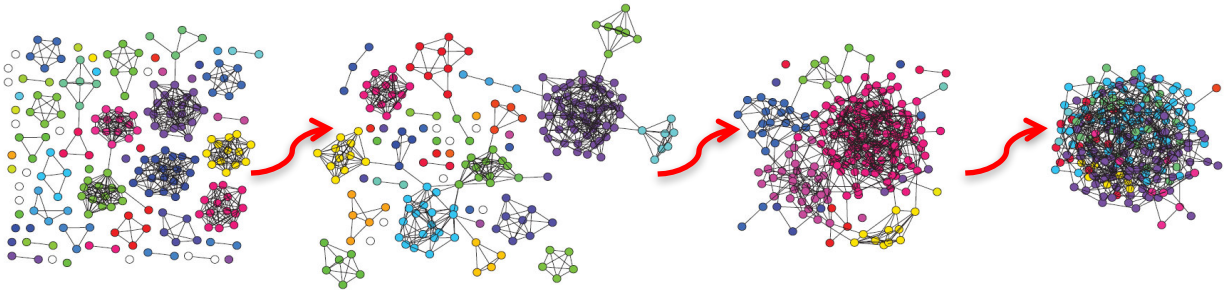
Link Prediction Problem



Modeling network evolution:

- Preferential attachment model
- Small world model

Link prediction: Given a network, can we predict which edges will be formed in the future?

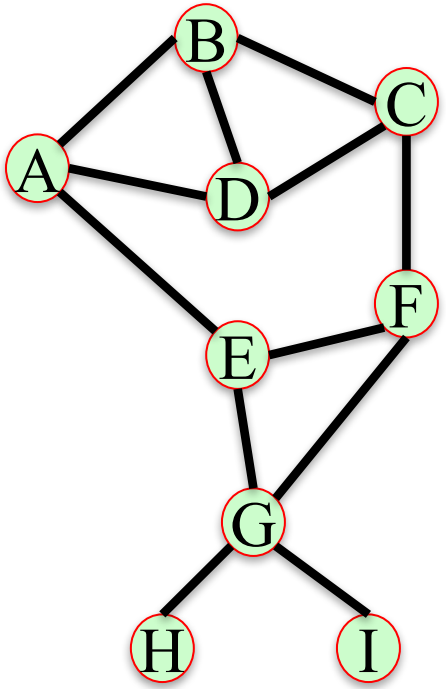
Link Prediction

What new edges are likely to form in this network?

Given a pair of nodes, how to assess whether they are likely to connect?

Triadic closure: the tendency for people who share connections in a social network to become connected.

Measure 1: number of common neighbors.



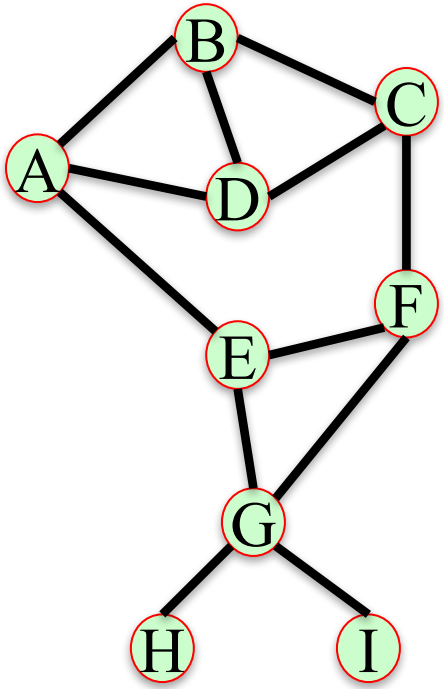
Measure 1: Common Neighbors

The number of common neighbors of nodes X and Y is

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

where $N(X)$ is the set of neighbors of node X

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

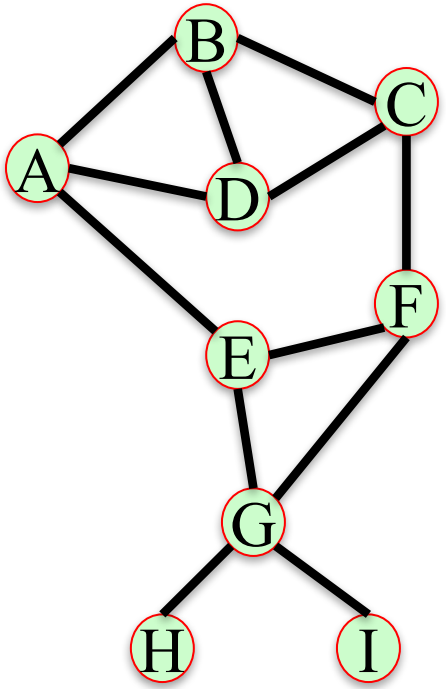


Measure 1: Common Neighbors

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

In: sorted(common_neigh,key=operator.itemgetter(2), reverse = True
In: print (common_neigh)

Out: [('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), ('F', 'H', 1), ('I', 'H', 1), ('A', 'I', 0), ('A', 'H', 0), ('C', 'I', 0), ('C', 'H', 0), ('B', 'I', 0), ('B', 'H', 0), ('B', 'G', 0), ('D', 'I', 0), ('D', 'H', 0), ('D', 'G', 0)]

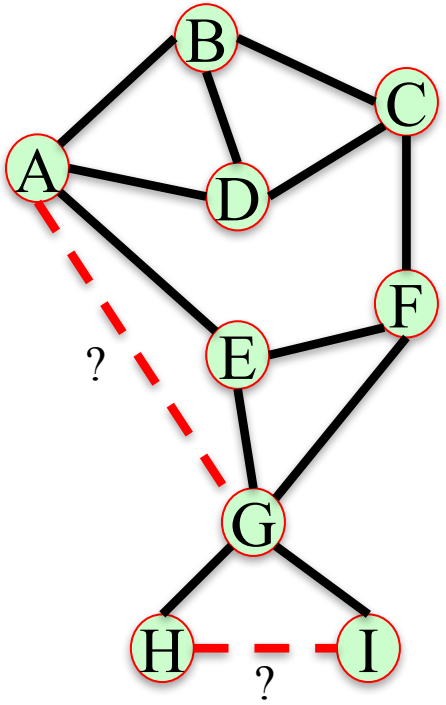


Measure 1: Common Neighbors

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

In: sorted(common_neigh,key=operator.itemgetter(2), reverse = True
In: print (common_neigh)

Out: [('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), ('F', 'H', 1), ('I', 'H', 1), ('A', 'I', 0), ('A', 'H', 0), ('C', 'I', 0), ('C', 'H', 0), ('B', 'I', 0), ('B', 'H', 0), ('B', 'G', 0), ('D', 'I', 0), ('D', 'H', 0), ('D', 'G', 0)]

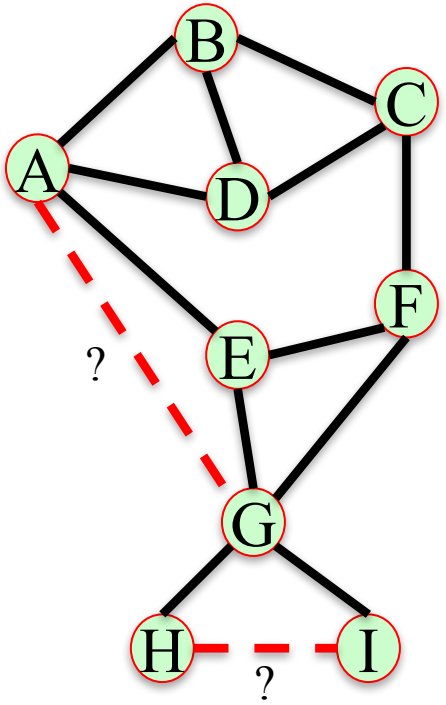


Measure 1: Common Neighbors

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

In: sorted(common_neigh,key=operator.itemgetter(2), reverse = True
In: print (common_neigh)

Out: [('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), ('F', 'H', 1), **('I', 'H', 1)**, ('A', 'I', 0), ('A', 'H', 0), ('C', 'I', 0), ('C', 'H', 0), ('B', 'I', 0), ('B', 'H', 0), ('B', 'G', 0), ('D', 'I', 0), ('D', 'H', 0), ('D', 'G', 0)]



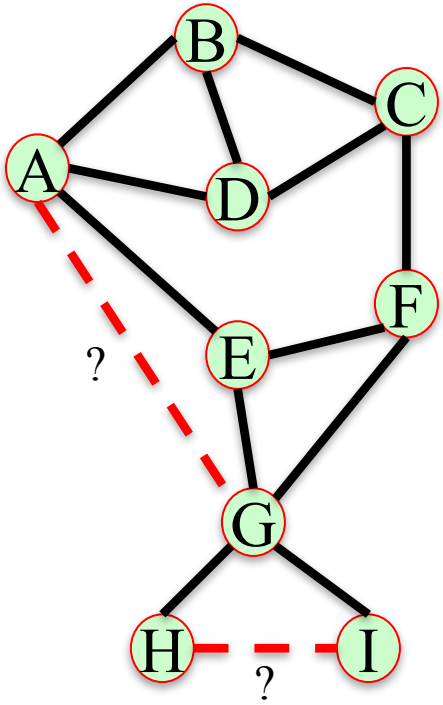
Measure 2: Jaccard Coefficient

Number of common neighbors normalized by the total number of neighbors.

The Jaccard coefficient of nodes X and Y is

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

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



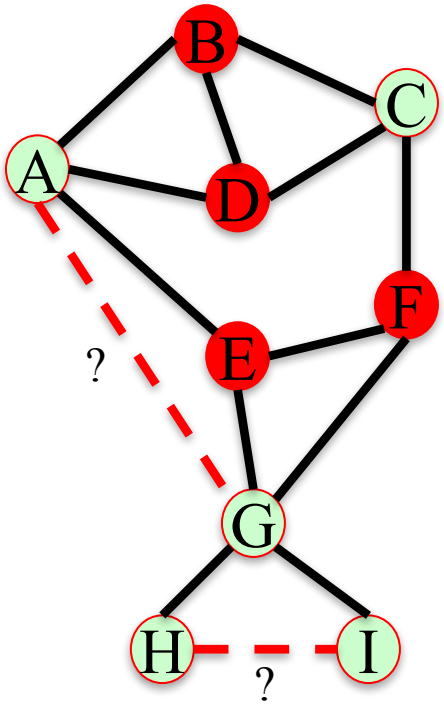
Measure 2: Jaccard Coefficient

Number of common neighbors normalized by the total number of neighbors.

The Jaccard coefficient of nodes X and Y is

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

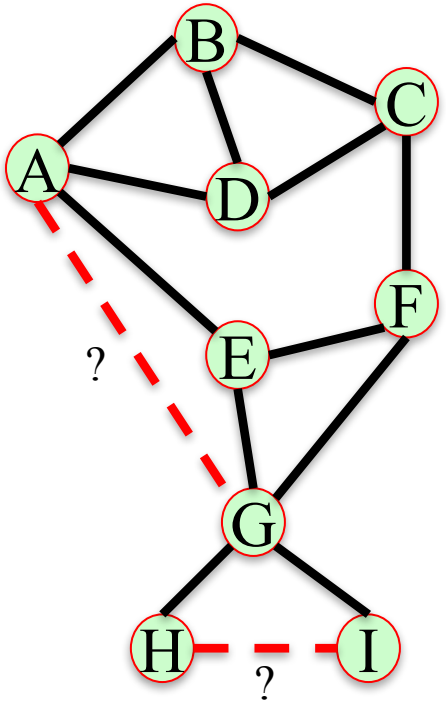
$$\text{jacc\_coeff}(A, C) = \frac{|\{B, D\}|}{|\{B, D, E, F\}|} = \frac{2}{4} = \frac{1}{2}$$



Measure 2: Jaccard Coefficient

Number of common neighbors normalized by the total number of neighbors.

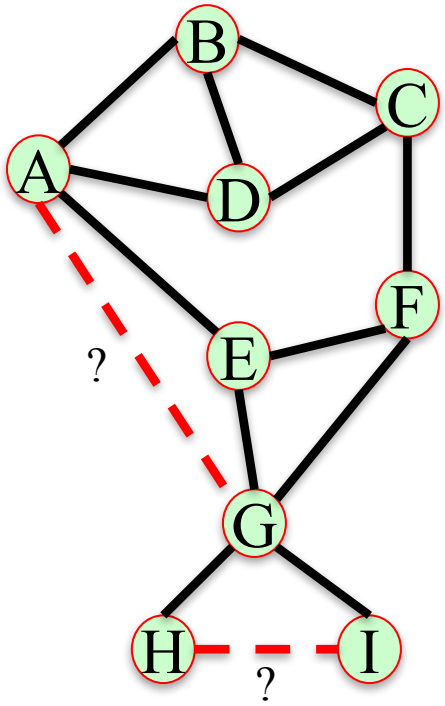
```
In: L = list(nx.jaccard_coefficient(G))
In: L.sort(key=operator.itemgetter(2), reverse = True)
In: print(L)
Out: [('I', 'H', 1.0), ('A', 'C', 0.5), ('E', 'I', 0.3333333333333333), ('E', 'H',
0.3333333333333333), ('F', 'I', 0.3333333333333333), ('F', 'H',
0.3333333333333333), ('A', 'F', 0.2), ('C', 'E', 0.2), ('B', 'E', 0.2), ('B', 'F',
0.2), ('E', 'D', 0.2), ('D', 'F', 0.2), ('A', 'G', 0.16666666666666666), ('C', 'G',
0.16666666666666666), ('A', 'I', 0.0), ('A', 'H', 0.0), ('C', 'I', 0.0), ('C', 'H',
0.0), ('B', 'I', 0.0), ('B', 'H', 0.0), ('B', 'G', 0.0), ('D', 'I', 0.0), ('D', 'H', 0.0),
('D', 'G', 0.0)]
```



Measure 2: Jaccard Coefficient

Number of common neighbors normalized by the total number of neighbors.

```
In: L = list(nx.jaccard_coefficient(G))
In: L.sort(key=operator.itemgetter(2), reverse = True)
In: print(L)
Out: [('I', 'H', 1.0), ('A', 'C', 0.5), ('E', 'I', 0.3333333333333333), ('E', 'H',
0.3333333333333333), ('F', 'I', 0.3333333333333333), ('F', 'H',
0.3333333333333333), ('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', 0.16666666666666666), ('C', 'G',
0.16666666666666666), ('A', 'I', 0.0), ('A', 'H', 0.0), ('C', 'I', 0.0), ('C', 'H',
0.0), ('B', 'I', 0.0), ('B', 'H', 0.0), ('B', 'G', 0.0), ('D', 'I', 0.0), ('D', 'H', 0.0),
('D', 'G', 0.0)]
```



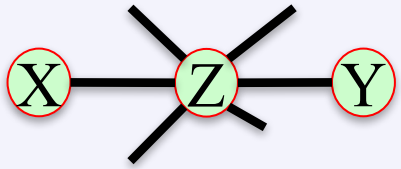
Measure 3: Resource Allocation

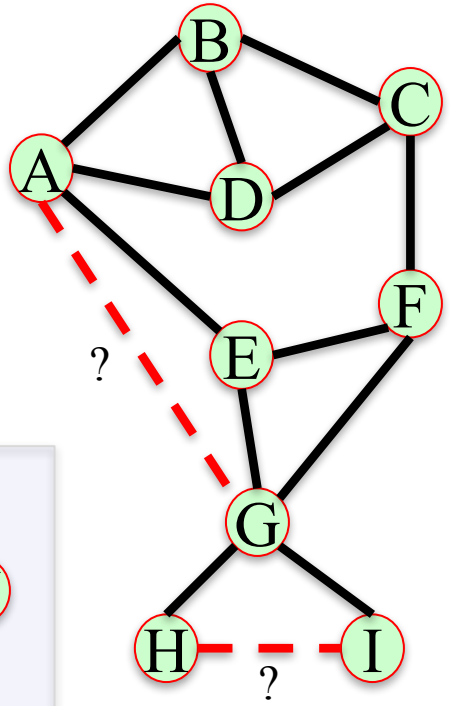
Fraction of a "resource" that a node can send to another through their common neighbors.

The Resource Allocation index of nodes X and Y is

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

Z has n neighbors
X sends 1 unit to Z, Z distributes the unit evenly among all neighbors
→ Y receives $1/n$ of the unit.





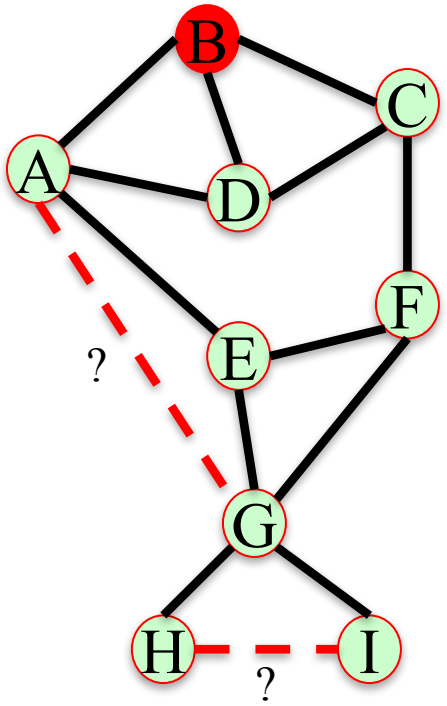
Measure 3: Resource Allocation

Fraction of a "resource" that a node can send to another through their common neighbors.

The Resource Allocation index of nodes X and Y is

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

$$\text{res\_alloc}(A, C) = \frac{1}{3} +$$



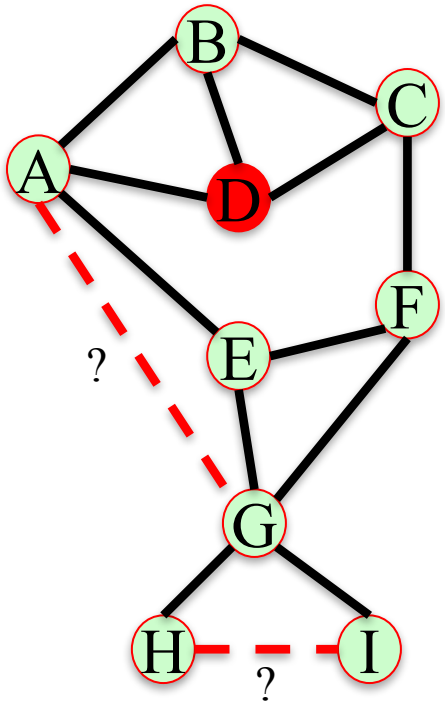
Measure 3: Resource Allocation

Fraction of a "resource" that a node can send to another through their common neighbors.

The Resource Allocation index of nodes X and Y is

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

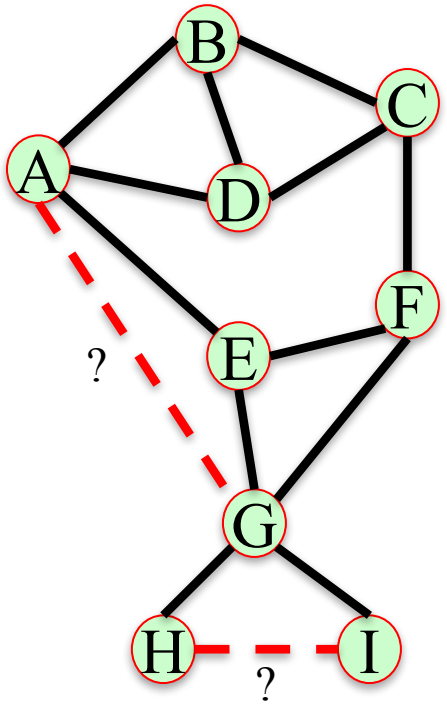
$$\text{res\_alloc}(A, C) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$$



Measure 3: Resource Allocation

Fraction of a "resource" that a node can send to another through their common neighbors.

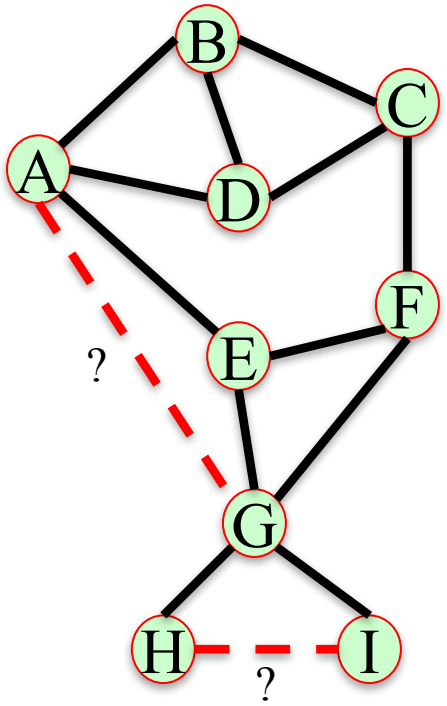
```
In: L = list(nx.resource_allocation_index(G))
In: L.sort(key=operator.itemgetter(2), reverse = True)
In: print(L)
Out: [('A', 'C', 0.6666666666666666), ('A', 'G',
0.3333333333333333), ('A', 'F', 0.3333333333333333), ('C', 'E',
0.3333333333333333), ('C', 'G', 0.3333333333333333), ('B', 'E',
0.3333333333333333), ('B', 'F', 0.3333333333333333), ('E', 'D',
0.3333333333333333), ('D', 'F', 0.3333333333333333), ('E', 'I', 0.25),
('E', 'H', 0.25), ('F', 'I', 0.25), ('F', 'H', 0.25), ('I', 'H', 0.25), ('A', 'I', 0),
('A', 'H', 0), ('C', 'I', 0), ('C', 'H', 0), ('B', 'I', 0), ('B', 'H', 0), ('B', 'G', 0),
('D', 'I', 0), ('D', 'H', 0), ('D', 'G', 0)]
```



Measure 3: Resource Allocation

Fraction of a "resource" that a node can send to another through their common neighbors.

```
In: L = list(nx.resource_allocation_index(G))
In: L.sort(key=operator.itemgetter(2), reverse = True)
In: print(L)
Out: [('A', 'C', 0.6666666666666666), ('A', 'G',
0.3333333333333333), ('A', 'F', 0.3333333333333333), ('C', 'E',
0.3333333333333333), ('C', 'G', 0.3333333333333333), ('B', 'E',
0.3333333333333333), ('B', 'F', 0.3333333333333333), ('E', 'D',
0.3333333333333333), ('D', 'F', 0.3333333333333333), ('E', 'I', 0.25),
('E', 'H', 0.25), ('F', 'I', 0.25), ('F', 'H', 0.25), ('I', 'H', 0.25), ('A', 'I', 0),
('A', 'H', 0), ('C', 'I', 0), ('C', 'H', 0), ('B', 'I', 0), ('B', 'H', 0), ('B', 'G', 0),
('D', 'I', 0), ('D', 'H', 0), ('D', 'G', 0)]
```



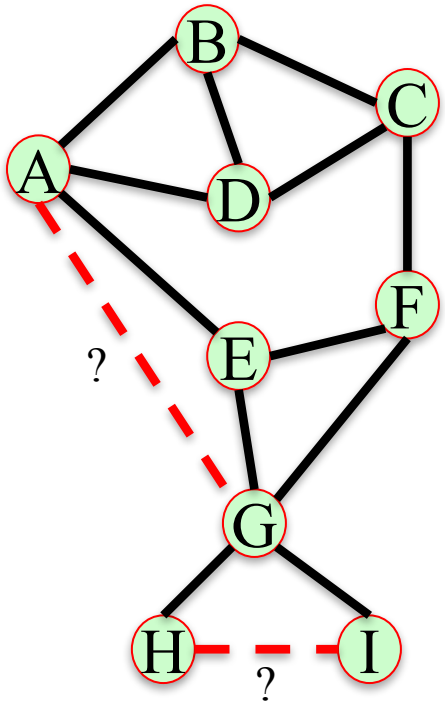
Measure 4: Adamic-Adar Index

Similar to resource allocation index, but with log in the denominator.

The Adamic-Adar index of nodes X and Y is

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

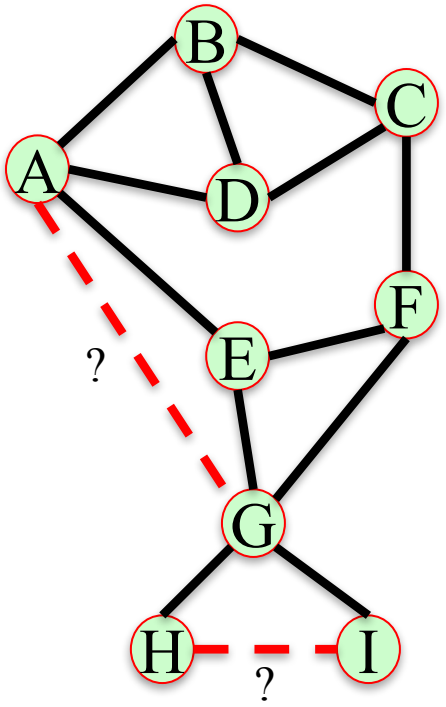
$$\text{adamic\_adar}(A, C) = \frac{1}{\log(3)} + \frac{1}{\log(3)} = 1.82$$



Measure 4: Adamic-Adar Index

Similar to resource allocation index, but with log in the denominator.

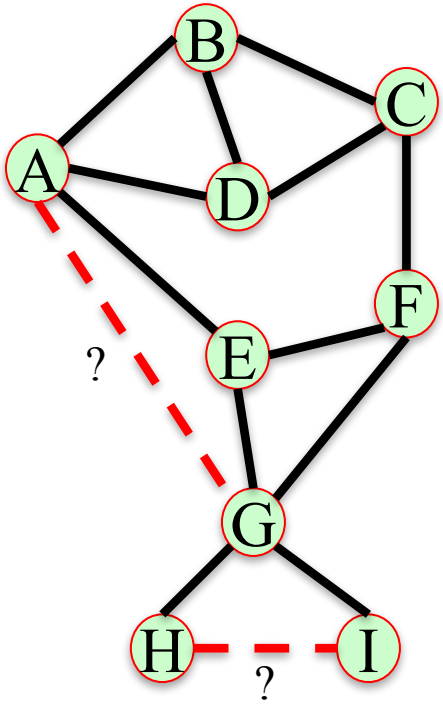
In: L = list(nx.**adamic_adar_index**(G))
In: L.sort(key=operator.itemgetter(2), reverse = True)
In: print(L)
Out: [('A', 'C', 1.8204784532536746), ('A', 'G', 0.9102392266268373), ('A', 'F', 0.9102392266268373), ('C', 'E', 0.9102392266268373), ('C', 'G', 0.9102392266268373), ('B', 'E', 0.9102392266268373), ('B', 'F', 0.9102392266268373), ('E', 'D', 0.9102392266268373), ('D', 'F', 0.9102392266268373), ('E', 'I', 0.7213475204444817), ('E', 'H', 0.7213475204444817), ('F', 'I', 0.7213475204444817), ('F', 'H', 0.7213475204444817), ('I', 'H', 0.7213475204444817), ('A', 'I', 0), ('A', 'H', 0), ('C', 'I', 0), ('C', 'H', 0), ('B', 'I', 0), ('B', 'H', 0), ('B', 'G', 0), ('D', 'I', 0), ('D', 'H', 0), ('D', 'G', 0)]



Measure 4: Adamic-Adar Index

Similar to resource allocation index, but with log in the denominator.

In: L = list(nx.adamic_adar_index(G))
In: L.sort(key=operator.itemgetter(2), reverse = True)
In: print(L)
Out: [('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), ('F', 'H', 0.7213475204444817), **('I', 'H', 0.7213475204444817)**, ('A', 'I', 0), ('A', 'H', 0), ('C', 'I', 0), ('C', 'H', 0), ('B', 'I', 0), ('B', 'H', 0), ('B', 'G', 0), ('D', 'I', 0), ('D', 'H', 0), ('D', 'G', 0)]



Measure 5: Pref. Attachment

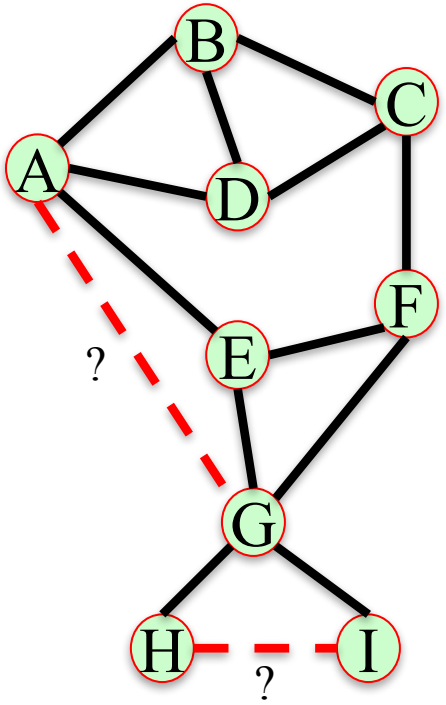
In the preferential attachment model, nodes with high degree get more neighbors.

Product of the nodes' degree.

The preferential attachment score of nodes X and Y is

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

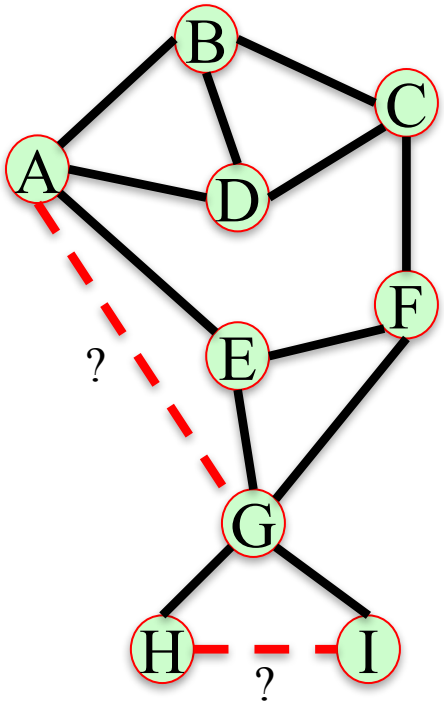
$\text{pref\_attach}(A, C) = 3 * 3 = 9$



Measure 5: Pref. Attachment

Product of the nodes' degree.

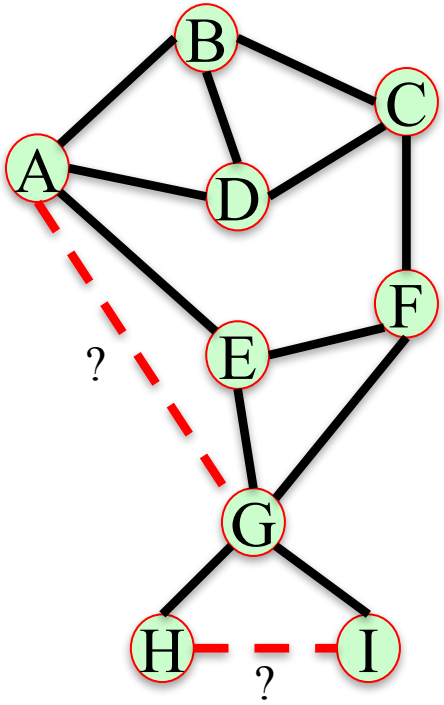
```
In: L = list(nx.preferential_attachment(G))
In: L.sort(key=operator.itemgetter(2), reverse = True)
In: print(L)
Out: [('A', 'G', 12), ('C', 'G', 12), ('B', 'G', 12), ('D', 'G', 12), ('A', 'C',
9), ('A', 'F', 9), ('C', 'E', 9), ('B', 'E', 9), ('B', 'F', 9), ('E', 'D', 9), ('D', 'F',
9), ('A', 'I', 3), ('A', 'H', 3), ('C', 'I', 3), ('C', 'H', 3), ('B', 'I', 3), ('B', 'H',
3), ('E', 'I', 3), ('E', 'H', 3), ('D', 'I', 3), ('D', 'H', 3), ('F', 'I', 3), ('F', 'H',
3), ('I', 'H', 1)]
```



Measure 5: Pref. Attachment

Product of the nodes' degree.

In: L = list(nx.preferential_attachment(G))
In: L.sort(key=operator.itemgetter(2), reverse = True)
In: print(L)
Out: [**('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), ('A', 'H', 3), ('C', 'I', 3), ('C', 'H', 3), ('B', 'I', 3), ('B', 'H', 3), ('E', 'I', 3), ('E', 'H', 3), ('D', 'I', 3), ('D', 'H', 3), ('F', 'I', 3), ('F', 'H', 3), **('I', 'H', 1)**]

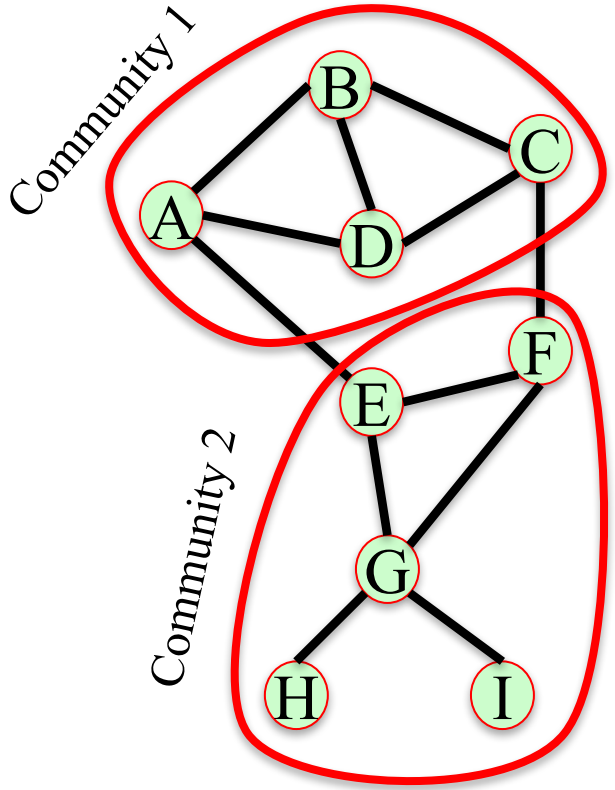


Community Structure

Some measures consider the community structure of the network for link prediction.

Assume the nodes in this network belong to different communities (sets of nodes).

Pairs of nodes who belong to the same community and have many common neighbors in their community are likely to form an edge.

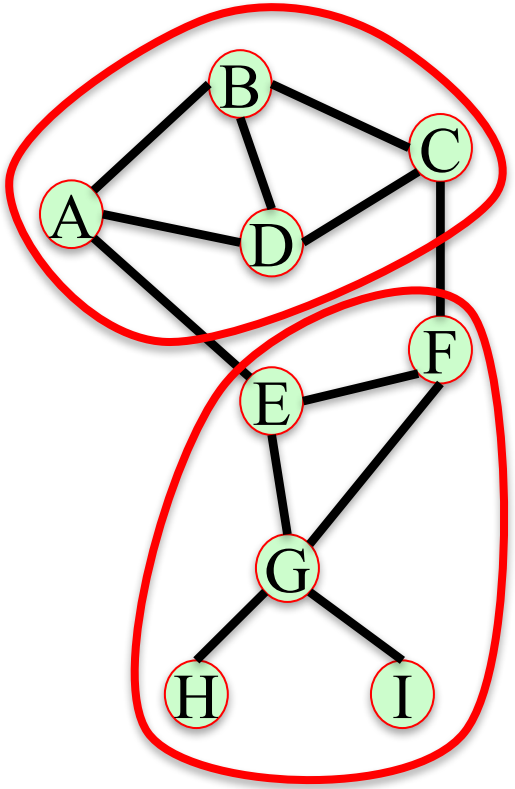


Measure 6: Community Common Neighbors

Number of common neighbors with bonus for neighbors in same community.

The Common Neighbor Soundarajan-Hopcroft score of nodes X and Y is:
$\text{cn\_soundarajan\_hopcroft}(X, Y) = |N(X) \cap N(Y)| + \sum_{u \in N(X) \cap N(Y)} f(u)$,

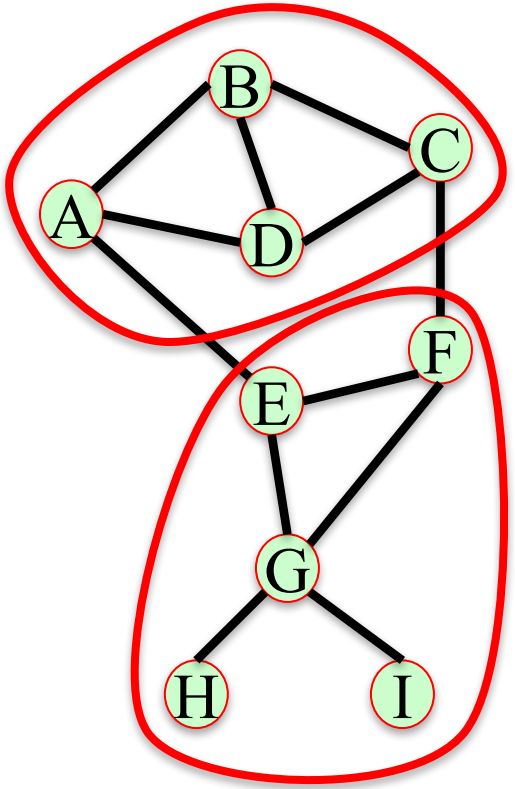
$$where\ f(u) = \begin{cases} 1, & u \text{ in same comm. as } X \text{ and } Y \\ 0, & \text{otherwise} \end{cases}$$



Measure 6: Community Common Neighbors

Number of common neighbors with bonus for neighbors in same community.

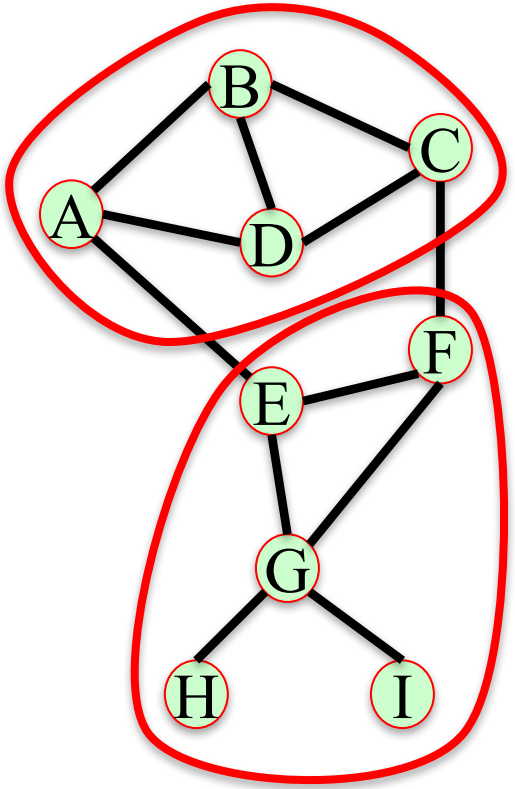
$$\text{cn\_soundarajan\_hopcroft}(A, C) = 2 + 2 = 4$$
$$\text{cn\_soundarajan\_hopcroft}(E, I) = 1 + 1 = 2$$
$$\text{cn\_soundarajan\_hopcroft}(A, G) = 1 + 0 = 1$$



Measure 6: Community Common Neighbors

Assign nodes to communities with attribute node "community"

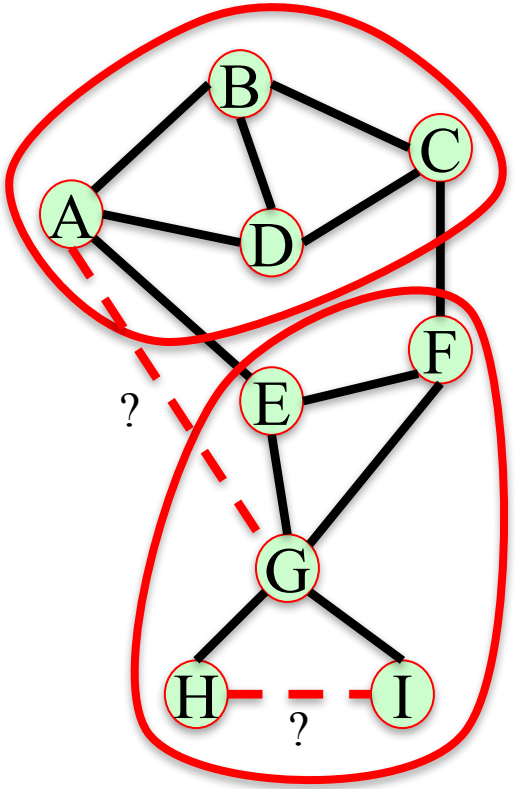
```
G.node['A']['community'] = 0
G.node['B']['community'] = 0
G.node['C']['community'] = 0
G.node['D']['community'] = 0
G.node['E']['community'] = 1
G.node['F']['community'] = 1
G.node['G']['community'] = 1
G.node['H']['community'] = 1
G.node['I']['community'] = 1
```



Measure 6: Community Common Neighbors

In: L = list(nx.**cn_soundarajan_hopcroft**(G))
In: L.sort(key=operator.itemgetter(2), reverse = True)
In: print(L)

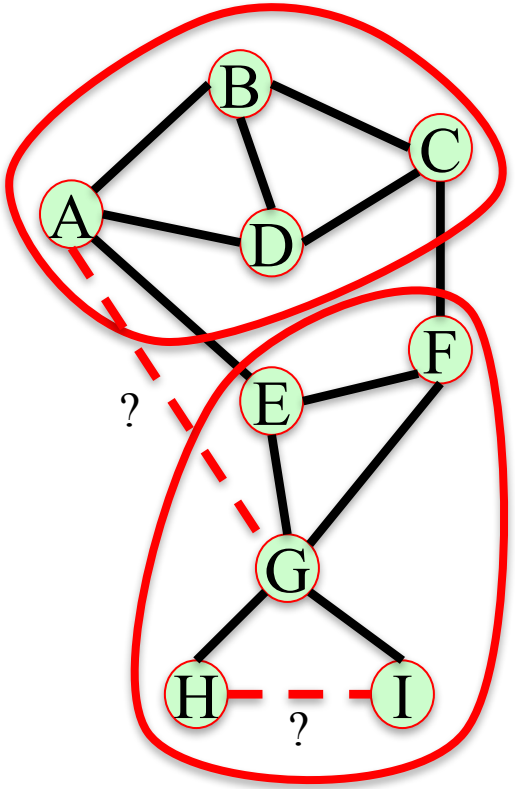
Out: [('A', 'C', 4), ('E', 'I', 2), ('E', 'H', 2), ('F', 'I', 2), ('F', 'H', 2), ('I', 'H', 2), ('A', 'G', 1), ('A', 'F', 1), ('C', 'E', 1), ('C', 'G', 1), ('B', 'E', 1), ('B', 'F', 1), ('E', 'D', 1), ('D', 'F', 1), ('A', 'I', 0), ('A', 'H', 0), ('C', 'I', 0), ('C', 'H', 0), ('B', 'I', 0), ('B', 'H', 0), ('B', 'G', 0), ('D', 'I', 0), ('D', 'H', 0), ('D', 'G', 0)]



Measure 6: Community Common Neighbors

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

Out: [('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), ('D', 'F', 1), ('A', 'I', 0), ('A', 'H', 0), ('C', 'I', 0), ('C', 'H', 0), ('B', 'I', 0), ('B', 'H', 0), ('B', 'G', 0), ('D', 'I', 0), ('D', 'H', 0), ('D', 'G', 0)]



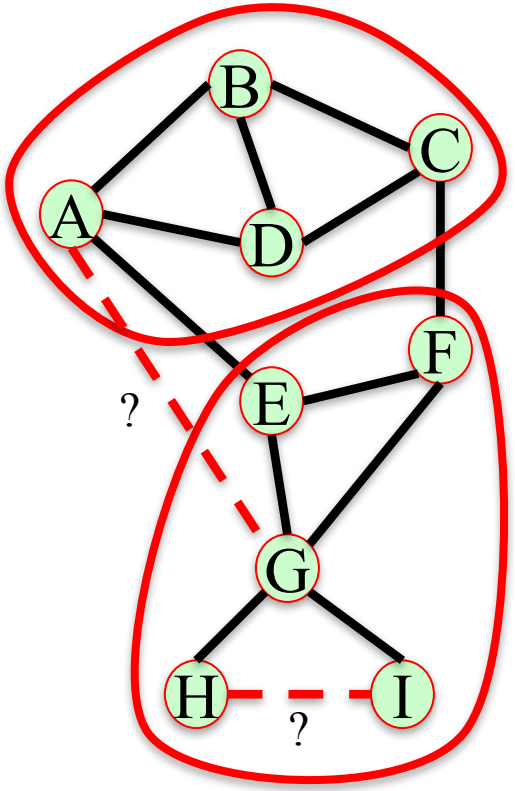
Measure 7: Community Resource Allocation

Similar to resource allocation index, but only considering nodes in the same community

The Resource Allocation Soundarajan-Hopcroft score of nodes X and Y is:

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

$$where\ f(u) = \begin{cases} 1, & u \text{ in same comm. as } X \text{ and } Y \\ 0, & \text{otherwise} \end{cases}$$



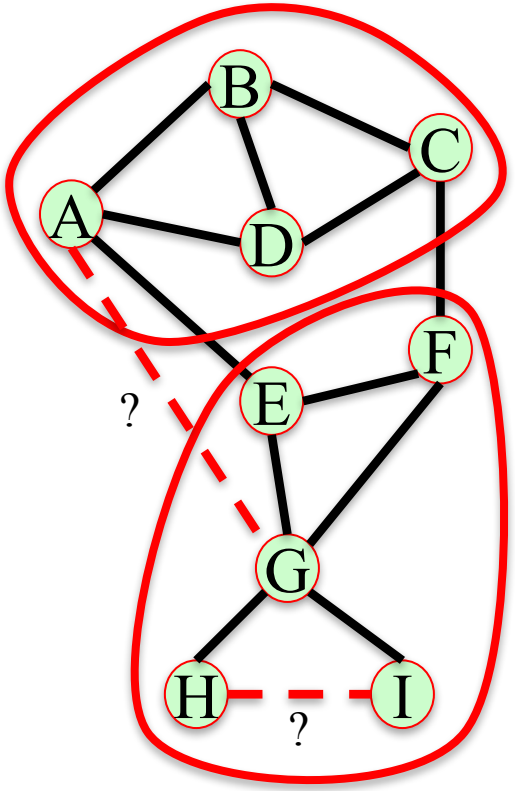
Measure 7: Community Resource Allocation

Similar to resource allocation index, but only considering nodes in the same community

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

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

$$\text{ra\_soundarajan\_hopcroft}(A, G) = 0$$

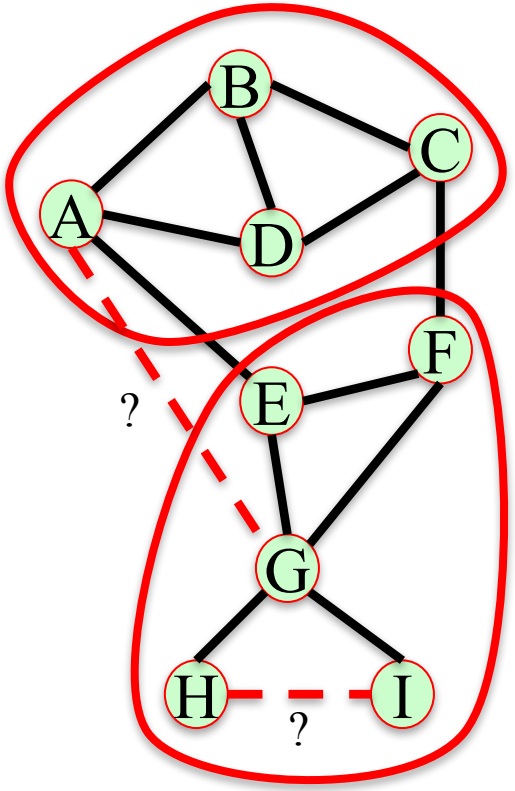


Measure 7: Community Resource Allocation

Similar to resource allocation index, but only considering nodes in the same community

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

```
Out: [('A', 'C', 0.6666666666666666), ('E', 'I', 0.25), ('E', 'H', 0.25), ('F', 'I',
0.25), ('F', 'H', 0.25), ('I', 'H', 0.25), ('A', 'I', 0), ('A', 'H', 0), ('A', 'G', 0),
('A', 'F', 0), ('C', 'I', 0), ('C', 'H', 0), ('C', 'E', 0), ('C', 'G', 0), ('B', 'I', 0), ('B',
'H', 0), ('B', 'E', 0), ('B', 'G', 0), ('B', 'F', 0), ('E', 'D', 0), ('D', 'I', 0), ('D', 'H',
0), ('D', 'G', 0), ('D', 'F', 0)]
```

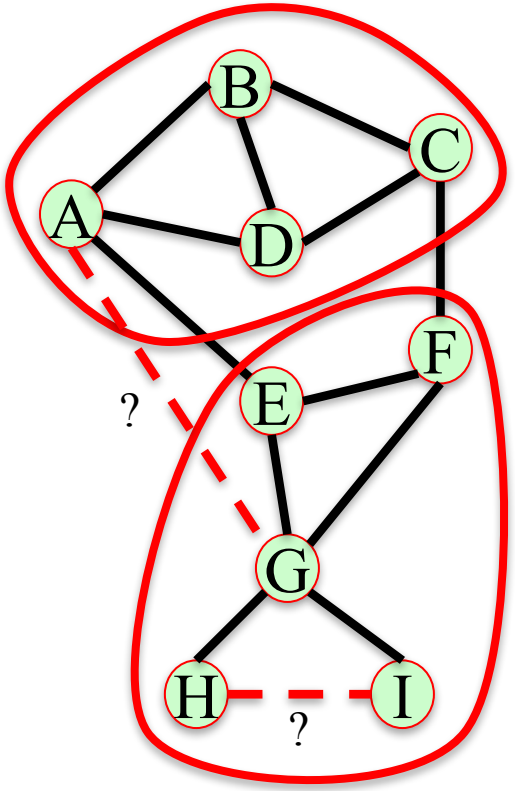


Measure 7: Community Resource Allocation

Similar to resource allocation index, but only considering nodes in the same community

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

Out: [('A', 'C', 0.6666666666666666), ('E', 'I', 0.25), ('E', 'H', 0.25), ('F', 'I', 0.25), ('F', 'H', 0.25), **('I', 'H', 0.25)**, ('A', 'I', 0), ('A', 'H', 0), **('A', 'G', 0)**, ('A', 'F', 0), ('C', 'I', 0), ('C', 'H', 0), ('C', 'E', 0), ('C', 'G', 0), ('B', 'I', 0), ('B', 'H', 0), ('B', 'E', 0), ('B', 'G', 0), ('B', 'F', 0), ('E', 'D', 0), ('D', 'I', 0), ('D', 'H', 0), ('D', 'G', 0), ('D', 'F', 0)]



Summary

- Link prediction problem: Given a network, predict which edges will be formed in the future.
- 5 basic measures:
  - Number of Common Neighbors
  - Jaccard Coefficient
  - Resource Allocation Index
  - Adamic-Adar Index
  - Preferential Attachment Score
- 2 measures that require community information:
  - Common Neighbor Soundarajan-Hopcroft Score
  - Resource Allocation Soundarajan-Hopcroft Score