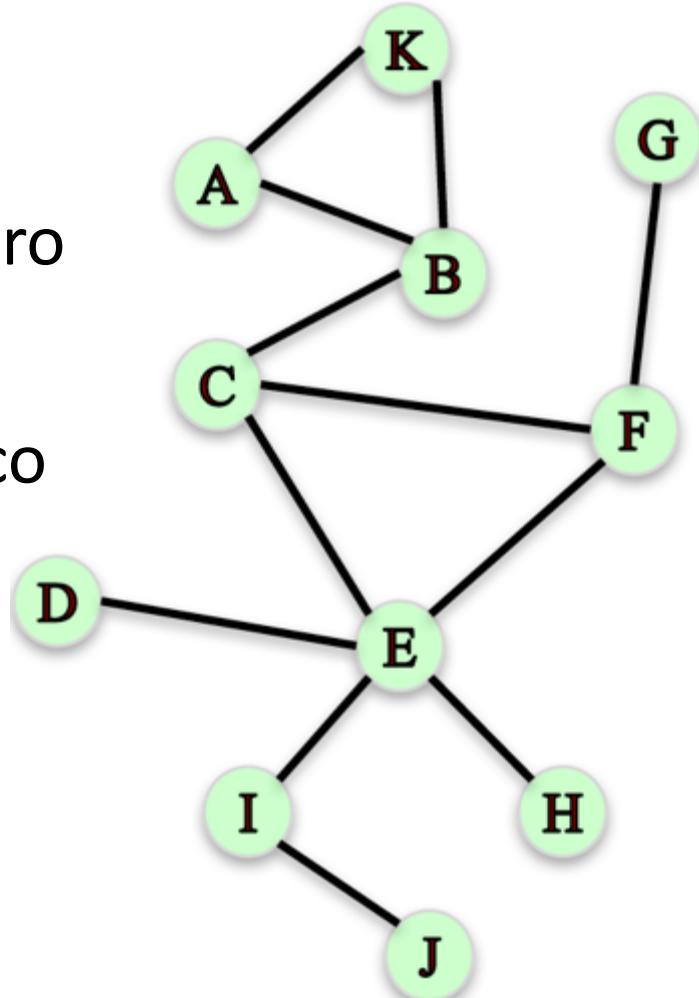
```
nx.shortest_path(G,'A', 'H')
```

```
nx.shortest_path_length(G,'A', 'H')
```

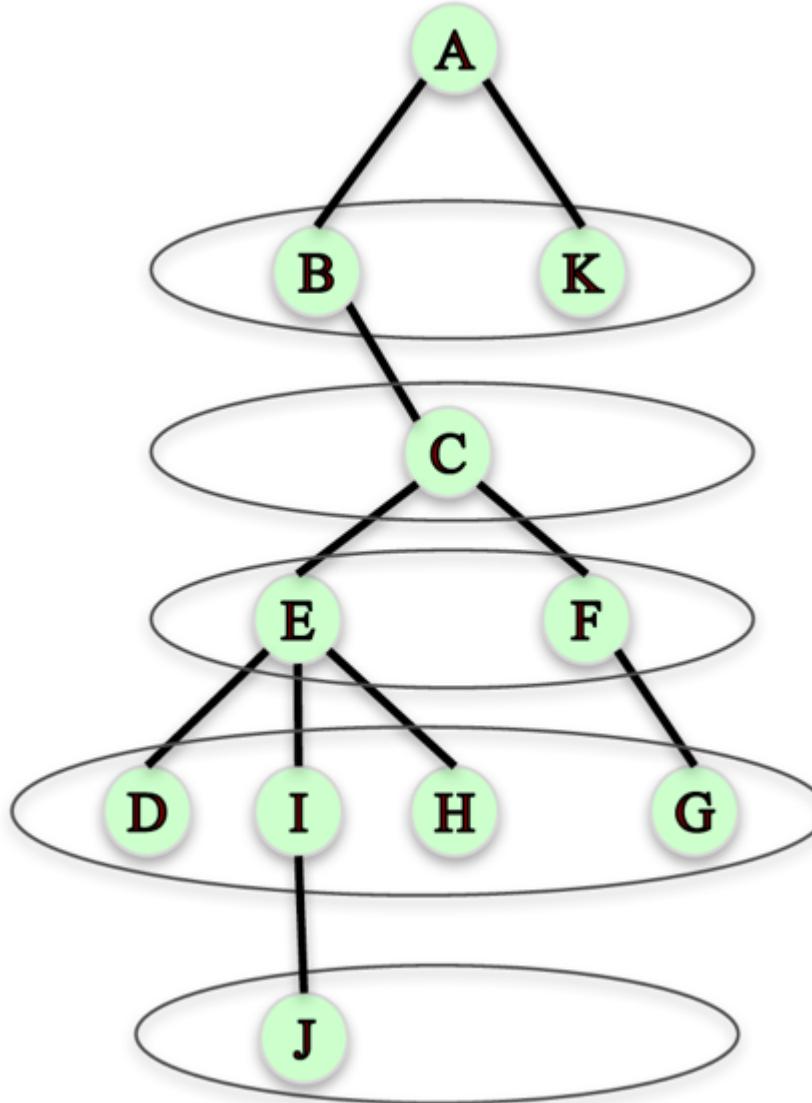
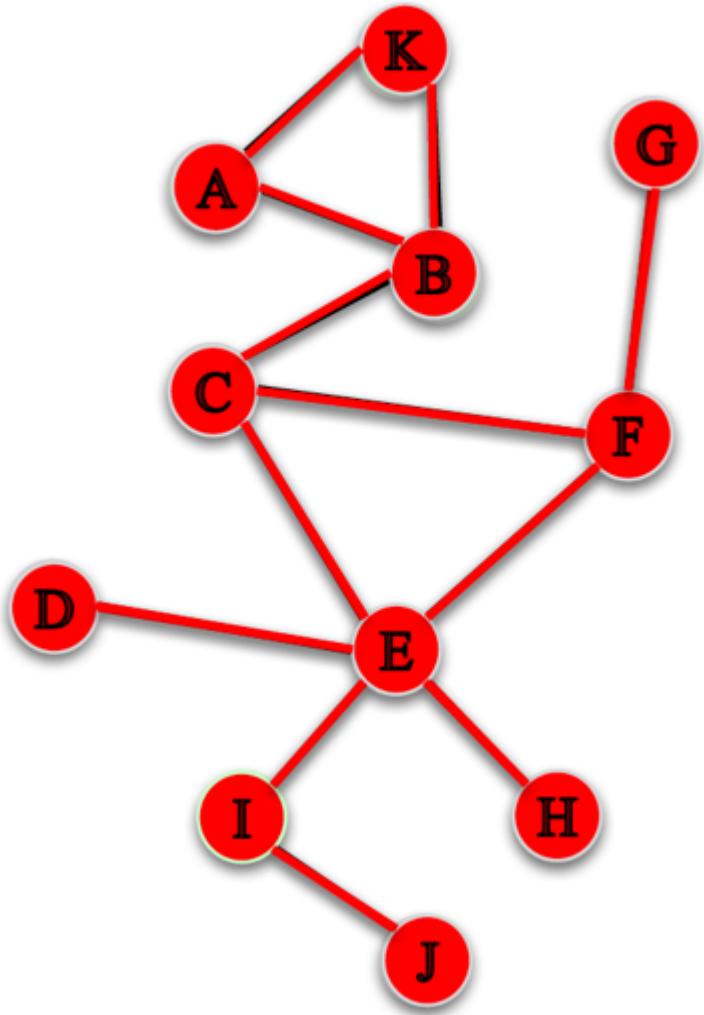


Distancia

- Encontrar la distancia desde el nodo **A** a cualquier otro nodo.
- Fácil de hacer manualmente en grafos pequeños pero tedioso en grafos grandes (reales).
- **Búsqueda en anchura:** un procedimiento sistemático y eficiente para calcular distancias desde un nodo a todos los demás nodos en una red grande "descubriendo" nodos en capas.



Búsqueda en anchura (Breadth-first search)



Distancia 1

Distancia 2

Distancia 3

Distancia 4

Distancia 5

Búsqueda en anchura (Breadth-first search)

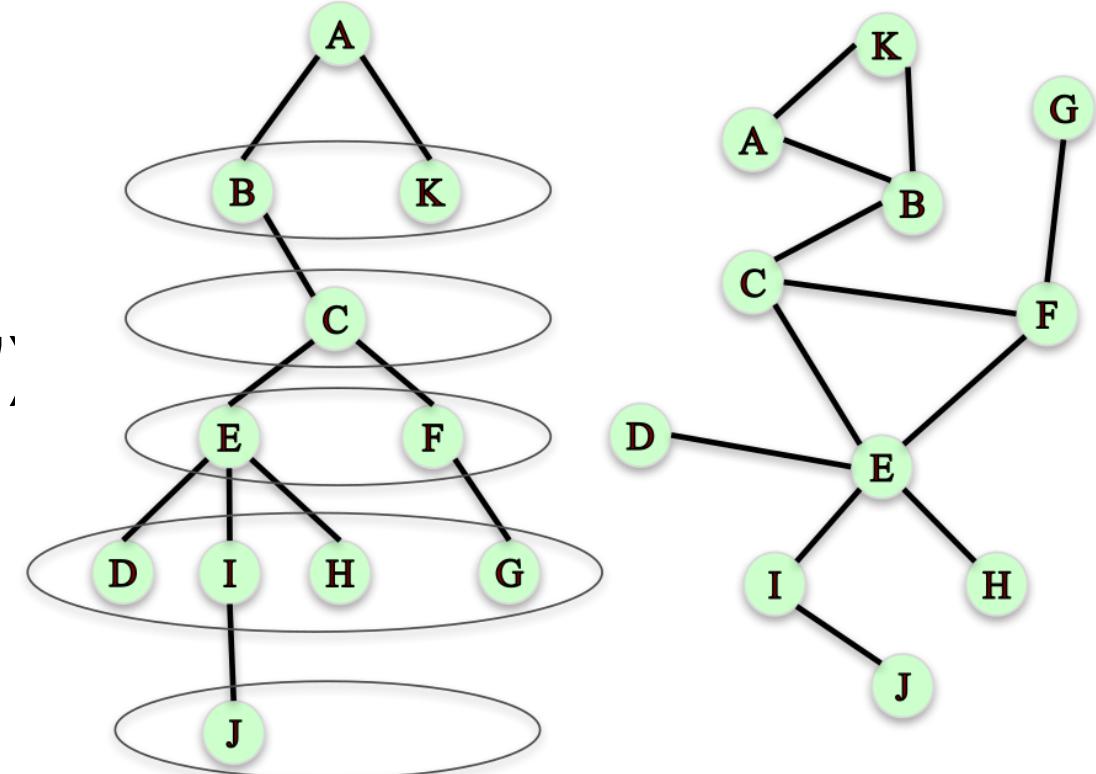
```
In: T = nx.bfs_tree(G, 'A')
```

```
In: T.edges()
```

```
Out: [('A', 'K'), ('A', 'B'), ('B', 'C'), ('C', 'E'), ('C', 'F'), ('E', 'I'), ('E', 'H'), ('E', 'D'), ('F', 'G'), ('I', 'J')]
```

```
In: nx.shortest_path_length(G,'A')
```

```
Out: {'A': 0, 'B': 1, 'C': 2, 'D': 4, 'E': 3, 'F': 3, 'G': 4, 'H': 4, 'I': 4, 'J': 5, 'K': 1}
```



Medidas de distancia

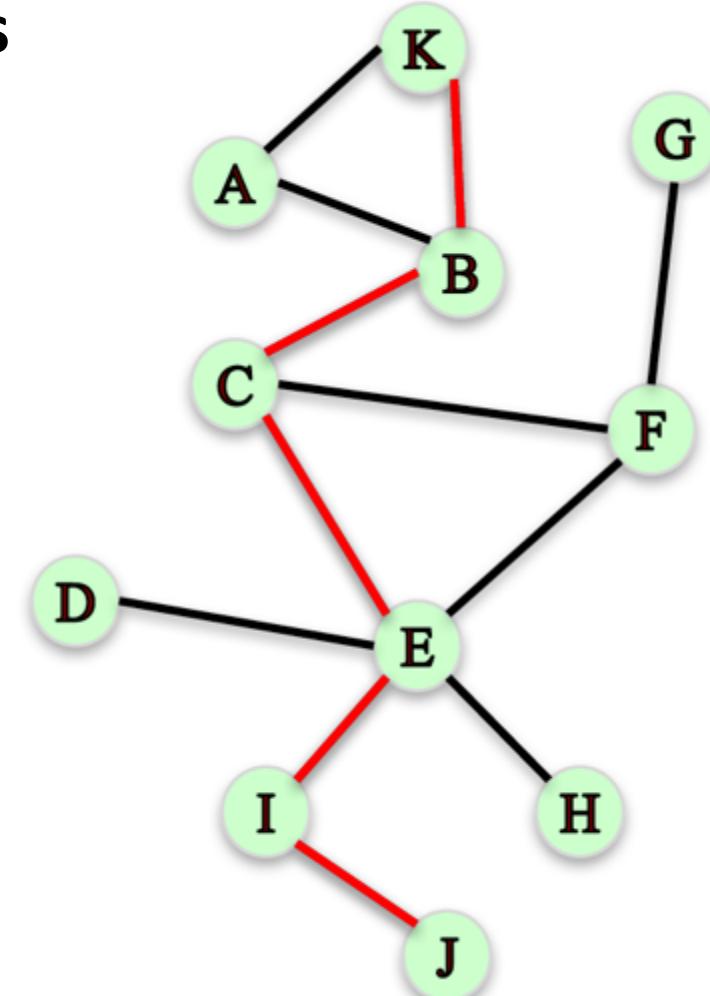
¿Cómo caracterizar la distancia entre todos los pares de nodos en un grafos?

Distancia promedio entre cada par de nodos.

```
nx.average_shortest_path_length(G)
```

2.52727272727

Diámetro: distancia máxima entre cualquier par de nodos. `nx.diameter(G)`



Medidas de distancia

¿Cómo caracterizar la distancia entre todos los pares de nodos en un grafos?

La **excentricidad** de un nodo n es la distancia más grande entre n y todos los demás nodos.

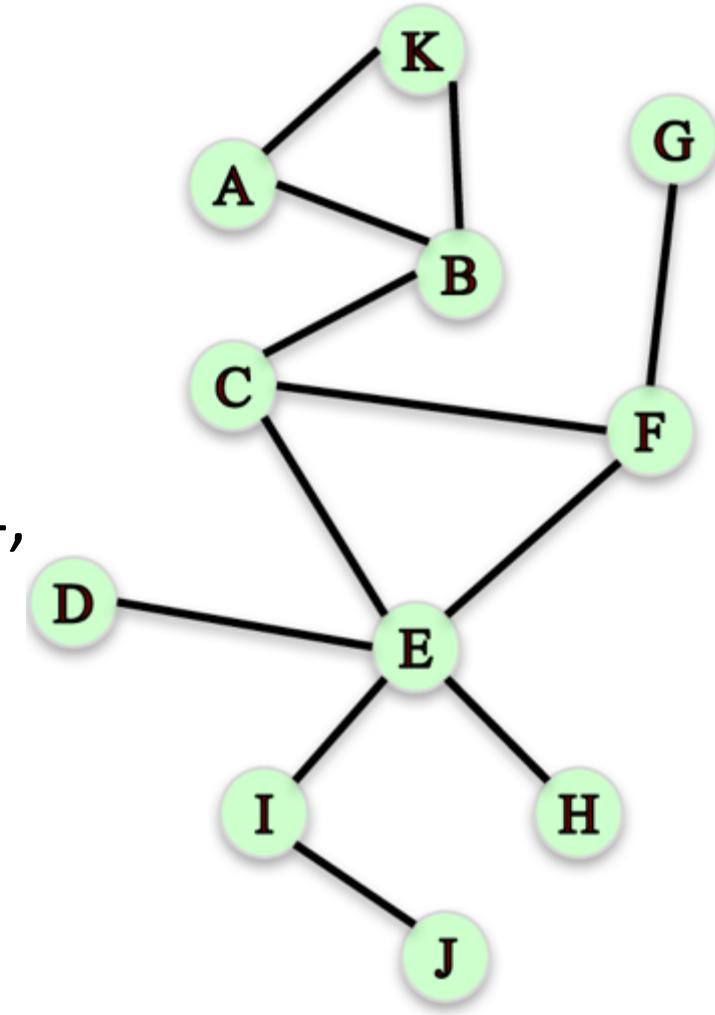
In: `nx.eccentricity(G)`

Out: {'A': 5, 'B': 4, 'C': 3, 'D': 4, 'E': 3, 'F': 3, 'G': 4, 'H': 4, 'I': 4, 'J': 5, 'K': 5}

El **radio** de un grafo es la excentricidad mínima.

In: `nx.radius(G)`

Out: 3



Medidas de distancia

¿Cómo caracterizar la distancia entre todos los pares de nodos en un grafos?

La **periferia** de un grafo es el conjunto de nodos que tienen excentricidad igual al diámetro.

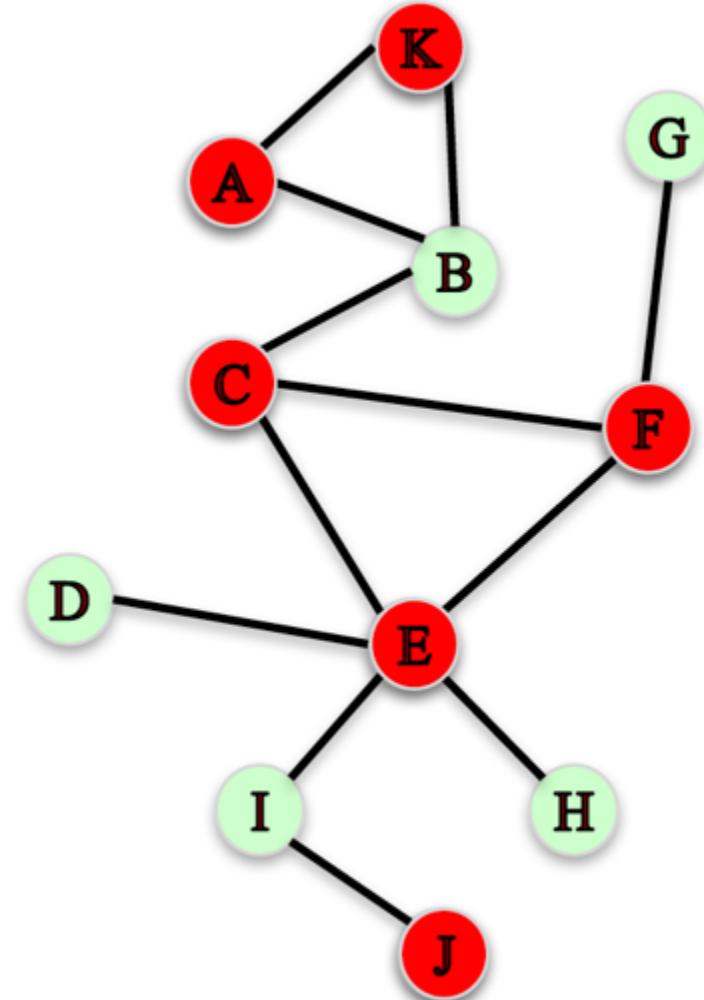
`nx.periphery(G)`

`['A', 'K', 'J']`

El **centro** de un grafo es el conjunto de nodos que tienen excentricidad igual al radio.

`nx.center(G)`

`['C', 'E', 'F']`



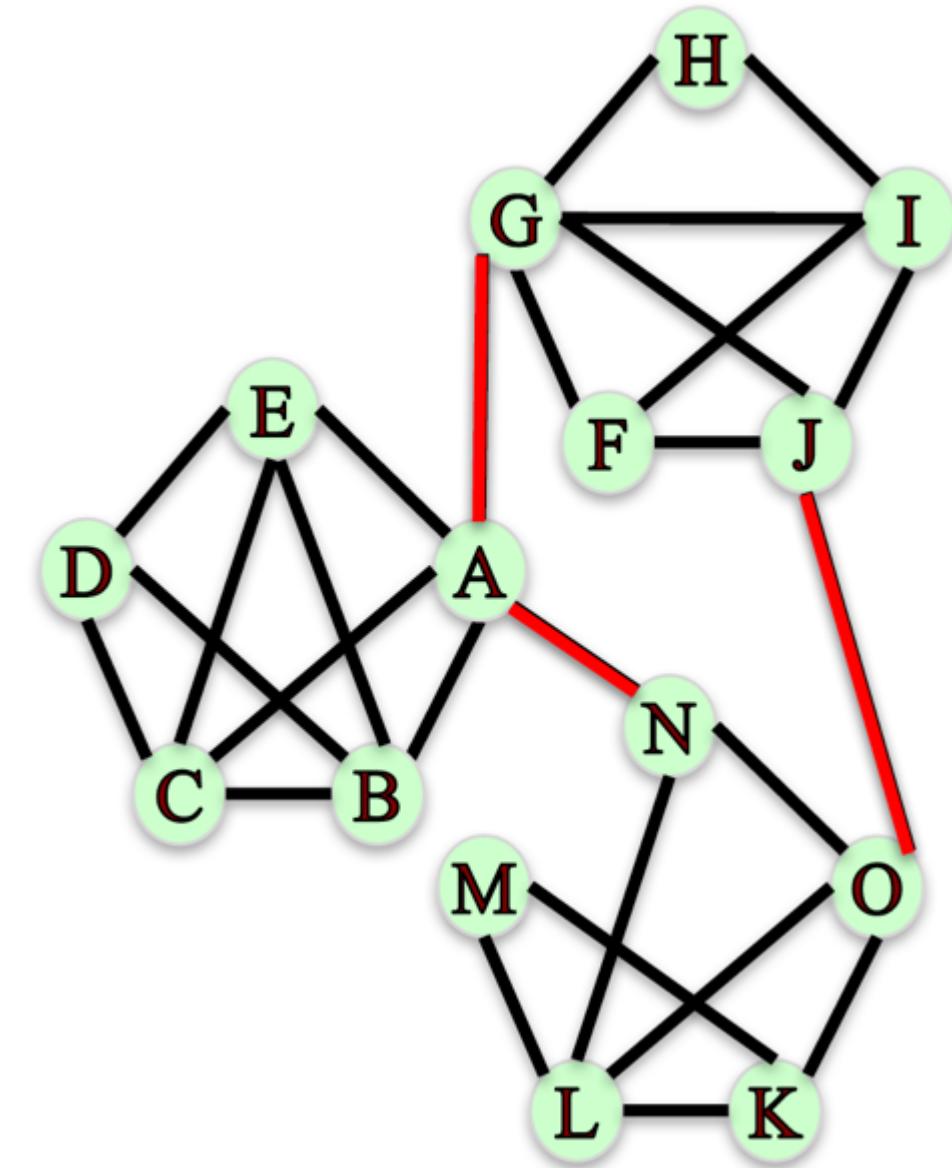
Conectividad de grafos

Un grafo no dirigido está **conectado** si, para cada par de nodos, hay un camino entre ellos.

In: `nx.is_connected(G)`

Out: True

Sin embargo, si eliminamos las aristas A — G, A — N y J — O, el grafo se **desconecta**.

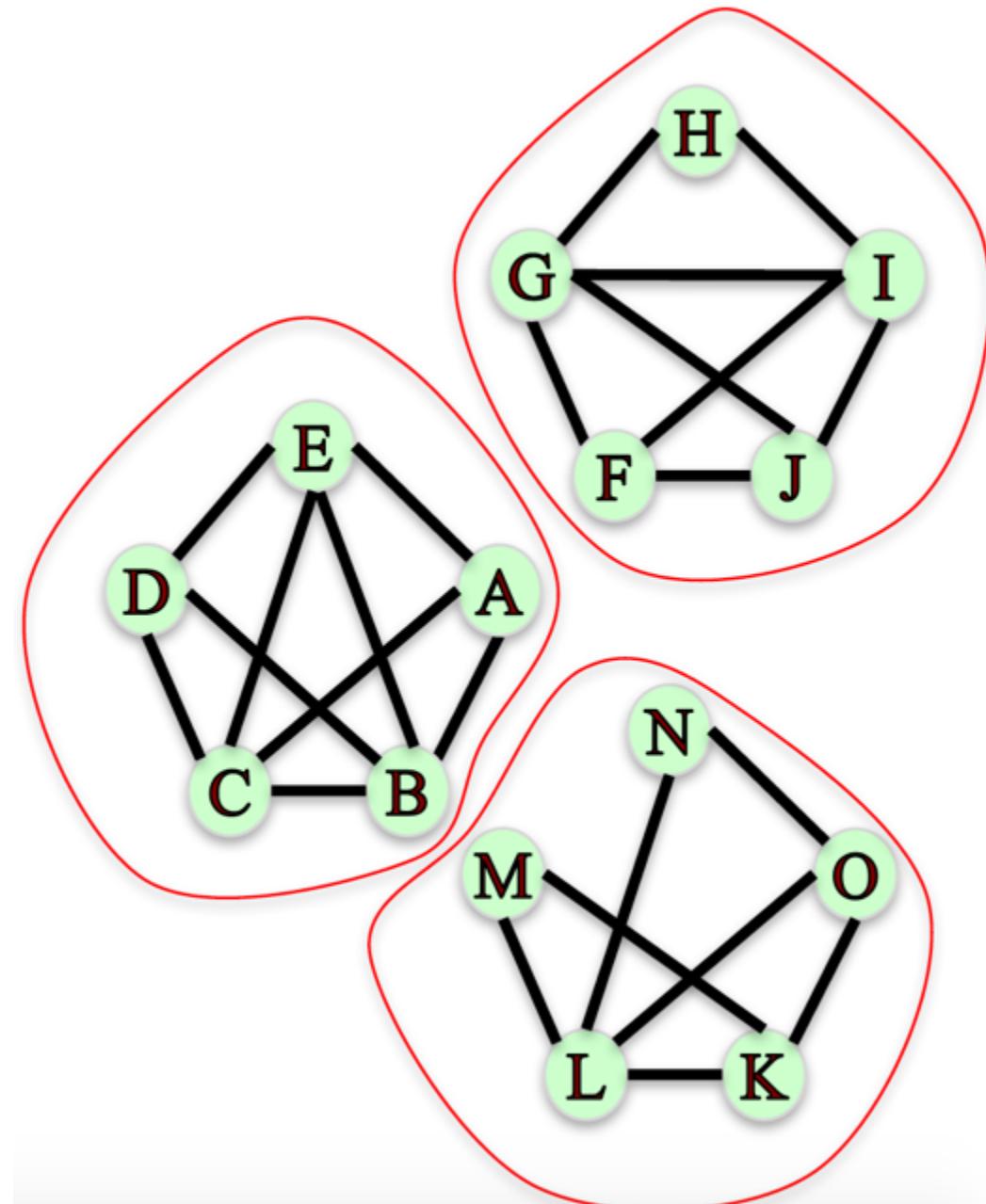


Conectividad de grafos

Un grafo no dirigido está **conectado** si, para cada par de nodos, hay un camino entre ellos.

Sin embargo, si eliminamos las aristas A — G, A — N y J — O, el grafo se **desconecta**.

No hay camino entre los nodos en las tres “comunidades” diferentes.



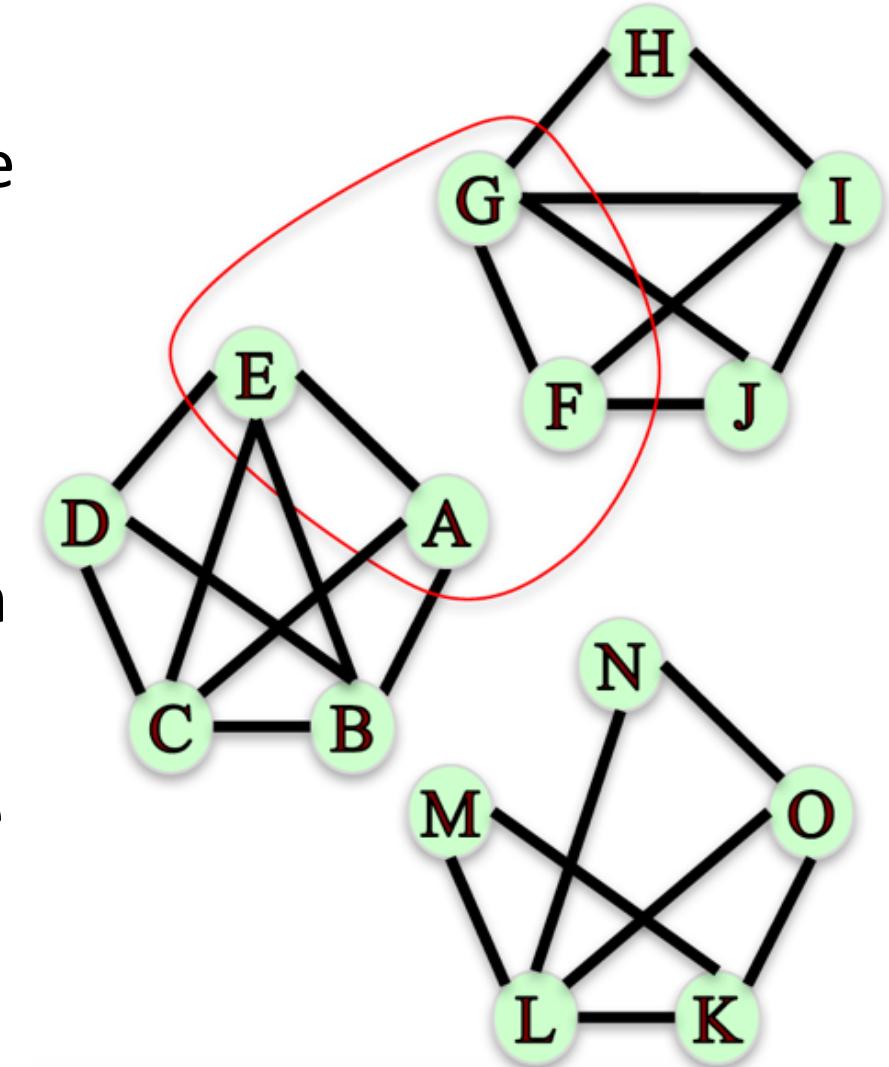
Componentes del grafo

Componente conectado es un subconjunto de nodos con las siguientes características:

1. Cada nodo en el subconjunto tiene un camino hacia cada otro nodo.
2. Ningún otro nodo tiene un camino a ningún nodo en otro subconjunto.

¿El subconjunto $\{E, A, G, F\}$ es un componente conectado?

No, no hay ruta entre los nodos A y F.



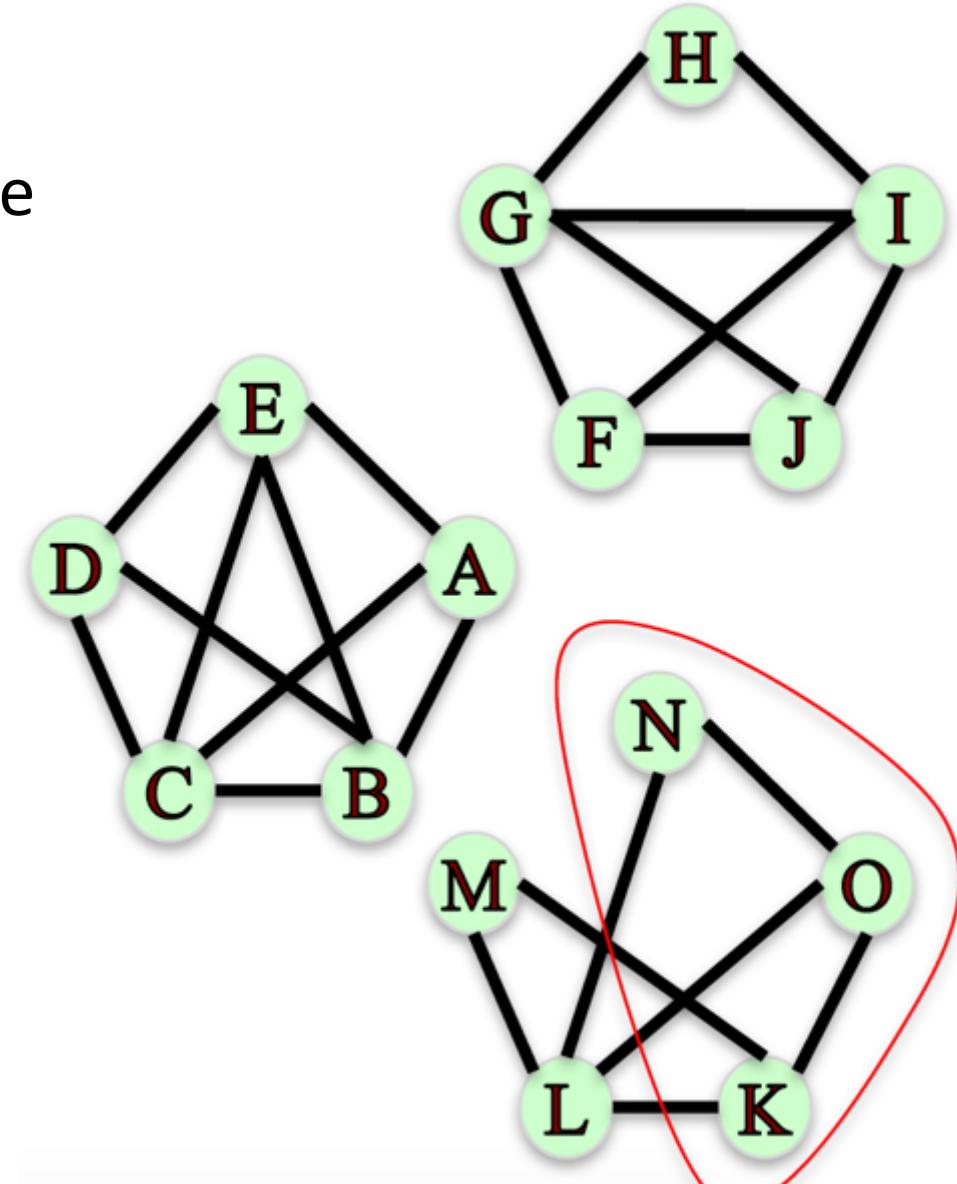
Componentes del grafo

Componente conectado es un subconjunto de nodos con las siguientes características:

1. Cada nodo en el subconjunto tiene un camino hacia cada otro nodo.
2. Ningún otro nodo tiene una ruta a ningún nodo en otro subconjunto.

¿El subconjunto {N, O, K} es un componente conectado?

No, el nodo L no tiene camino a N, O, ni K.



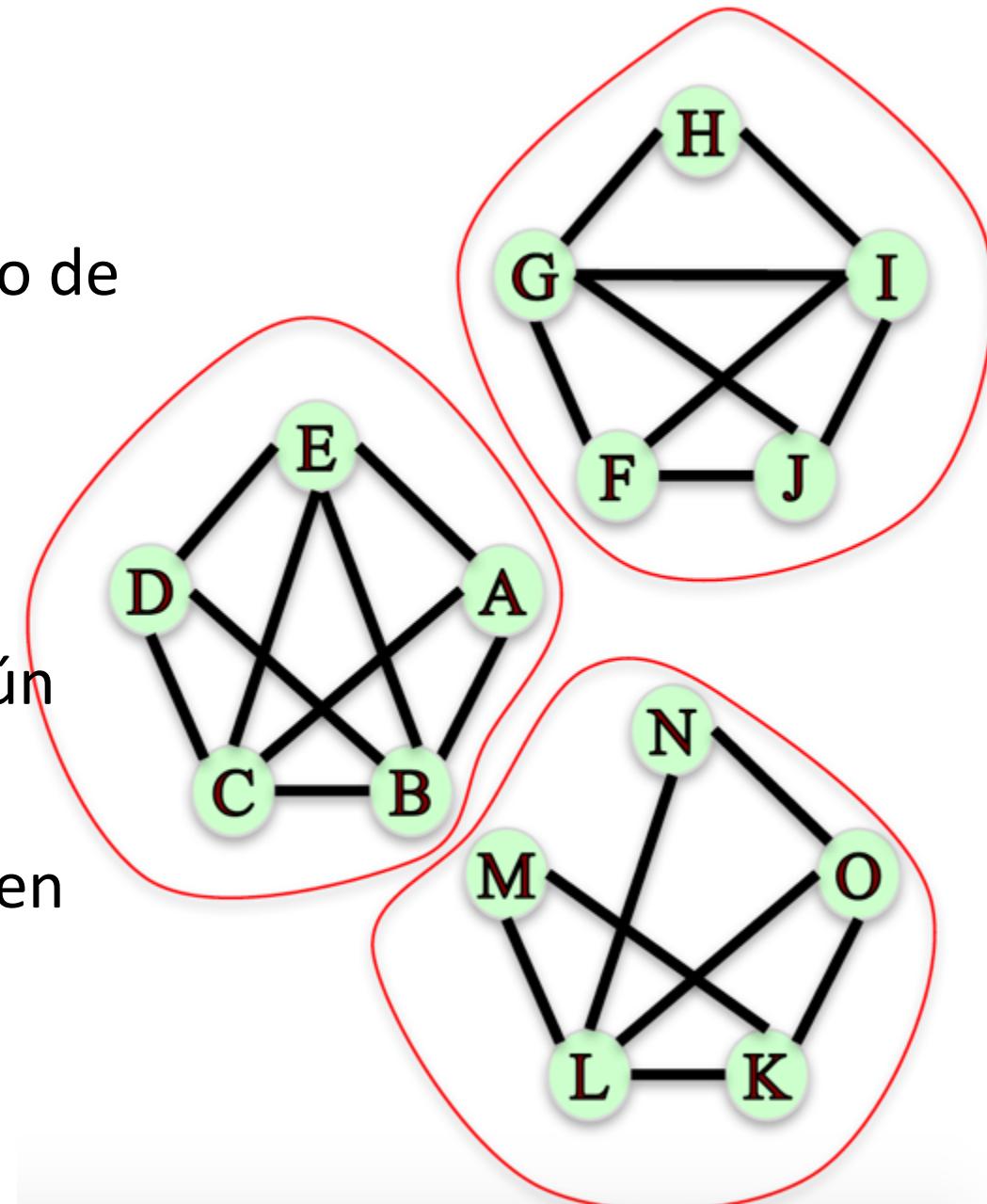
Componentes del grafo

Componente conectado es un subconjunto de nodos con las siguientes características:

1. Cada nodo en el subconjunto tiene un camino hacia cada otro nodo.
2. Ningún otro nodo tiene una ruta a ningún nodo en el subconjunto.

¿Cuáles son los componentes conectados en este grafo?

{A, B, C, D, E}, {F, G, H, I, J}, {K, L, M, N, O}



Componentes del grafo

In: `nx.number_connected_components(G)`

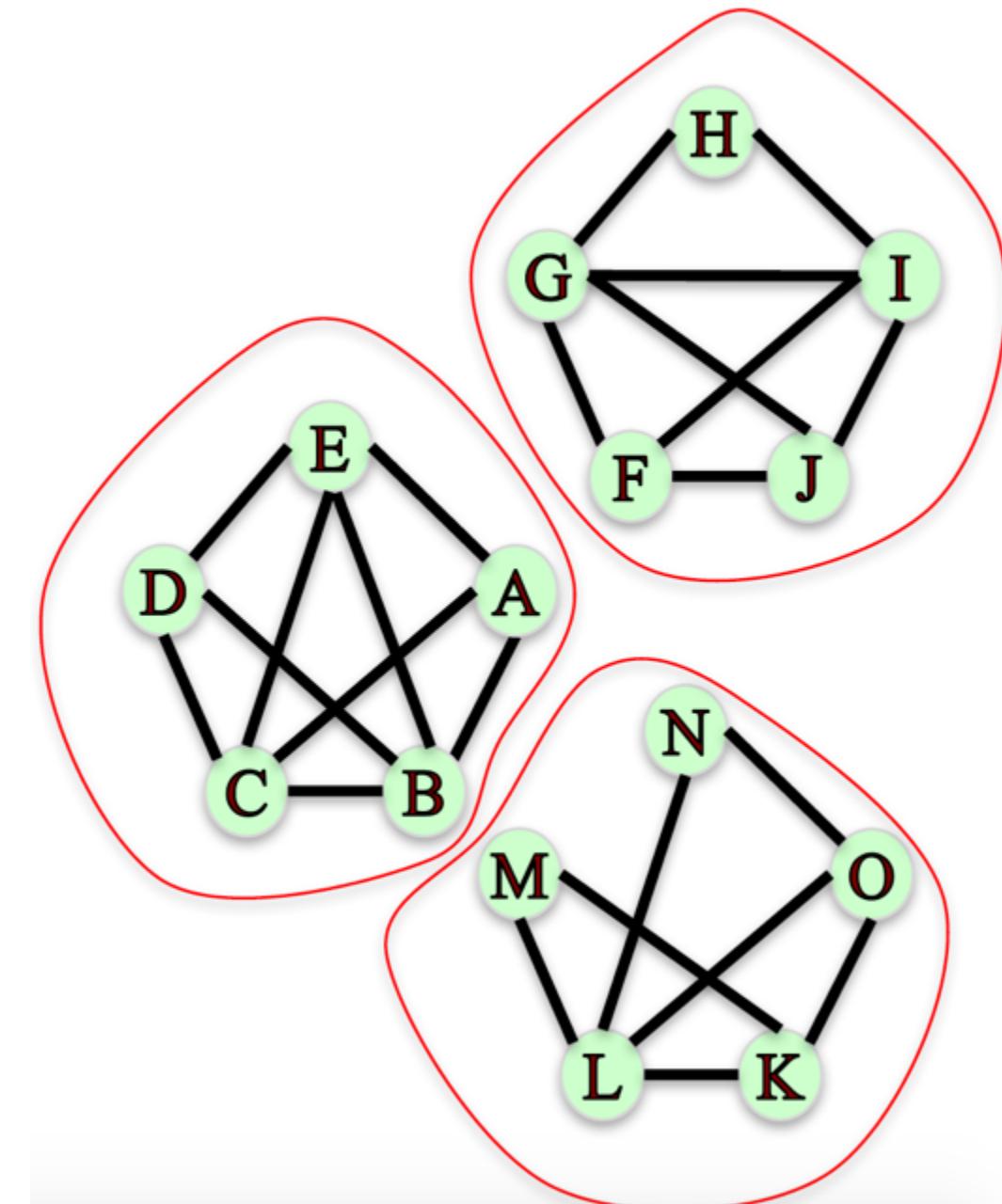
Out: 3

In: `sorted(nx.connected_components(G))`

Out: `[{'A', 'B', 'C', 'D', 'E'},
 {'F', 'G', 'H', 'I', 'J'},
 {'K', 'L', 'M', 'N', 'O'}]`

In: `nx.node_connected_component(G, 'M')`

Out: `{'K', 'L', 'M', 'N', 'O'}`



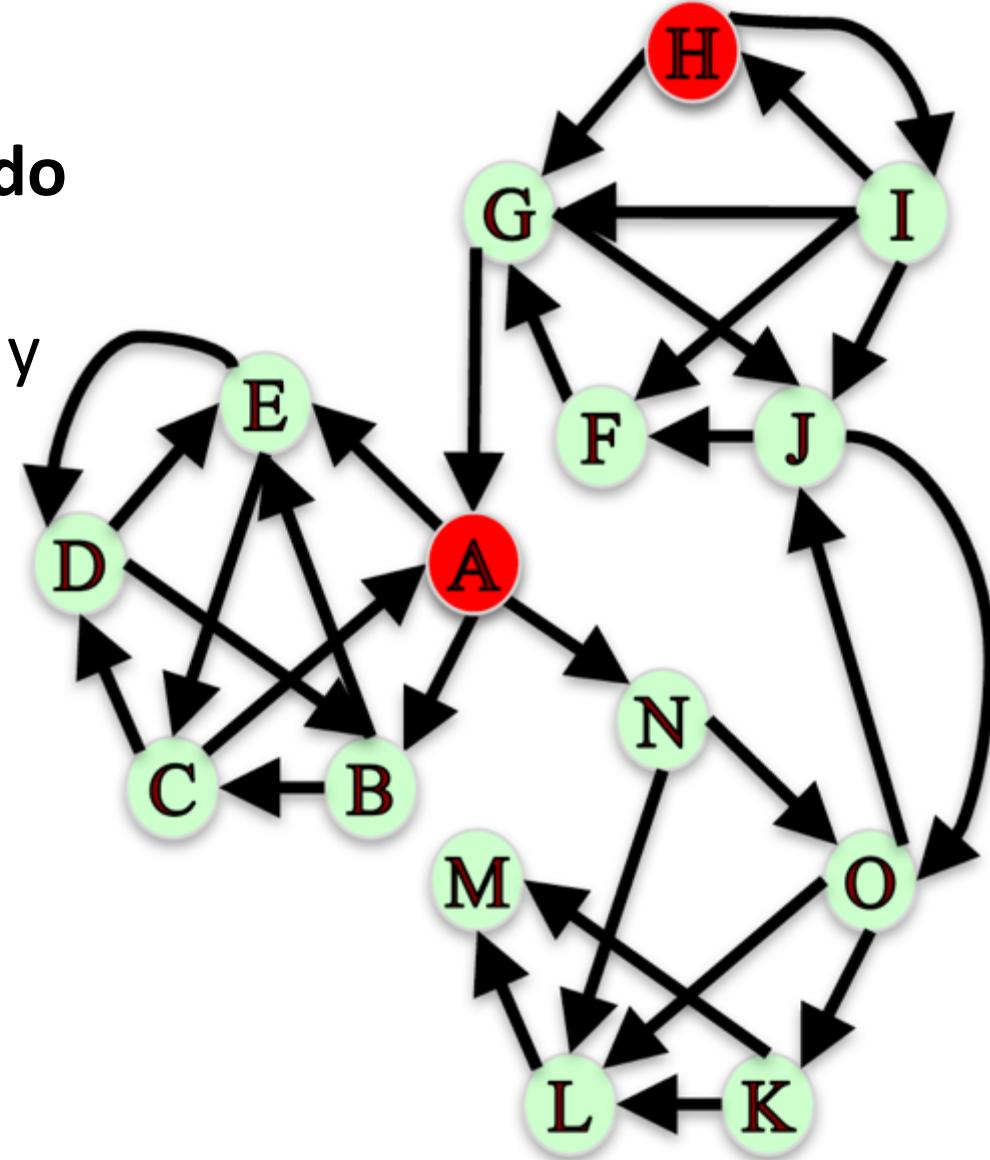
Conectividad en grafos dirigidos

Un grafo dirigido está **fuertemente conectado** (strongly connected) si, para cada par de nodos u y v , hay un camino dirigido de u a v y un camino dirigido de v a u .

¿Este grafo está **fuertemente conectado**?

```
nx.is_strongly_connected(G)
```

No, no hay una ruta dirigida de A a H



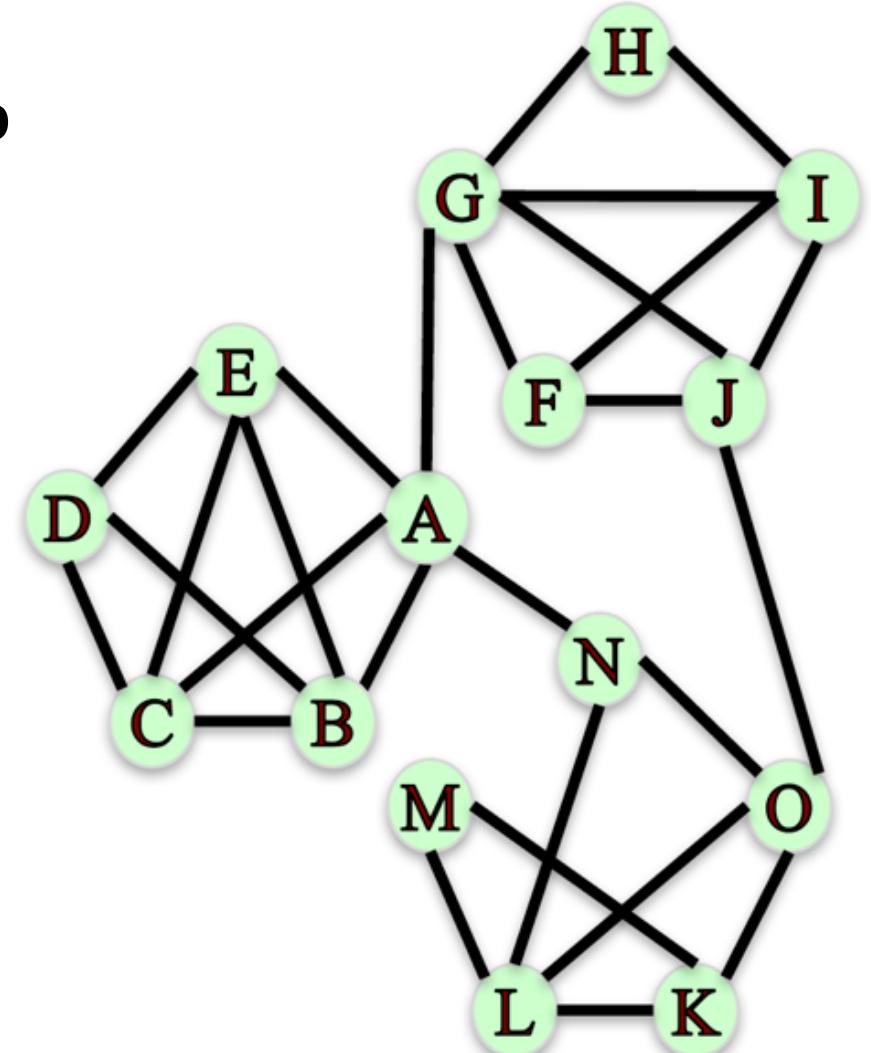
Conectividad en grafos dirigidos

Un grafo dirigido está **débilmente conectado** si la sustitución de todas las aristas dirigidas por aristas no dirigidas produce un grafo conectado no dirigido.

¿Este grafo está **debilmente conectado**?

In: `nx.is_weakly_connected(G)`

Si



Conectividad en grafos dirigidos

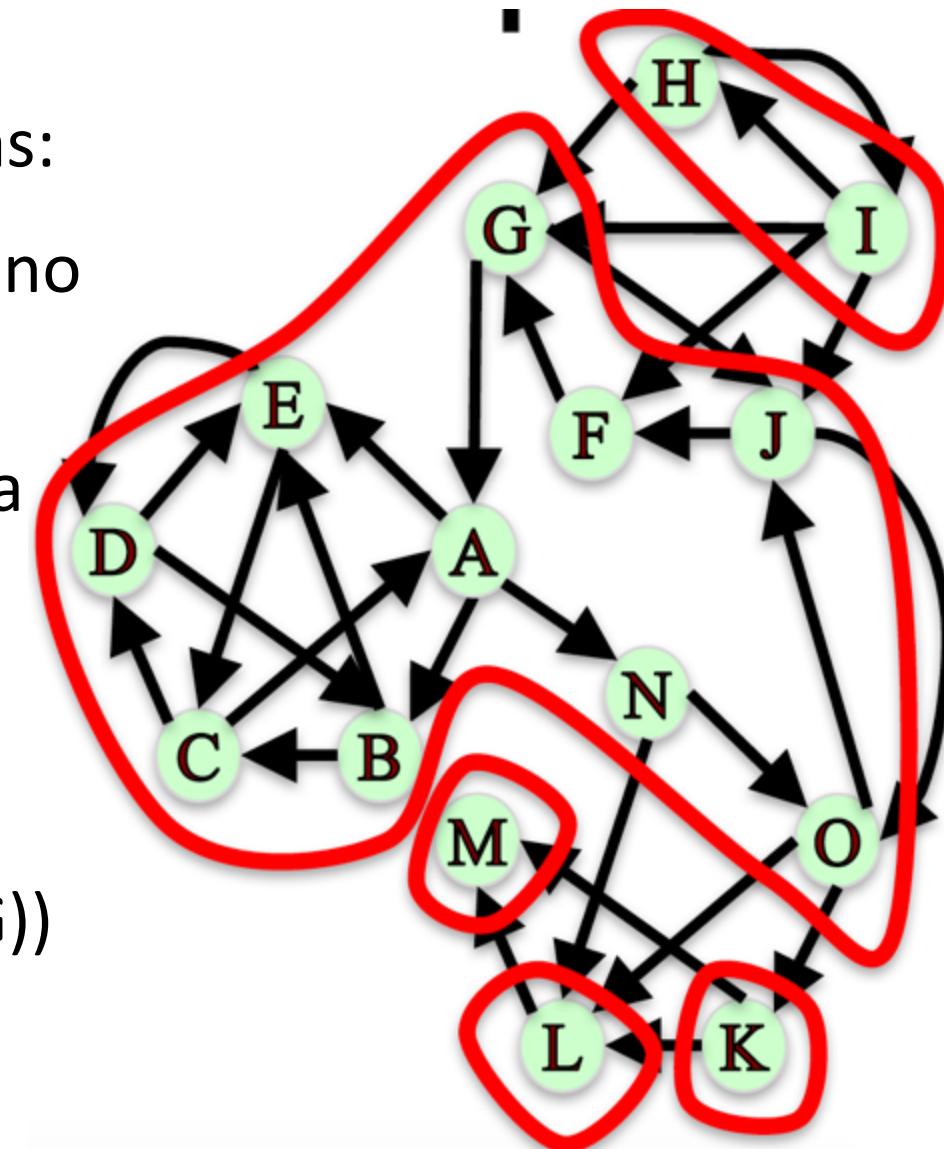
Componente fuertemente conectado es un subconjunto de nodos con estas características:

1. Cada nodo en el subconjunto tiene un camino **directo** a todos los demás nodos.
2. Ningún otro nodo tiene un camino **directo** a cada nodo en el subconjunto.

¿Cuáles son los componentes fuertemente conectados en este grafo?

```
sorted(nx.strongly_connected_components(G))
```

```
{M}, {L}, {K}, {A, B, C, D, E, F, G, J, N, O}, {H, I}
```



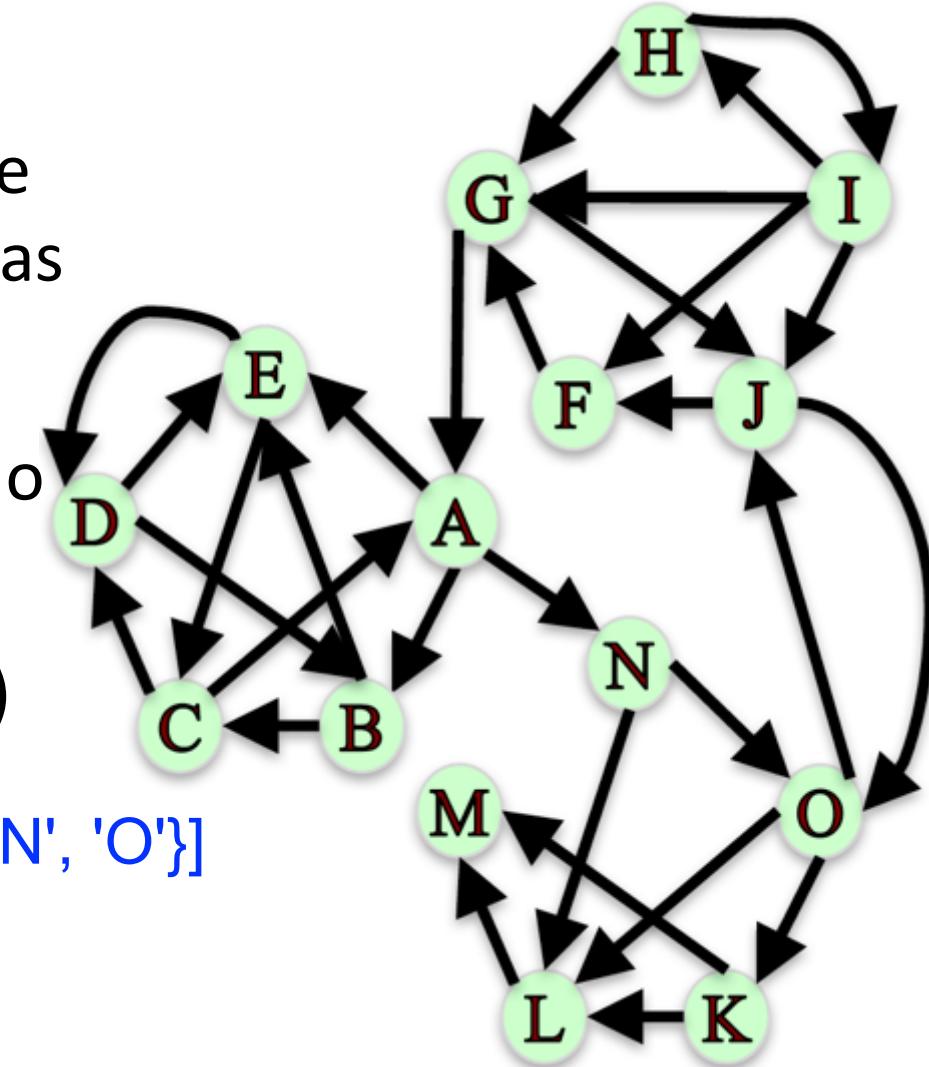
Conectividad en grafos dirigidos

Componente débilmente conectado: son los componentes conectados del grafo después de reemplazar todas las aristas dirigidos con aristas no dirigidas.

Como el grafo está **débilmente conectado**, solo tiene un componente débilmente conectado.

```
sorted(nx.weakly_connected_components(G))
```

```
[{'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O'}]
```



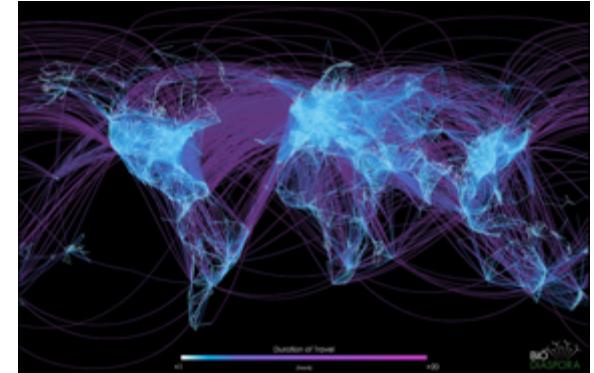
Conectividad y robustez en las redes.

Robustez de la red: la capacidad de una red (grafo) para mantener sus propiedades estructurales generales cuando enfrenta fallas o ataques.

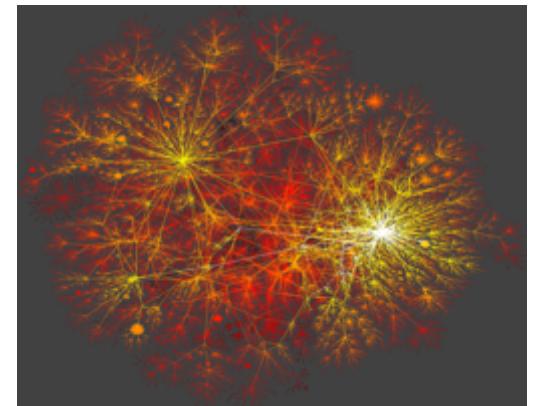
Tipo de ataques: eliminación de nodos o aristas.

Propiedades estructurales: conectividad.

Ejemplos: cierres de aeropuertos, fallas en el enrutador de Internet, fallas en la línea eléctrica.



Red de vuelos directos
alrededor del mundo
[Bio.Diaspora]



Conectividad en internet [K. C. Claffy]

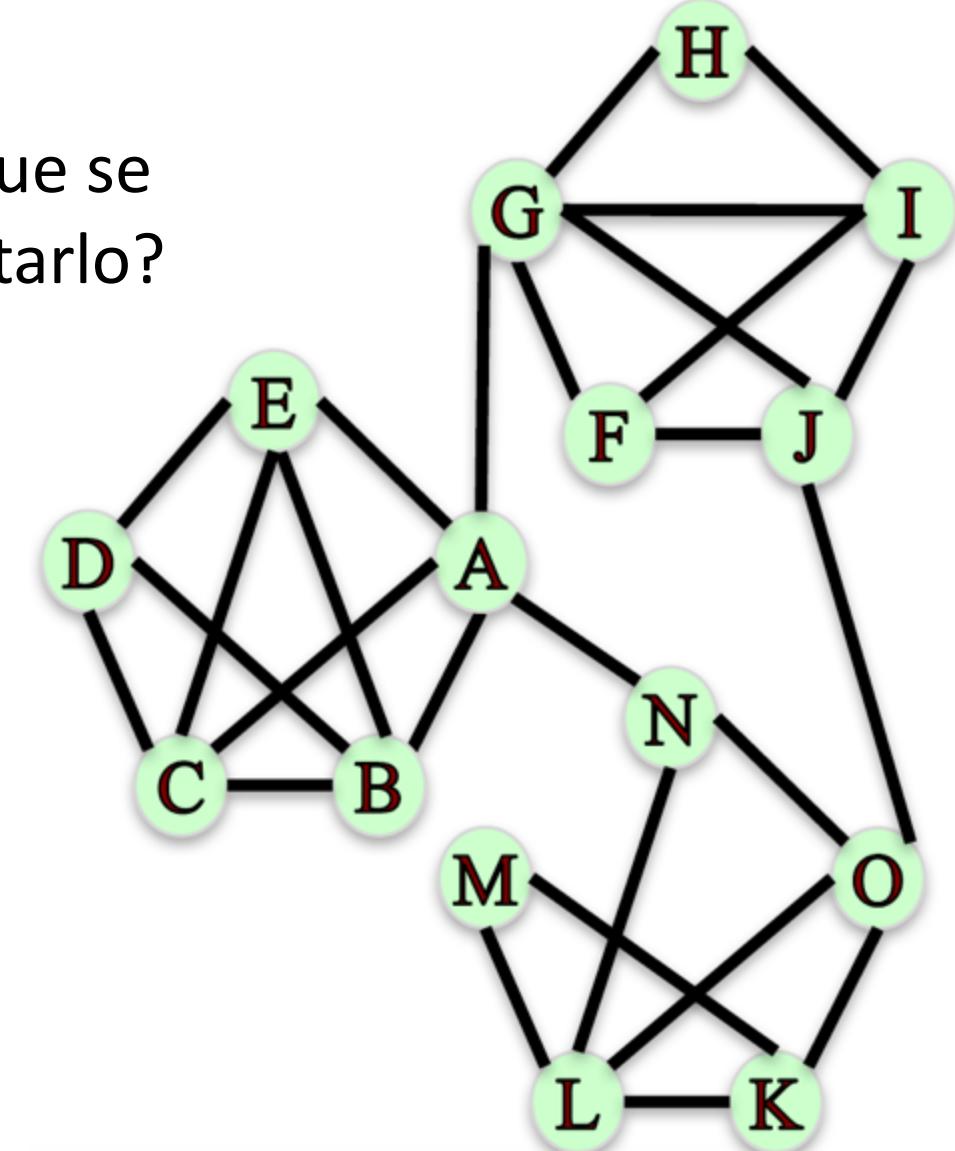
Desconectando un grafo

¿Cuál es el número más pequeño de **nodos** que se pueden eliminar de este grafo para desconectararlo?

`nx.node_connectivity(G)`

¿Cuál nodo?

`nx.minimum_node_cut(G)`



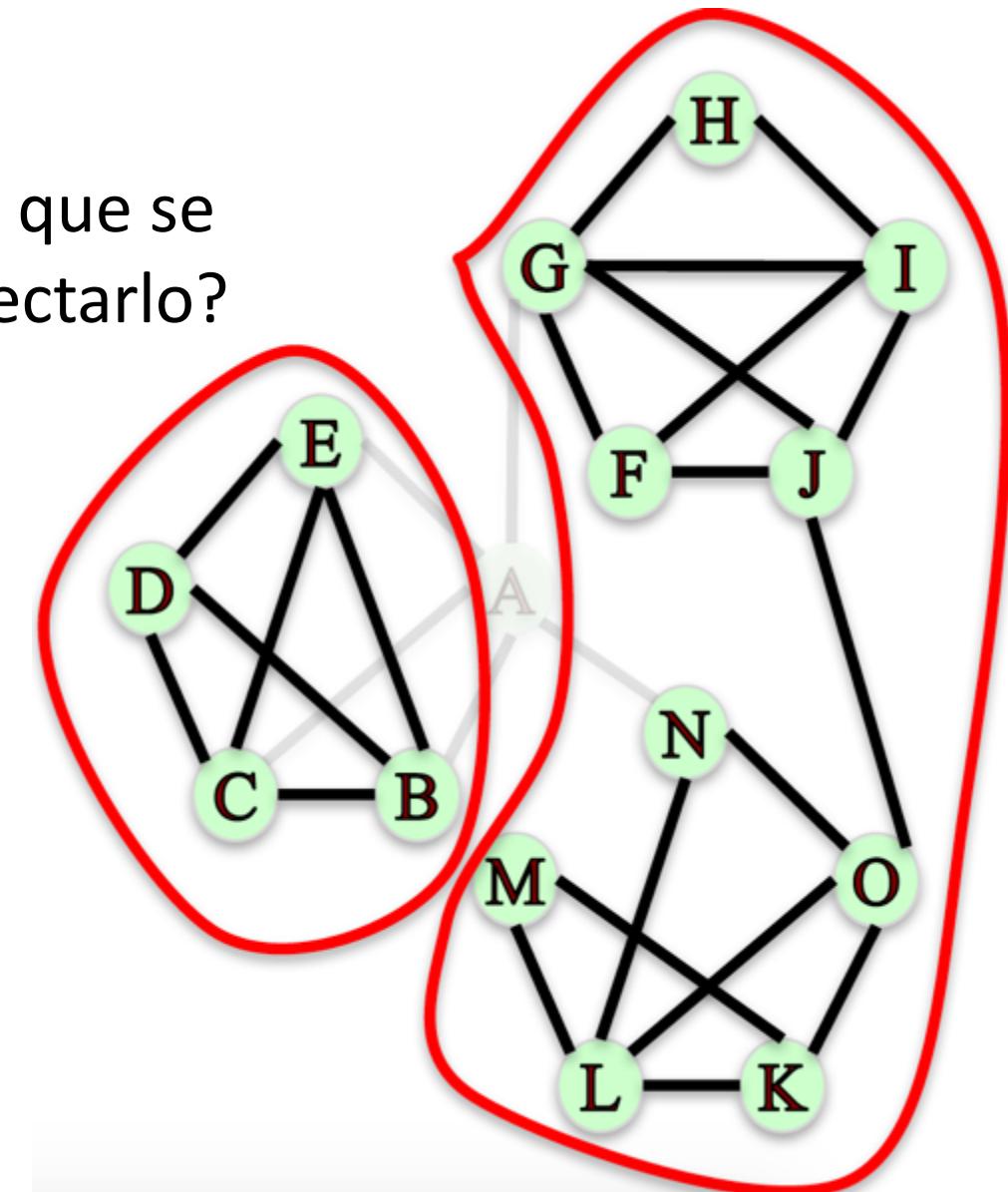
Desconectando un grafo

¿Cuál es el número más pequeño de **nodos** que se pueden eliminar de este grafo para desconectararlo?

1

Cual nodo?

{A}



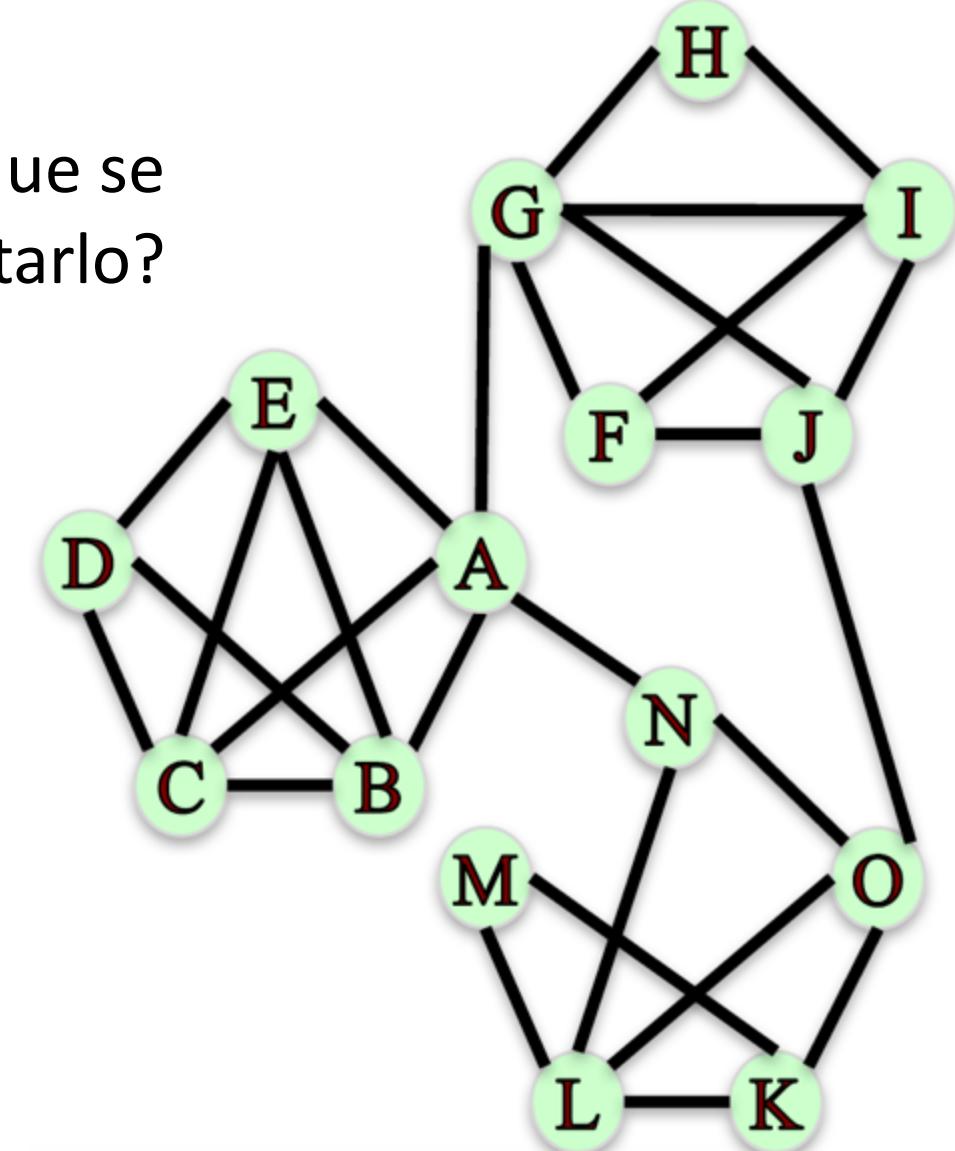
Desconectando un grafo

¿Cuál es el número más pequeño de **aristas** que se pueden eliminar de este grafo para desconectararlo?

`nx.edge_connectivity(G)`

¿Cuáles aristas?

`nx.minimum_edge_cut(G)`



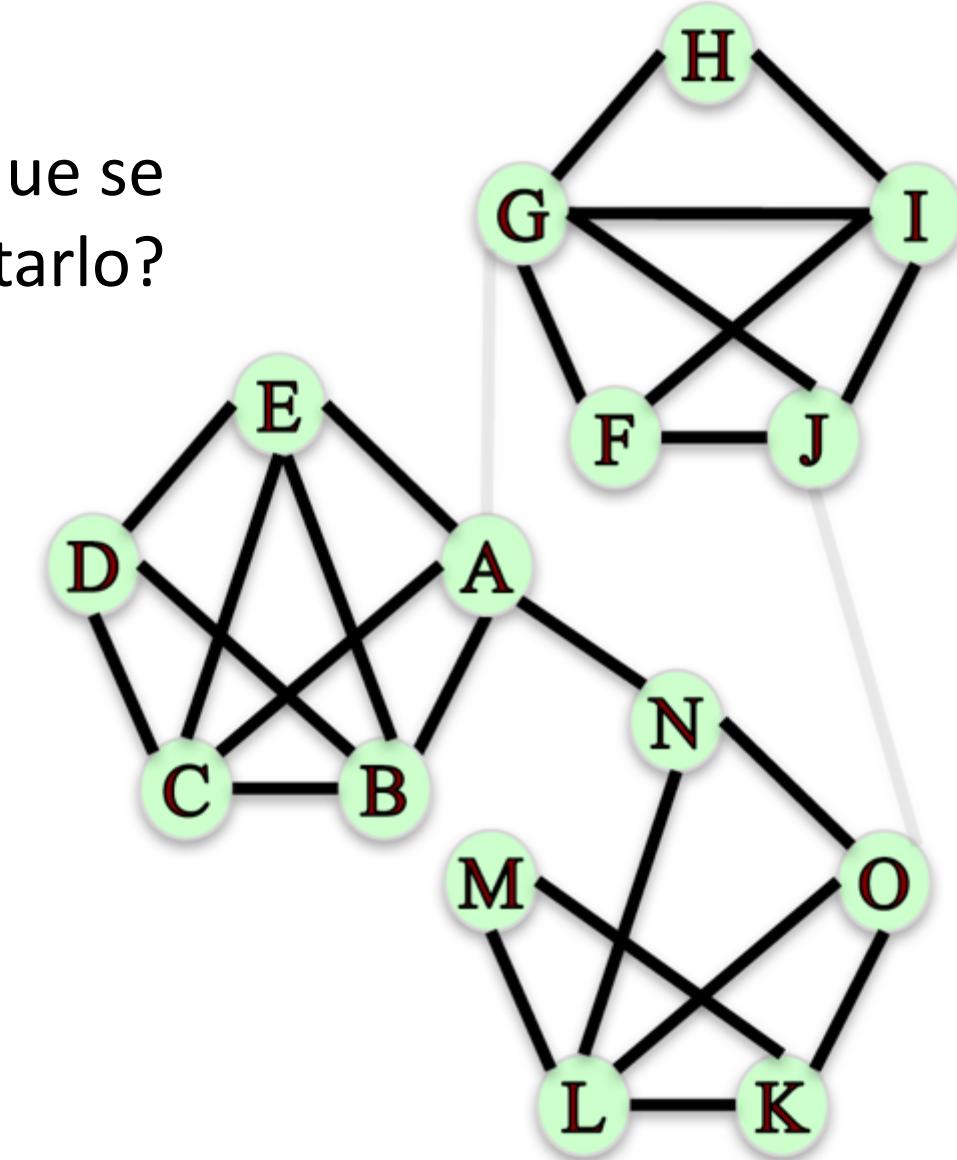
Desconectando un grafo

¿Cuál es el número más pequeño de **aristas** que se pueden eliminar de este grafo para desconectararlo?

2

Cuales aristas?

$\{('A', 'G'), ('O', 'J')\}$



Desconectando un grafo

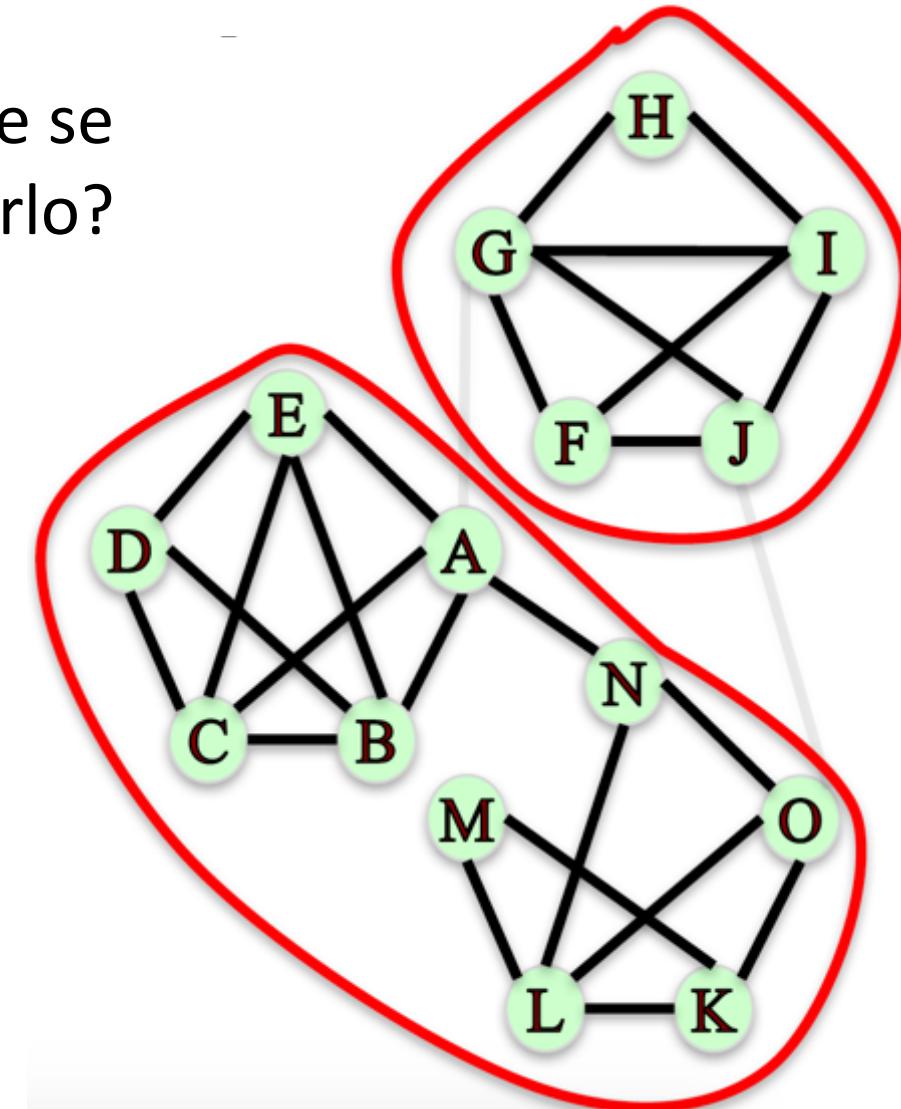
¿Cuál es el número más pequeño de **aristas** que se pueden eliminar de este grafo para desconectararlo?

2

Cuales aristas?

`{('A', 'G'), ('O', 'J')}`

Las redes robustas tienen grandes cortes mínimos de nodo y arista.

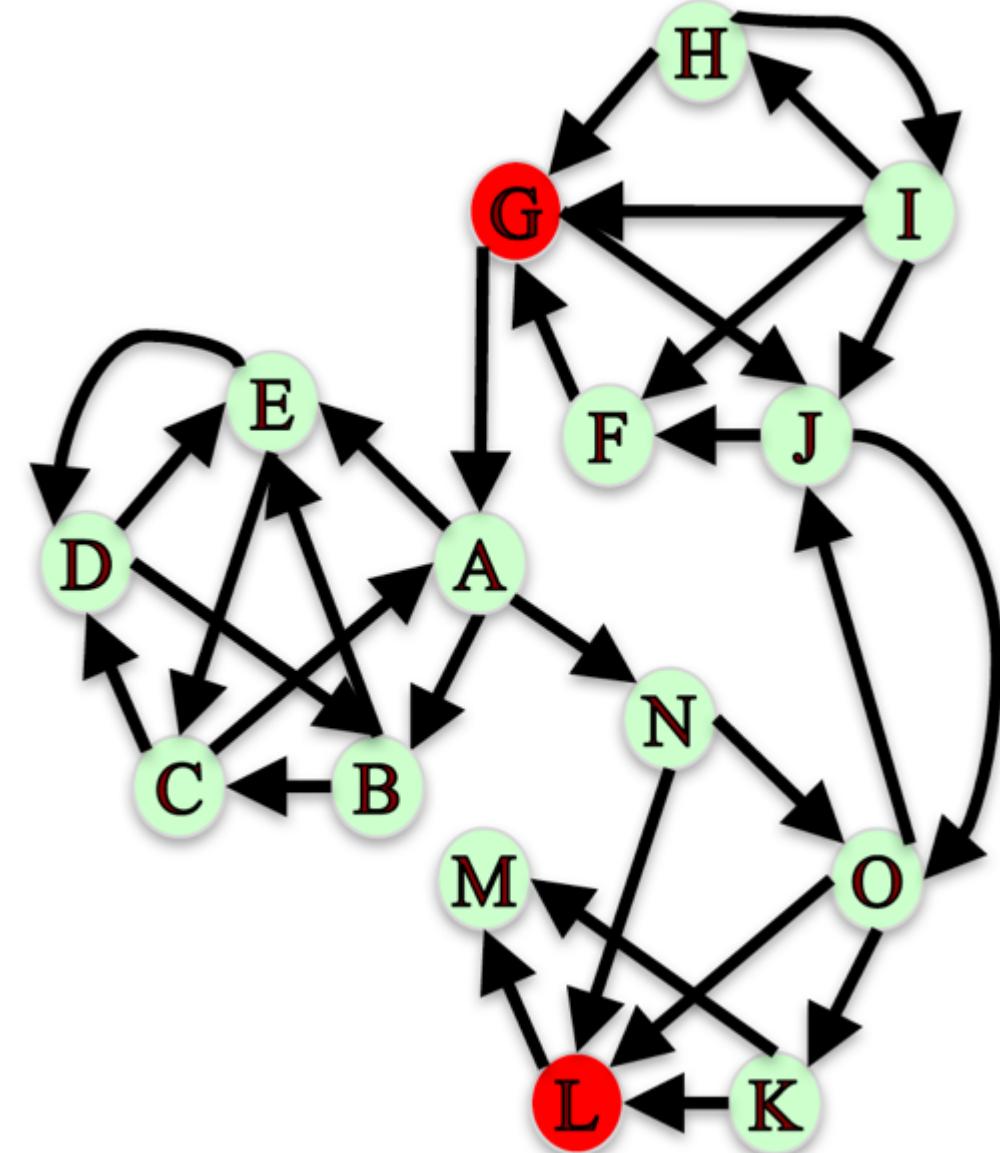


Caminos simples

Imagina que el nodo **G** desea enviar un mensaje al nodo **L** pasándolo a otros nodos en esta red.

¿Qué opciones tiene **G** para entregar el mensaje?

```
sorted(nx.all_simple_paths(G, 'G', 'L'))
```

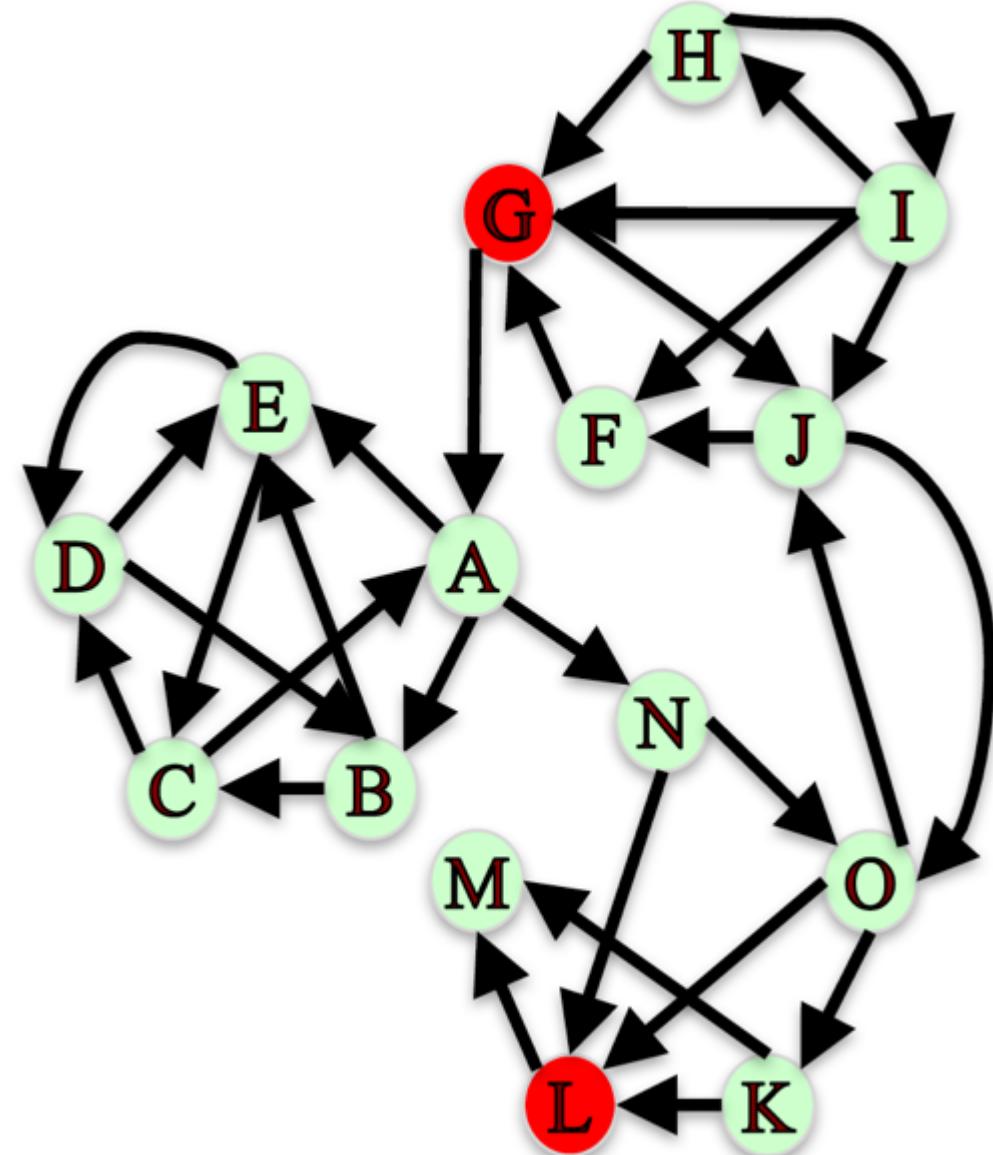


Caminos simples

Imagina que el nodo **G** desea enviar un mensaje al nodo **L** pasándolo a otros nodos en esta red.

¿Qué opciones tiene **G** para entregar el mensaje?

- ['G', 'A', 'N', 'L'],
- ['G', 'A', 'N', 'O', 'K', 'L'],
- ['G', 'A', 'N', 'O', 'L'],
- ['G', 'J', 'O', 'K', 'L'],
- ['G', 'J', 'O', 'L']

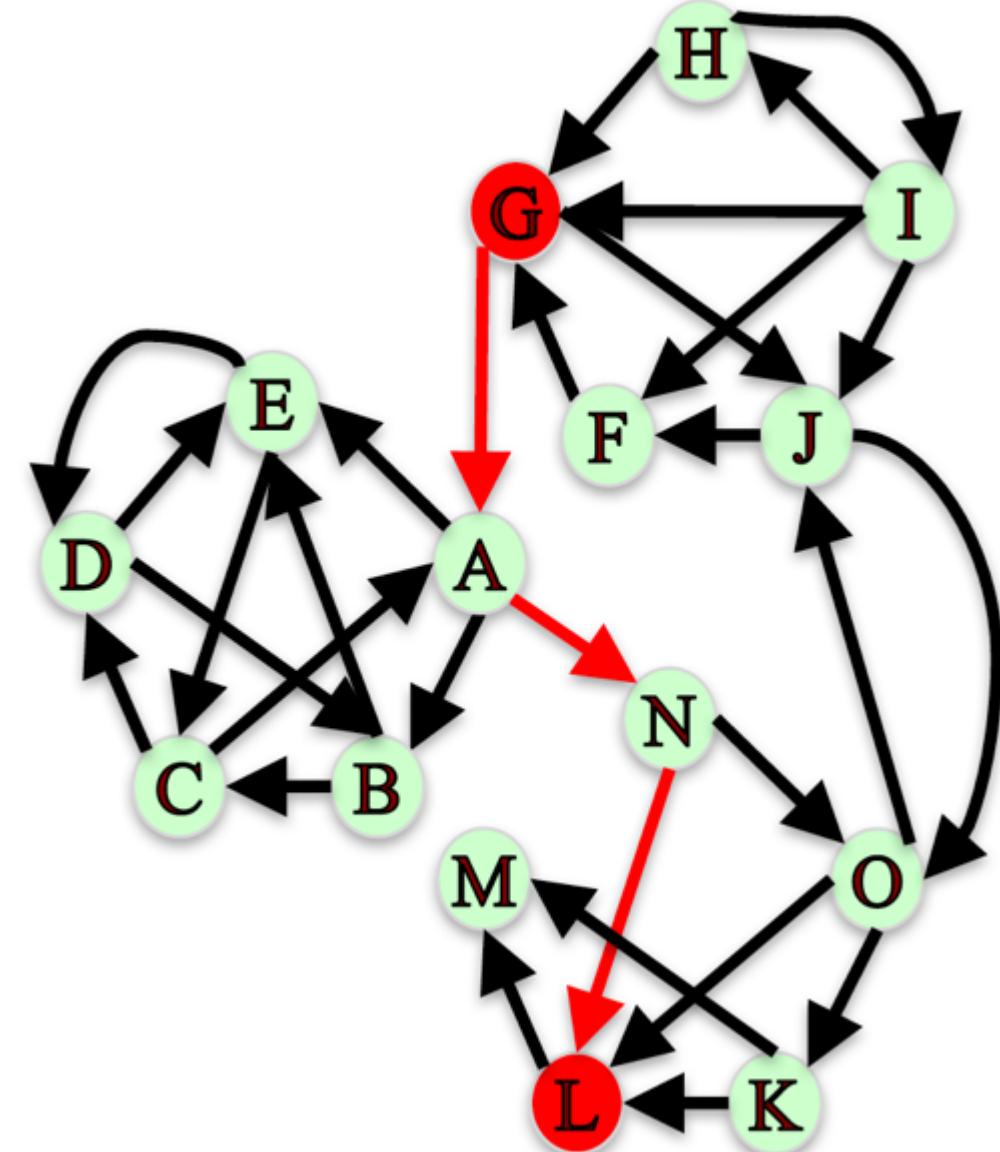


Caminos simples

Imagina que el nodo **G** desea enviar un mensaje al nodo **L** pasándolo a otros nodos en esta red.

¿Qué opciones tiene **G** para entregar el mensaje?

- ['G', 'A', 'N', 'L'],
- ['G', 'A', 'N', 'O', 'K', 'L'],
- ['G', 'A', 'N', 'O', 'L'],
- ['G', 'J', 'O', 'K', 'L'],
- ['G', 'J', 'O', 'L']

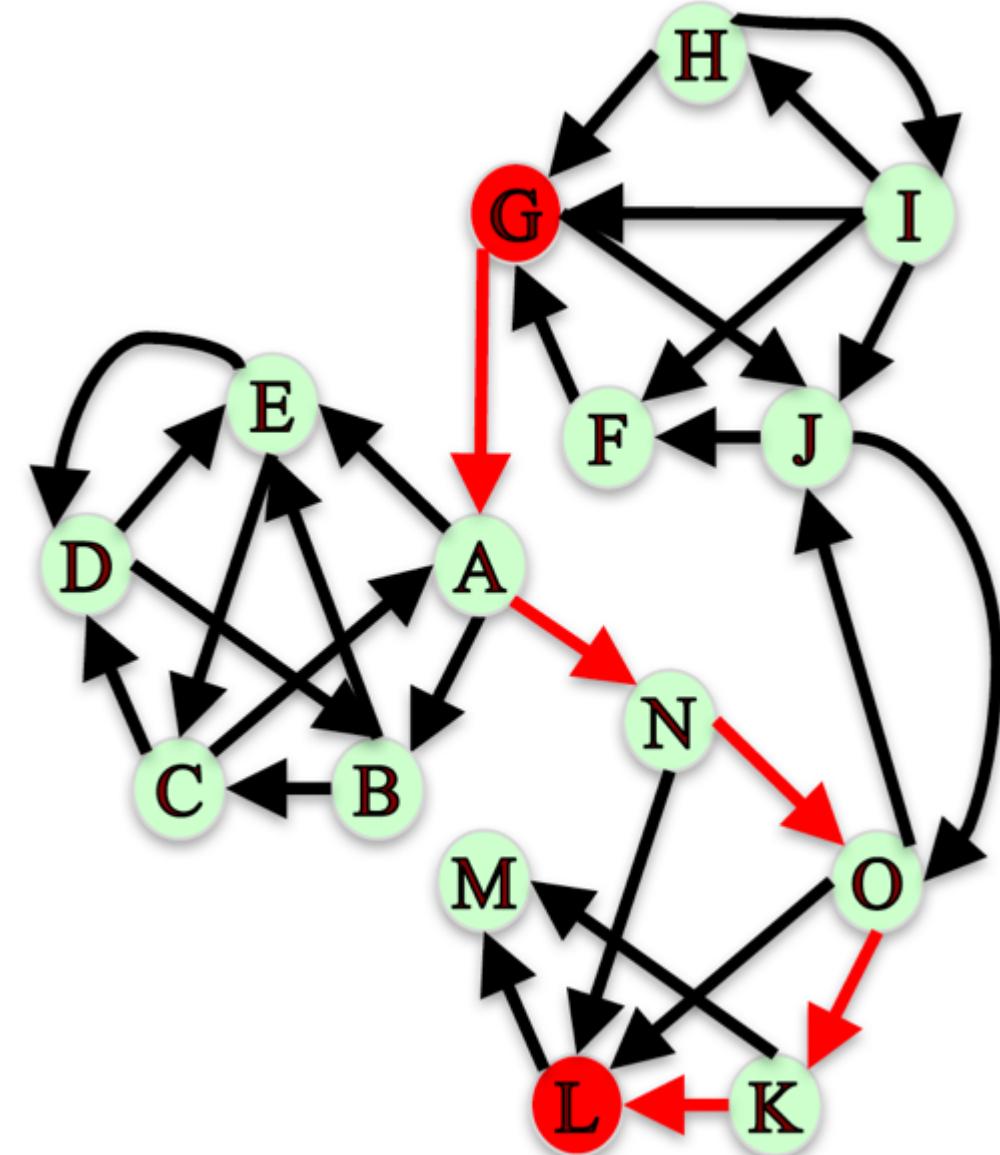


Caminos simples

Imagina que el nodo **G** desea enviar un mensaje al nodo **L** pasándolo a otros nodos en esta red.

¿Qué opciones tiene **G** para entregar el mensaje?

- ['G', 'A', 'N', 'L'],
- ['G', 'A', 'N', 'O', 'K', 'L'],
- ['G', 'A', 'N', 'O', 'L'],
- ['G', 'J', 'O', 'K', 'L'],
- ['G', 'J', 'O', 'L']

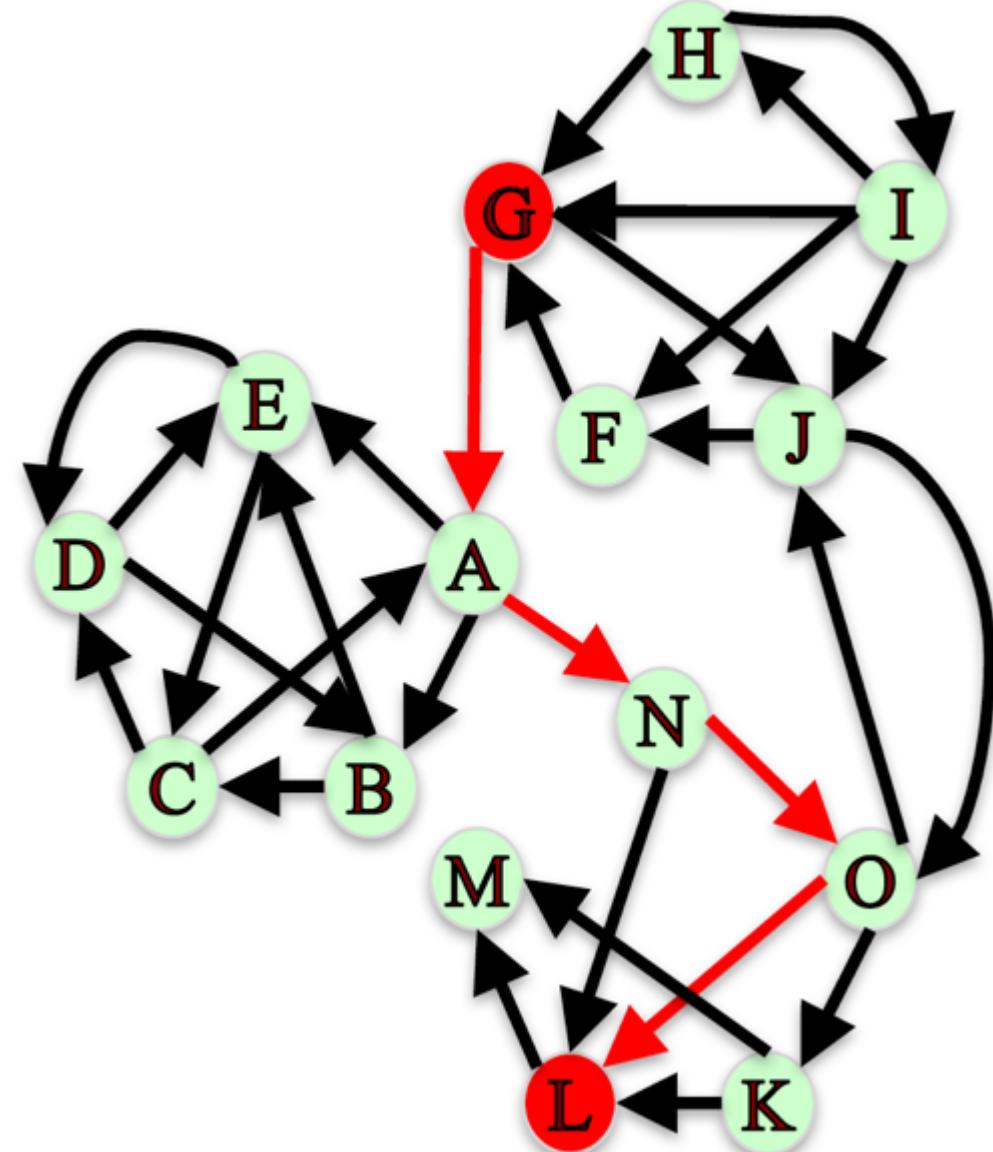


Caminos simples

Imagina que el nodo **G** desea enviar un mensaje al nodo **L** pasándolo a otros nodos en esta red.

¿Qué opciones tiene **G** para entregar el mensaje?

- ['G', 'A', 'N', 'L'],
- ['G', 'A', 'N', 'O', 'K', 'L'],
- ['G', 'A', 'N', 'O', 'L'],
- ['G', 'J', 'O', 'K', 'L'],
- ['G', 'J', 'O', 'L']

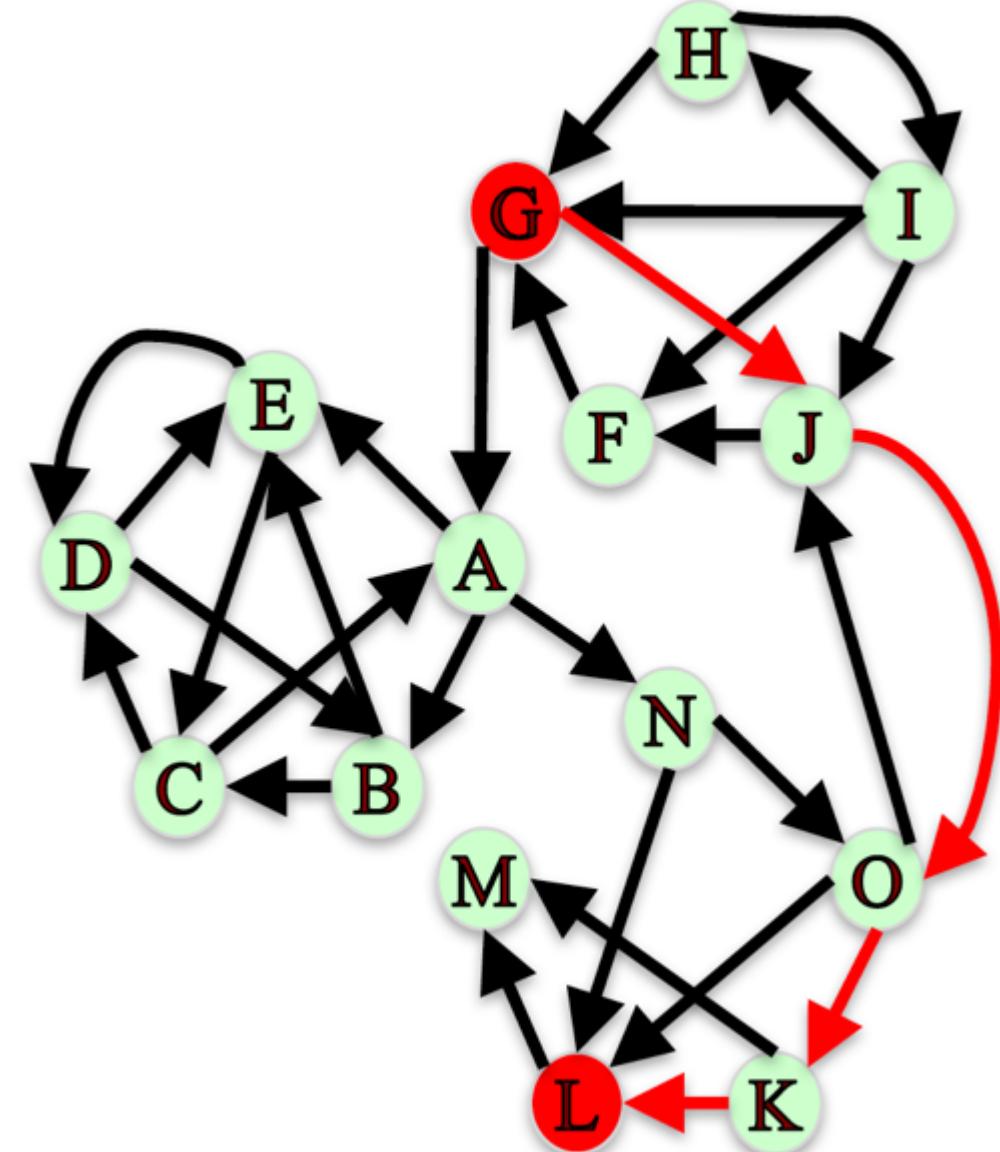


Caminos simples

Imagina que el nodo **G** desea enviar un mensaje al nodo **L** pasándolo a otros nodos en esta red.

¿Qué opciones tiene **G** para entregar el mensaje?

- ['G', 'A', 'N', 'L'],
- ['G', 'A', 'N', 'O', 'K', 'L'],
- ['G', 'A', 'N', 'O', 'L'],
- ['G', 'J', 'O', 'K', 'L'],
- ['G', 'J', 'O', 'L']

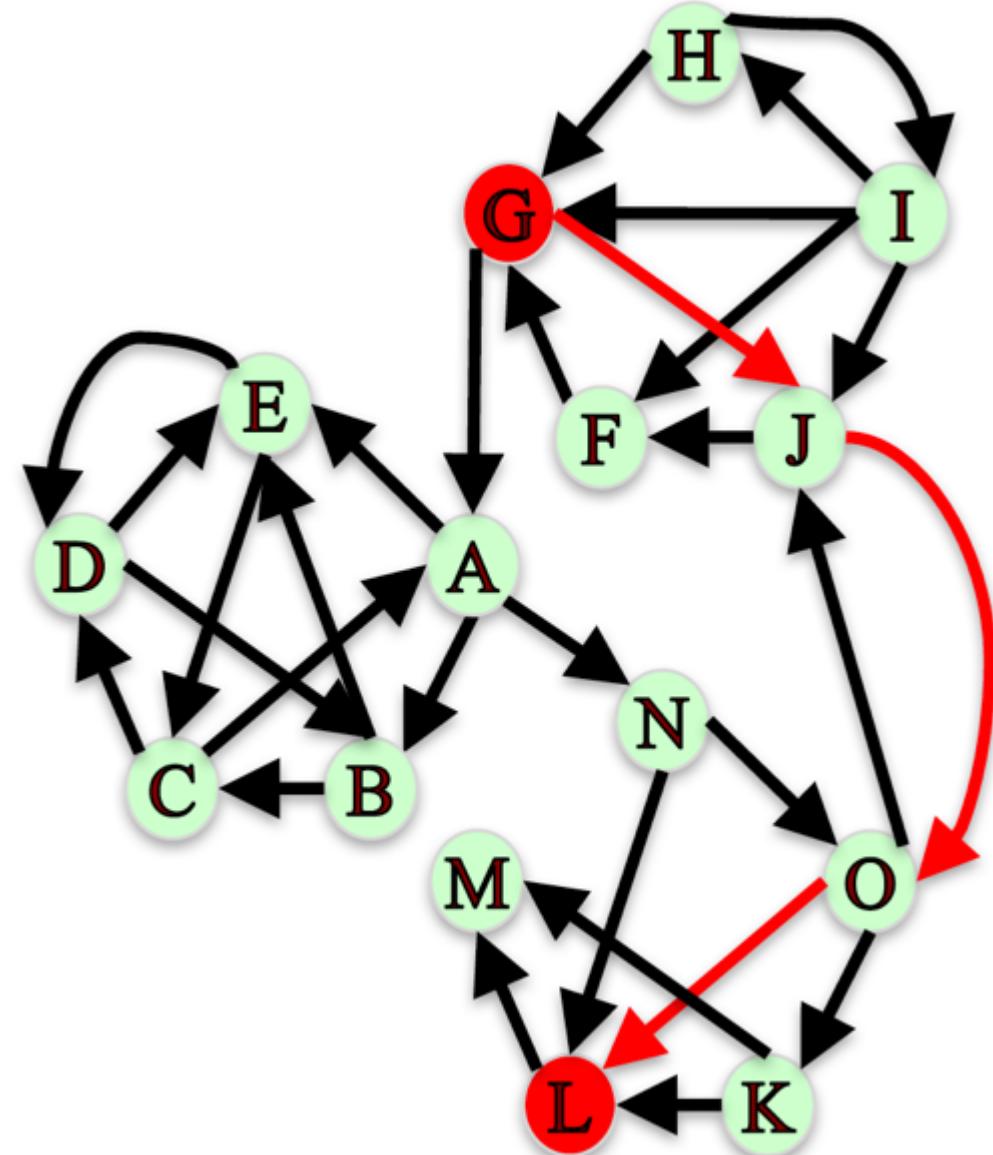


Caminos simples

Imagina que el nodo **G** desea enviar un mensaje al nodo **L** pasándolo a otros nodos en esta red.

¿Qué opciones tiene **G** para entregar el mensaje?

- ['G', 'A', 'N', 'L'],
- ['G', 'A', 'N', 'O', 'K', 'L'],
- ['G', 'A', 'N', 'O', 'L'],
- ['G', 'J', 'O', 'K', 'L'],
- ['G', 'J', 'O', 'L']



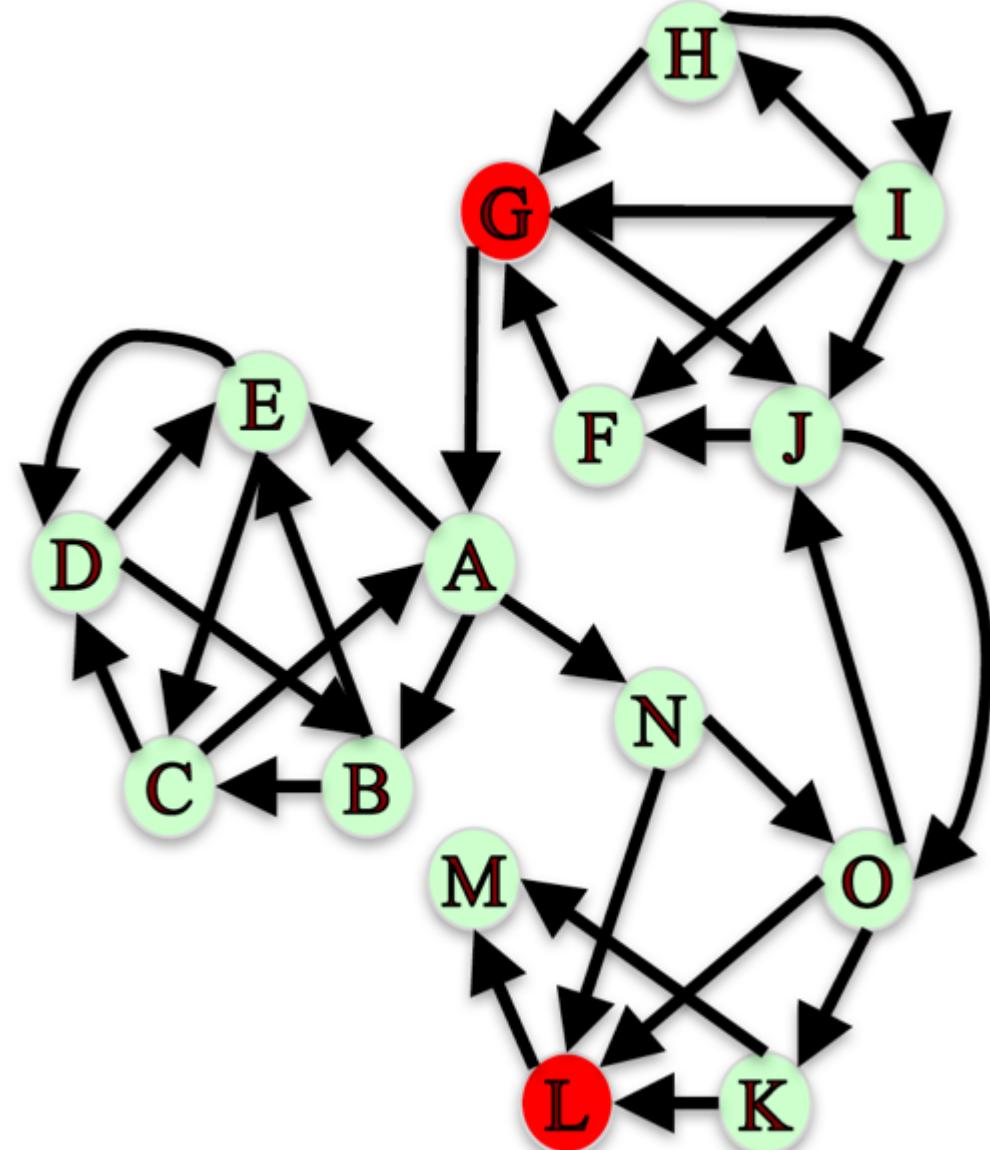
Conectividad de nodo

Si quisiéramos bloquear el mensaje de **G** a **L** mediante la eliminación de **nodos** del grafo, ¿cuántos **nodos** tendríamos que eliminar?

```
nx.node_connectivity(G, 'G', 'L')
```

¿Cuáles nodos?

```
nx.minimum_node_cut(G, 'G', 'L')
```



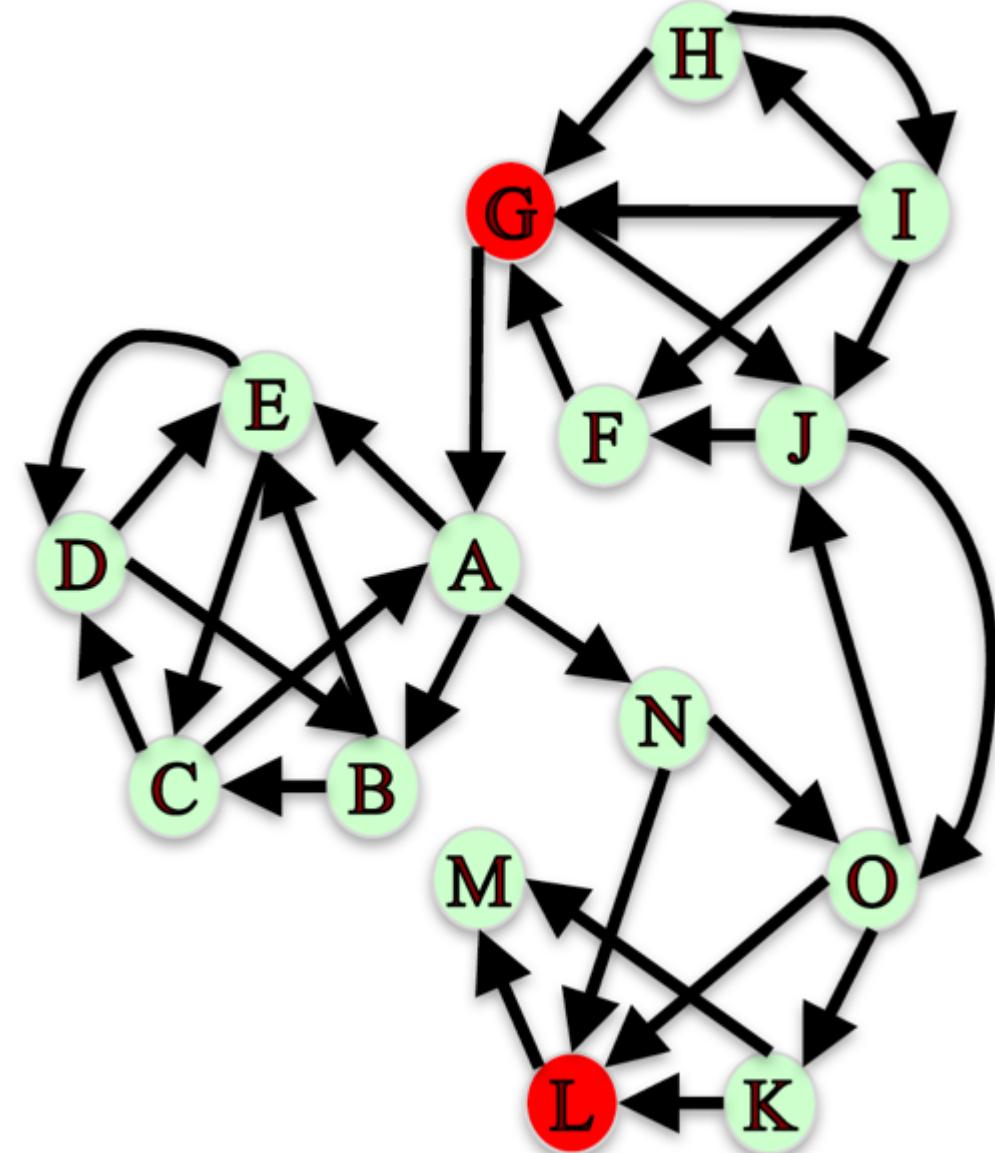
Conectividad de nodo

Si quisiéramos bloquear el mensaje de **G** a **L** mediante la eliminación de **nodos** del grafo, ¿cuántos **nodos** tendríamos que eliminar?

2

¿Cuáles nodos?

{'N', 'O'}



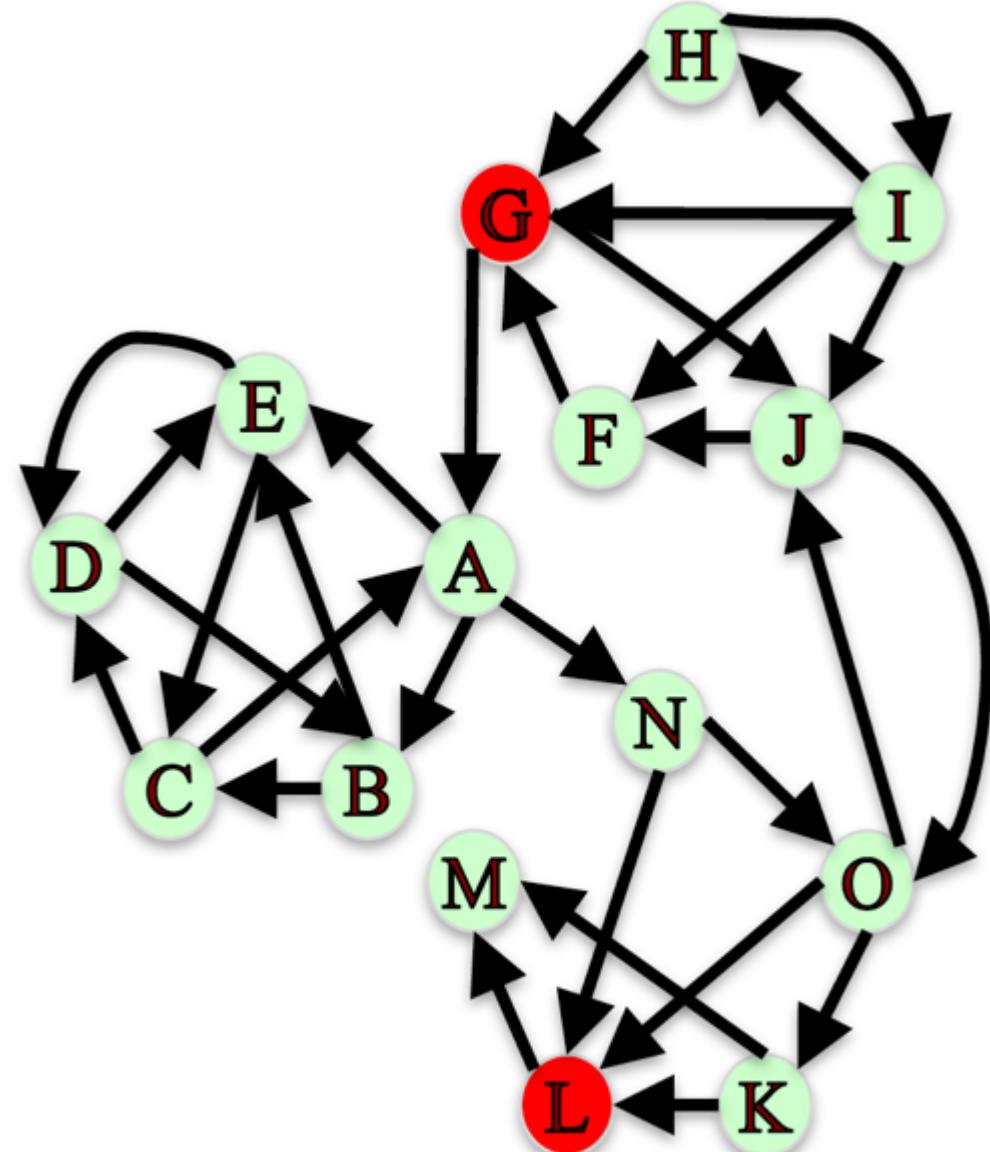
Conectividad de aristas

Si quisiéramos bloquear el mensaje de **G** a **L** mediante la eliminación de **aristas** del grafo, ¿cuántas **aristas** tendríamos que eliminar?

```
nx.edge_connectivity(G, 'G', 'L')
```

¿Cuáles aristas?

```
nx.minimum_edge_cut(G, 'G', 'L')
```



Conectividad de aristas

Si quisiéramos bloquear el mensaje de **G** a **L** mediante la eliminación de **aristas** del grafo, ¿cuántas **aristas** tendríamos que eliminar?

2

¿Cuáles aristas?

$\{('A', 'N'), ('J', 'O')\}$

