CIPFP Mislata
Centre Integrat Públic
Formació Professional Superior

:: U4 ::

Análisis de estructuras en red

2. Distancias, conectividad y comunidades

Curso 2023-24

Apartados

1. Distancias
2. Conectividad
3. Comunidades
4. Consistencia

Apartados

1. Distancias
2. Conectividad
3. Comunidades
4. Consistencia

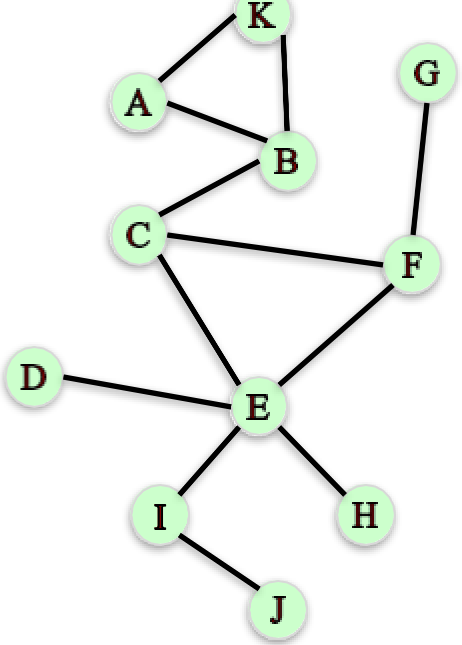
1. Distancias

¿Cuánto de "lejos" está el nodo **A** del nodo **H**?

¿Están los nodos lejos o cerca unos de otros en esta red?

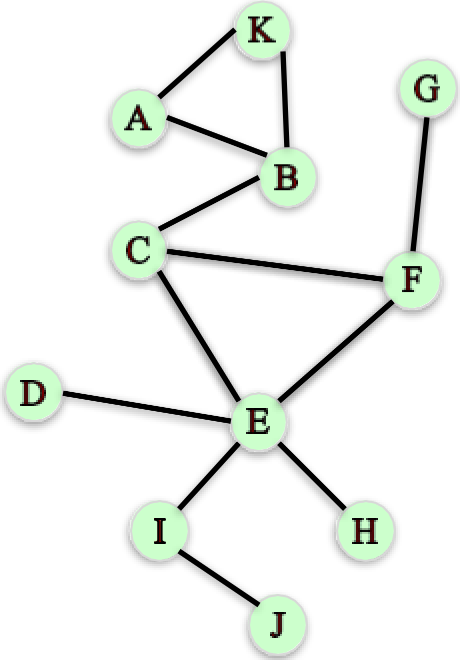
¿Qué nodos están "más cerca" y "más lejos" de otros nodos?

Necesitamos pensar en términos de **distancia** entre los nodos para responder a estas preguntas …



1. Distancias

Camino: secuencia de nodos conectados por un enlace.

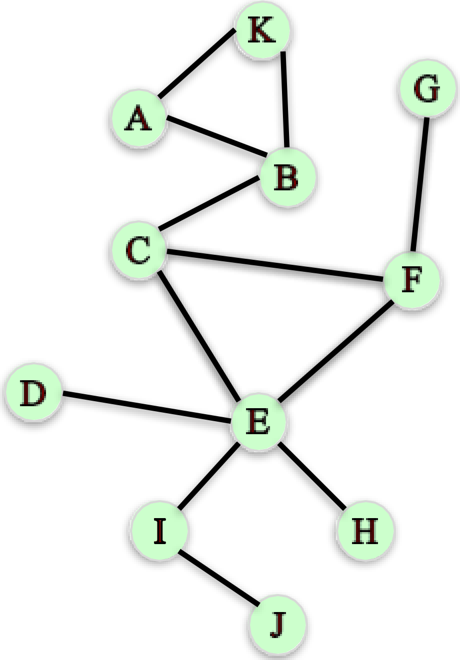


1. Distancias

CIPFP Mislata
Centre Integrat Públic
Formació Professional Superior

Camino: secuencia de nodos conectados por un enlace.

Encuentra dos caminos desde el nodo **G** al nodo **C**:

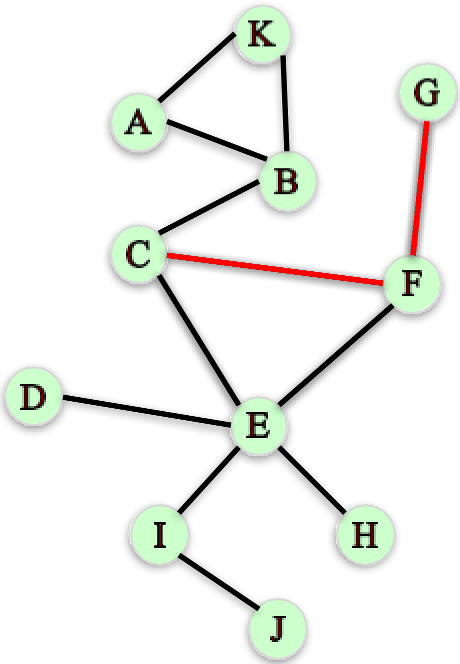


1. Distancias

Camino: secuencia de nodos conectados por un enlace.

Encuentra dos caminos desde el nodo **G** al nodo **C**:

G – F – C



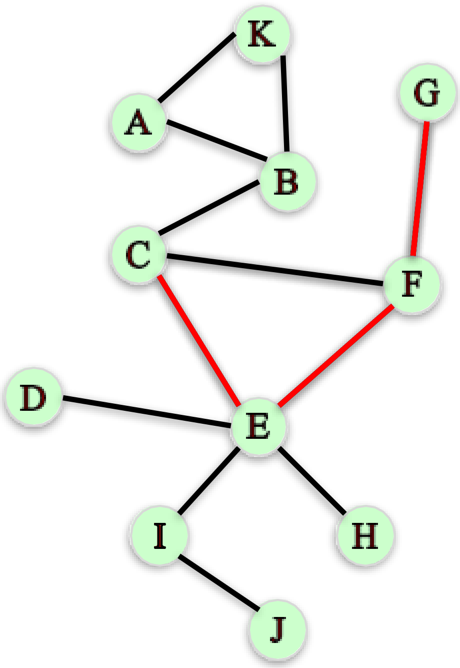
1. Distancias

Camino: secuencia de nodos conectados por un enlace.

Encuentra dos caminos desde el nodo **G** al nodo **C**:

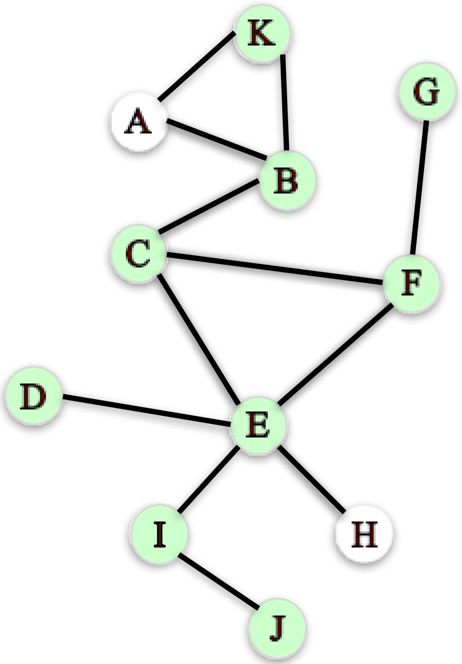
G – F – C

G – F – E – C



1. Distancias

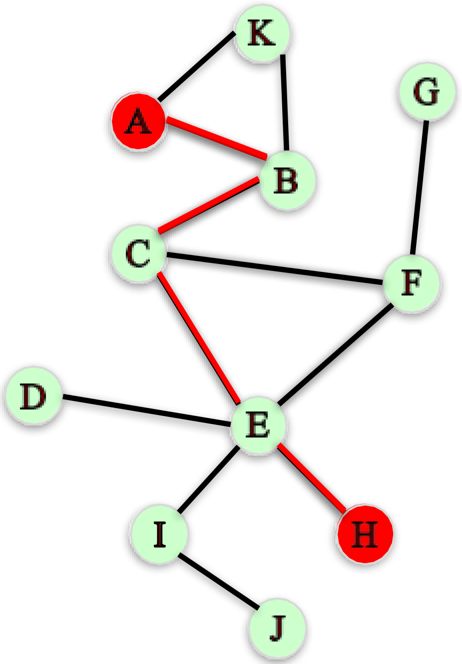
¿A qué distancia está el nodo **A** del nodo **H**?



1. Distancias

¿A qué distancia está el nodo **A** del nodo **H**?

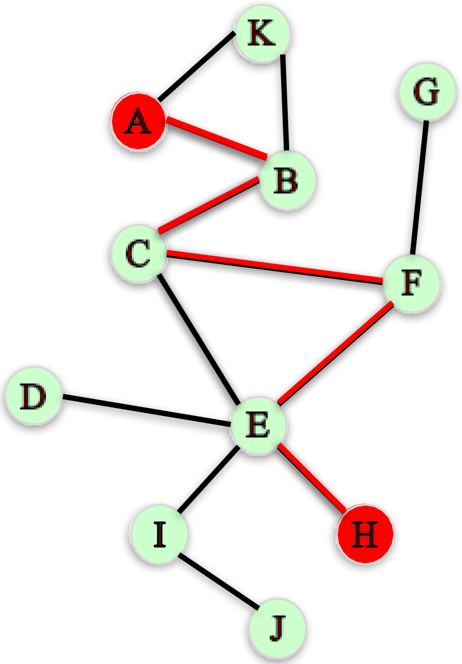
Ruta 1: A – B – C – E – H (4 “saltos”)



1. Distancias

¿A qué distancia está el nodo **A** del nodo **H**?

Ruta 1: A – B – C – E – H (4 “saltos”)
Ruta 2: A – B – C – F – E – H (5 “saltos”)



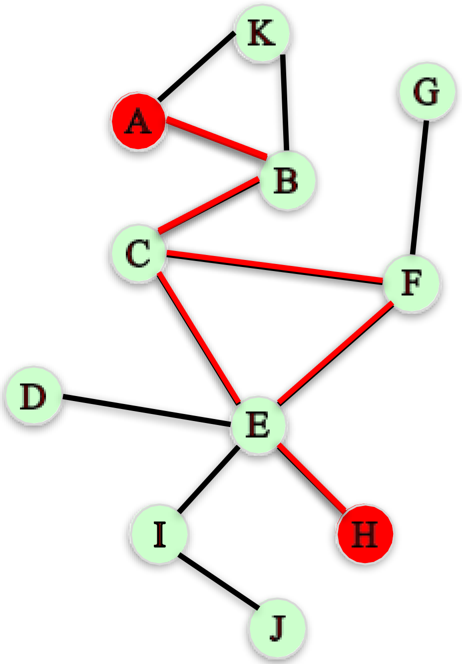
1. Distancias

¿A qué distancia está el nodo **A** del nodo **H**?

Ruta 1: A – B – C – E – H (4 “saltos”)
Ruta 2: A – B – C – F – E – H (5 “saltos”)

Longitud del camino: Número de pasos que contiene de principio a fin.

La ruta 1 tiene una longitud de 4, la ruta 2 tiene una longitud de 5



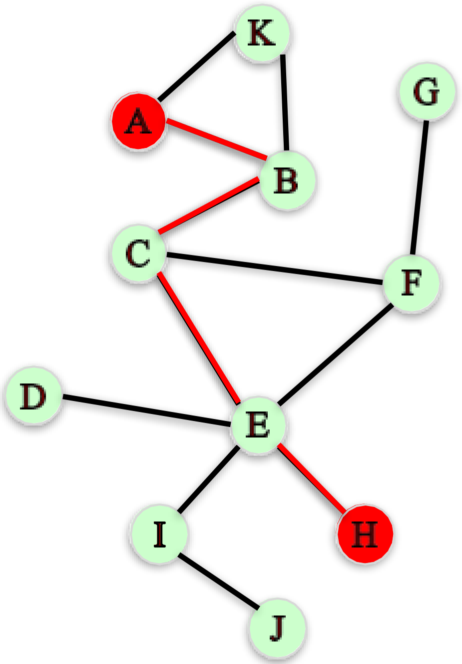
1. Distancias

Distancia entre dos nodos: la longitud del camino más corto entre ellos.

La distancia entre el nodo **A** y **H** es **4**

In: **nx.shortest_path(G,'A', 'H')**
Out: **['A', 'B', 'C', 'E', 'H']**

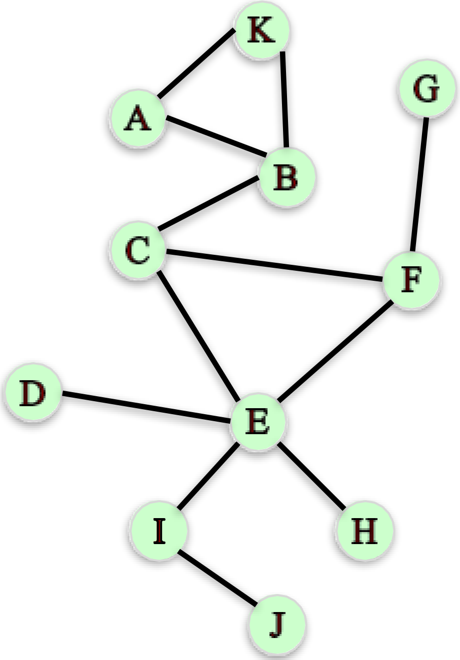
In: **nx.shortest_path_length(G,'A', 'H')**
Out: **4**



1. Distancias

¿Cómo podemos hallar la distancia del nodo **A** a todos los demás nodos?

Fácil de hacer manualmente en redes pequeñas, pero tedioso en redes grandes (reales).



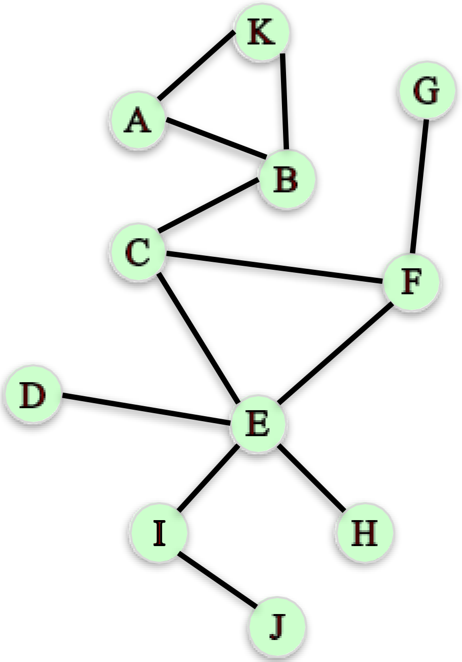
1. Distancias

¿Cómo podemos hallar la distancia del nodo **A** a todos los demás nodos?

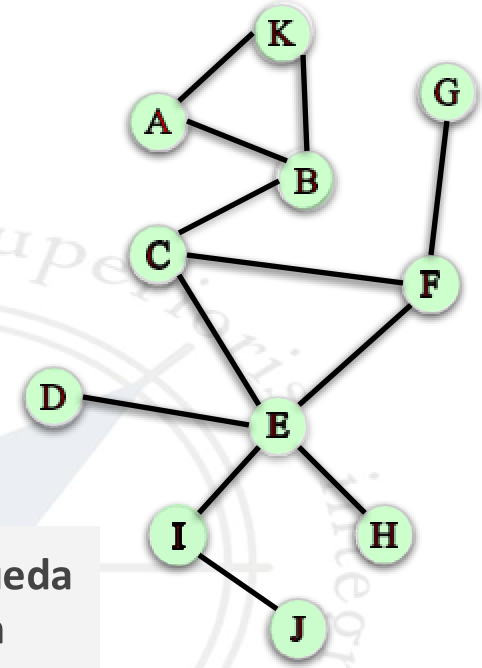
Fácil de hacer manualmente en redes pequeñas, pero tedioso en redes grandes (reales).

Búsqueda en amplitud:

Procedimiento sistemático para calcular las distancias de un nodo a todos los demás nodos en una red grande mediante el "descubrimiento" de nodos por niveles.

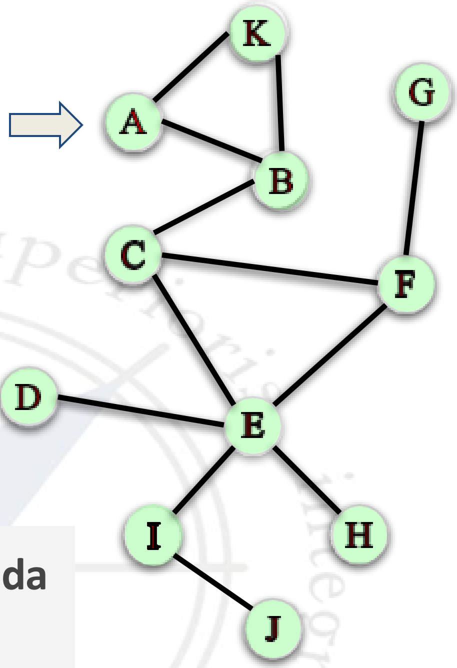


1. Distancias



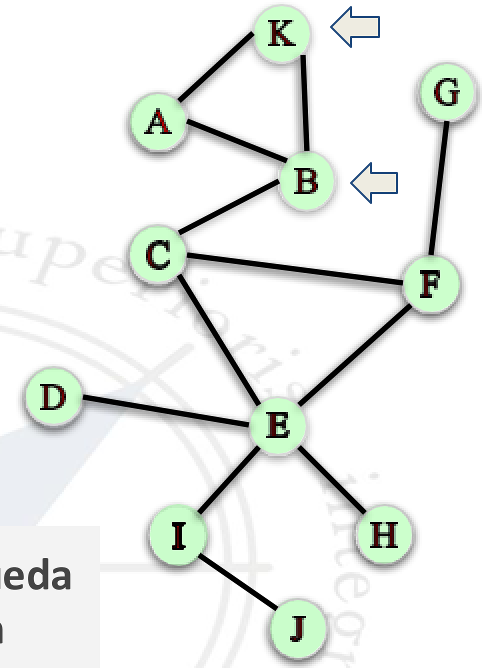
Búsqueda en amplitud

1. Distancias

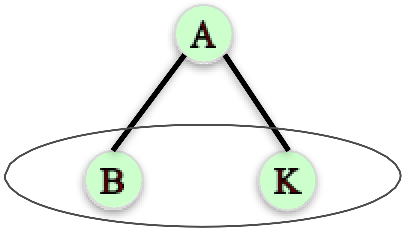


Búsqueda en amplitud

1. Distancias

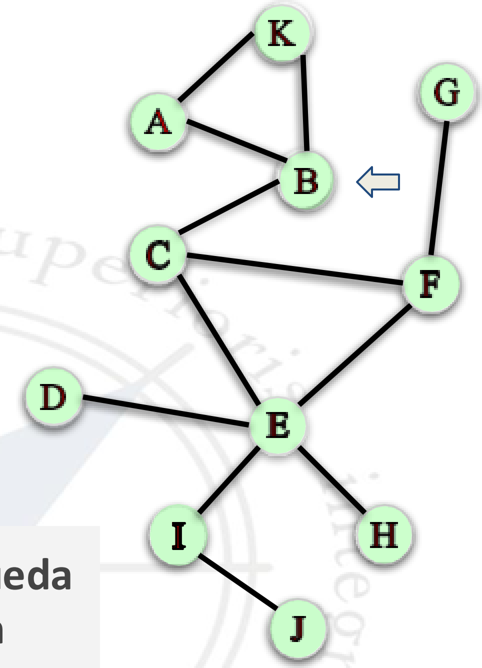


Búsqueda en amplitud

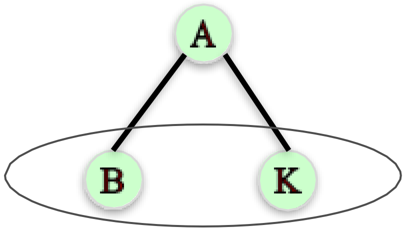


Distancia 1

1. Distancias

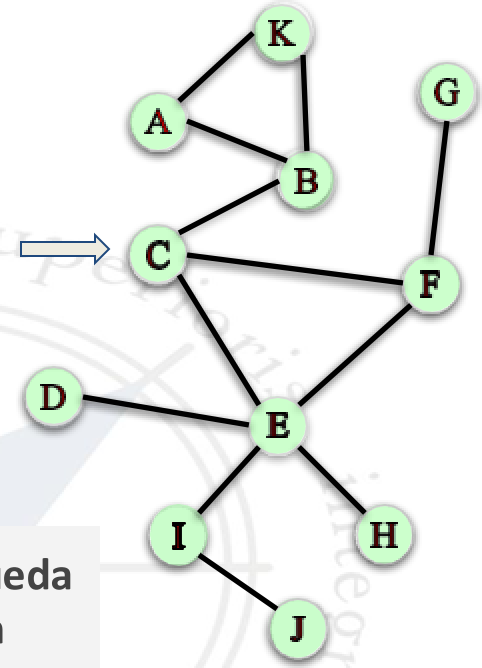


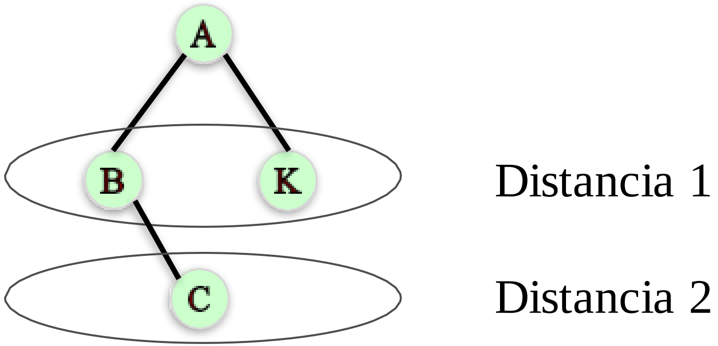
Búsqueda en amplitud



Distancia 1

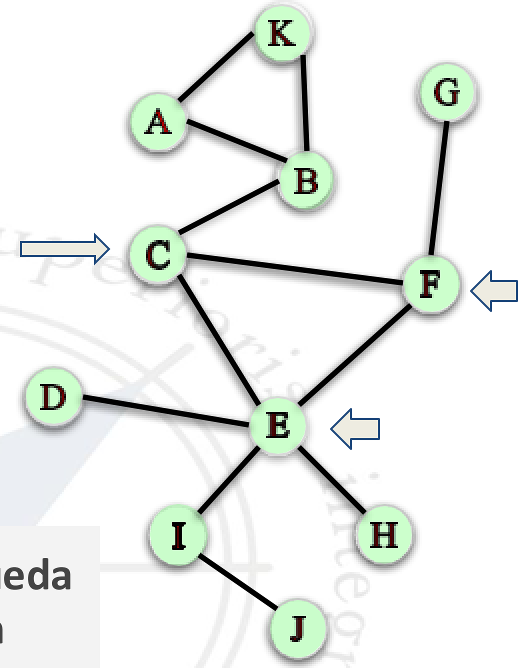
1. Distancias



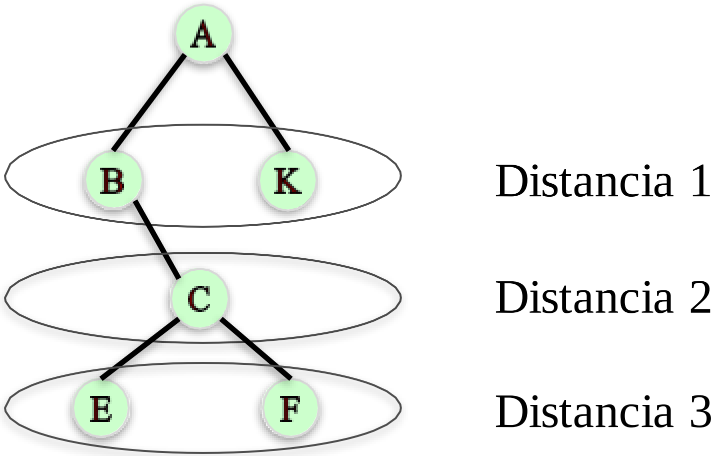


Búsqueda en amplitud

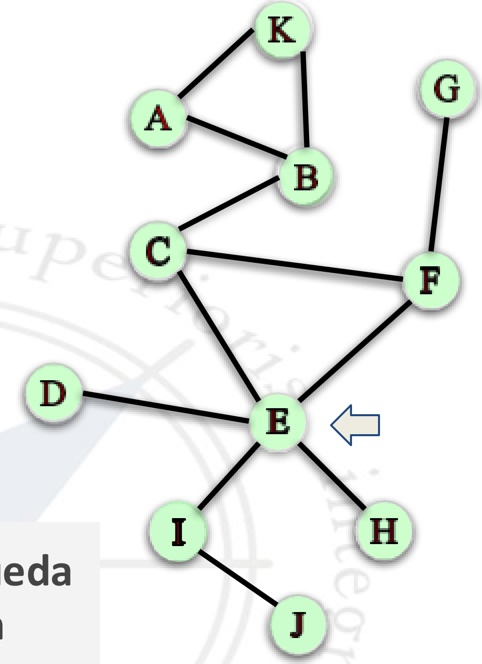
1. Distancias

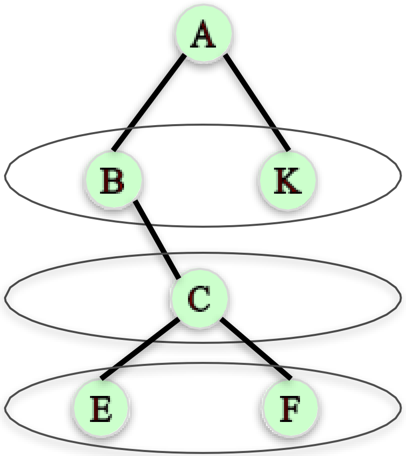


Búsqueda en amplitud



1. Distancias





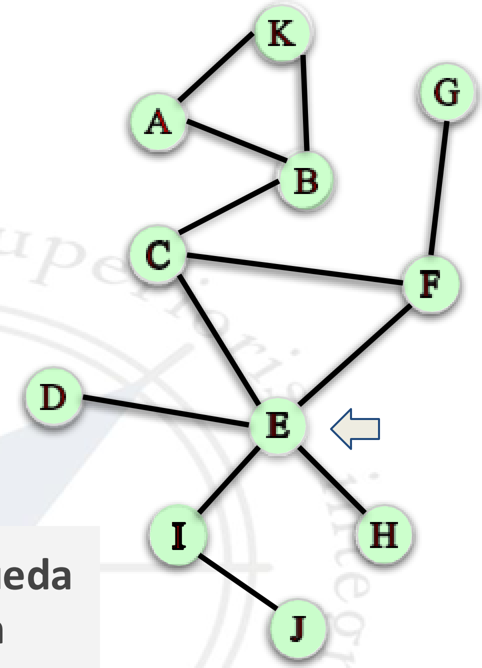
Distancia 1

Distancia 2

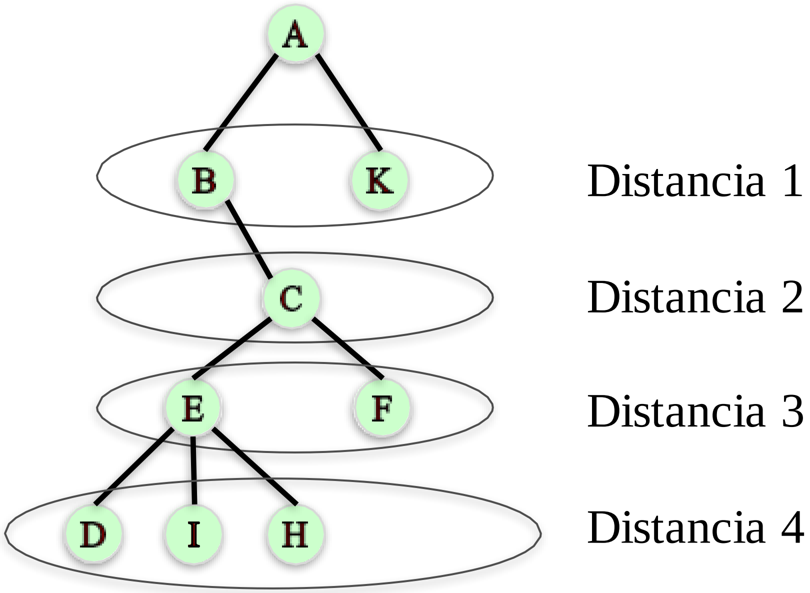
Distancia 3

Búsqueda en amplitud

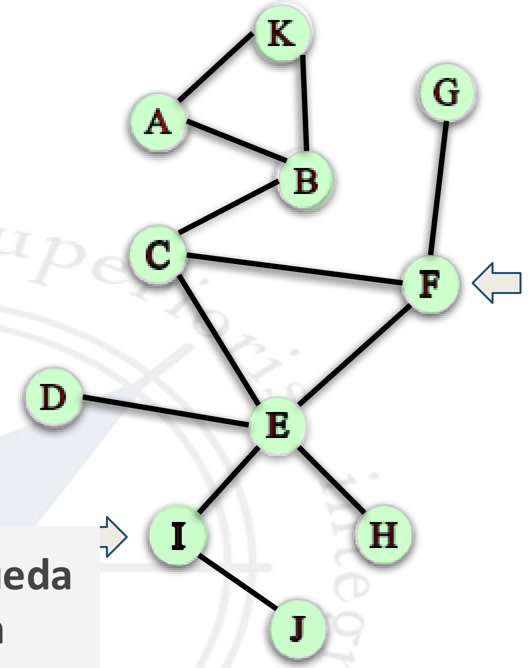
1. Distancias



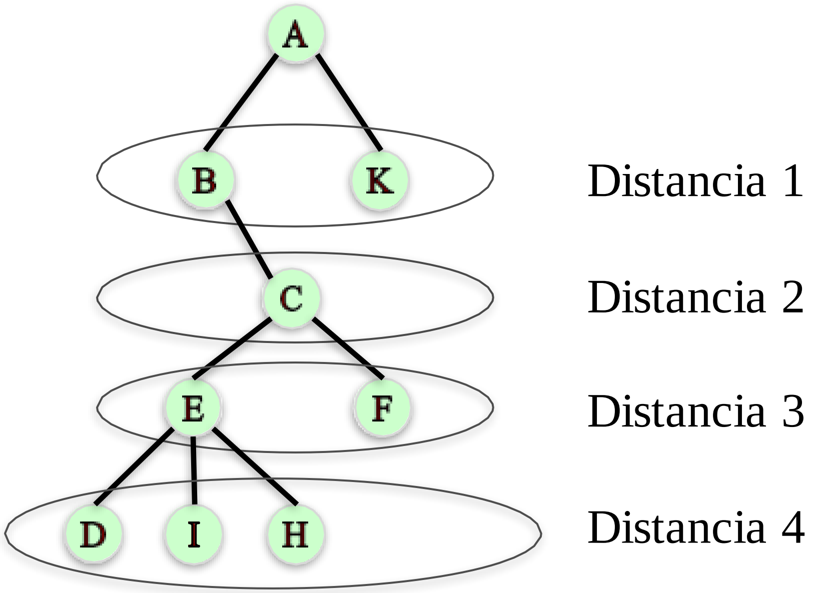
Búsqueda en amplitud



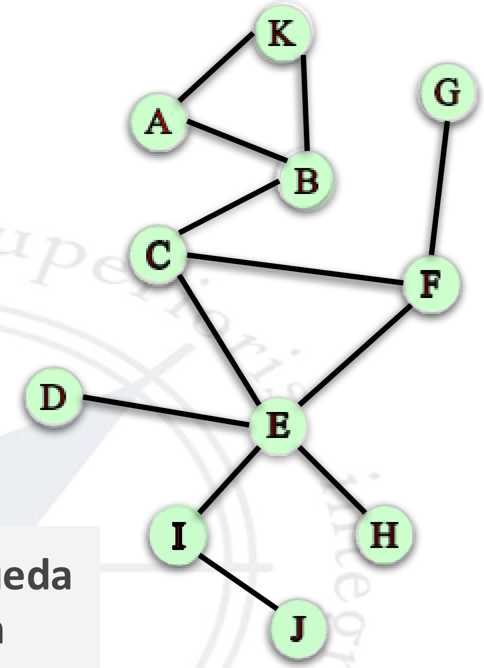
1. Distancias



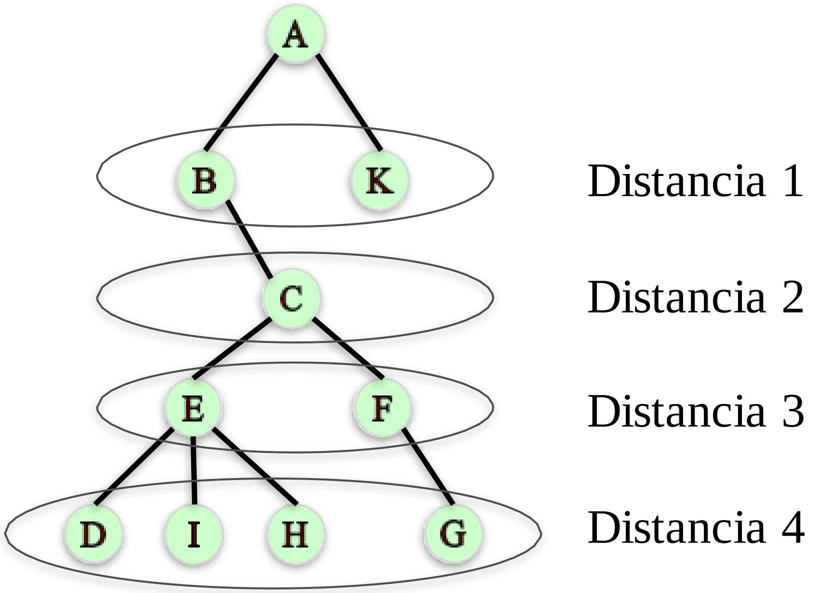
Búsqueda en amplitud



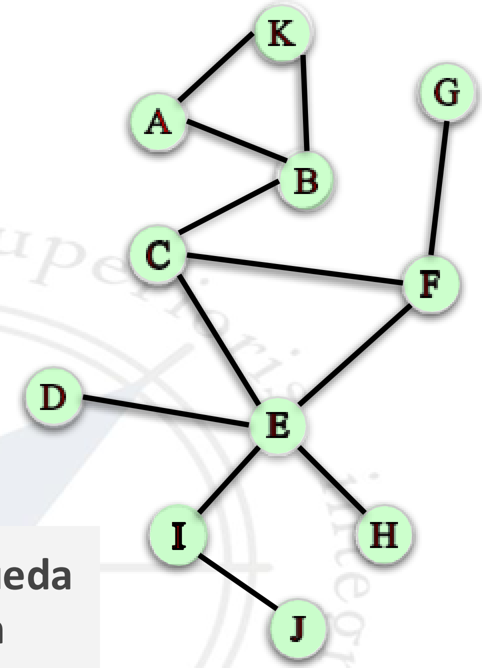
1. Distancias



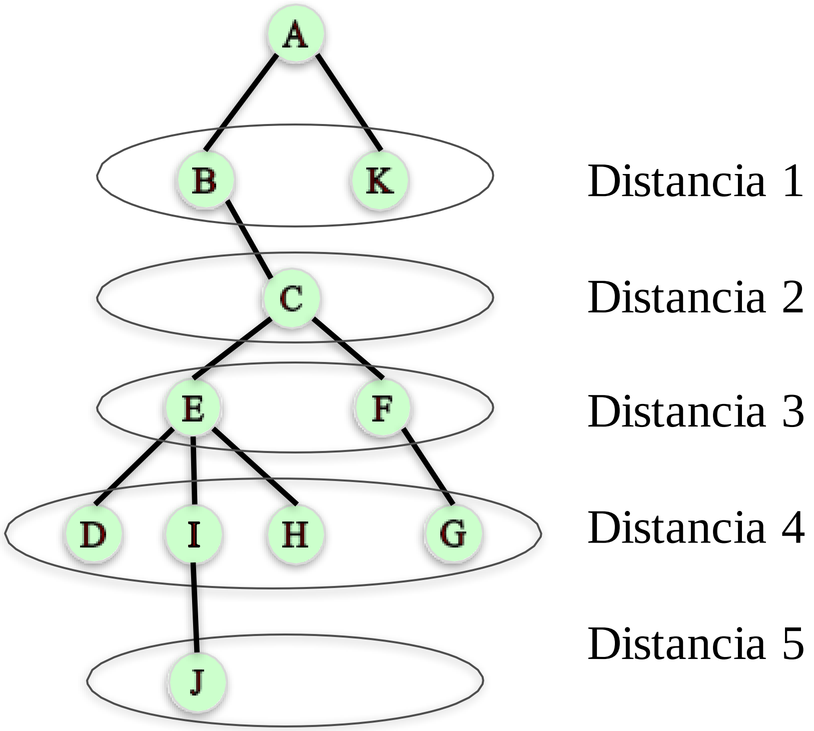
Búsqueda en amplitud



1. Distancias



Búsqueda en amplitud



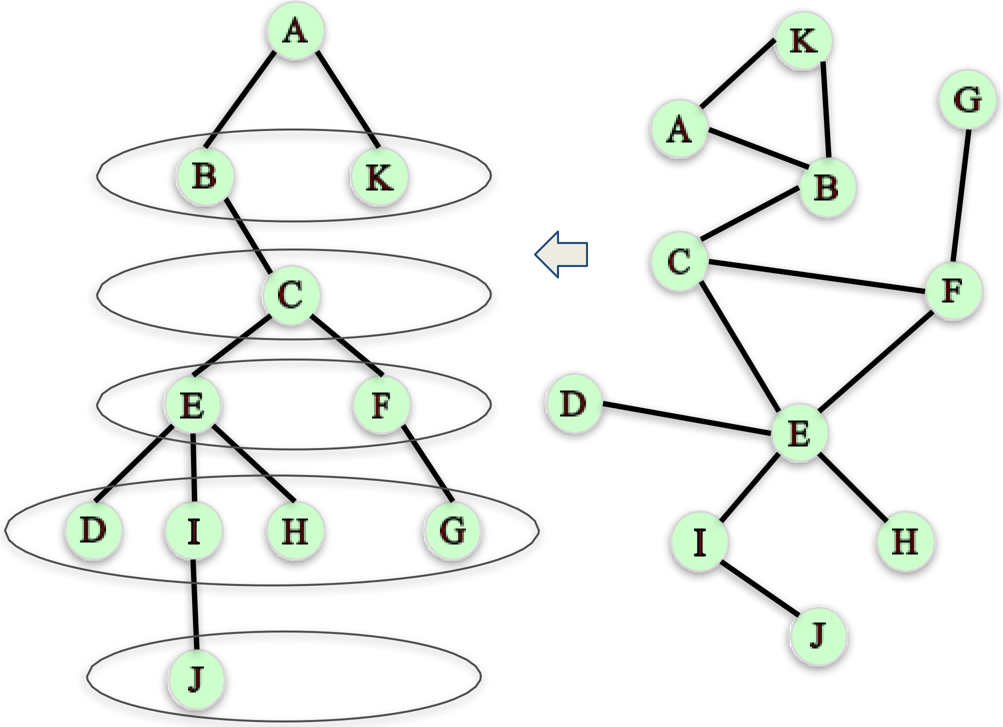
1. Distancias

CIPFP Mislata
Centre Integrat Públic
Formació Professional Superior

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

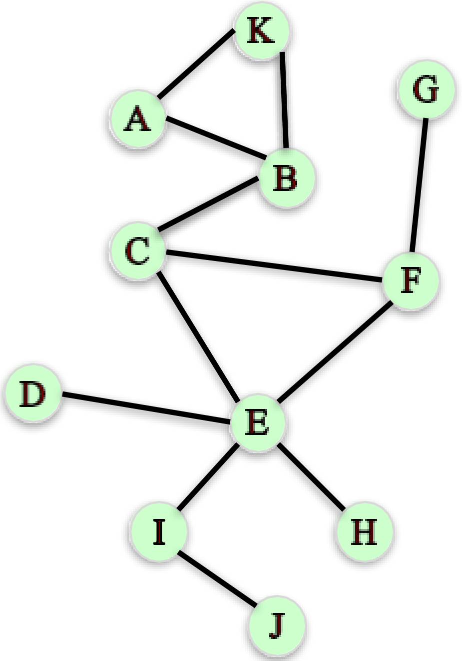
In: T.edges()
Out: [('A', 'K'), ('A', 'B'), ('B', 'C'), ('C', 'E'), ('C', 'F'), ('E', 'I'), ('E', 'H'), ('E', 'D'), ('F', 'G'), ('I', 'J')]

In: nx.shortest_path_length(G,'A')
Out: {'A': 0, 'B': 1, 'C': 2, ‘E': 3, 'D': 4, 'F': 3, 'G': 4, 'H': 4, 'I': 4, 'J': 5, 'K': 1}



1. Distancias

¿Cómo caracterizamos la distancia entre **todos** los pares de nodos en un grafo?

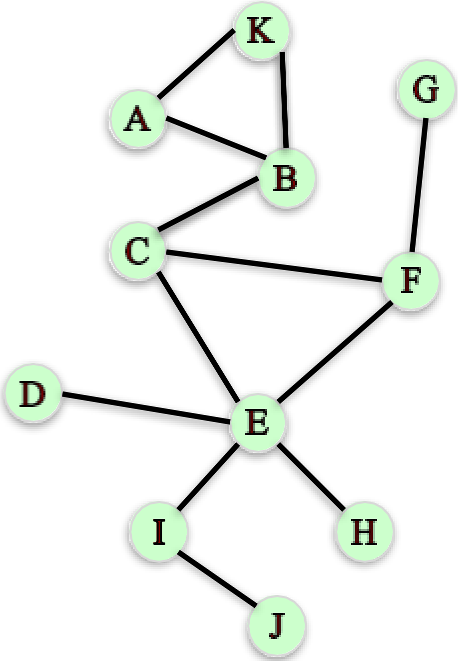


1. Distancias

¿Cómo caracterizamos la distancia entre todos los pares de nodos en un grafo?

Distancia media (más corta) entre cada par de nodos

In: nx.average_shortest_path_length(G)
Out: 2.52727272727



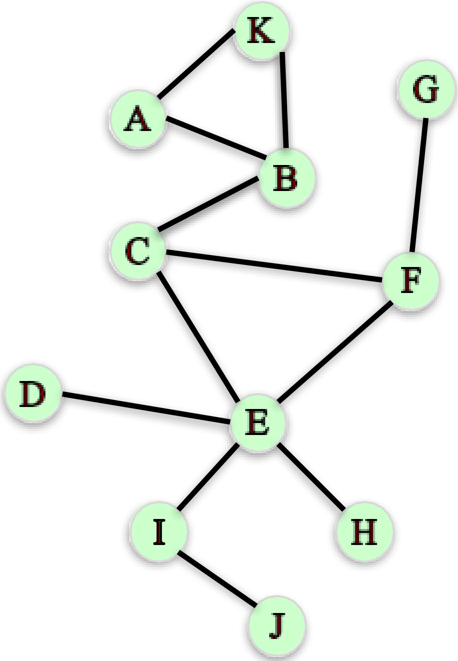
1. Distancias

¿Cómo caracterizamos la distancia entre todos los pares de nodos en un grafo?

Distancia media (más corta) entre cada par de nodos

In: **nx.average_shortest_path_length(G)**
Out: 2.52727272727

Diámetro: distancia máxima entre cualquier par de nodos



1. Distancias

CIPFP Mislata
Centre Integrat Públic
Formació Professional Superior

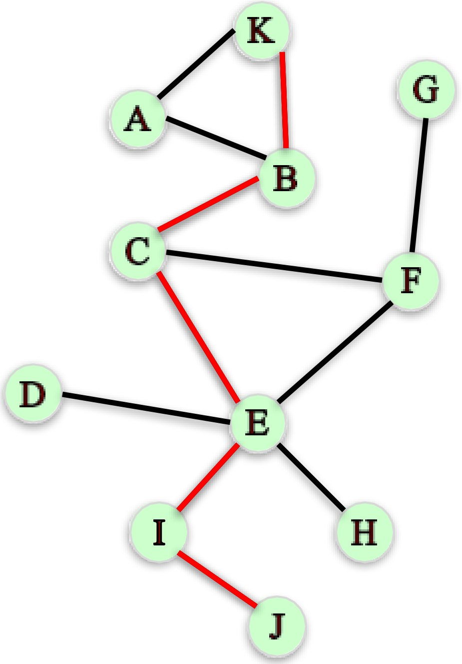
¿Cómo caracterizamos la distancia entre todos los pares de nodos en un grafo?

Distancia media (más corta) entre cada par de nodos

In: nx.average_shortest_path_length(G)
Out: 2.52727272727

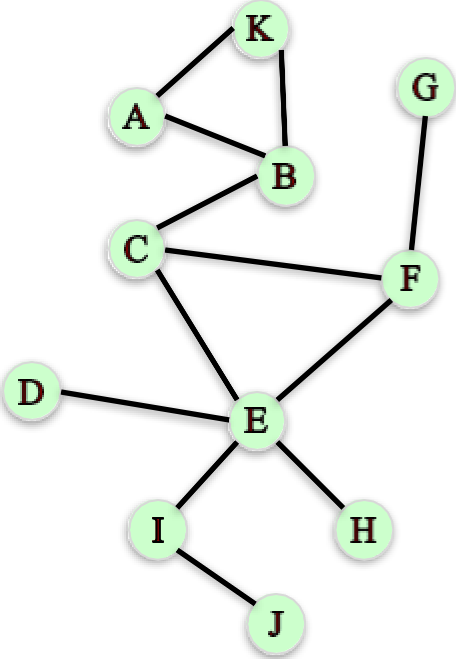
Diámetro: distancia máxima entre cualquier par de nodos

In: nx.diameter (G)
Out: 5



1. Distancias

¿Cómo resumir las distancias entre todos los pares de nodos en un grafo?

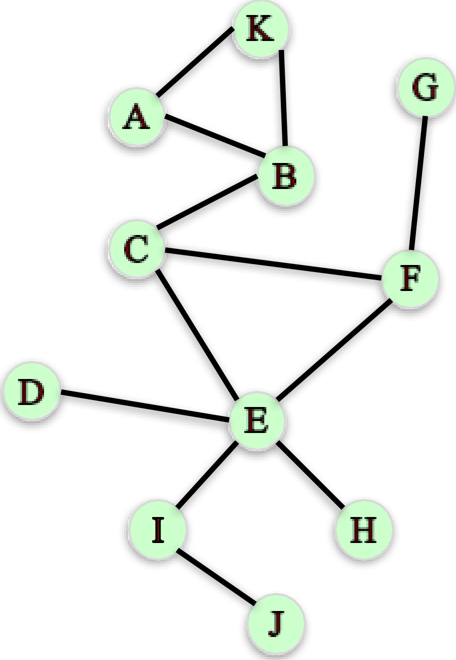


1. Distancias

¿Cómo resumir las distancias entre todos los pares de nodos en un grafo?

La **excentricidad** de un nodo 'n' es la mayor distancia 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}



1. Distancias

CIPFP Mislata
Centre Integrat Públic
Formació Professional Superior

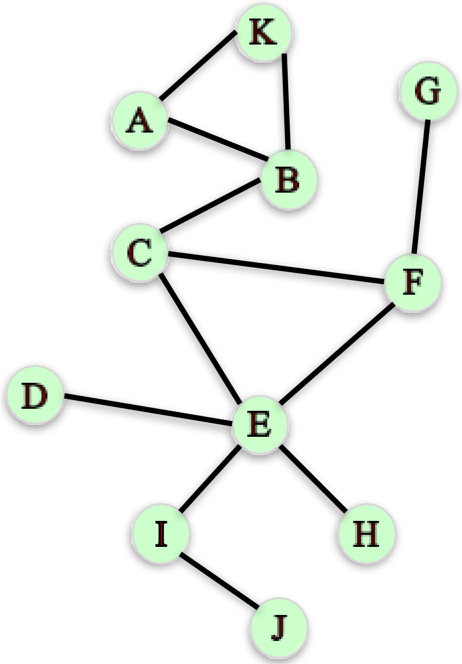
¿Cómo resumir las distancias entre todos los pares de nodos en un grafo?

La **excentricidad** de un nodo ‘n’ es la mayor distancia 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}

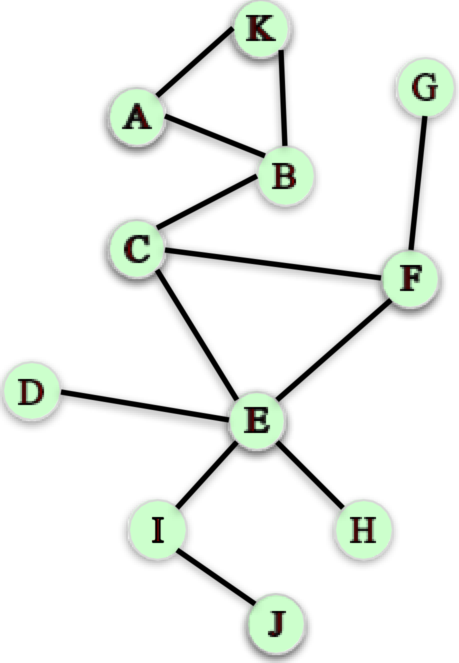
El **radio** de un grafo es la mínima excentricidad.

In: nx.radius(G)
Out: 3



1. Distancias

¿Cómo resumir las distancias entre todos los pares de nodos en un grafo?

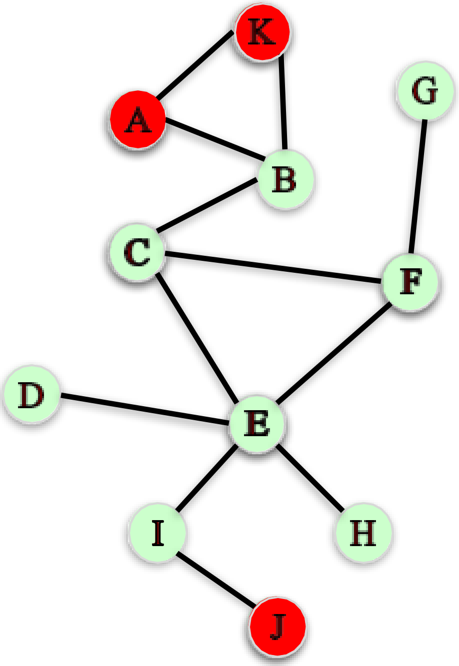


1. Distancias

¿Cómo resumir las distancias entre todos los pares de nodos en un grafo?

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

In: nx.periphery(G)
Out: ['A', 'K', 'J']



1. Distancias

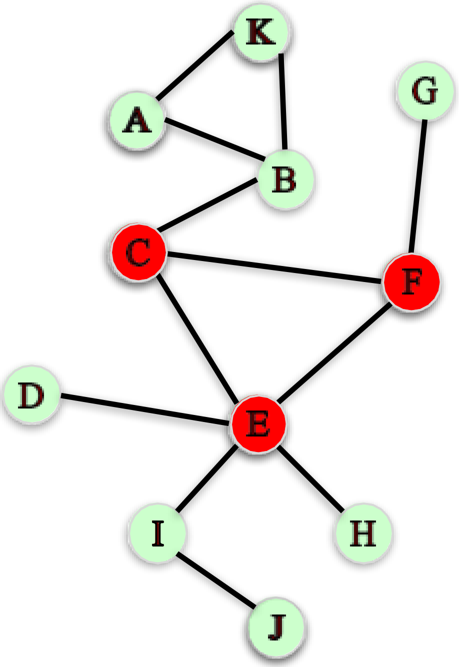
¿Cómo resumir las distancias entre todos los pares de nodos en un grafo?

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

In: nx.periphery(G)
Out: ['A', 'K', 'J']

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

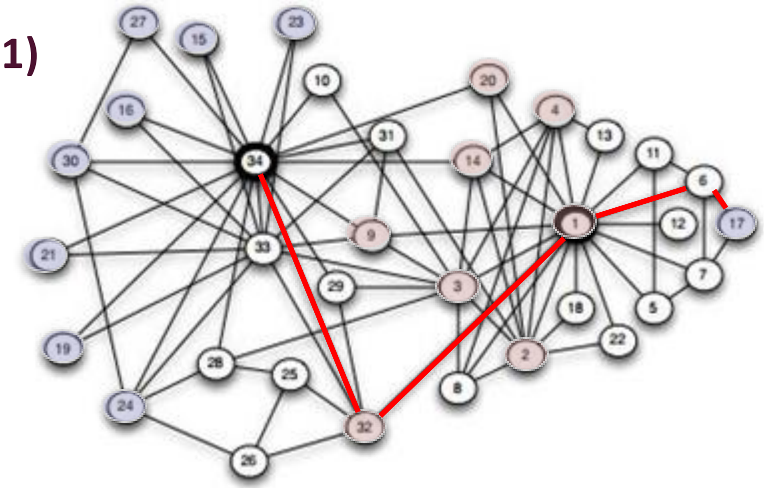
In: nx.center(G)
Out: ['C', 'E', 'F']



Ejemplo

Red de un club de kárate

```
G = nx.karate_club_graph()
G = nx.convert_node_labels_to_integers(G, first_label=1)
```



Ruta media más corta = 2,41

Radio = 3

Diámetro = 5

Centro = [1, 2, 3, 4, 9, 14, 20, 32]

Periferia: [15, 16, 17, 19, 21, 23, 24, 27, 30]

Red de amistad en un club de kárate de **34** personas

El nodo 34 parece bastante "central". Sin embargo, tiene una distancia de 4 al nodo 17.

1. Distancias

Resumen

Distancia entre dos nodos: longitud del camino más corto entre ellos.

La **excentricidad** de un nodo n: la mayor distancia entre n y todos los demás nodos.

Caracterización de **distancias** en una red:

- **Distancia media (más corta) entre cada par de nodos**
- **Diámetro**: distancia máxima entre cualquier par de nodos
- **Radio**: la mínima excentricidad en el gráfico

Identificación de nodos centrales y periféricos:

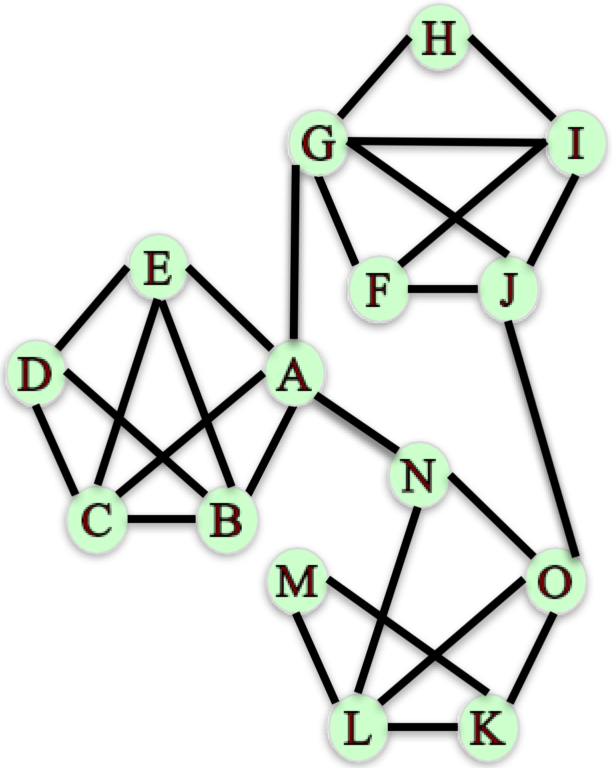
- La **periferia** es el conjunto de nodos con excentricidad = diámetro
- El **centro** es el conjunto de nodos con excentricidad = radio

Apartados

1. Distancias
2. Conectividad
3. Comunidades
4. Consistencia

2. Conectividad
Grafos no dirigidos

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

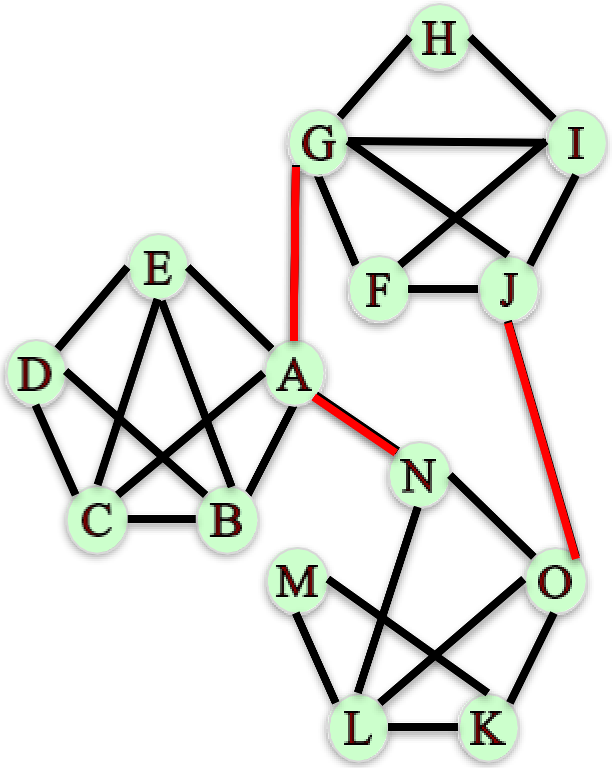


2. Conectividad
Grafos no dirigidos

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

In: **nx.is_connected(G)**

Out: **True**



2. Conectividad

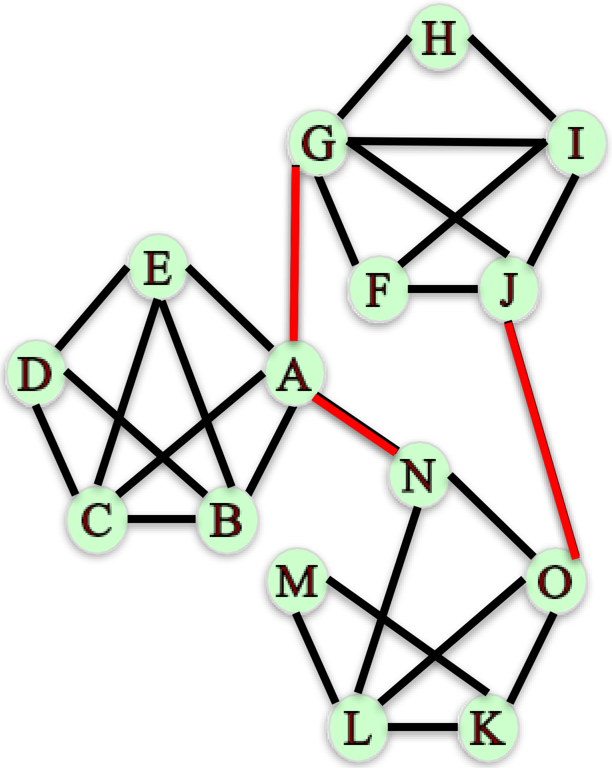
Grafos no dirigidos

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

In: **nx.is_connected(G)**

Out: **True**

¿Qué pasa si eliminamos los enlaces **A—G**, **A—N** y **J—O?**



2. Conectividad

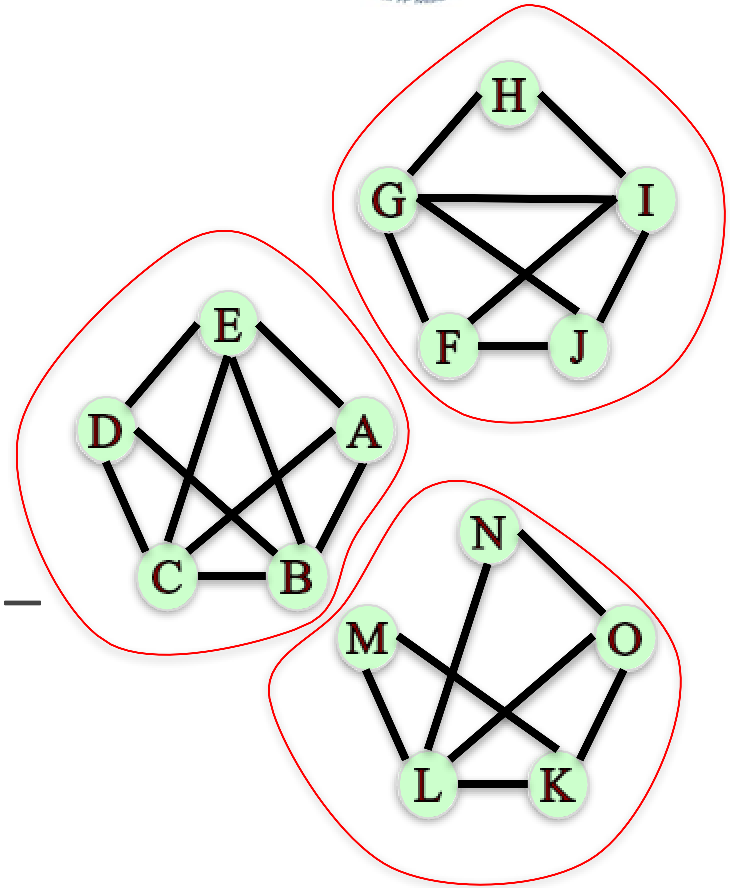
Grafos no dirigidos

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

In: **nx.is_connected(G)**
Out: **True**

¿Qué pasa si eliminamos los enlaces **A—G**, **A—N** y **J—O?**

El grafo se desconecta: no hay camino entre los nodos en las tres "comunidades" diferentes.



2. Conectividad

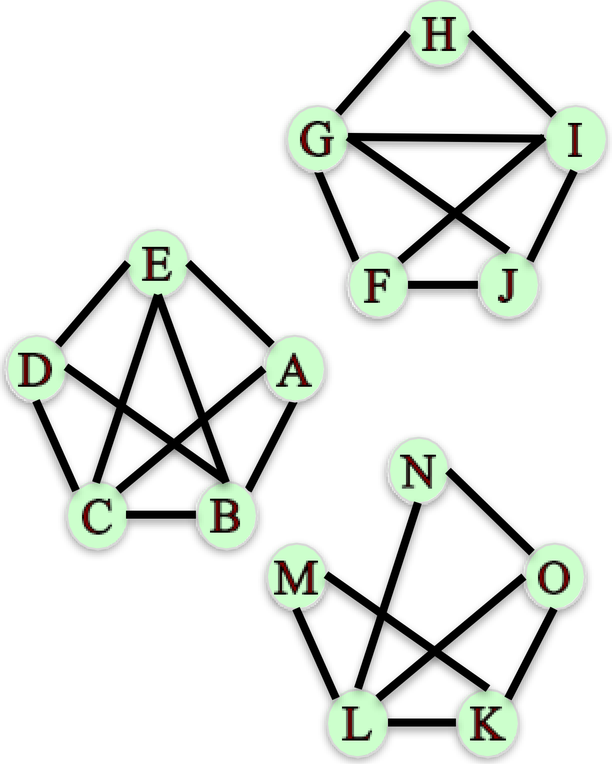
Grafos no dirigidos

Componente conectado

Un subconjunto de nodos donde:

1. Cada nodo del subconjunto tiene una ruta a todos los demás nodos.

1. Ningún otro nodo tiene un camino a ningún nodo del subconjunto.



2. Conectividad

Grafos no dirigidos

CIPFP Mislata
Centre Integrat Públic
Formació Professional Superior

Componente conectado

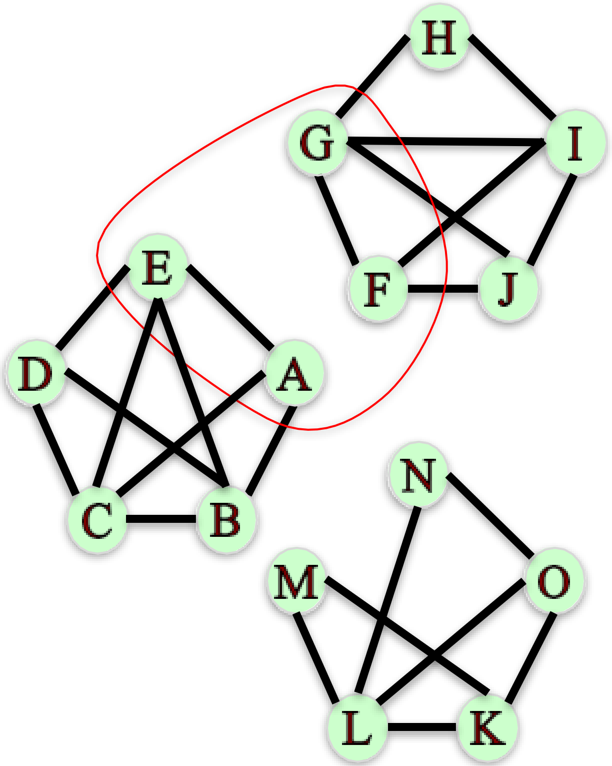
Un subconjunto de nodos donde:

1. Cada nodo del subconjunto tiene una ruta a todos los demás nodos.

1. Ningún otro nodo tiene un camino a ningún nodo del subconjunto.

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

> No, no hay camino entre los nodos A y F.



2. Conectividad

Grafos no dirigidos

Componente conectado

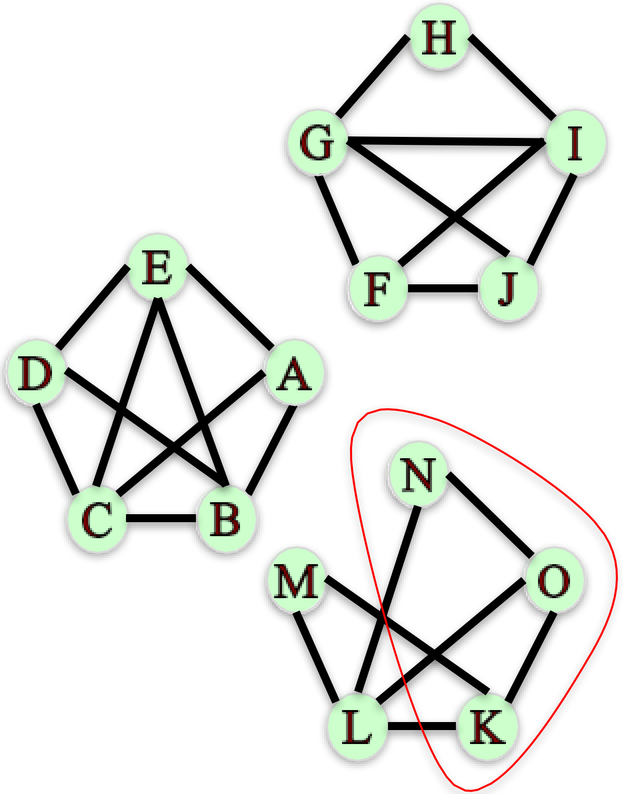
Un subconjunto de nodos donde:

1. Cada nodo del subconjunto tiene una ruta a todos los demás nodos.

1. Ningún otro nodo tiene un camino a ningún nodo del subconjunto.

¿Es el subconjunto {N, O, K} un componente conectado?

> No, el nodo L tiene un camino hacia N, O y K.



2. Conectividad

Grafos no dirigidos

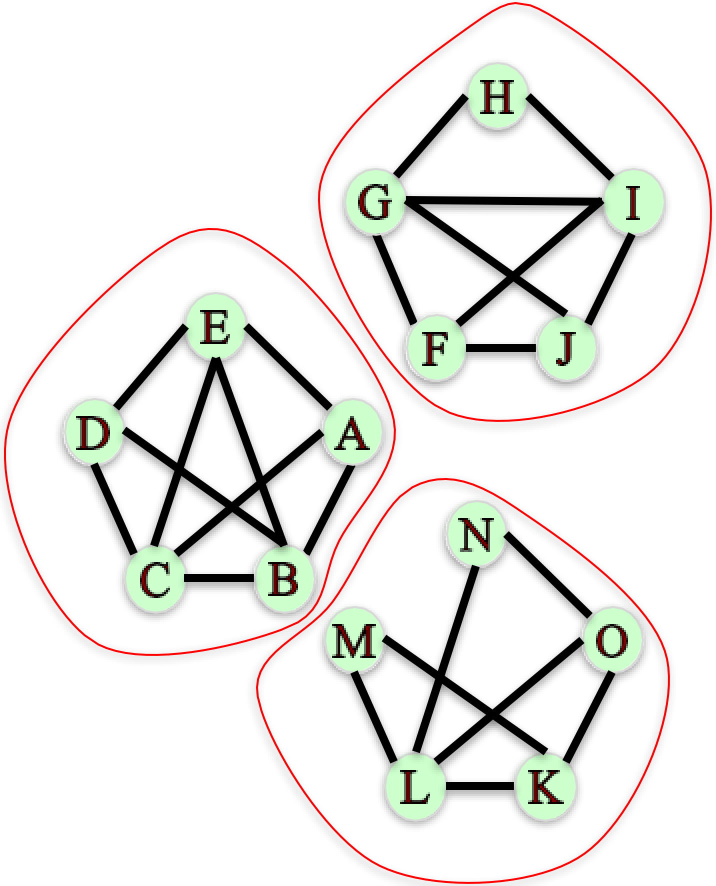
Componente conectado

Un subconjunto de nodos donde:

1. Cada nodo del subconjunto tiene una ruta a todos los demás nodos.
1. Ningún otro nodo tiene un camino a ningún nodo del subconjunto.

¿Cuáles son los componentes conectados en este gráfico?

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



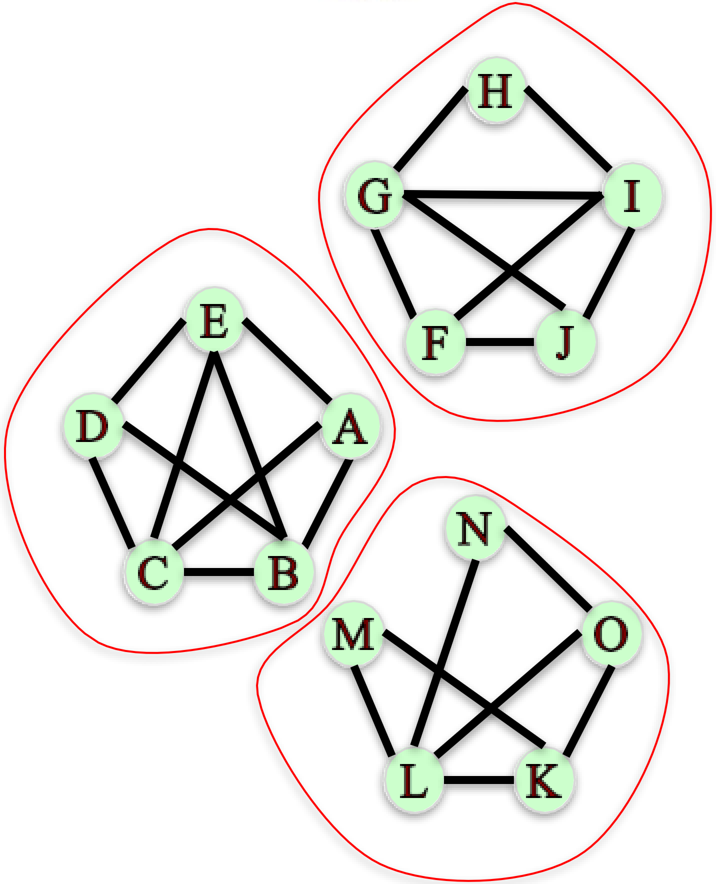
2. Conectividad

Grafos no dirigidos

In: **nx.number_connected_components(G)**
Out: 3

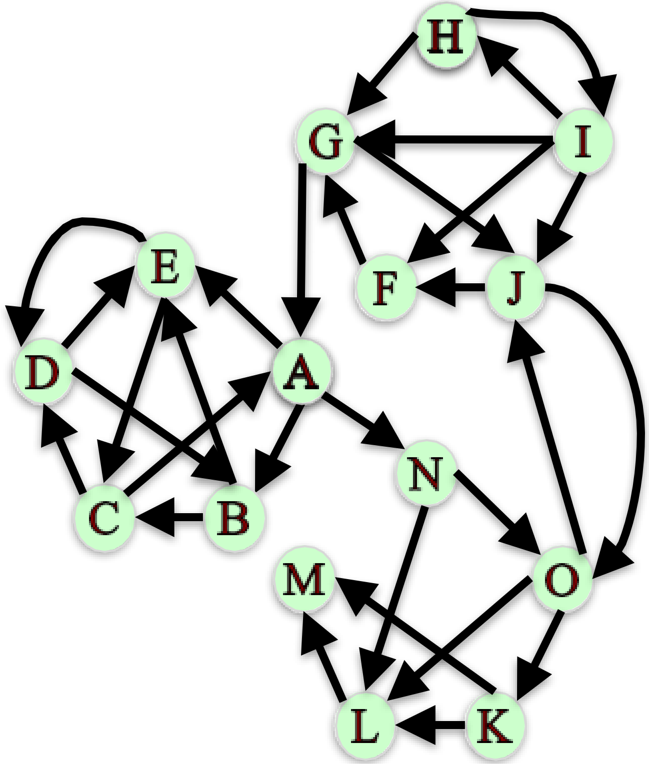
In: **sorted(nx.connected_components(G))**
Out: [{'A', 'B', 'C', 'D', 'E'}, {'F', 'G', 'H', 'I', 'J'}, {'K', 'L', 'M', 'N', 'O'}]

In: **nx.node_connected_component(G, 'M')**
Out: {'K', 'L', 'M', 'N', 'O'}



2. Conectividad
Grafos dirigidos

Un **grafo dirigido** está **fuertemente conectado** si, para cada par de nodos ‘u’ y ‘v’, hay un camino directo de ‘u’ a ‘v’ y un camino dirigido de ‘v’ a ‘u’.



2. Conectividad

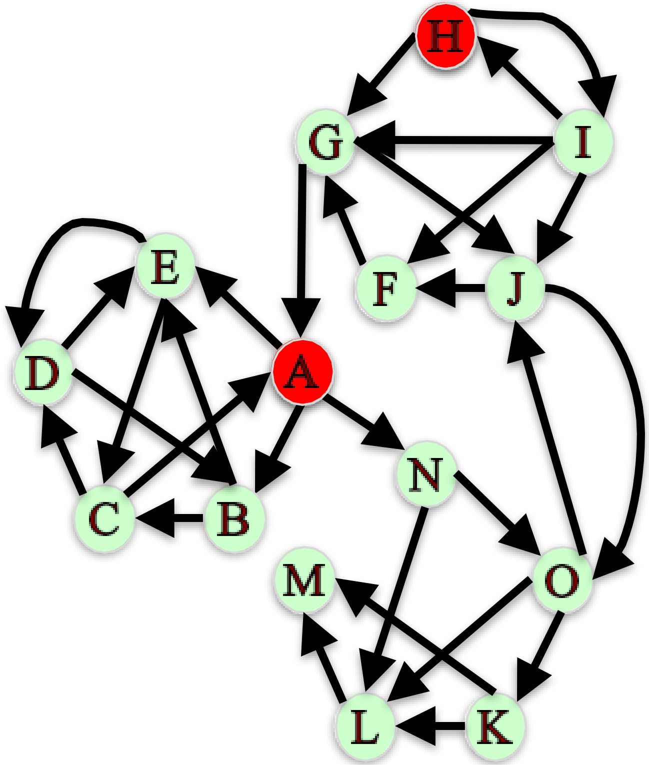
Grafos dirigidos

Un **grafo dirigido** está **fuertemente conectado** si, para cada par de nodos ‘u’ y ‘v’, hay un camino directo de ‘u’ a ‘v’ y un camino dirigido de ‘v’ a ‘u’.

In: nx.is_strongly_connected(G)

Out: False

Nota: Existe un camino dirigido de **H** a **A**, pero no uno directo desde **A** a **H**



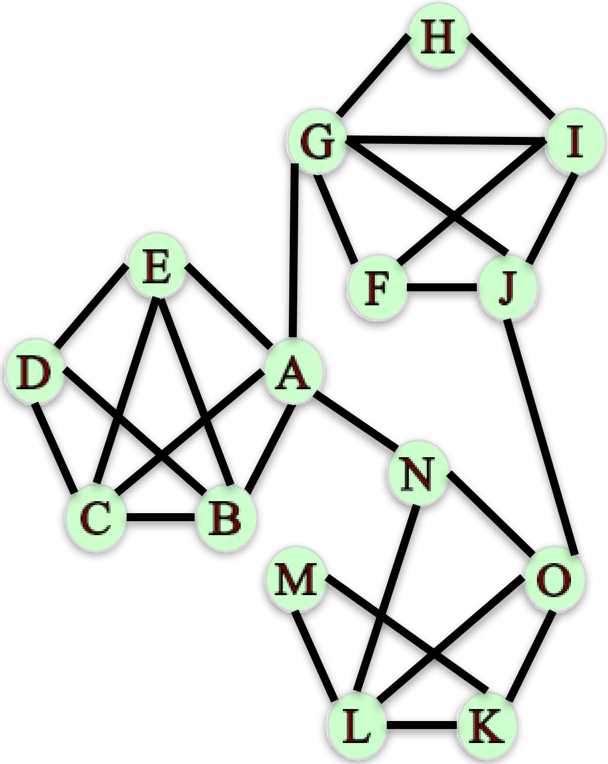
2. Conectividad

Grafos dirigidos

Un **grafo dirigido** está **débilmente conectado** si reemplazar todos los enlaces dirigidos con enlaces no dirigidos produce un grafo no dirigido conectado.

In: nx.is_weakly_connected(G)

Out: True



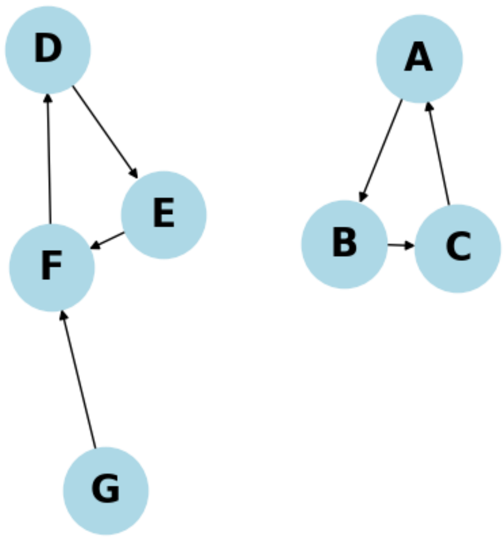
2. Conectividad

Grafos dirigidos

Componente fuertemente conectado:

```
import networkx as nx

G = nx.DiGraph()

G.add_nodes_from(['A', 'B', 'C', 'D', 'E', 'F', 'G'])

G.add_edges_from([('A', 'B'), ('B', 'C'), ('C', 'A'), ('D', 'E'), ('E', 'F'),
('F', 'D'), ('G', 'F')])

nx.draw(G, with_labels=True, node_color='lightblue', node_size=200
0, font_size=20, font_weight='bold')

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

[{'A', 'C', 'B'}, {'F', 'E', 'D'}, {'G'}]



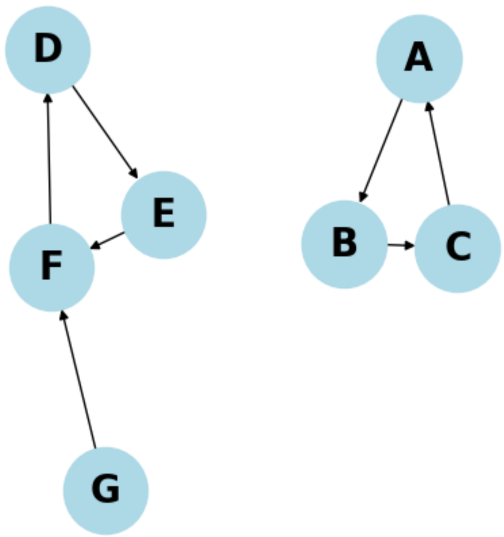
2. Conectividad

Grafos dirigidos

Componente débilmente conectado:

```
import networkx as nx

G = nx.DiGraph()

G.add_nodes_from(['A', 'B', 'C', 'D', 'E', 'F', 'G'])

G.add_edges_from([('A', 'B'), ('B', 'C'), ('C', 'A'), ('D', 'E'), ('E', 'F'),
('F', 'D'), ('G', 'F')])

nx.draw(G, with_labels=True, node_color='lightblue', node_size=200
0, font_size=20, font_weight='bold')

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

[{'A', 'C', 'B'}, {'F', 'E', 'G', 'D'}]

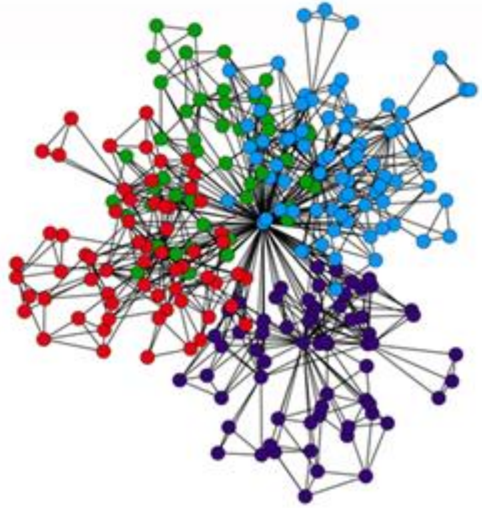


Apartados

1. Distancias
2. Conectividad
3. Comunidades
4. Consistencia

3. Comunidades

Detección de comunidades:
Identificar subconjuntos de nodos que están muy conectados entre ellos y, al mismo tiempo, poco conectados con el resto de nodos de la red.



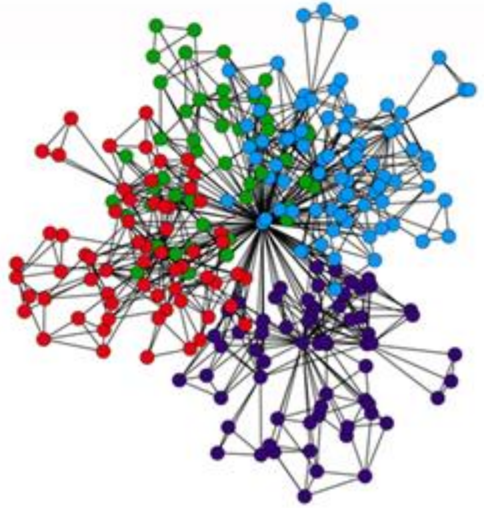
Características:
Los nodos dentro de una comunidad suelen compartir características similares, como intereses comunes o características estructurales. Además, las comunidades pueden ser superpuestas o no superpuestas.

Métodos de detección:
Algoritmos basados en la estructura del grafo, en la optimización de la modularidad y en algoritmos de aprendizaje automático

3. Comunidades

Ejemplo: algoritmo de Lovain

Se basa en la optimización de la **modularidad** del grafo.



¿A qué nos referimos con **modularidad**?

1. Mide la calidad de la partición de un grafo en comunidades

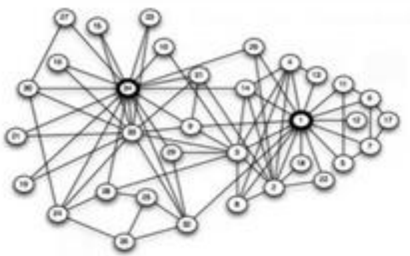
2. Premisa: una buena partición tendrá una densidad de aristas alta entre los nodos de la misma comunidad y baja entre los nodos de diferentes comunidades.

3. Comunidades

CIPFP Mislata
Centre Integrat Públic
Formació Professional Superior

Ejemplo: algoritmo de Lovain

```
import networkx as nx
import community

# Creamos un grafo de ejemplo
G = nx.karate_club_graph()

# Ejecutamos el algoritmo Lovain
partition = community.best_partition(G)

# Imprimimos la partición
print(partition)
```



Friendship network in a 34-person karate club
[Zachary 1977]

Out:

{0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 1, 9: 1, 10: 0, 11: 0, 12: 0, 13: 0, 14: 1, 15: 1, 16: 0, 17: 0, 18: 1, 19: 0, 20: 1, 21: 0, 22: 1, 23: 1, 24: 1, 25: 1, 26: 1, 27: 1, 28: 1, 29: 1, 30: 1, 31: 1, 32: 1, 33: 1}

Apartados

1. Distancias
2. Conectividad
3. Comunidades
4. Consistencia

4. Consistencia

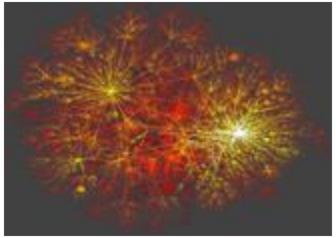
Consistencia de la red: la capacidad de una red para mantener sus propiedades estructurales generales cuando enfrenta fallas o ataques.

Tipo de ataques: eliminación de nodos o enlaces.

Propiedades estructurales: conectividad.

Ejemplos: cierres de aeropuertos, interrupciones en el enrutamiento en Internet, cortes en las líneas eléctricas o en carreteras, …

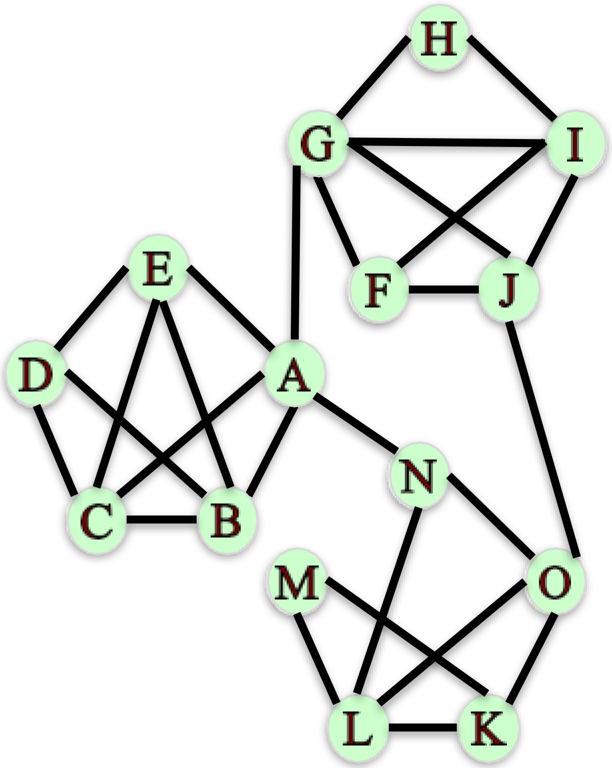
Red de vuelos directos alrededor del mundo
[Bio.Diaspora]



Conectividad en Internet [K. C. Claffy]

4. Consistencia

¿Cuál es el menor número de **nodos** que se puede quitar de este grafo para *desconectarlo*?



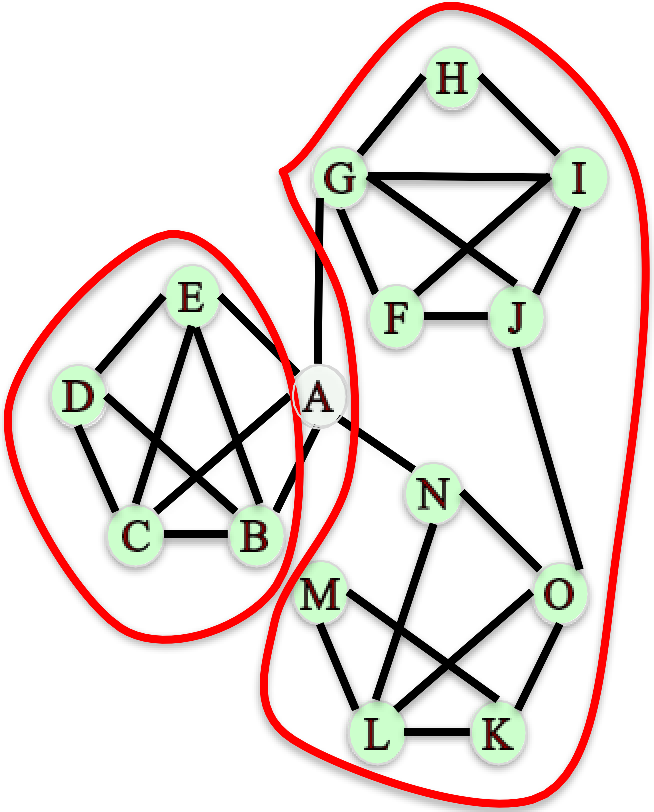
4. Consistencia

¿Cuál es el menor número de **nodos** que al ser eliminados hacen que el grafo quede desconectado?

In: **nx.node_connectivity(G_un)**
Out: 1

¿Qué nodo?

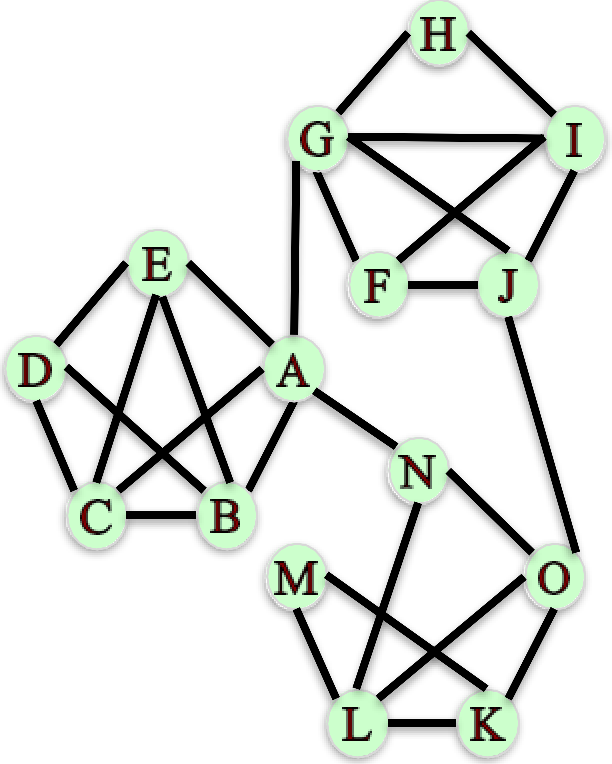
In: **nx.minimum_node_cut(G_un)**
Out: {'A'}



4. Consistencia

¿Cuál es el menor número de **enlaces** que de perderse hacen que el grafo quede desconectado?

In: **nx.edge_connectivity(G_un)**
Out: 2



4. Consistencia

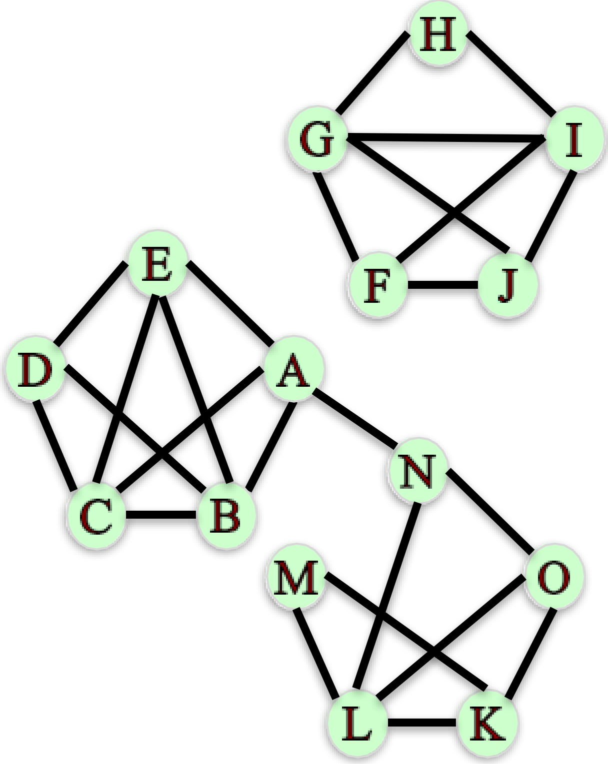
CIPFP Mislata
Centre Integrat Públic
Formació Professional Superior

¿Cuál es el menor número de **enlaces** que de perderse hacen que el grafo quede desconectado?

In: nx.edge_connectivity(G_un)
Out: 2

¿Qué enlaces serían?

In: nx.minimum_edge_cut(G_un)
Out: {('A', 'G'), ('O', 'J')}



4. Consistencia

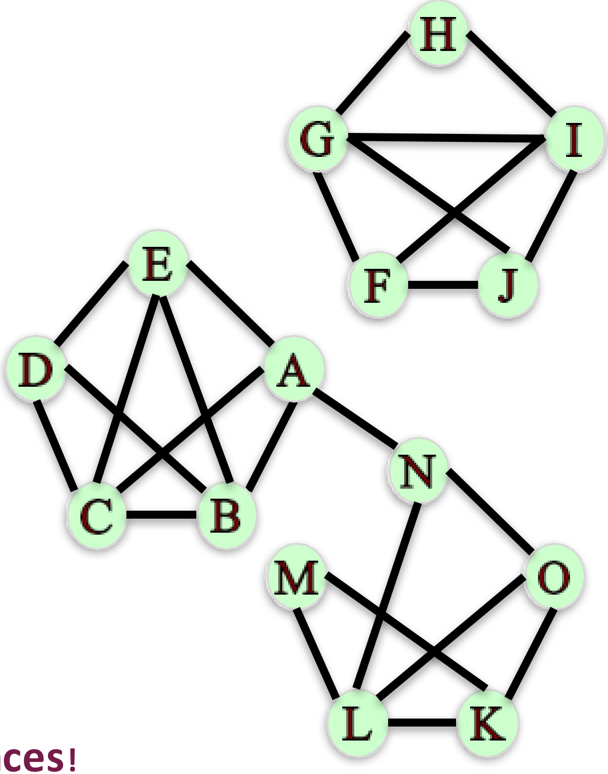
¿Cuál es el menor número de **enlaces** que de perderse hacen que el grafo quede desconectado?

In: nx.edge_connectivity(G_un)
Out: 2

¿Qué enlaces serían?

In: nx.minimum_edge_cut(G_un)
Out: {('A', 'G'), ('O', 'J')}

¡Las redes robustas tienen cortes mínimos de nodos y enlaces!

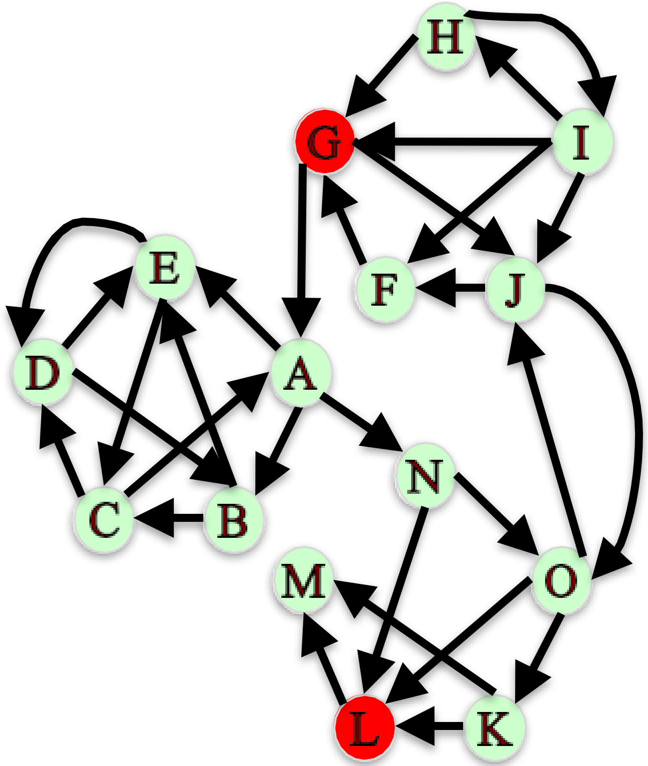


4. Consistencia

Imaginemos que el nodo **G** quiere enviar un mensaje al nodo **L** en esta red …

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

In: sorted(nx.all_simple_paths(G, 'G', 'L'))
Out:
[['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]



4. Consistencia

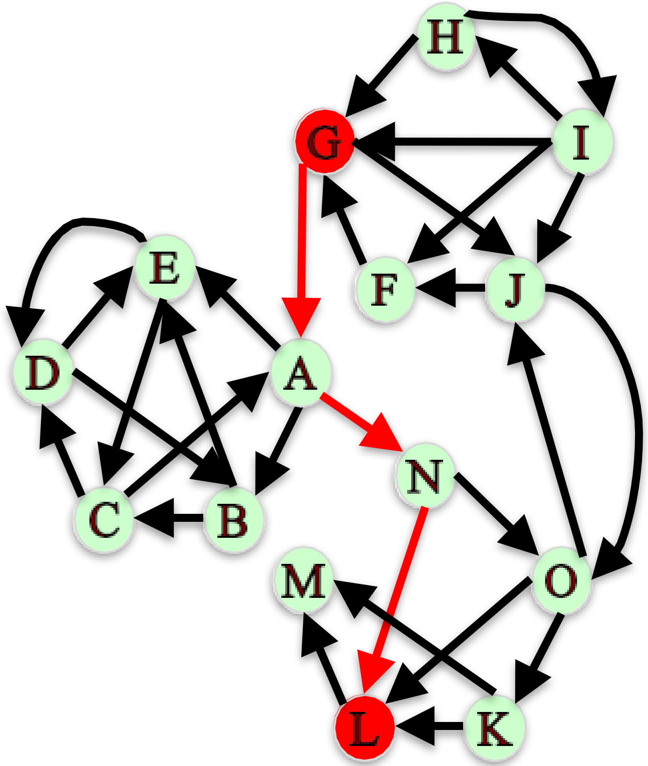
Imaginemos que el nodo **G** quiere enviar un mensaje al nodo **L** en esta red …

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

In: sorted(nx.all_simple_paths(G, 'G', 'L'))

Out:

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

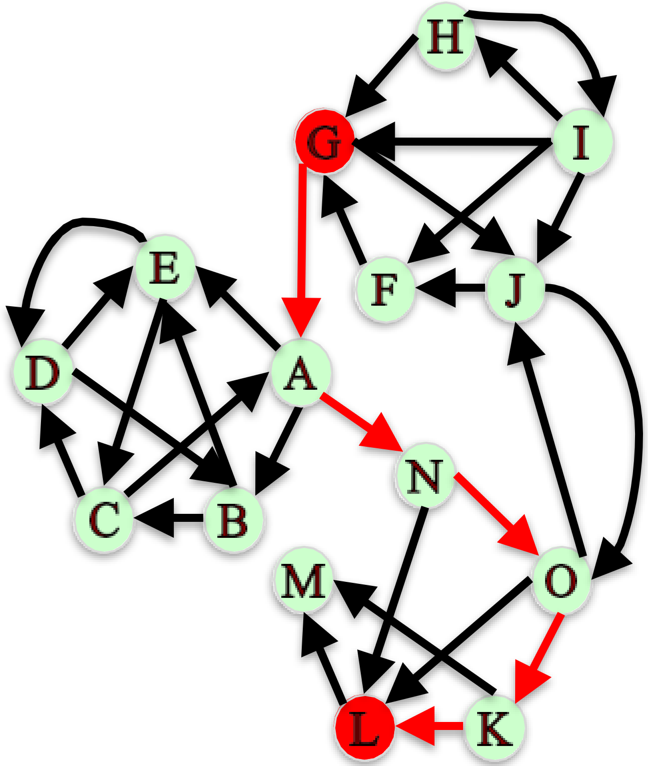


4. Consistencia

Imaginemos que el nodo **G** quiere enviar un mensaje al nodo **L** en esta red …

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

In: sorted(nx.all_simple_paths(G, 'G', 'L'))
Out:
[['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]

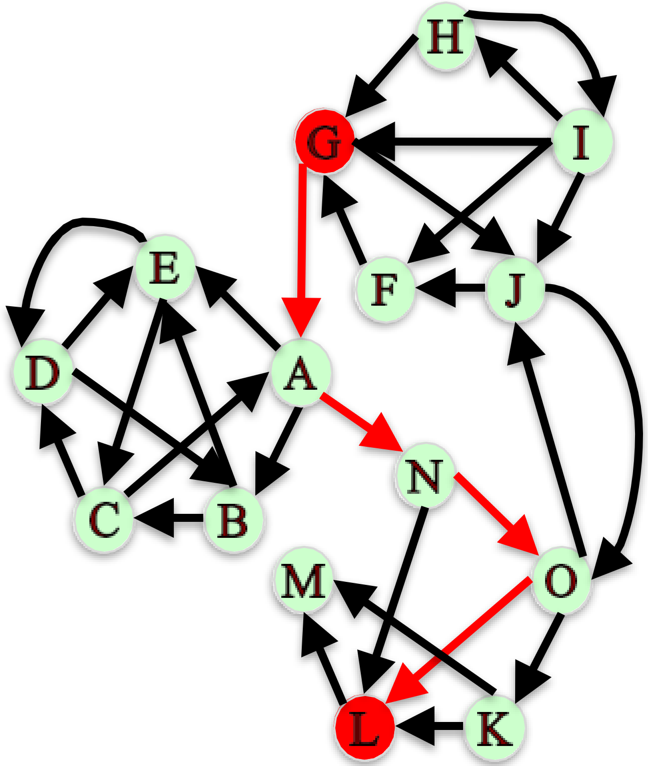


4. Consistencia

Imaginemos que el nodo **G** quiere enviar un mensaje al nodo **L** en esta red …

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

In: sorted(nx.all_simple_paths(G, 'G', 'L'))
Out:
[['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]

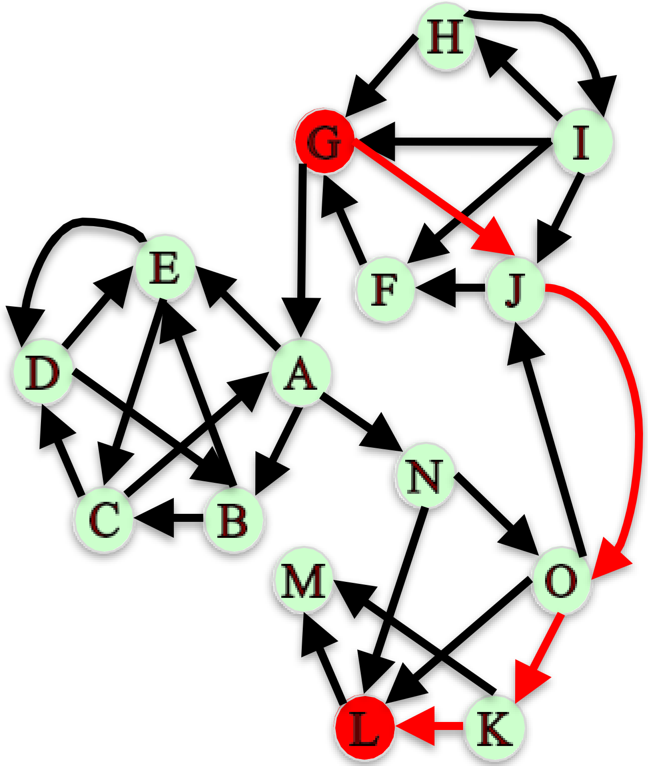


4. Consistencia

Imaginemos que el nodo **G** quiere enviar un mensaje al nodo **L** en esta red …

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

In: sorted(nx.all_simple_paths(G, 'G', 'L'))
Out:
[['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]

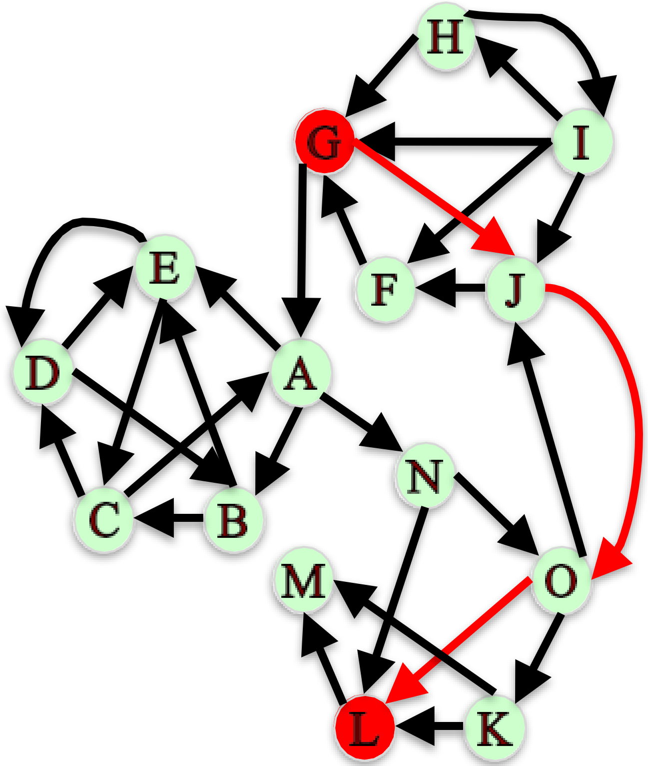


4. Consistencia

Imaginemos que el nodo **G** quiere enviar un mensaje al nodo **L** en esta red …

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

In: sorted(nx.all_simple_paths(G, 'G', 'L'))
Out:
[['G', 'A', 'N', 'L'],
['G', 'A', 'N', 'O', 'K', 'L'],
['G', 'A', 'N', 'O', 'L'],
['G', 'J', 'O', 'K', 'L'],
['G', 'J', 'O', 'L']]



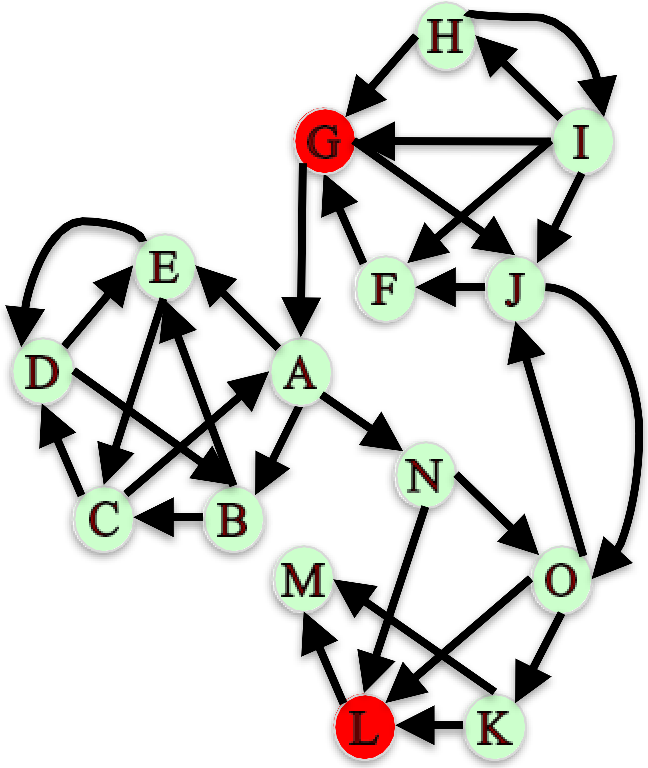
4. Consistencia

Si quisiéramos bloquear el mensaje de **G** a **L** eliminando **nodos** de la red …

¿Cuántos nodos tendríamos que eliminar?

In: nx.node_connectivity(G, 'G', 'L')

Out: 2



4. Consistencia

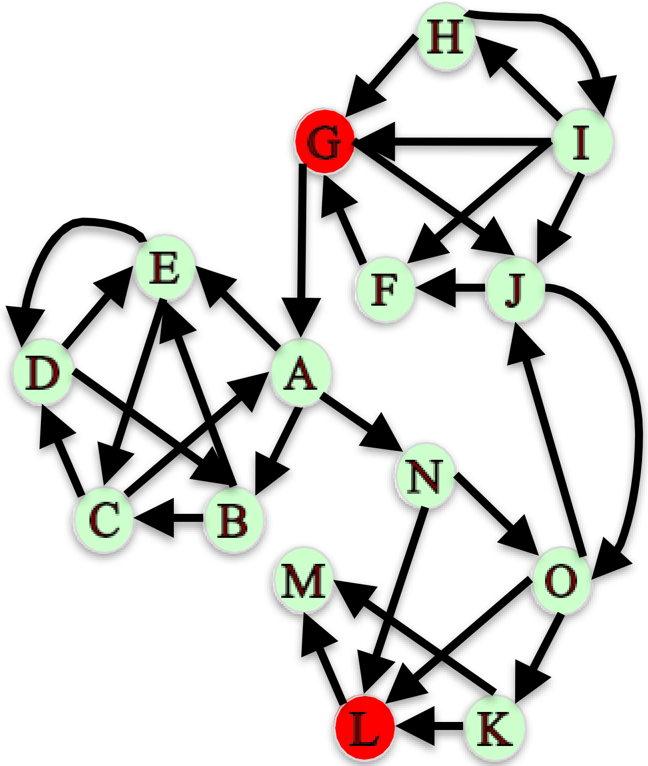
Si quisiéramos bloquear el mensaje de **G** a **L** eliminando **nodos** de la red ...

¿Cuántos nodos tendríamos que eliminar?

In: nx.node_connectivity(G, 'G', 'L')
Out: 2

¿Qué **nodos** serían?

In: nx.minimum_node_cut(G, 'G', 'L')
Out: {'N', 'O'}



4. Consistencia

Si quisiéramos bloquear el mensaje de **G** a **L** eliminando **nodos** de la red …

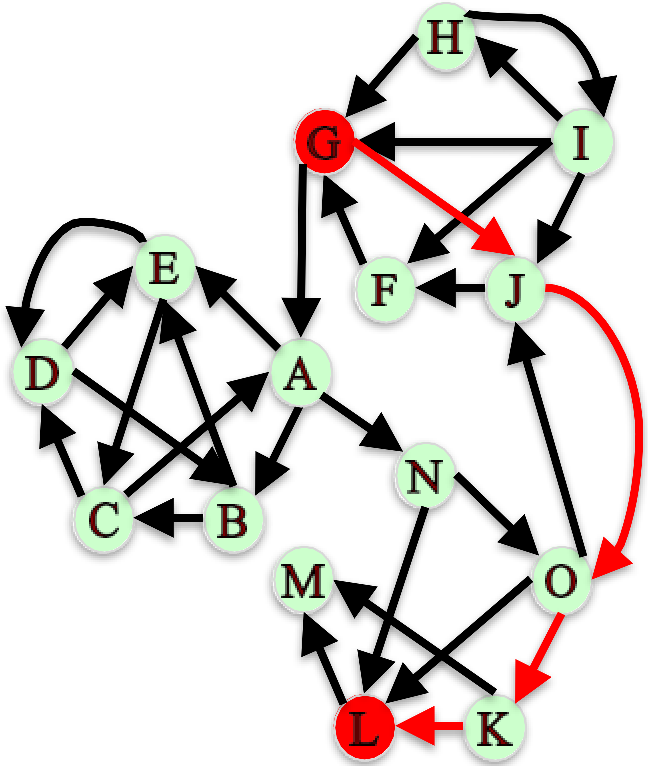
¿Cuántos nodos tendríamos que eliminar?

In: nx.node_connectivity(G, 'G', 'L')
Out: 2

¿Qué **nodos** serían?

In: nx.minimum_node_cut(G, 'G', 'L')
Out: {'N', 'O'}

Si sólo eliminamos el nodo **N**, el mensaje podría seguir el camino: **G → J → O → K → L**



4. Consistencia

Si quisiéramos bloquear el mensaje de **G** a **L** eliminando **nodos** de la red …

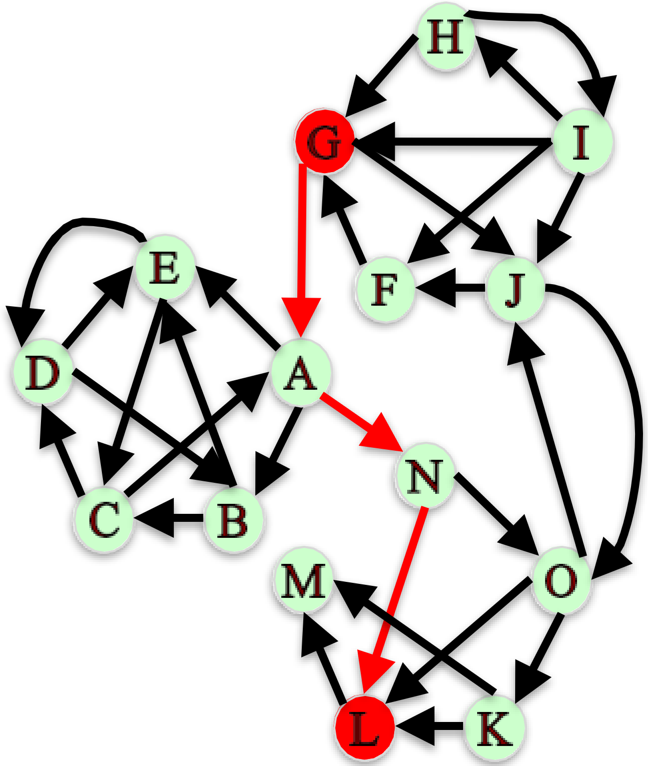
¿Cuántos nodos tendríamos que eliminar?

In: nx.node_connectivity(G, 'G', 'L')
Out: 2

¿Qué **nodos** serían?

In: nx.minimum_node_cut(G, 'G', 'L')
Out: {'N', 'O'}

Si sólo eliminamos el nodo **O**, el mensaje podría seguir el camino: **G → A → N → L**



4. Consistencia

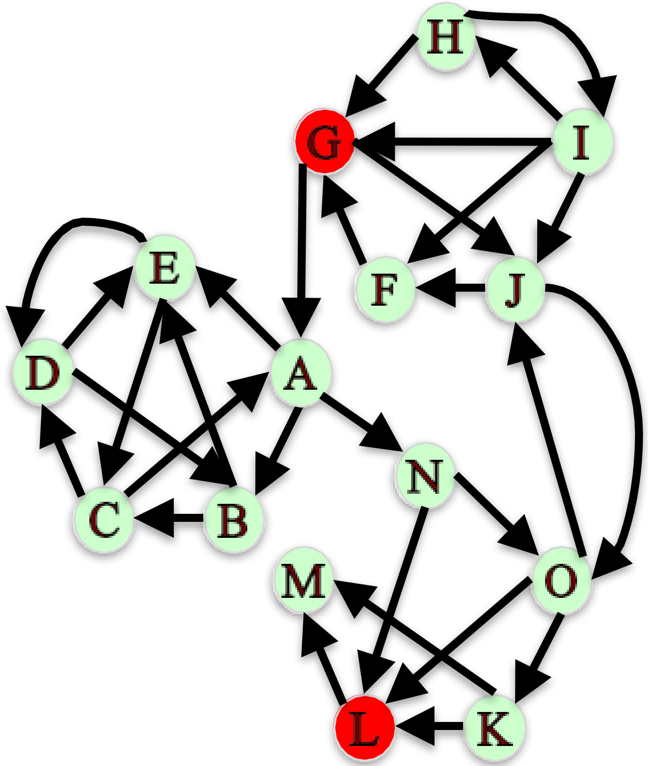
Y si quisiéramos bloquear el mensaje de **G** a **L** eliminando **enlaces** de la red …

¿Cuántos enlaces tendríamos que eliminar?

In: nx.node_connectivity(G, 'G', 'L')
Out: 2

¿Qué **enlaces** serían?

In: nx.minimum_edge_cut(G, 'G', 'L')
Out: {('A', 'N'), ('J', 'O')}



4. Consistencia

Y si quisiéramos bloquear el mensaje de **G** a **L** eliminando **enlaces** de la red …

¿Cuántos enlaces tendríamos que eliminar?

In: nx.node_connectivity(G, 'G', 'L')
Out: 2

¿Qué **enlaces** serían?

In: nx.minimum_edge_cut(G, 'G', 'L')
Out: {('A', 'N'), ('J', 'O')}

Necesitamos eliminar **A -> N** y **J -> O** para bloquear mensajes que vayan de **G** a **L**.