Introduction

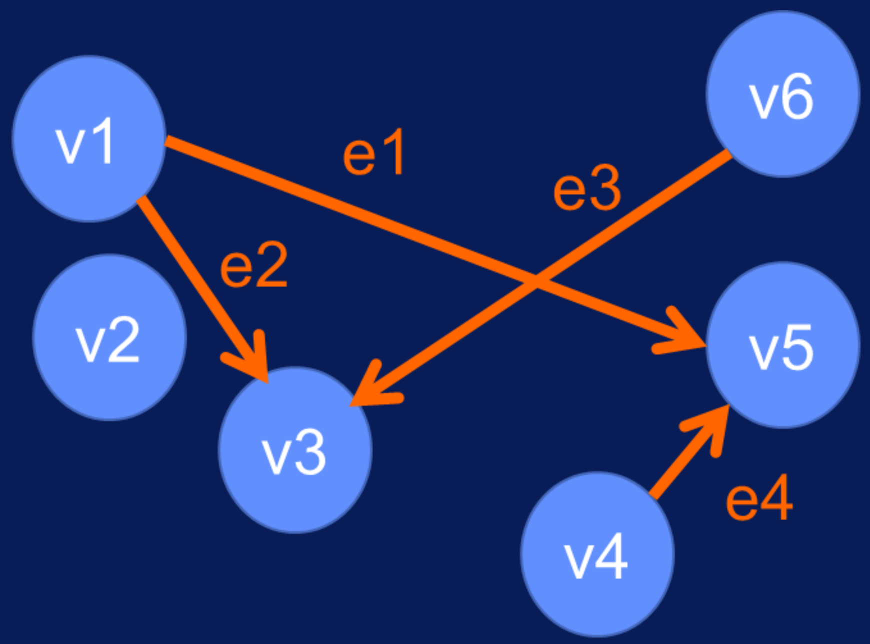
In this module you will learn a number of basic graph analytic techniques.

After this module you will be able to identify the right class of techniques to apply for a graph analytic problem

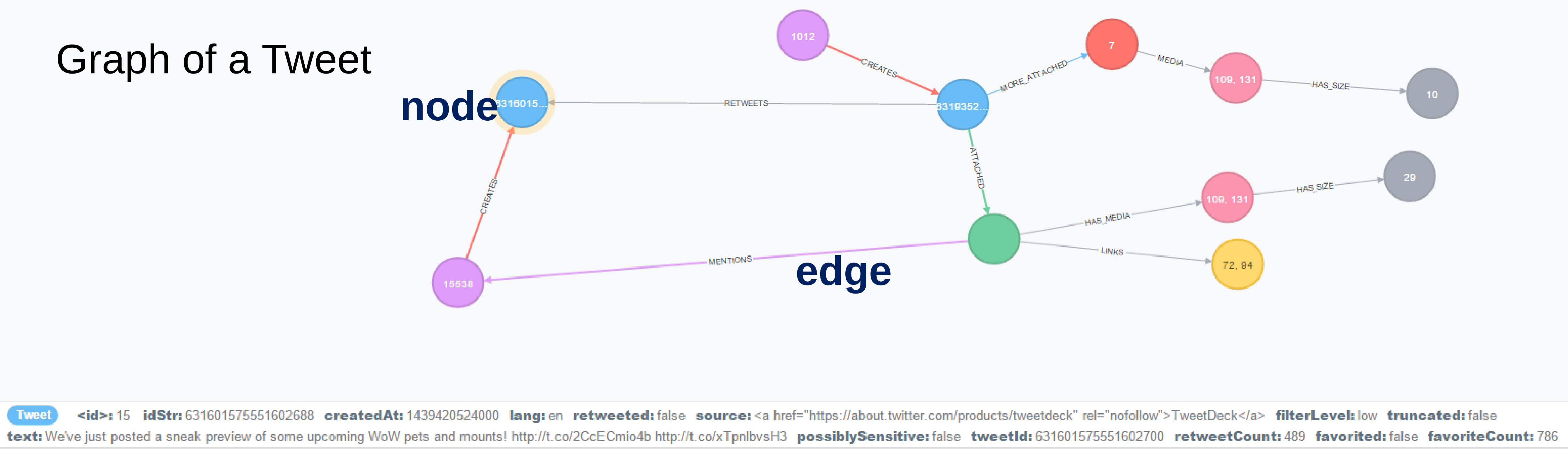
Module Plan

- **Focus on Analytic Techniques**
  - Big Data computing in Modules 3 and 4
- **Basic definitions**
- **Analytics**
  - Path Analytics
  - Connectivity Analytics
  - Community Analytics
  - Centrality Analytics

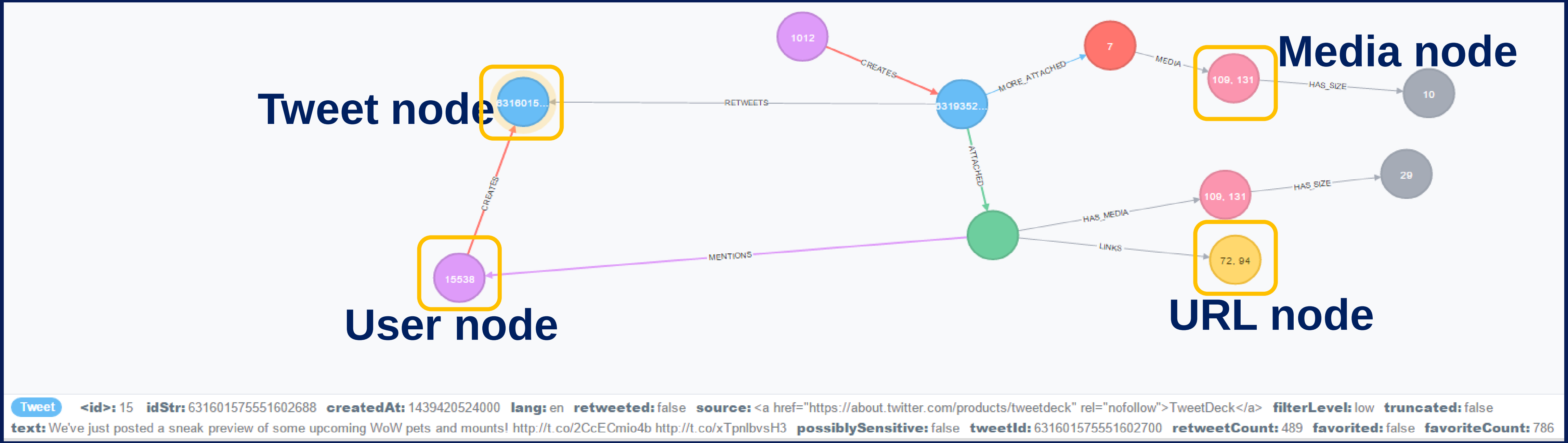
Our First Definition of Graphs



- V: a set of vertices
- E: a set of edges

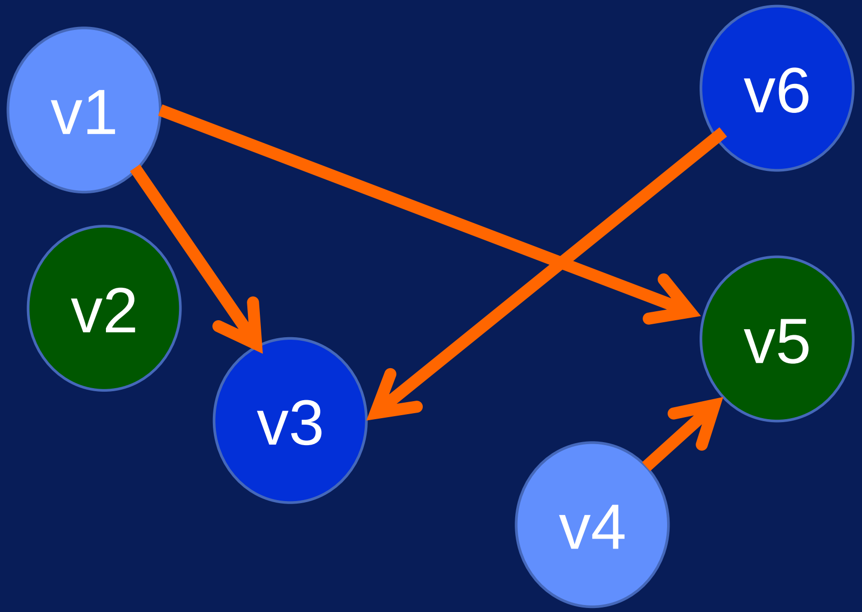


A Real Graph Has More Information Content

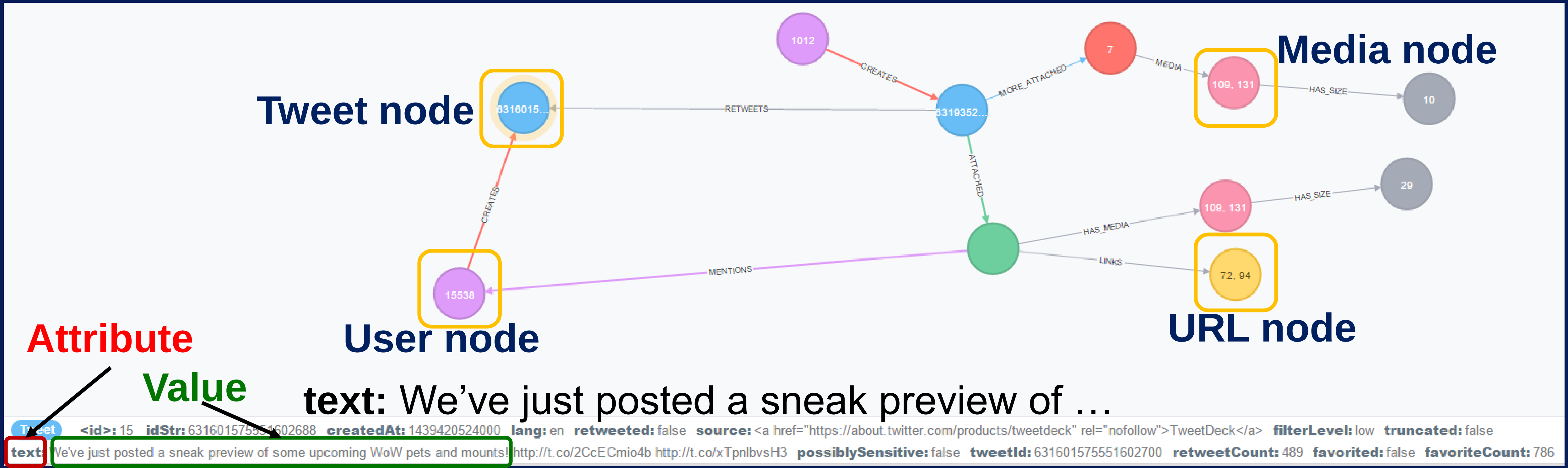


Node Types (aka Labels)

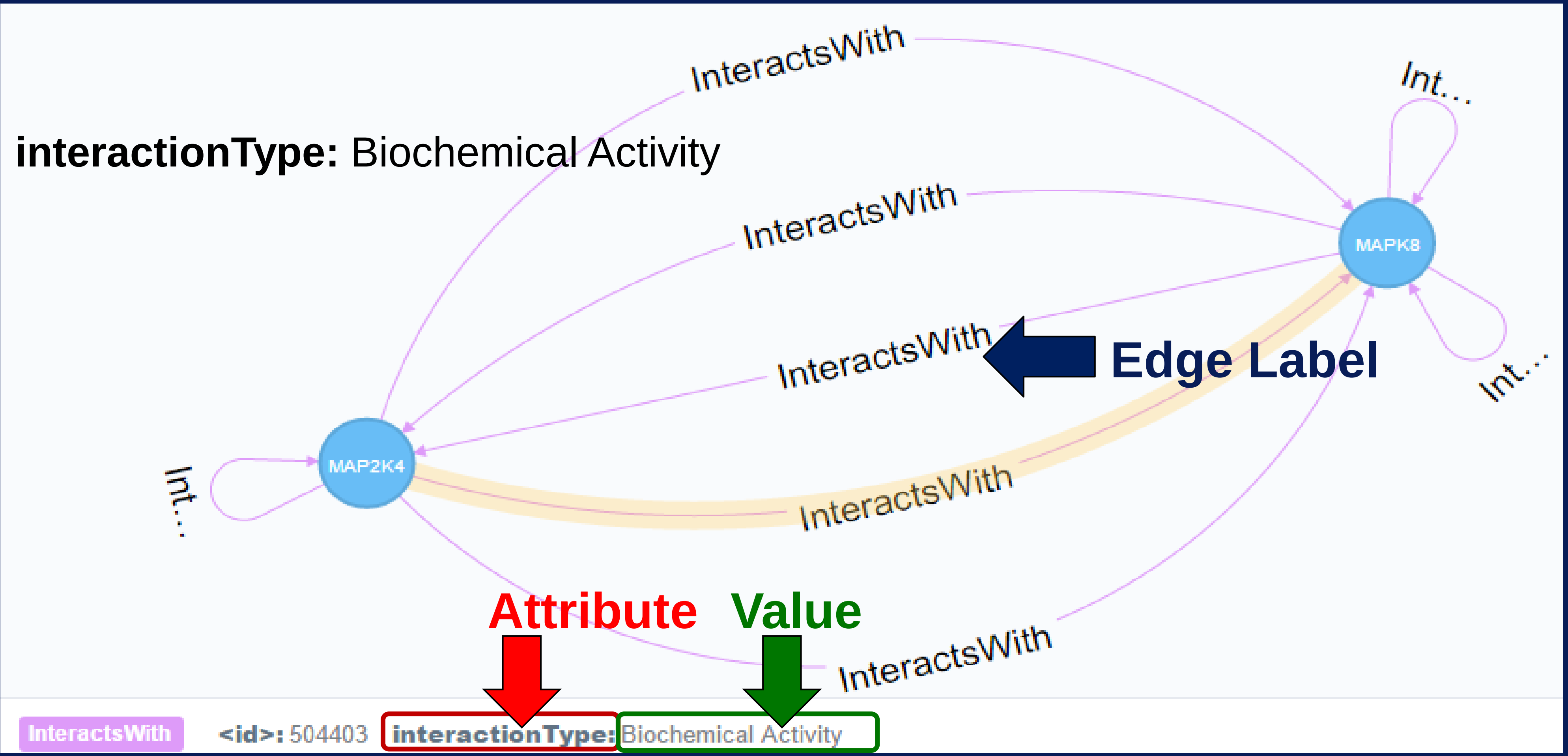
Graphs with Node Types



- V: a set of vertices
- E: a set of edges
- TN: a set of node types
- f (TN→V): type assignment to nodes



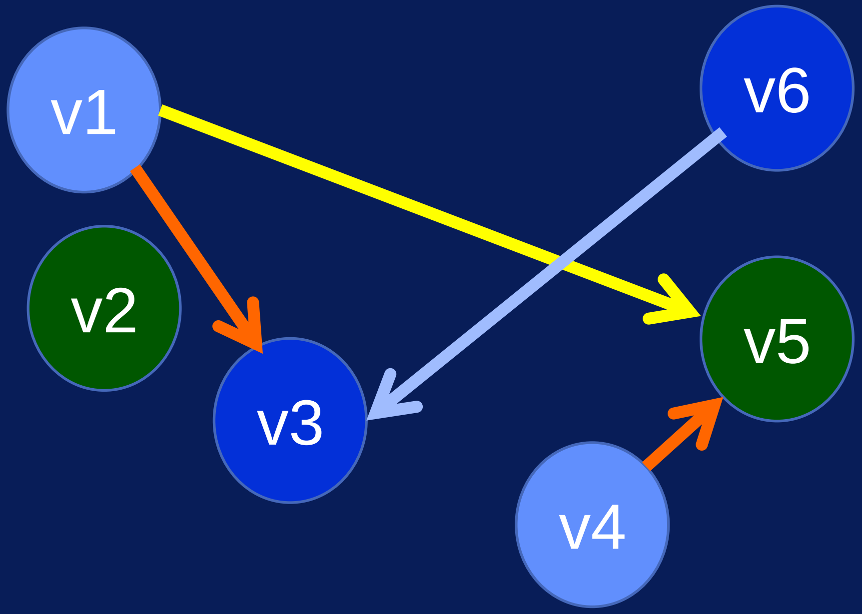
- **Node Schema = Properties (Attributes) with Values**



Edge Schema

interactionType: $\underbrace{\text{physical, genetic, biochemical, …}}_{\text{Domain of interactionType}}$

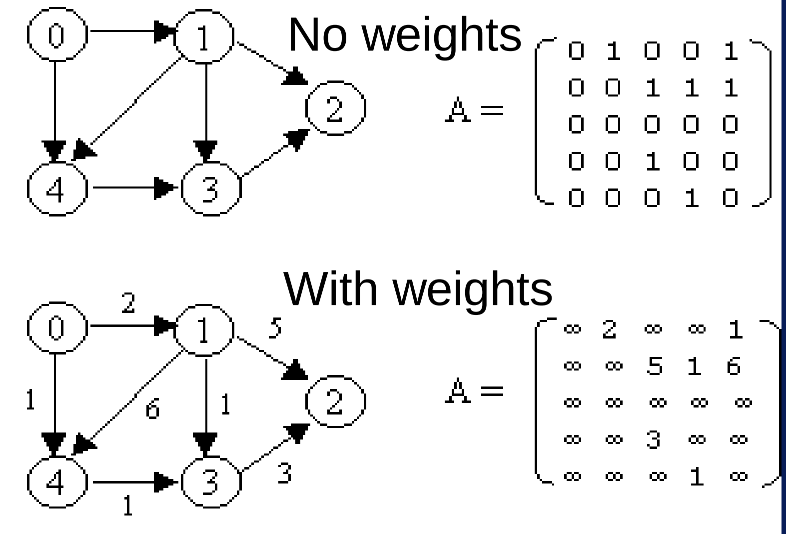
Extended Graph Model



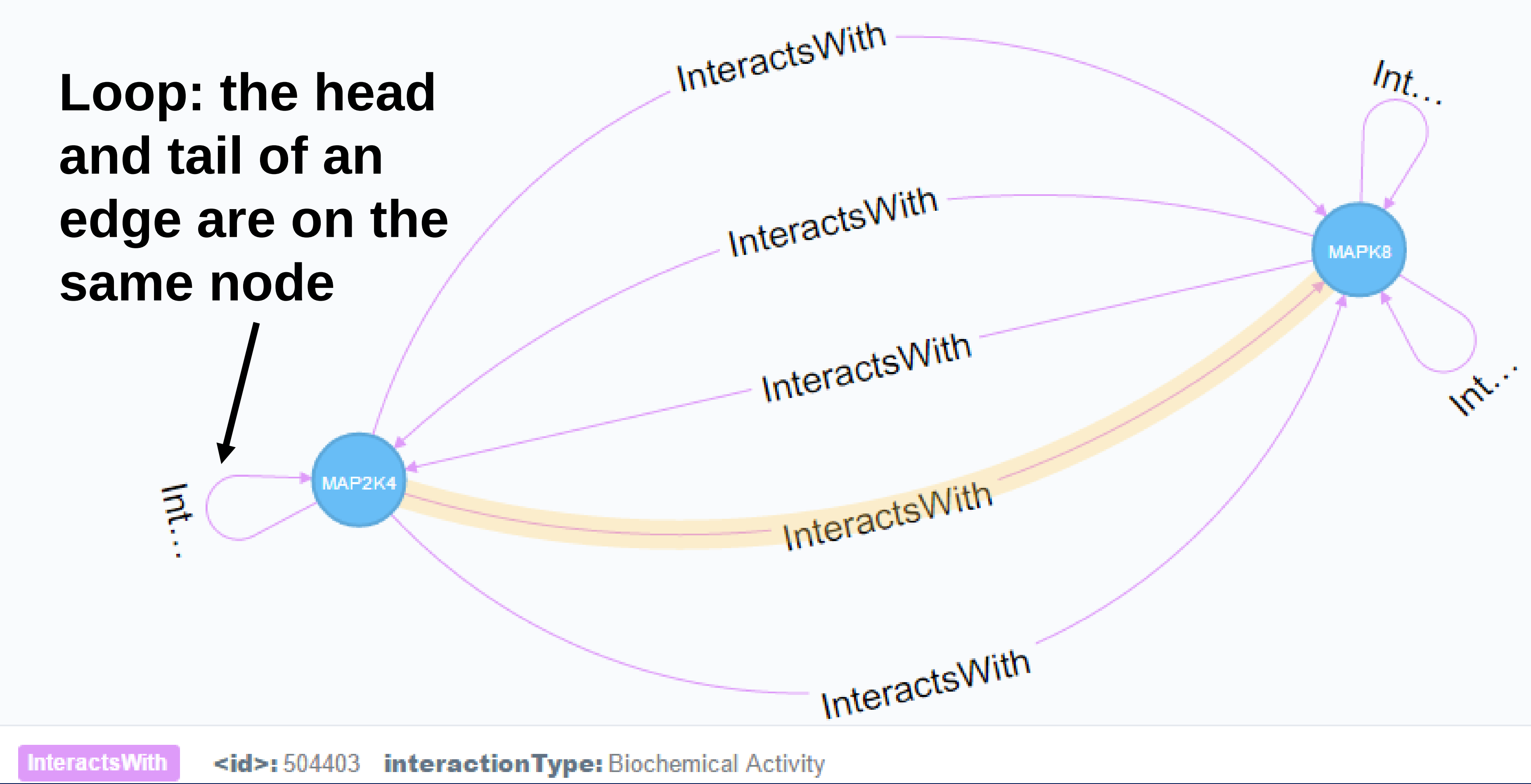
- V: set of vertices
- E: set of edges
- TN: set of node types
- f (TN→V): type assignment to nodes
- TE: a set of edge types
- g (TE→E): type assignment to edges
- AN: set of node attributes
- AE: set of edge attributes
- *dom*(AN[i]): domain of the i-th node attribute
- *dom*(AE[i]): domain of the i-th edge attribute

“Weight” – an Edge Property

- **Why weights?**
  - “Distance” in a road network
  - “Strength of Connection” in a personal network
  - “Likelihood of interaction” in a biological network
  - “Certainty of information” in a knowledge network



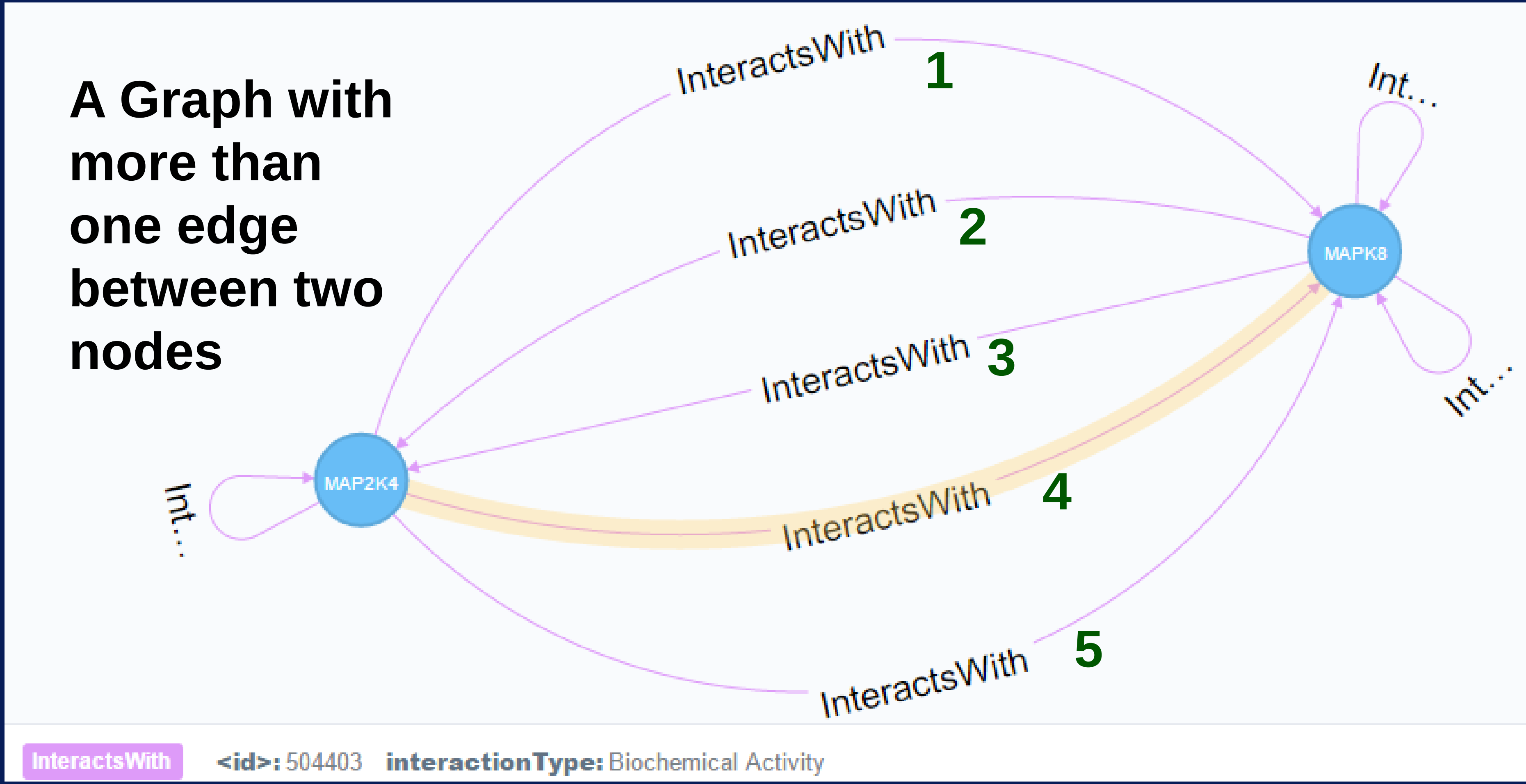
Loop: the head and tail of an edge are on the same node



Structural Property of a Graph

Multi-graph

A Graph with more than one edge between two nodes

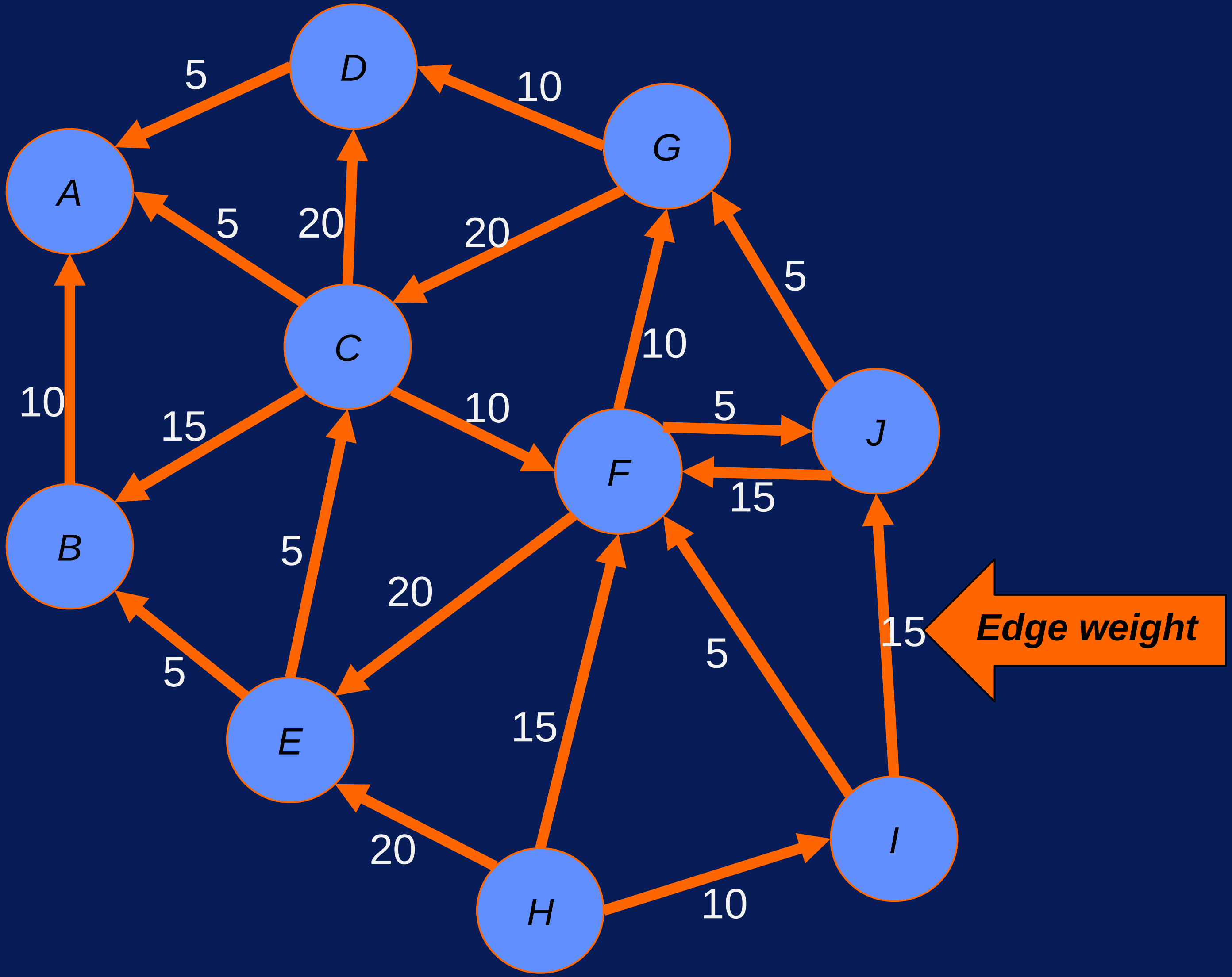


Why multiple edges?

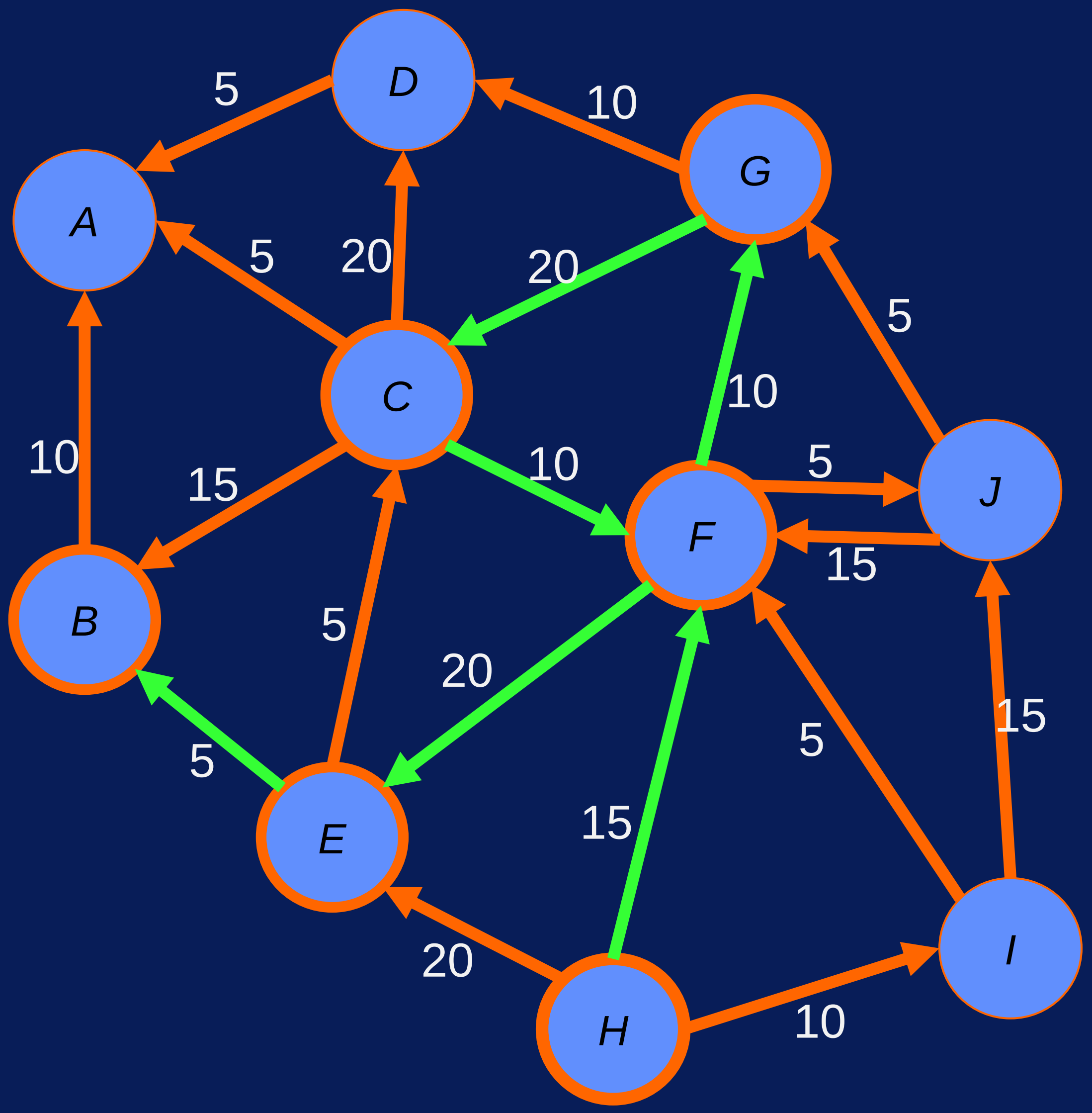
Each edge has a different information content

Path Analytics

A Walk in the Graph

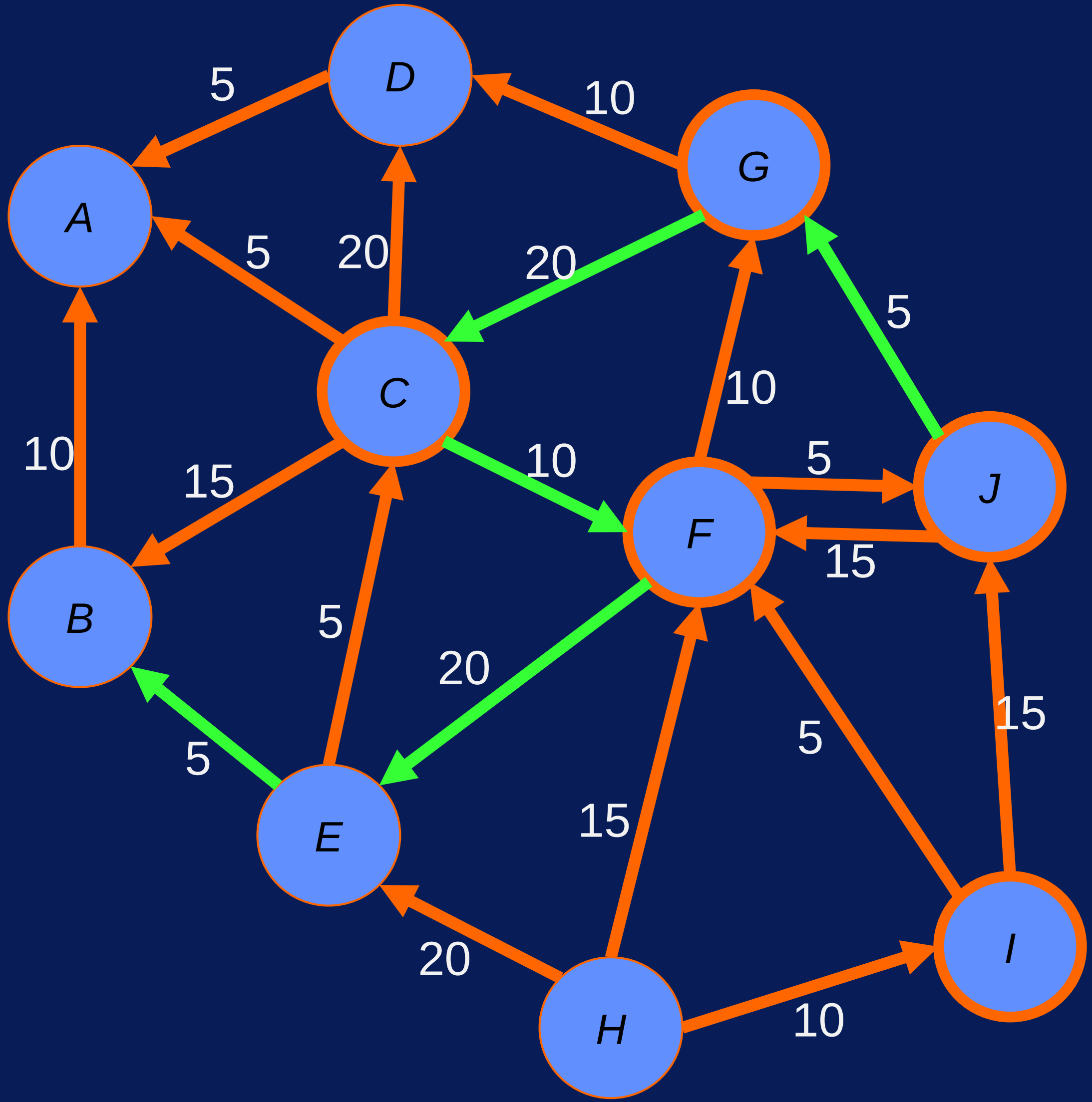


A Walk in the Graph



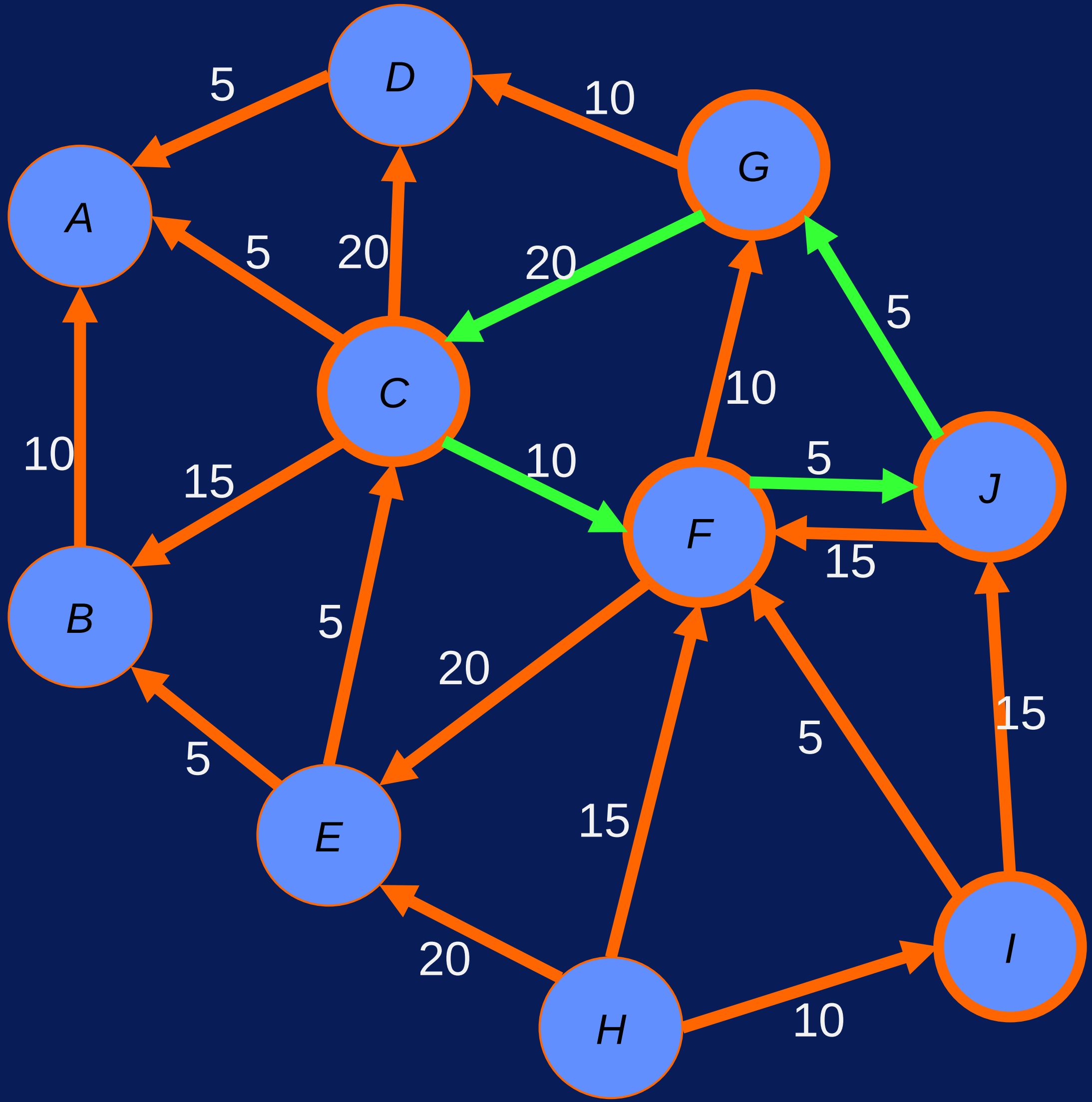
- ***Walk***
  - an alternating sequence of vertices and edges over a graph

A Walk in the Graph



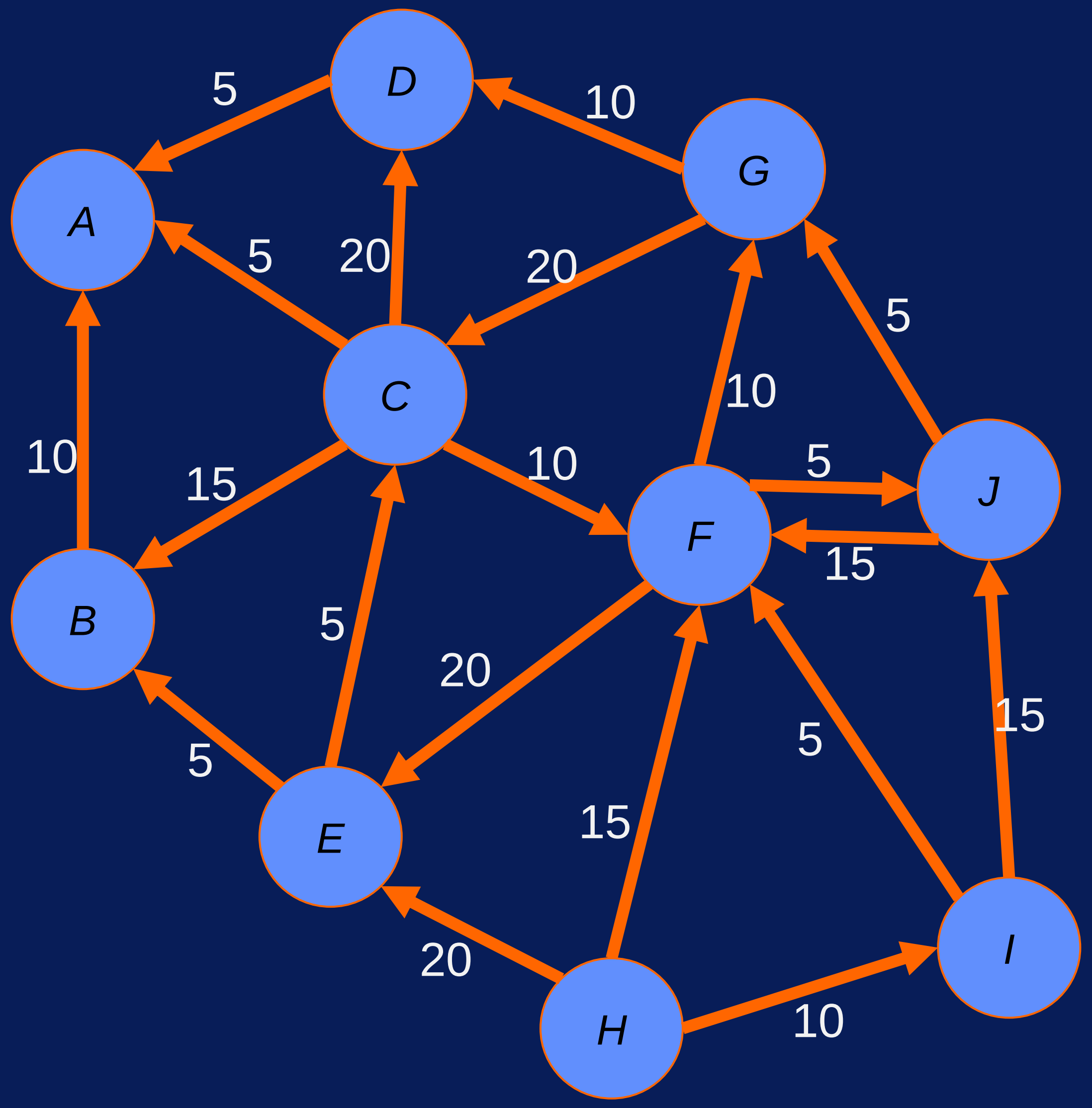
- ***Constraining a Walk***
  - Path
    - A walk with no repeating node except possibly for the first and last

A Walk in the Graph



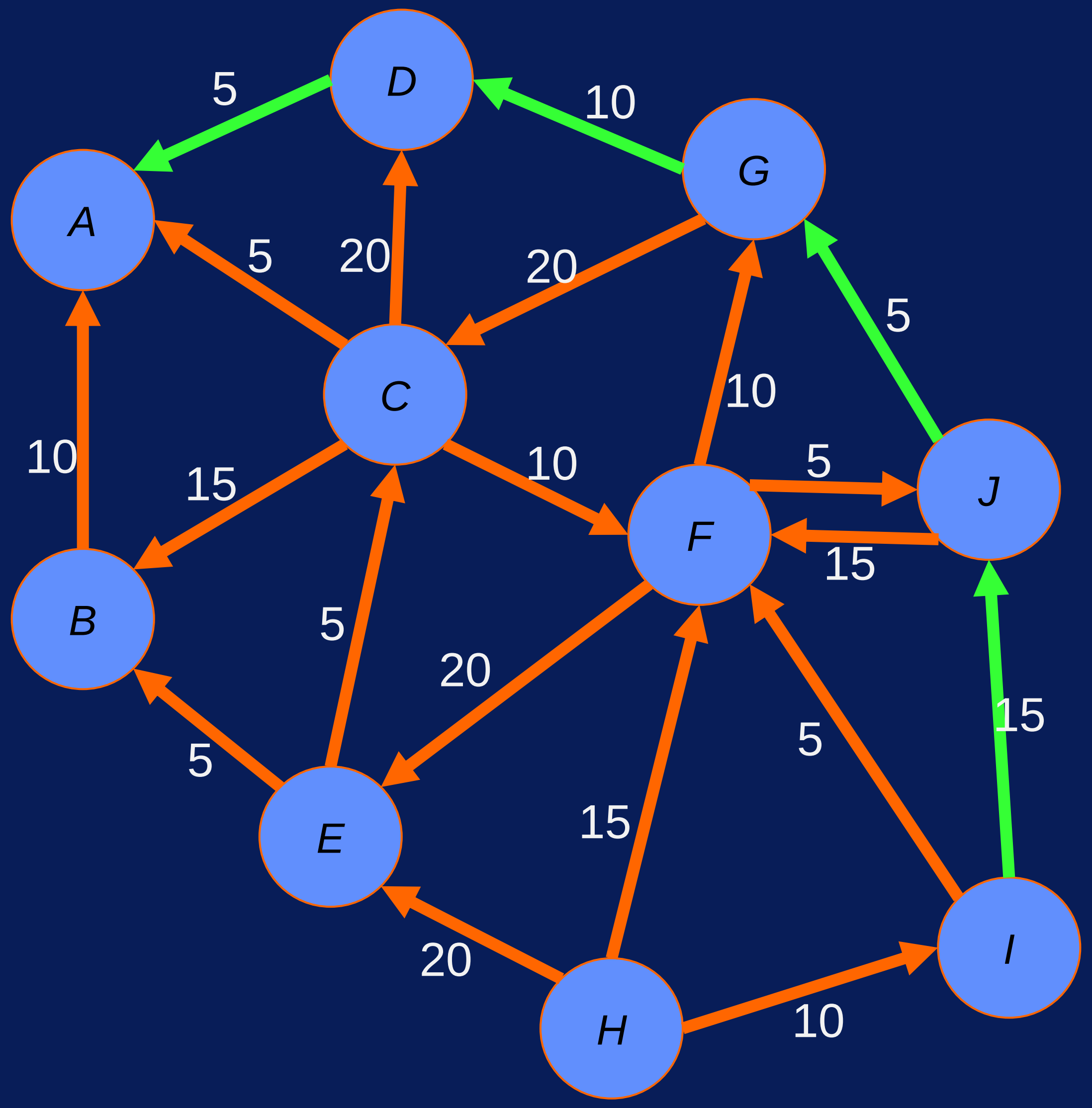
- ***Constraining a Walk***
  - Cycle
    - A path of length n ≥ 3 whose start and end vertices are the same
  - Acyclic
    - Graph with no cycles

A Walk in the Graph



- ***Constraining a Walk***
  - Trail
    - A walk with no repeating edge
      - H→F→G→C→F→E→C→F→J **not a trail**

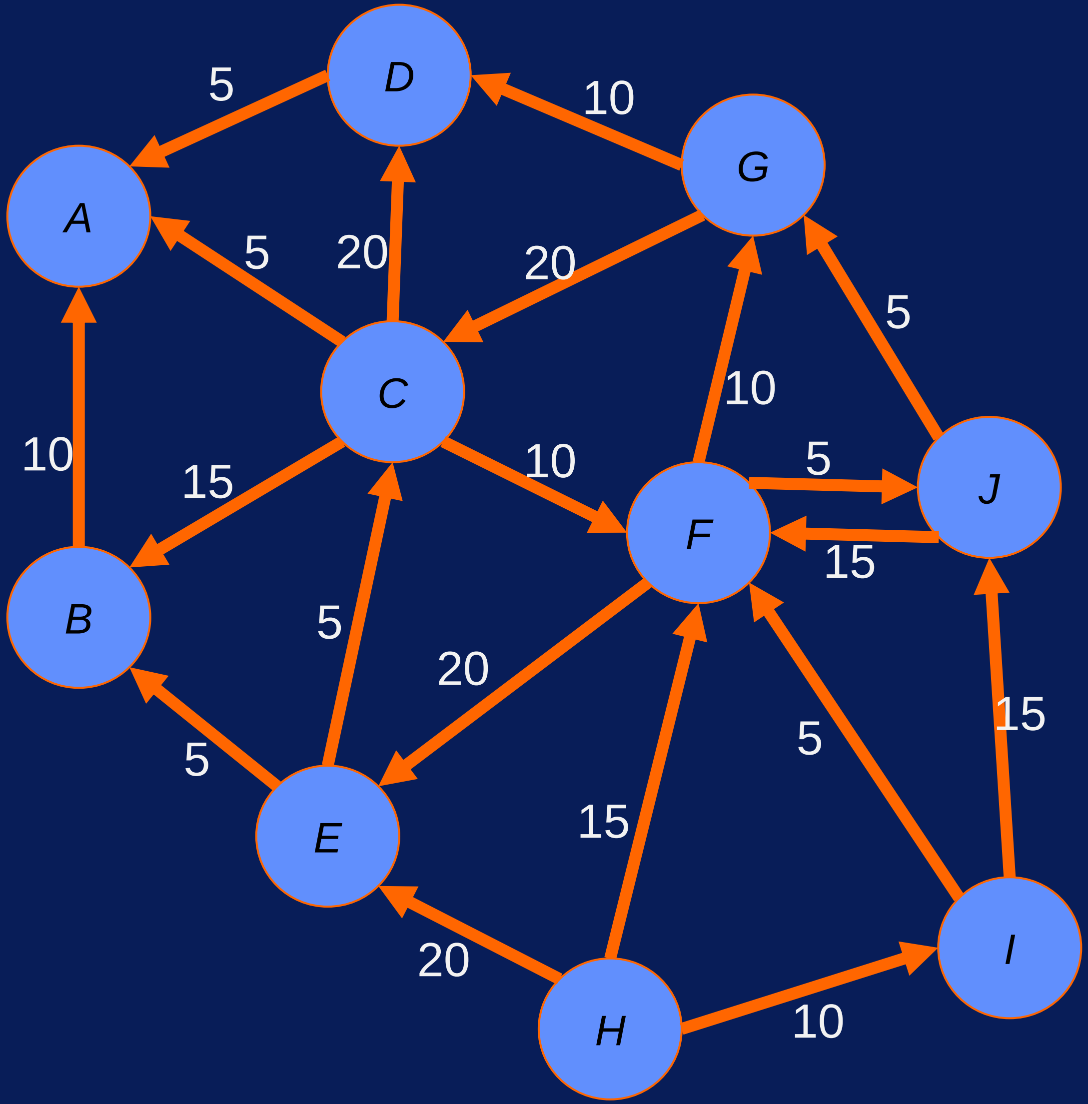
A Walk in the Graph



- ***Reachability***
  - node u is reachable from node v if there is a path from u to v
    - A is reachable from I
    - I is not reachable from A

Diameter of a Graph

Maximum pairwise distance between nodes



To

From

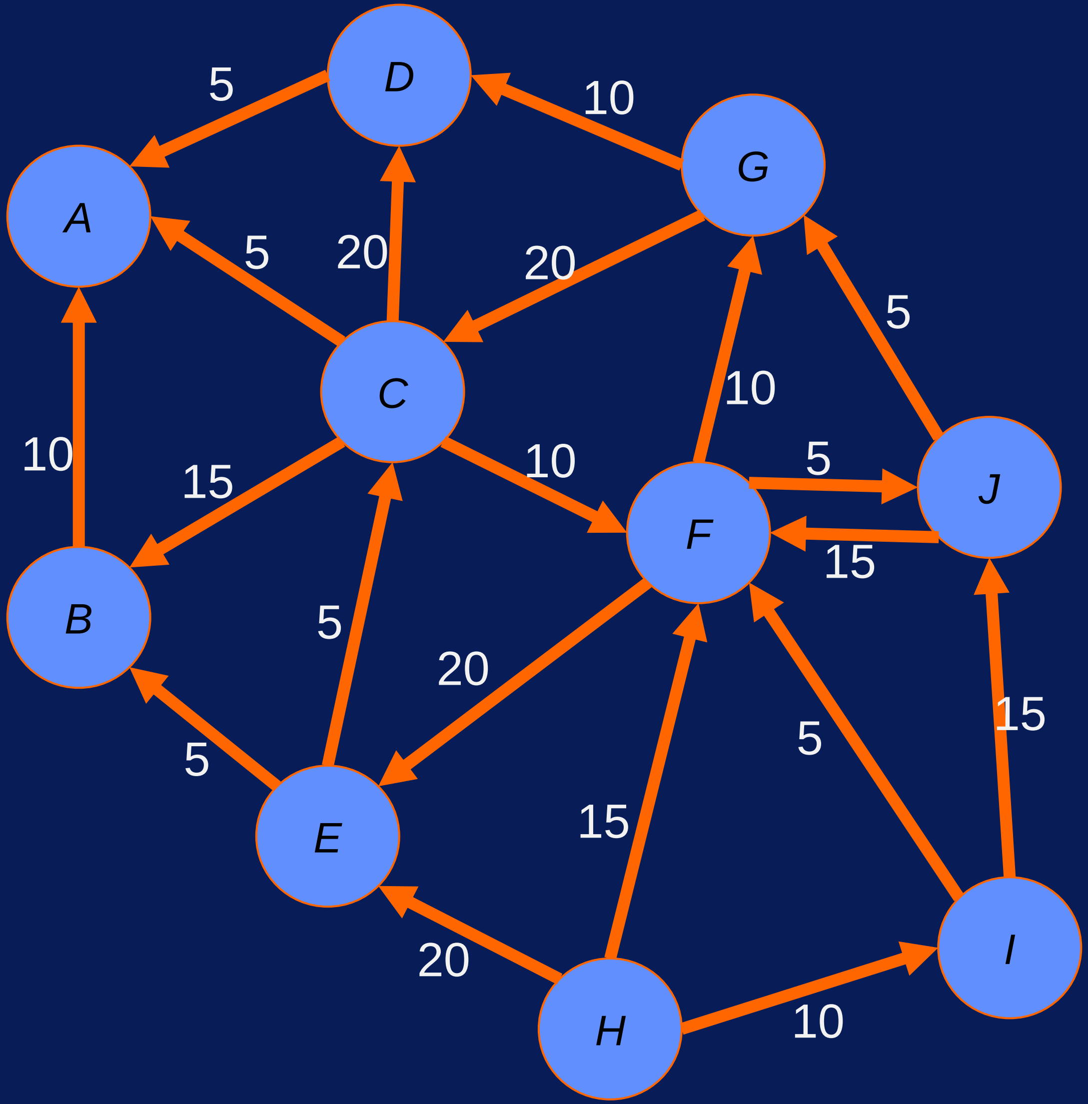
	A	B	C	D	E	F	G	H	I	J
A	0	∞	∞	∞	∞	∞	∞	∞	∞	∞
B	1	0	∞	∞	∞	∞	∞	∞	∞	∞
C	1	1	0	1	2	1	2	∞	∞	2
D	1	∞	∞	0	∞	∞	∞	∞	∞	∞
E	2	1	1	2	0	2	3	∞	∞	3
F	3	3	2	2	1	0	1	∞	∞	1
G	2	2	1	1	3	2	0	∞	∞	3
H	3	2	2	3	1	1	2	0	1	2
I	4	3	3	3	2	1	2	∞	0	1
J	3	3	2	2	2	1	1	∞	∞	0

Largest value

(Shortest-hop) Distance Matrix

Diameter of a Graph

Maximum pairwise distance between nodes



To

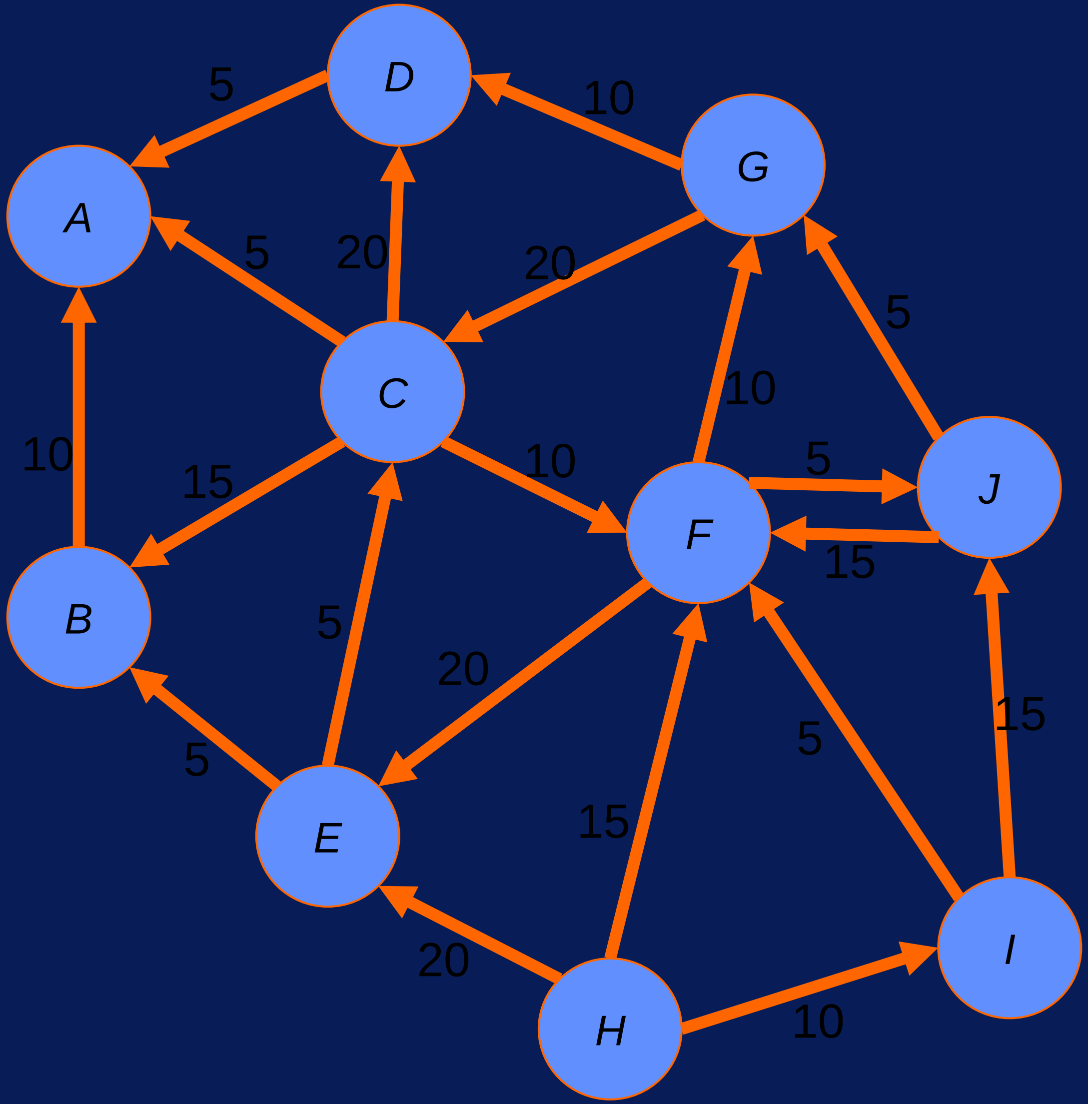
From

	A	B	C	D	E	F	G	H	I	J
A	0	∞	∞	∞	∞	∞	∞	∞	∞	∞
B	1	0	∞	∞	∞	∞	∞	∞	∞	∞
C	1	1	0	1	2	1	2	∞	∞	2
D	1	∞	∞	0	∞	∞	∞	∞	∞	∞
E	2	1	1	2	0	2	3	∞	∞	3
F	3	3	2	2	1	0	1	∞	∞	1
G	2	2	1	1	3	2	0	∞	∞	3
H	3	2	2	3	1	1	2	0	1	2
I	4	3	3	3	2	1	2	∞	0	1
J	3	3	2	2	2	1	1	∞	∞	0

Largest value

(Shortest-hop) Distance Matrix

pause



The Basic Path Analytics Question

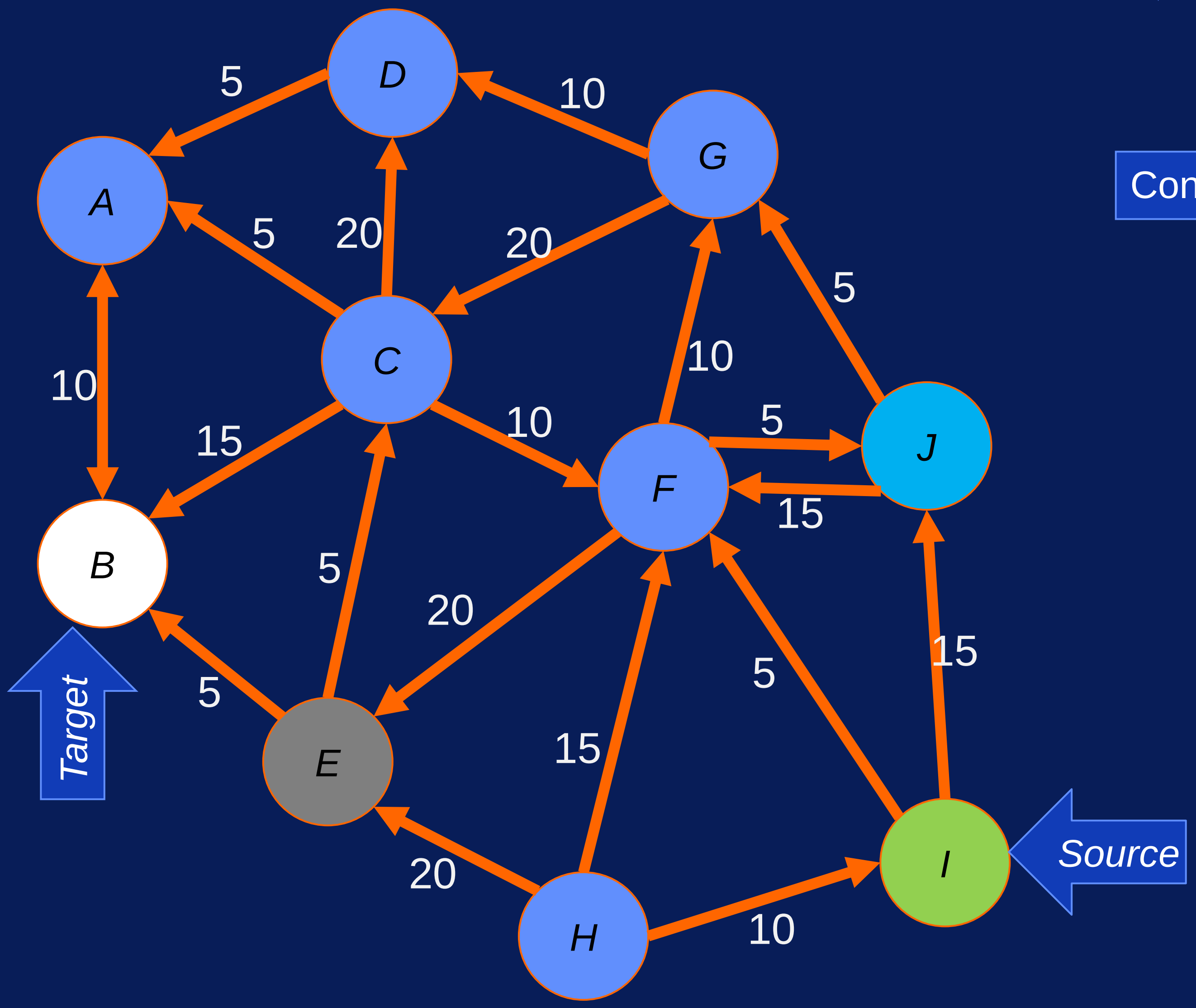
- **What is the “best” path to go from node 1 to node 2?**
  - Specification of “best” may include
    - Function to optimize
    - Nodes/edges to traverse
    - Nodes/edges to avoid
    - Preferences to satisfy

Optimization function

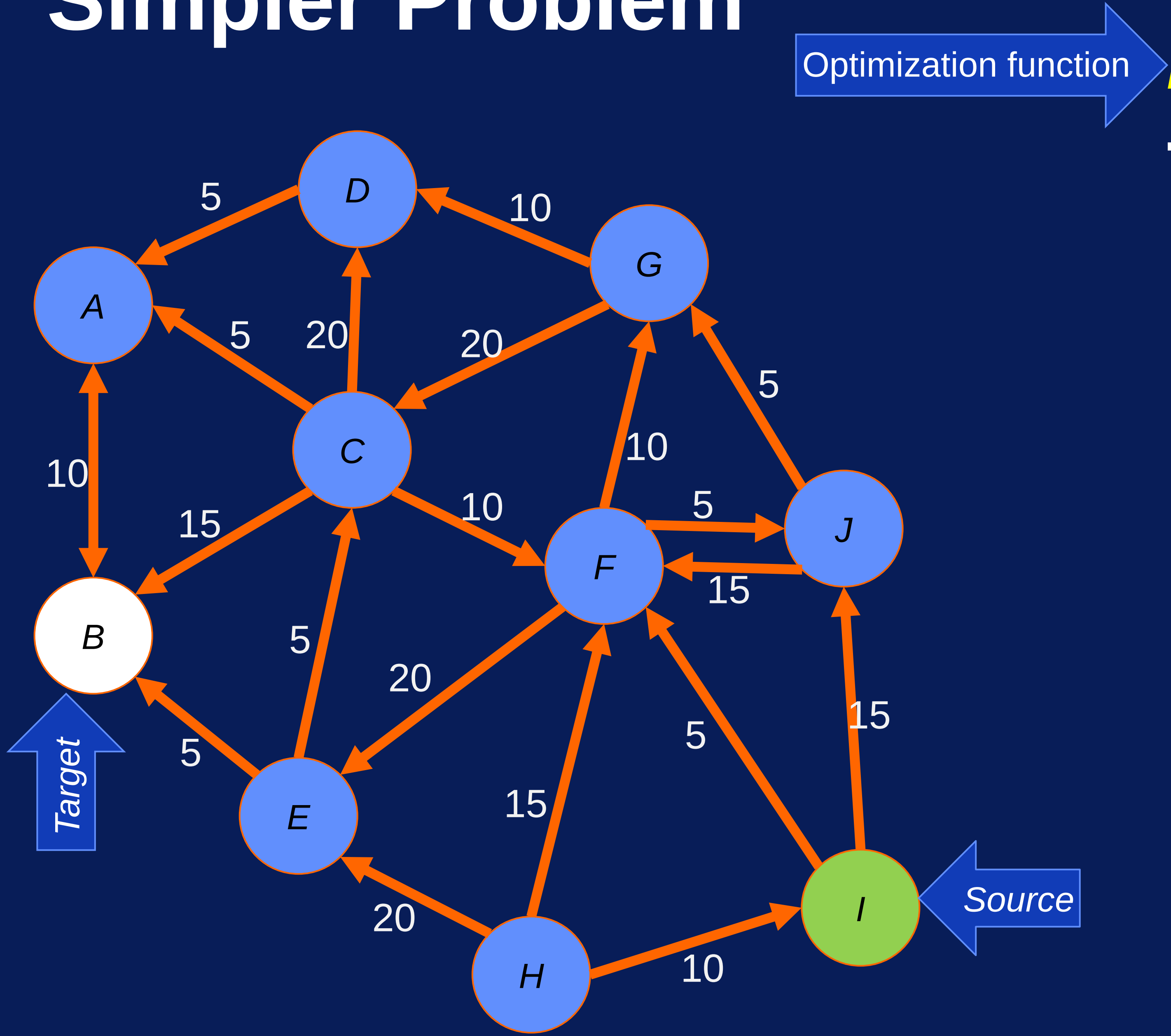
Find **least weight path** from I to B

Constraints

- <u>Avoid</u> paths through E
- <u>Must</u> go through J

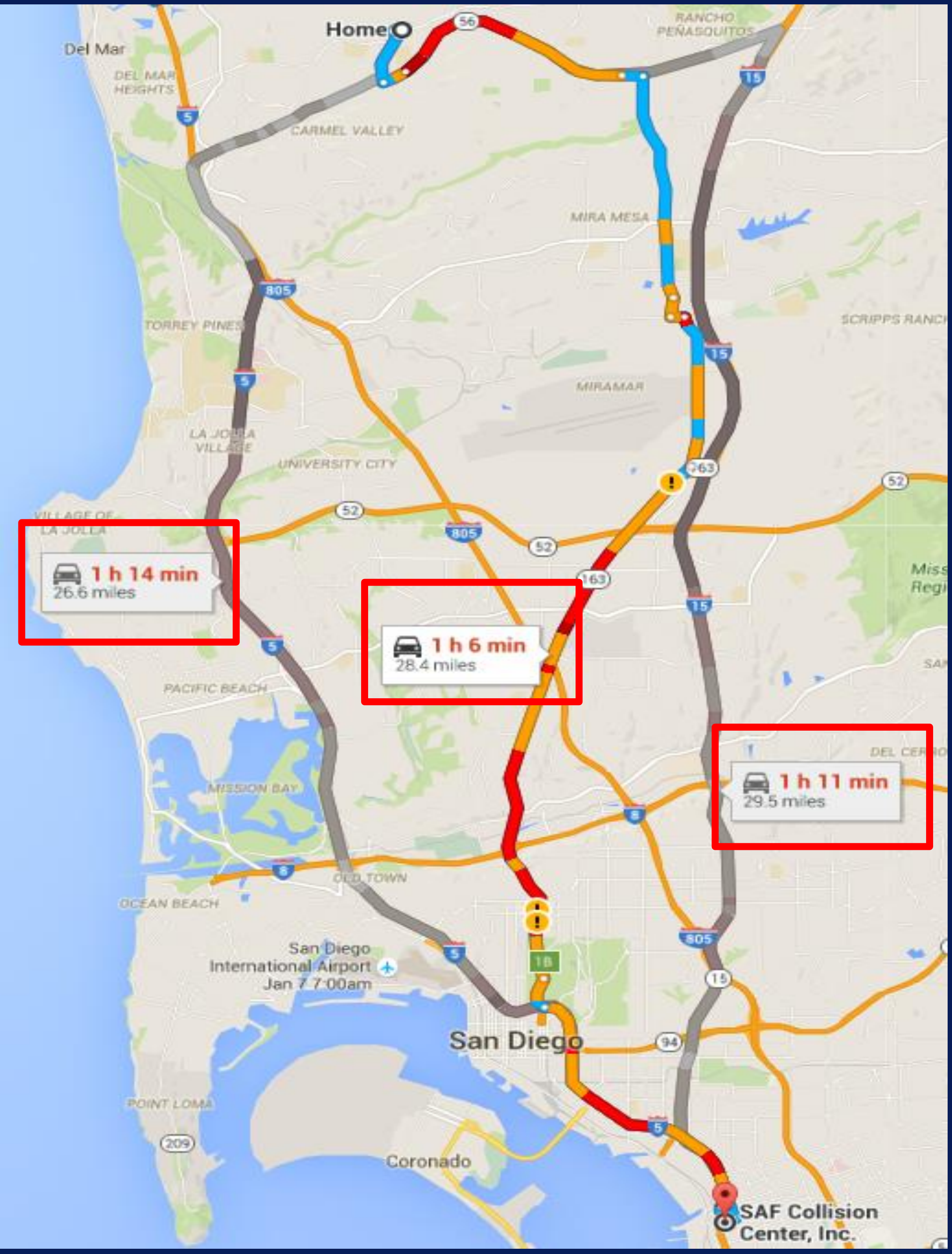


Simpler Problem



Find
least weight path
from I to B

Routes on Google Maps



- "Shortest" route would change based on weather, traffic, road condition…

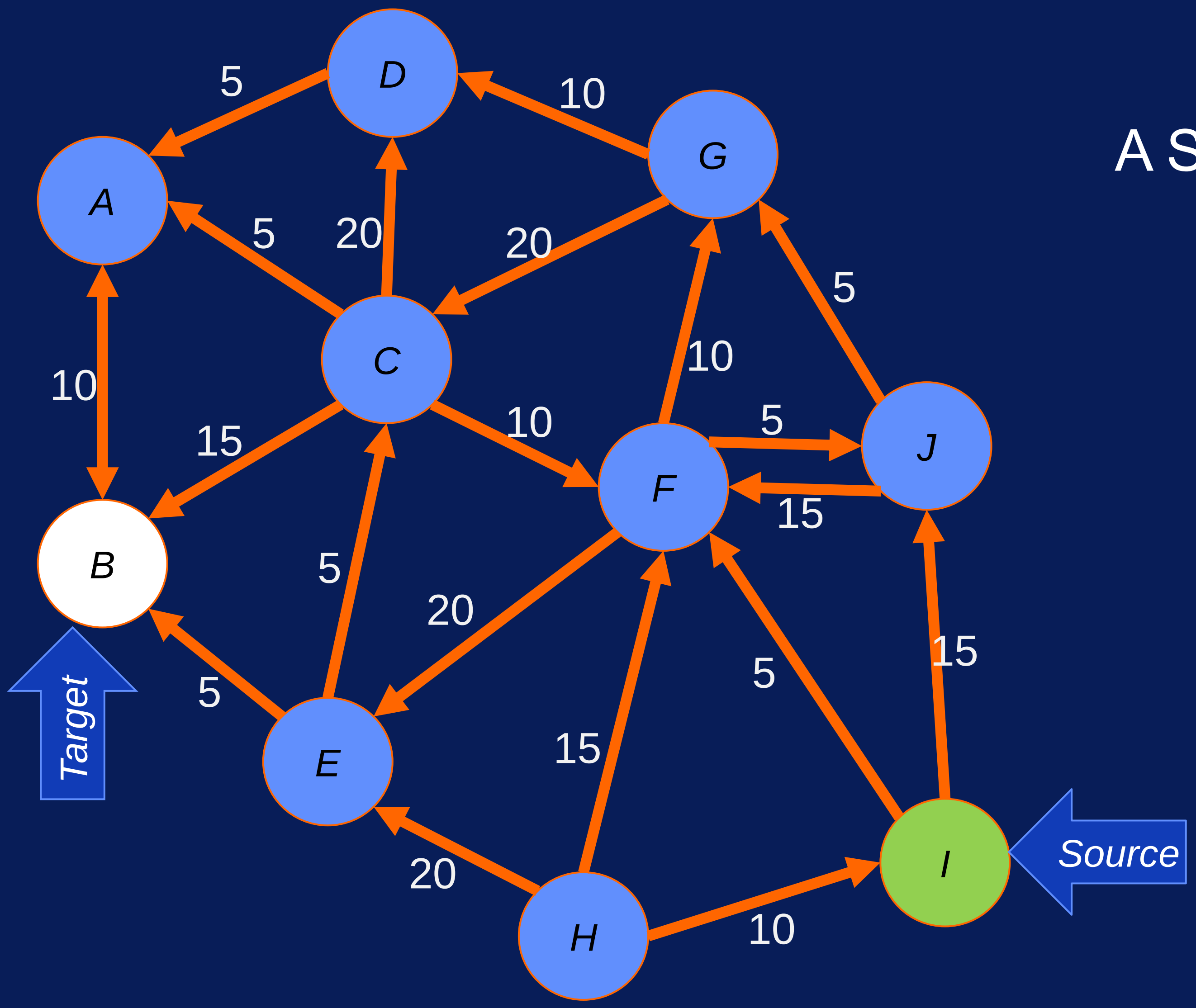
Simpler Problem

Optimization function

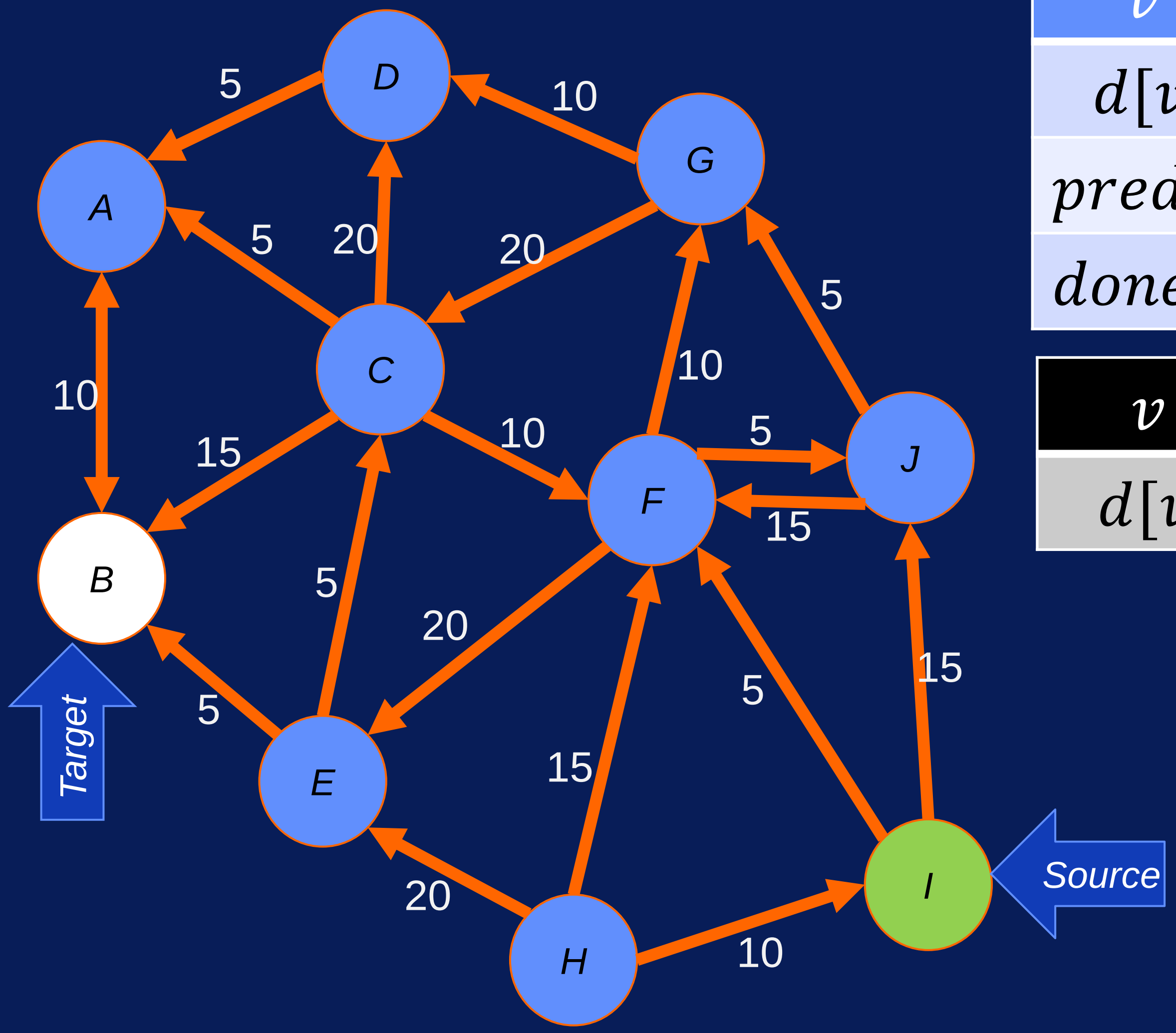
Find *least weight path* from I to B

A Standard Method: Dijkstra's algorithm

Many good online tutorials



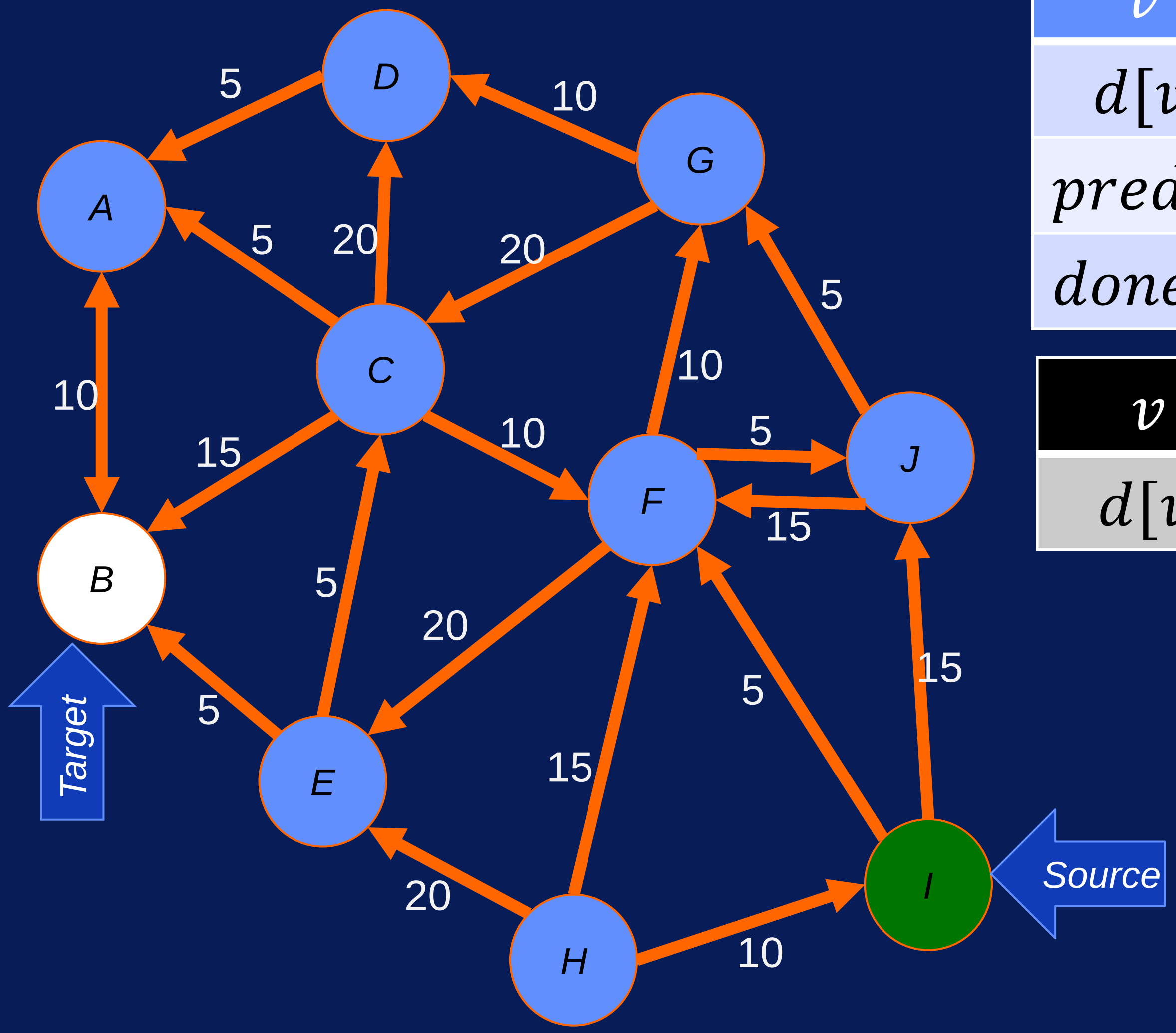
Dijkstra's Algorithm



v	I	A	B	C	D	E	F	G	H	J
$d[v]$	0	∞	∞	∞	∞	∞	∞	∞	∞	∞
$pred[v]$										
$done[v]$	N	N	N	N	N	N	N	N	N	N

v	I	A	B	C	D	E	F	G	H	J
$d[v]$	0	∞	∞	∞	∞	∞	∞	∞	∞	∞

Step 1



v	I	A	B	C	D	E	F	G	H	J
$d[v]$	0	∞	∞	∞	∞	∞	$\boxed{5}$	∞	∞	15
$pred[v]$							I			I
$done[v]$	Y	N	N	N	N	N	N	N	N	N

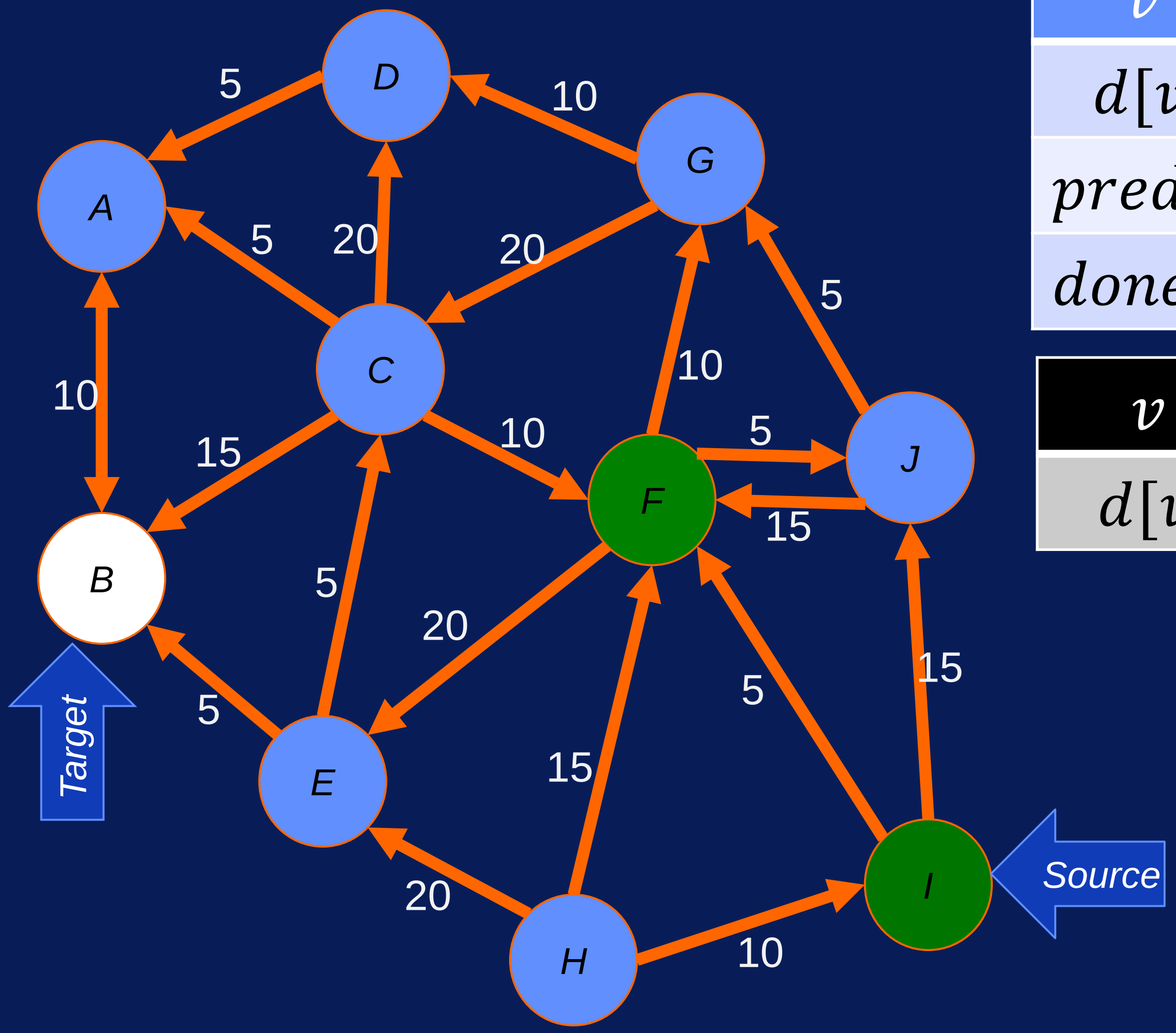
v	I	A	B	C	D	E	F	G	H	J
$d[v]$	0	∞	∞	∞	∞	∞	$\boxed{5}$	∞	∞	15

Step 2

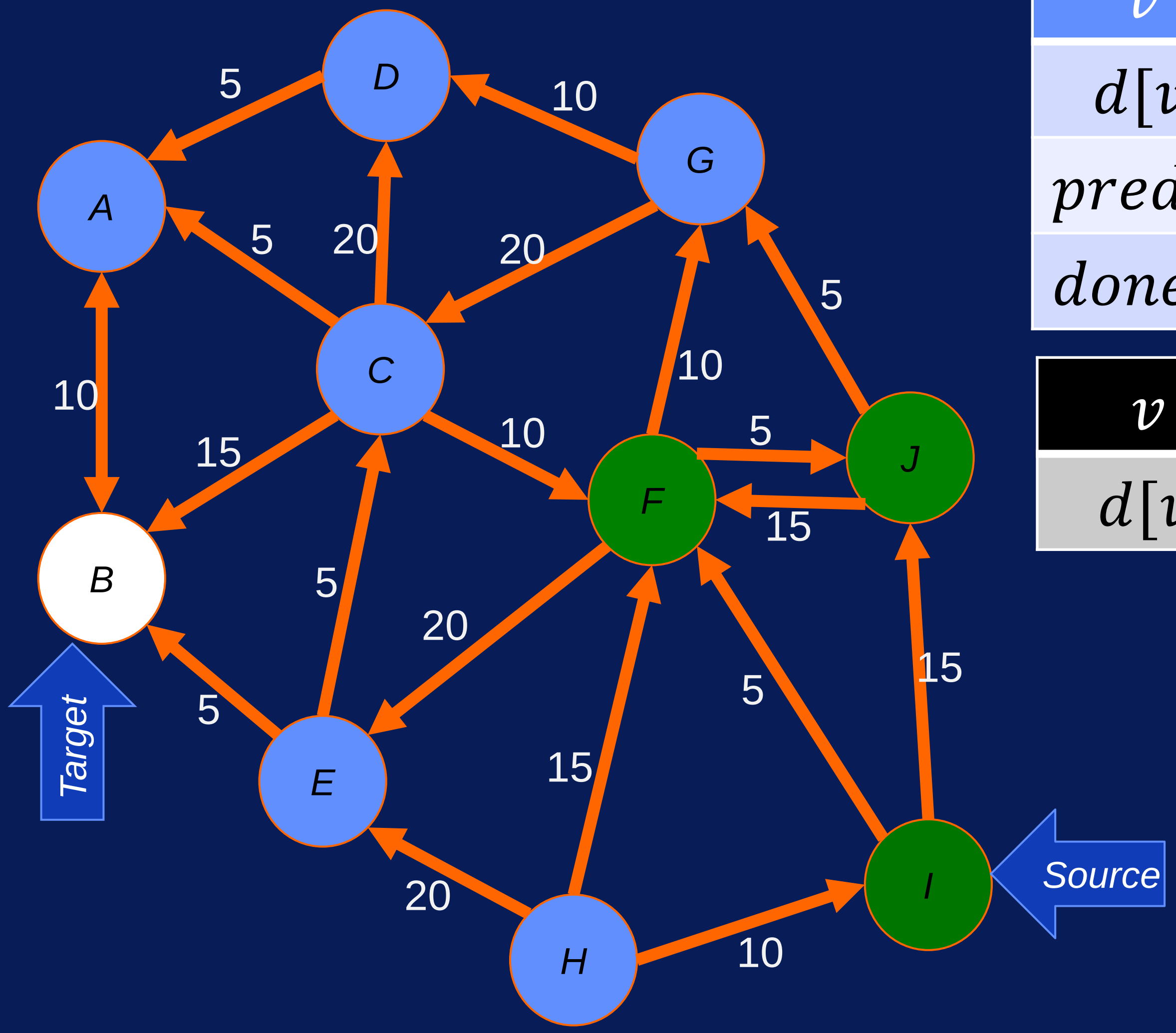
5+5

v	I	A	B	C	D	E	F	G	H	J
$d[v]$	0	∞	∞	∞	∞	25	5	15	∞	$\boxed{10}$
$pred[v]$						F	I	F		$\boxed{F}$
$done[v]$	Y	N	N	N	N	N	Y	N	N	N

v	A	B	C	D	E	F	G	H	J
$d[v]$	∞	∞	∞	∞	25	5	15	∞	$\boxed{10}$



Step 3



v	I	A	B	C	D	E	F	G	H	J
$d[v]$	0	∞	∞	∞	∞	25	**5**	**15**	∞	10
$pred[v]$						F	I	F		F
$done[v]$	Y	N	N	N	N	N	Y	N	N	Y

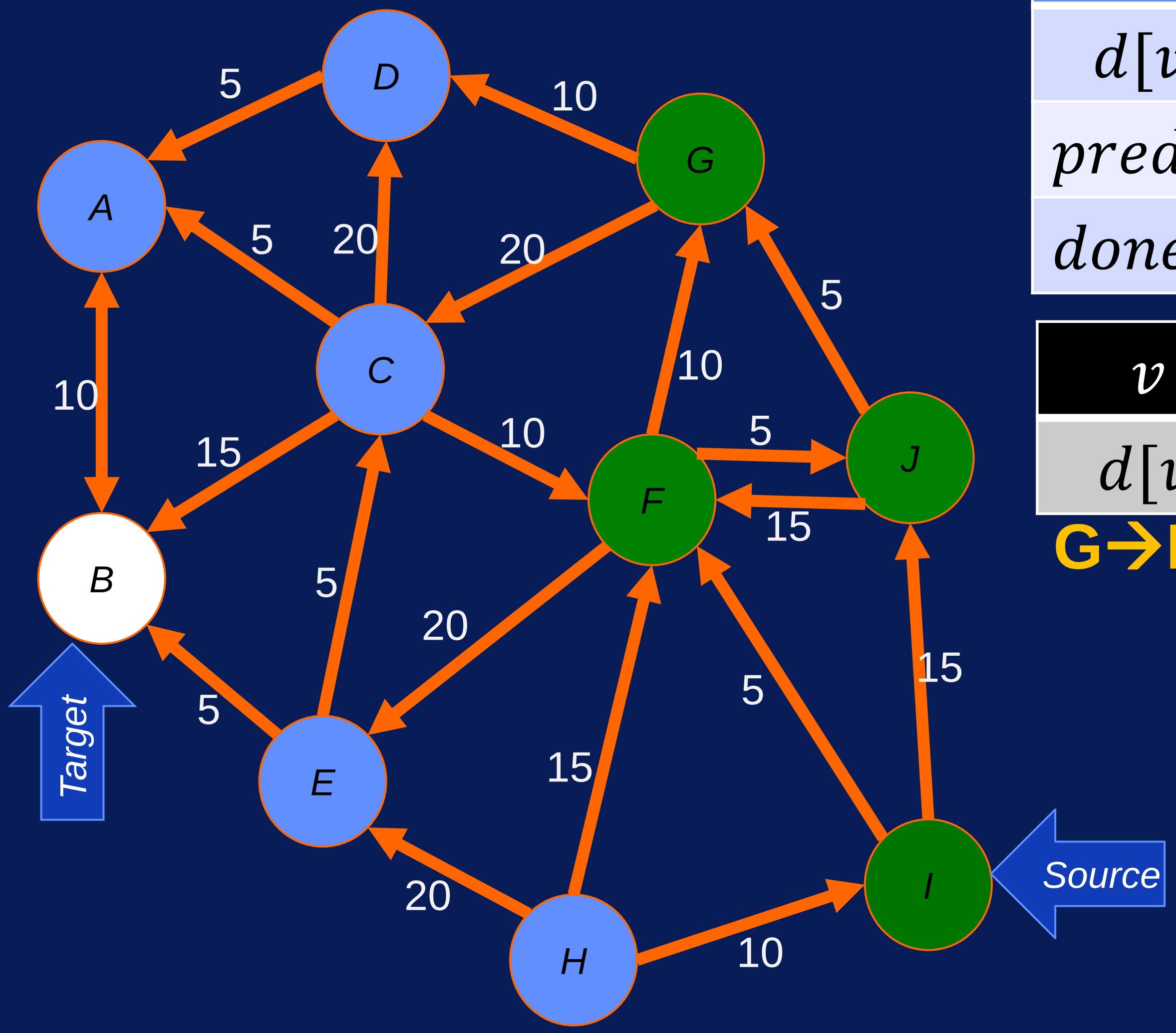
v	A	B	C	D	E	G	H	J
$d[v]$	∞	∞	∞	∞	25	15	∞	10

Step 4

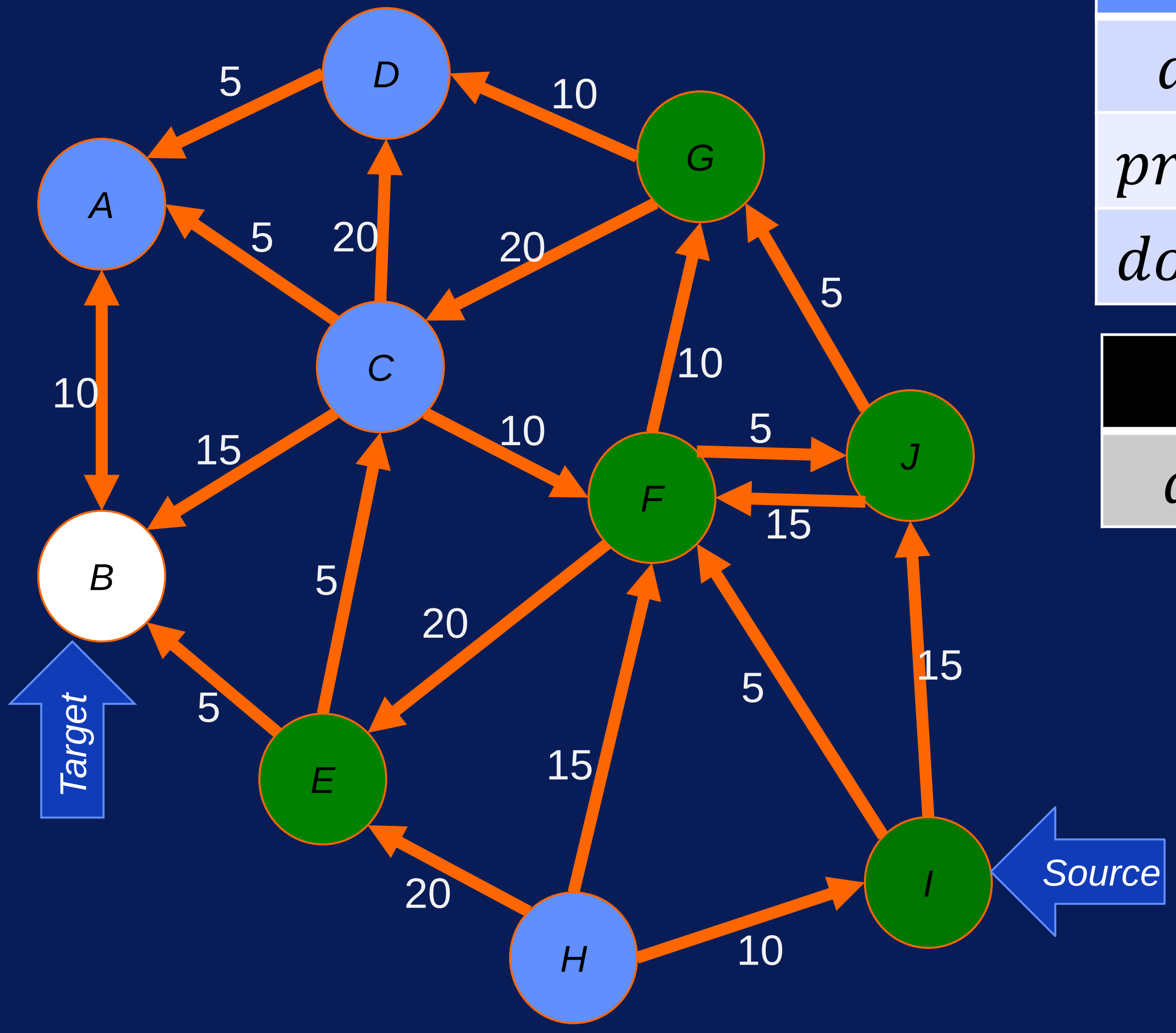
v	I	A	B	C	D	E	F	G	H	J
$d[v]$	0	∞	∞	35	25	25	5	15	∞	10
$pred[v]$				G	G	F	I	F		F
$done[v]$	Y	N	N	N	N	N	Y	Y	N	Y

v	A	B	C	D	E	G	H
$d[v]$	∞	∞	35	25	25	15	∞

G→D or F→E?



Step 5 (with E)



v	I	A	B	C	D	E	F	G	H	J
$d[v]$	0	∞	30	30	25	25	5	15	∞	10
$pred[v]$			E	E	G	F	I	F		F
$done[v]$	Y	N	N	Y	N	Y	Y	Y	N	Y

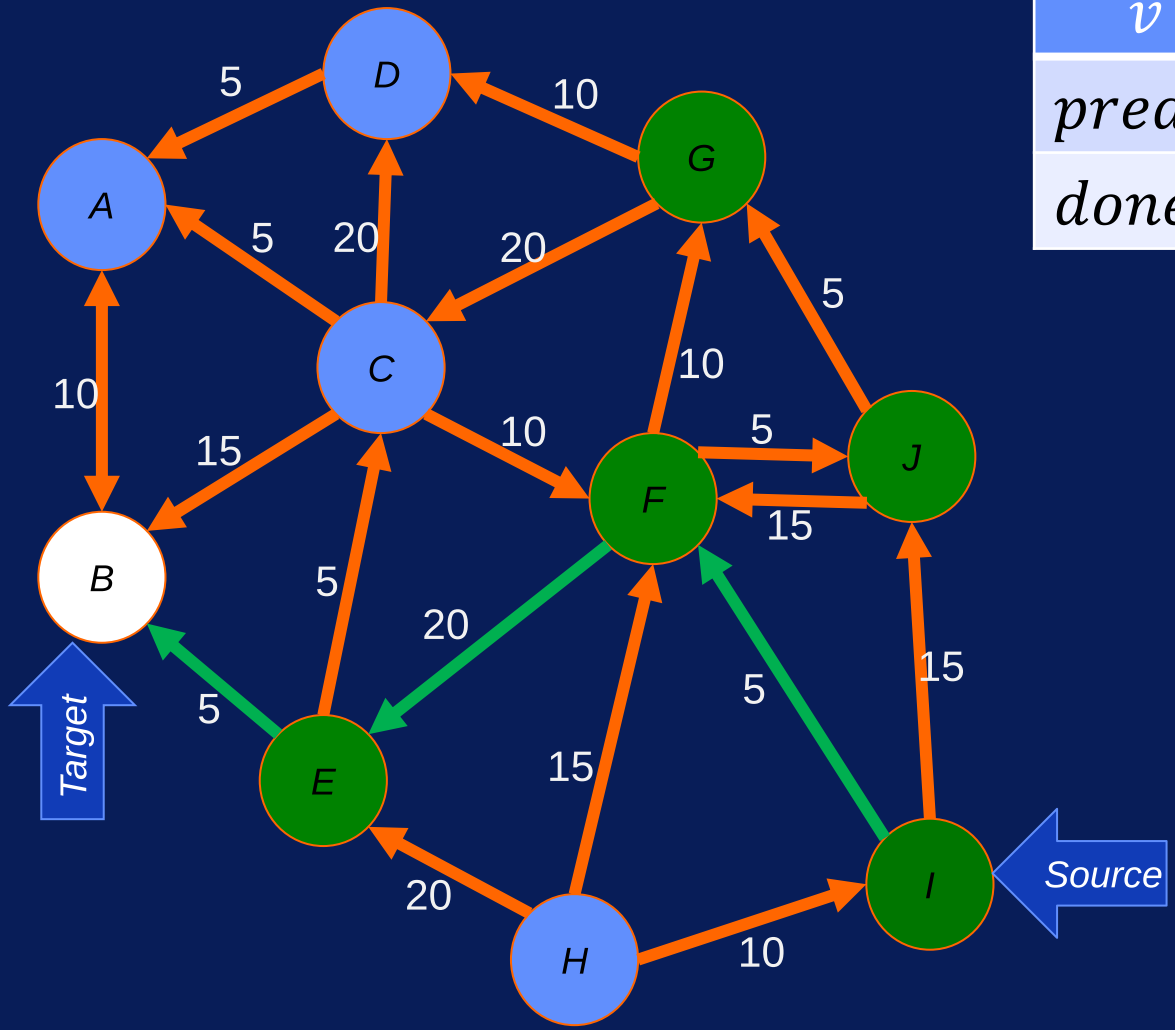
v	A	B	C	D	E	H
$d[v]$	∞	30	25	25	25	∞

Reached B!

Why is H still ∞?

H is a root and unreachable

Construct Path



v	I	A	B	C	D	E	F	G	H	J
$pred[v]$			E	E	G	F	I	F		F
$done[v]$	Y	N	Y	Y	N	Y	Y	Y	N	Y

Follow predecessors from target!

So how well does the algorithm work for big graphs?

Dijkstra and Big Graphs

- The worst-case complexity of Dijkstra is proportional to the number of edges times log(number of nodes)
    - For1 Million nodes and 10 Million edges, the worst case complexity is proportional to ~14 Million!!

That’s really high!!

We have two optional videos that you can go through.

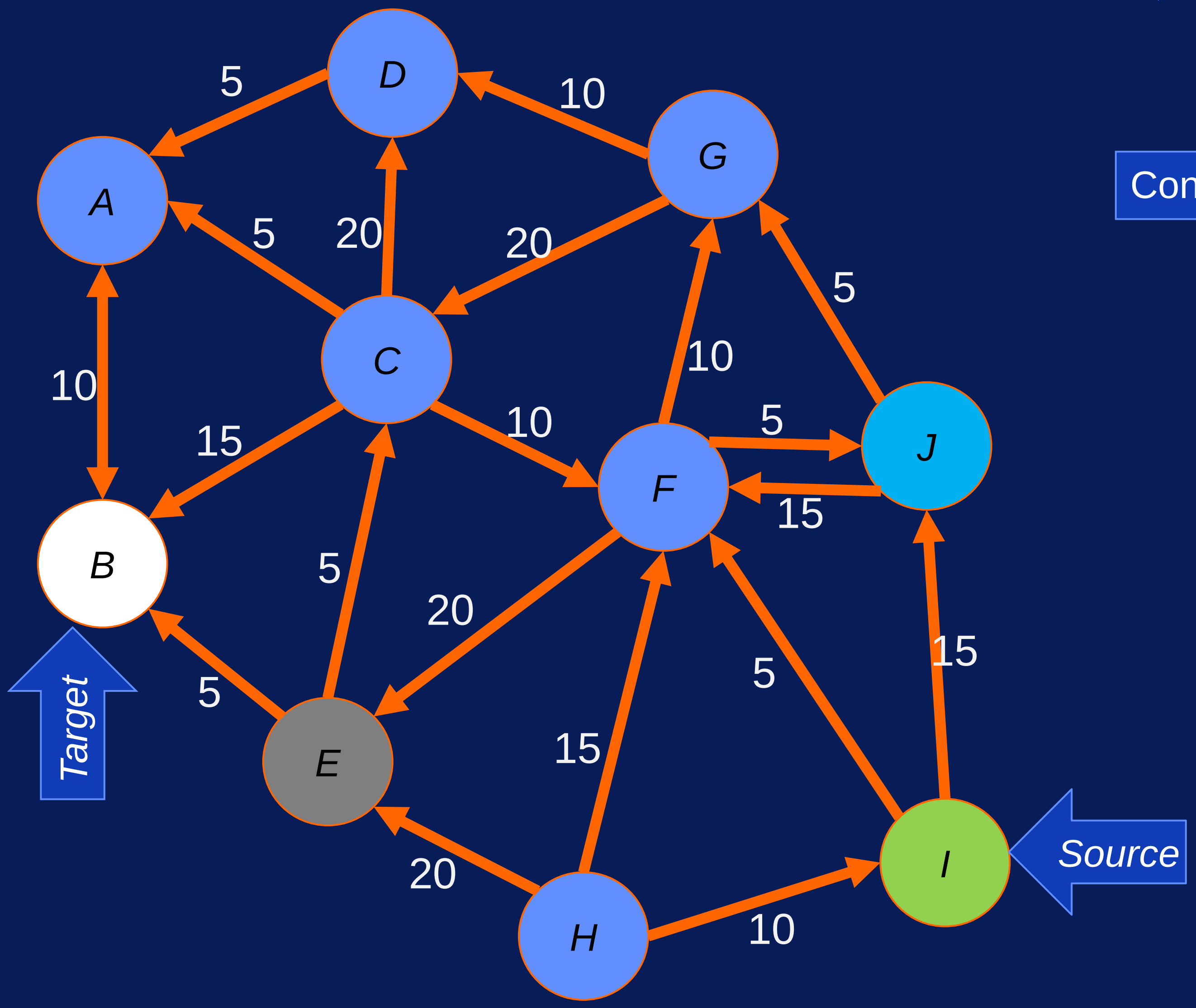
These videos present two modifications to improve Dijkstra’s algorithm practically

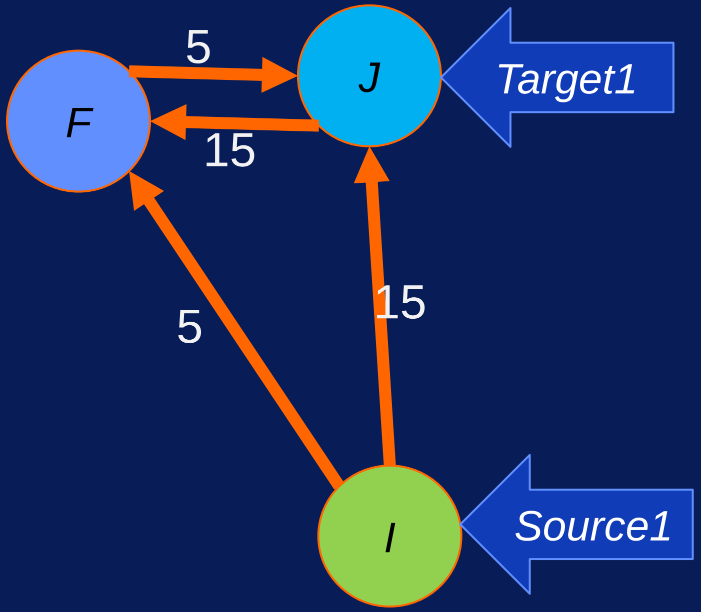
Optimization function

Find ***least weight path*** from I to B

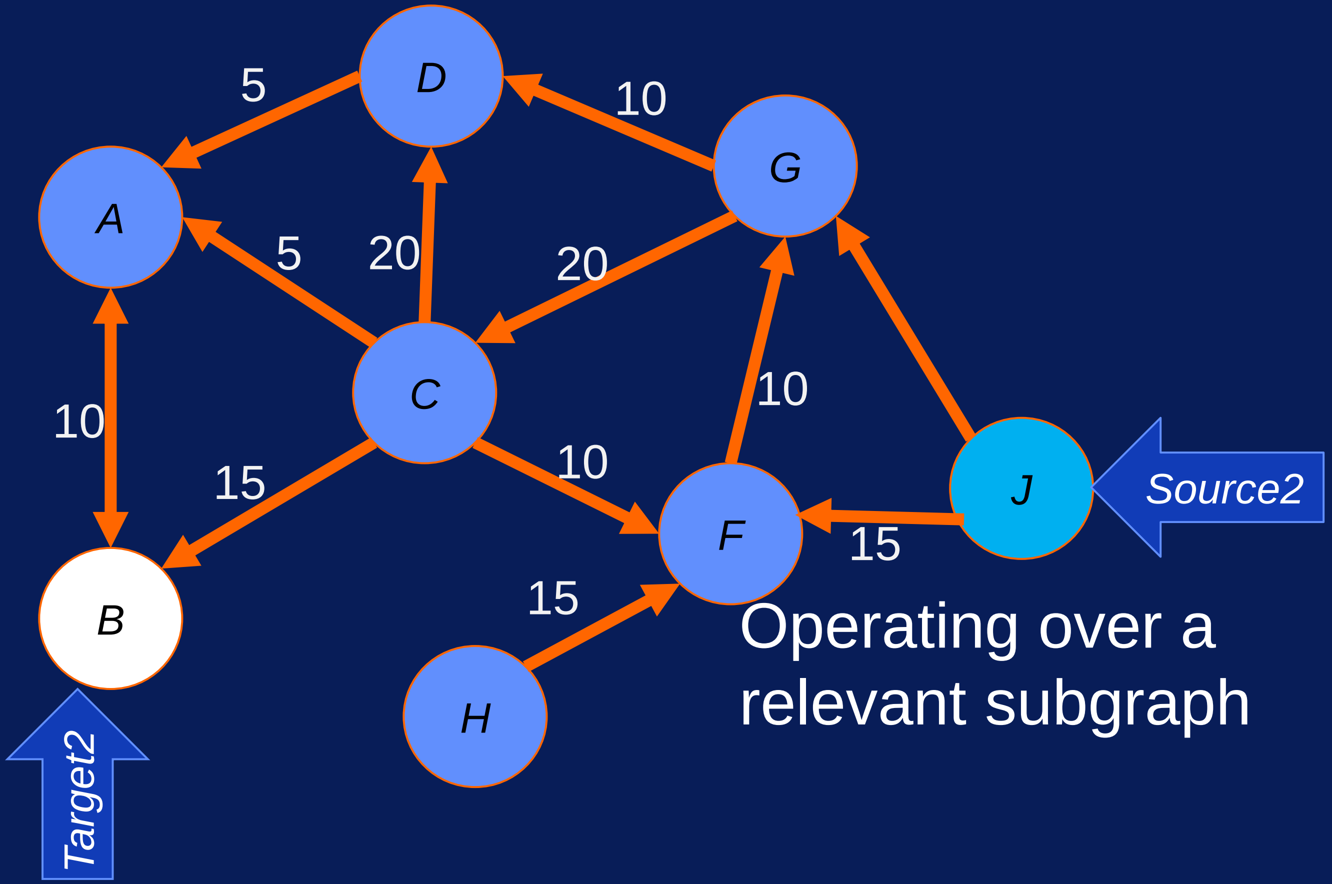
Constraints

- <u>Avoid</u> paths through E
- <u>Must</u> go through J





Splitting the task



Operating over a relevant subgraph

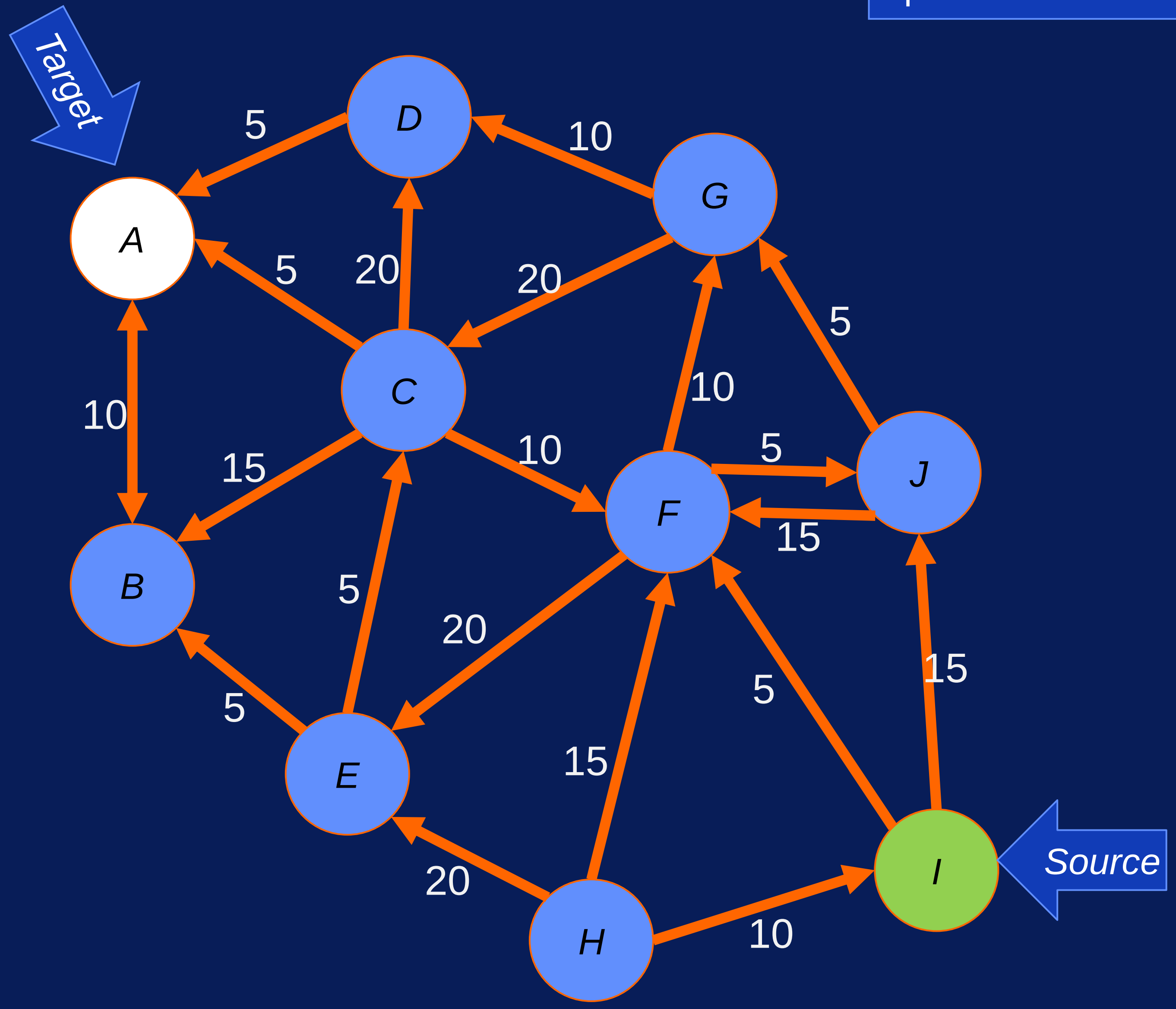
Optional 1

Can we do better?

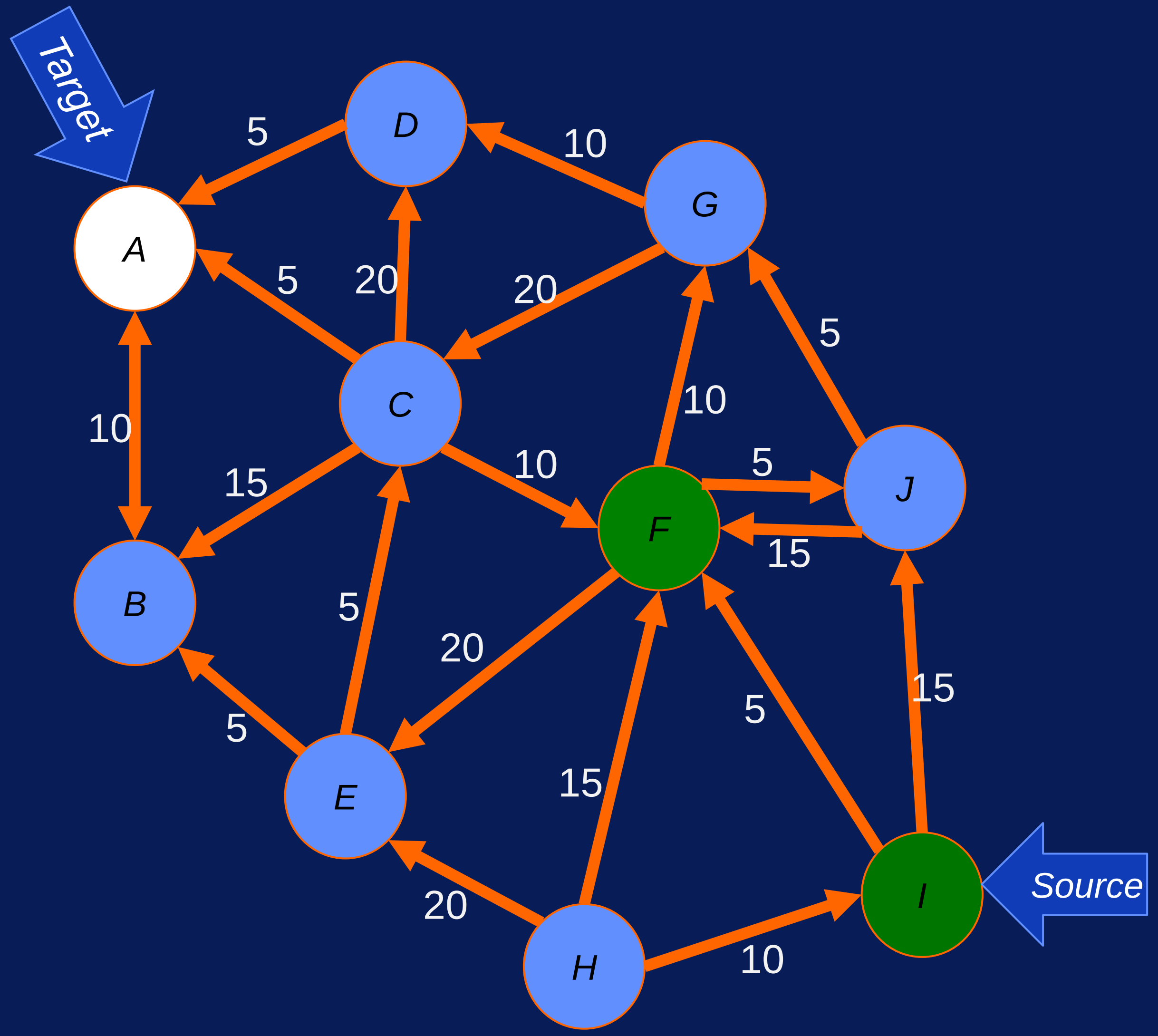
Optimization function

Find *least weight path* from I to A

Bi-directional Dijkstra algorithm

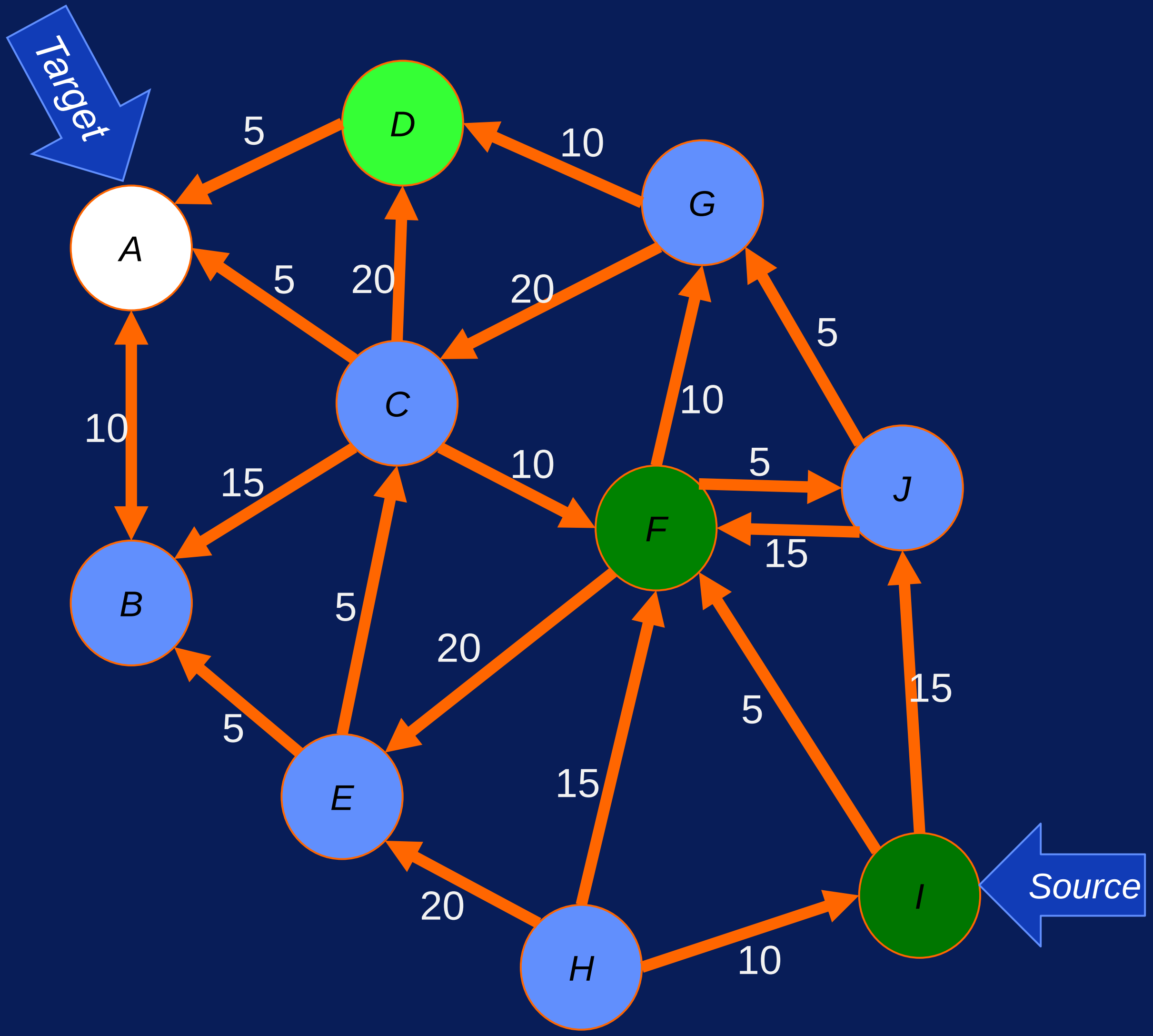


Step 1



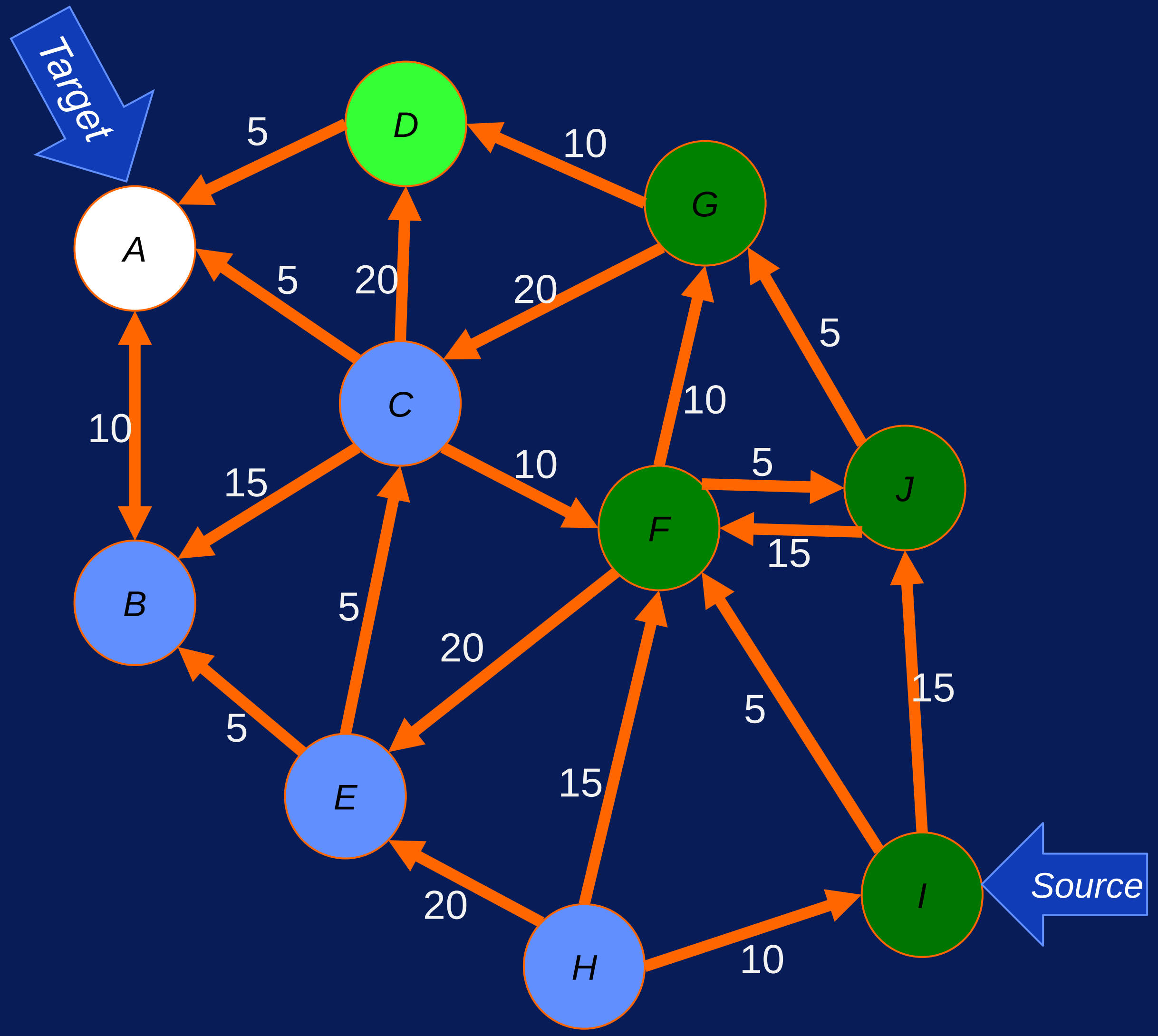
Start with the Source Node like before
Reach the first frontier
Length(I→F) = 5

Step 2



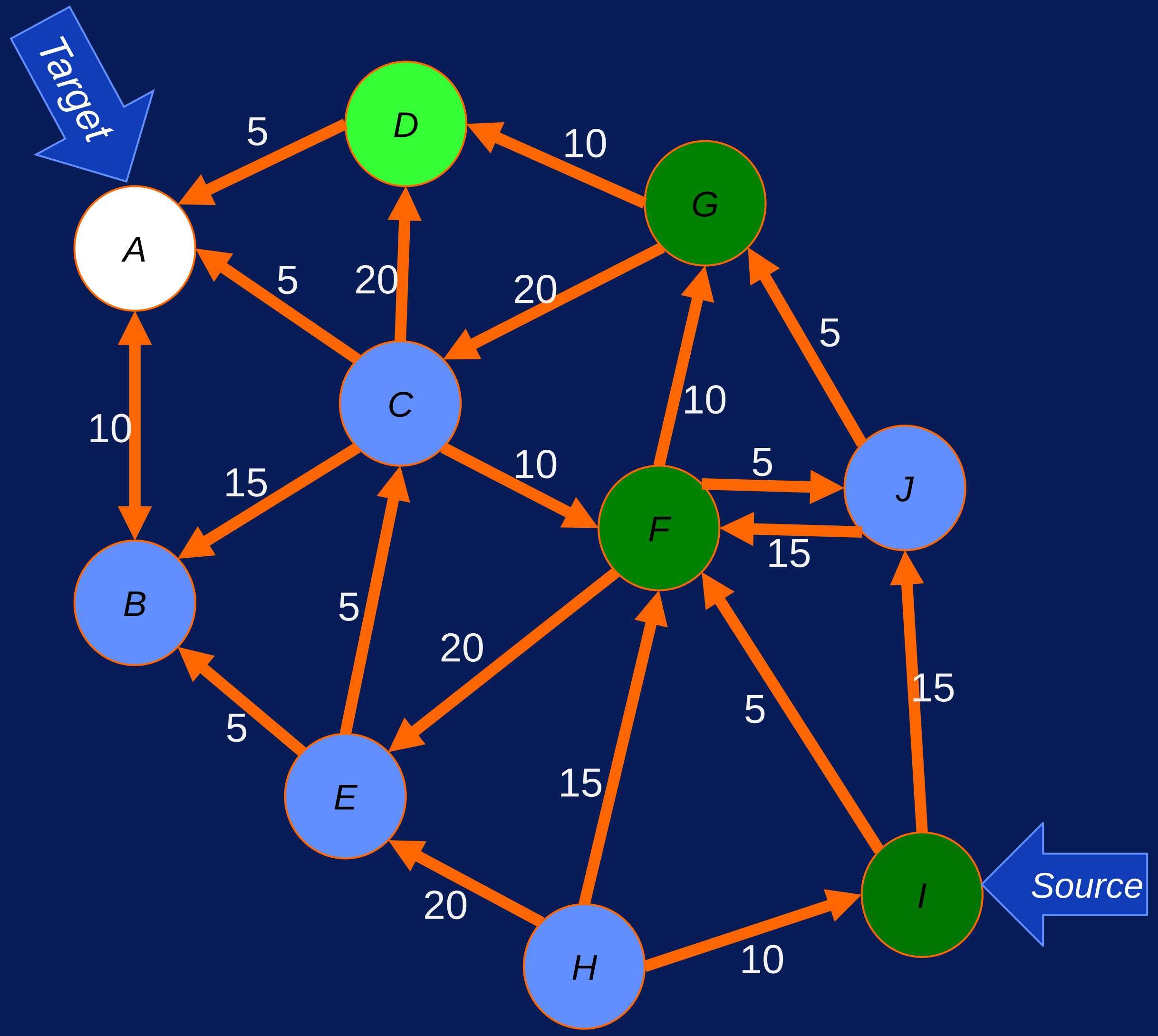
Start from Target Node
Pretend all edges are reversed
Reach the first reverse frontier
Length(A→D) = 5

Step 3



Alternate between forward frontier and reverse frontier
Length(I→F→G) = 15 (ignored J)
Length(A→D) = 5

Step 4

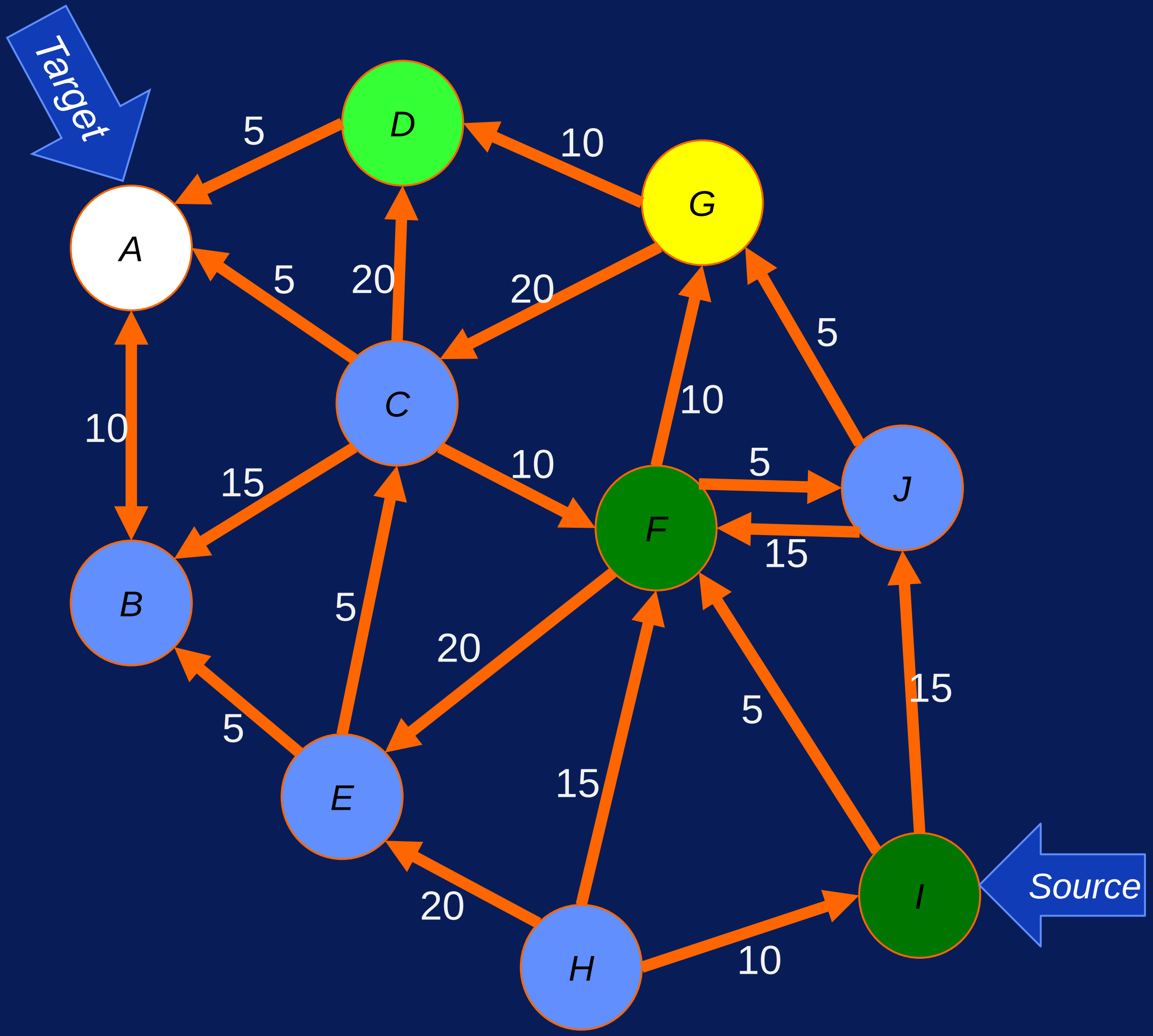


Alternate between forward frontier and reverse frontier

Length(I→F→G) = 15

Length(A→D→G) = 15

Step 5



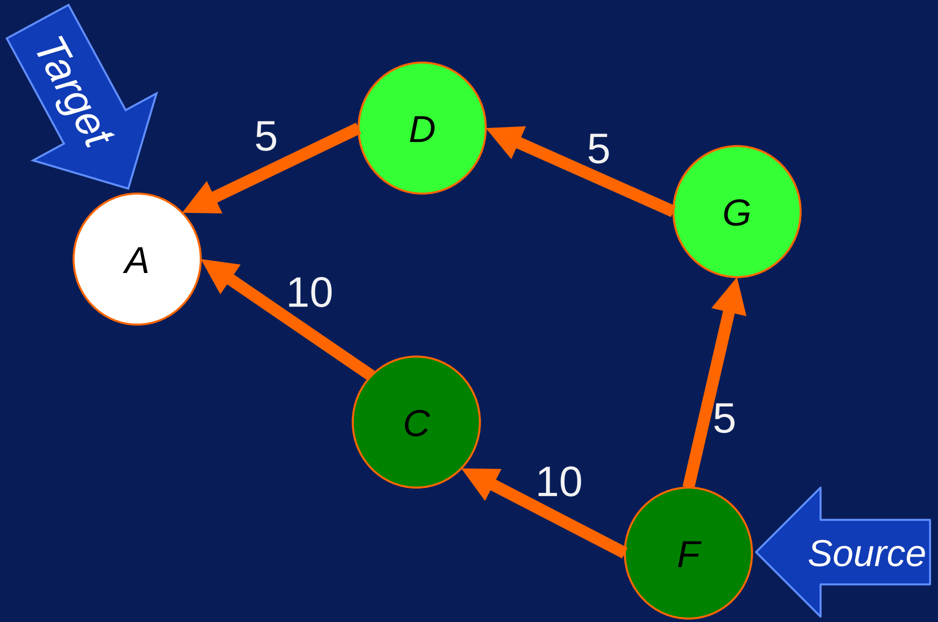
Until a front touches a node of the other front

Length(I→F→G) = 15

Length(A→D→G) = 15

Ensure that the sum of the weighted length is minimum

Least Weight, not Least Hop

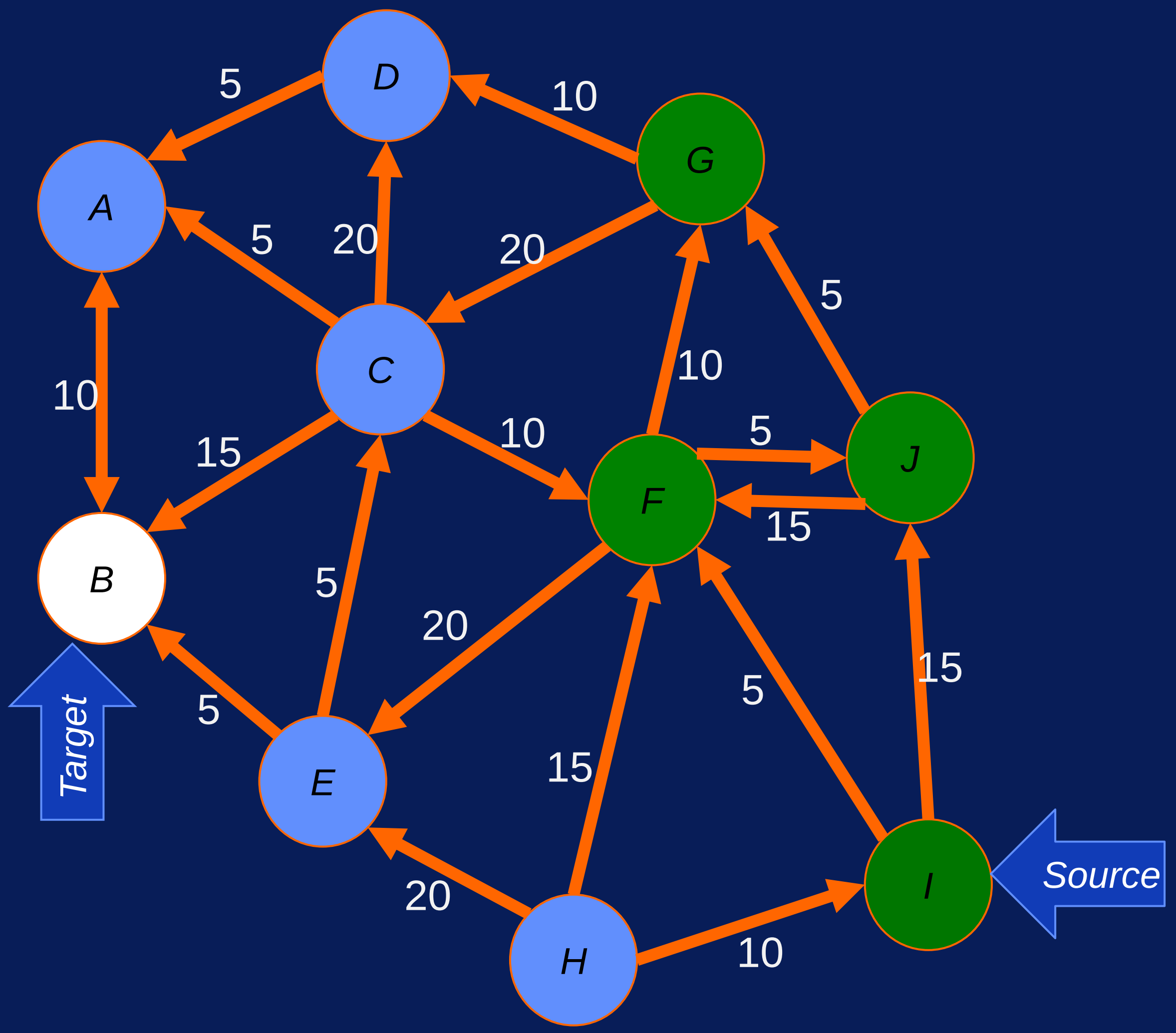


The 2-step path F→C→A
Has more weight than
the 3-step paths F→G→D→A

Count the sum of the shortest weighted path from both frontiers

Optional 2

Unresolved Issue

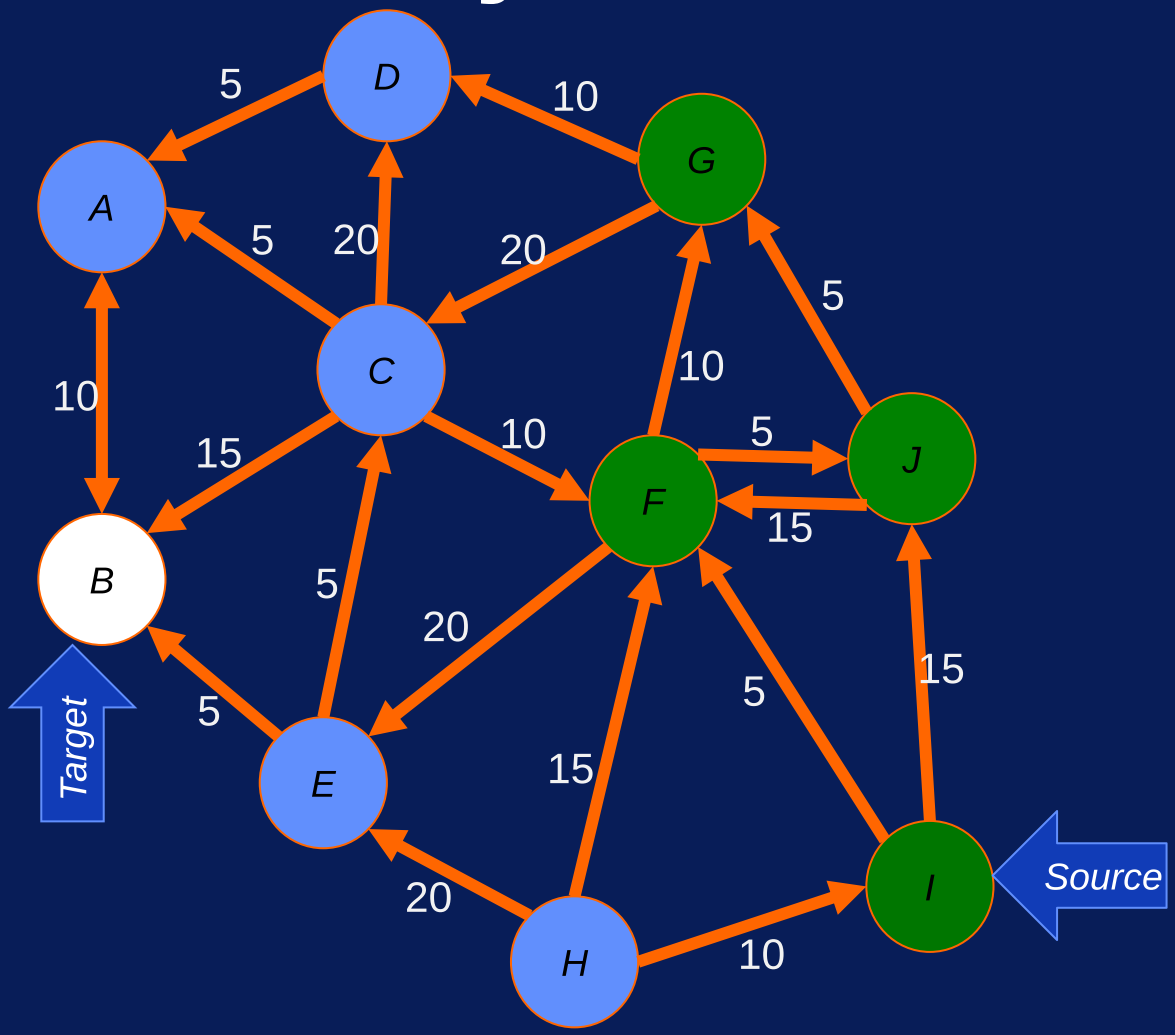


Path taken by algorithm: F→G

IF we could take F→E instead, it would terminate sooner

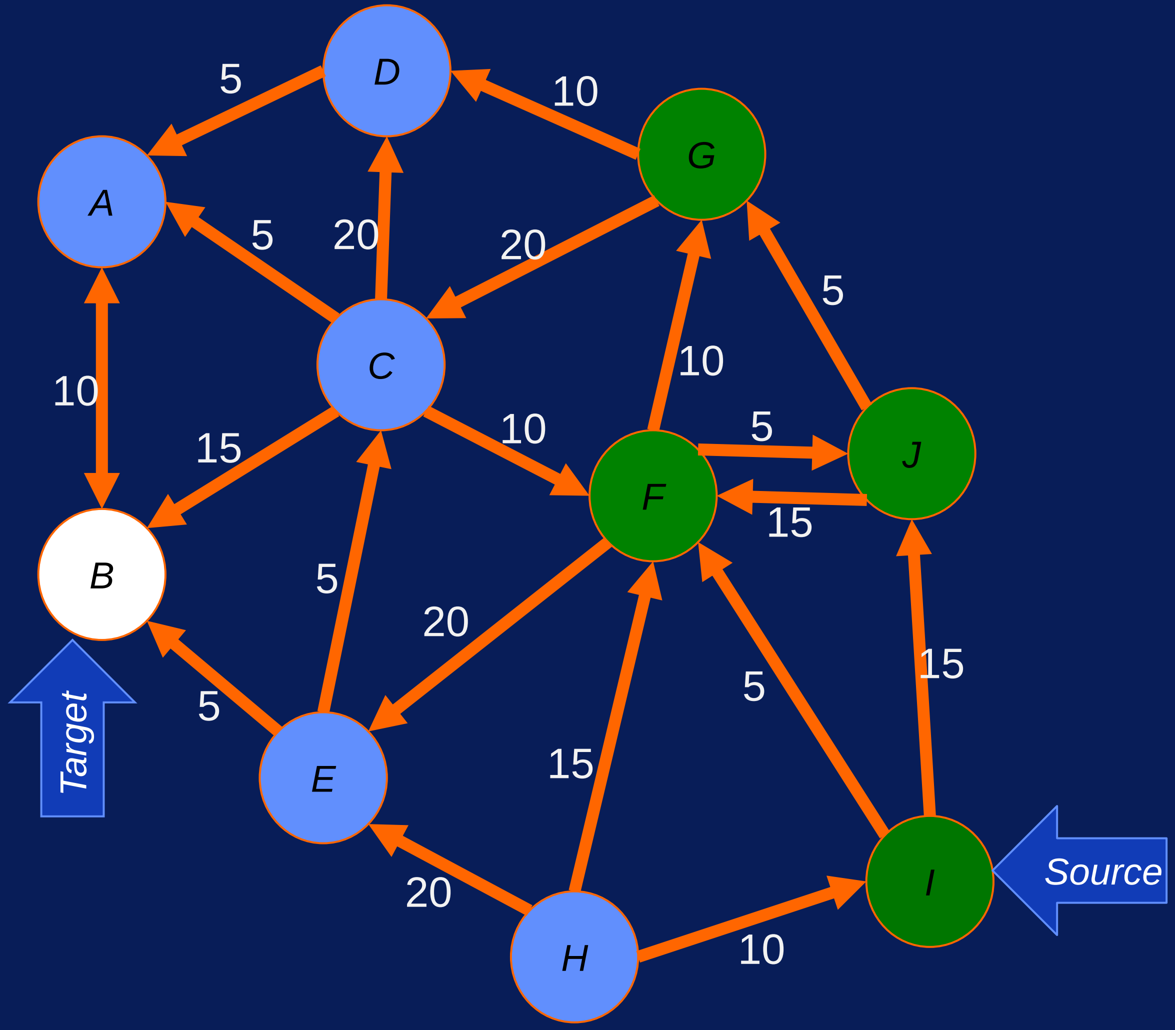
- **How can we change weights?**

Goal-Directed Dijkstra



- The target is B
  - Can we make a node closer to B less costly to get to?
- Use potential function λ
  - Modified weight
    - $w'(u,v) = w(u,v) - \lambda(u) + \lambda(v)$
    - Should not be negative

Goal-Directed Dijkstra

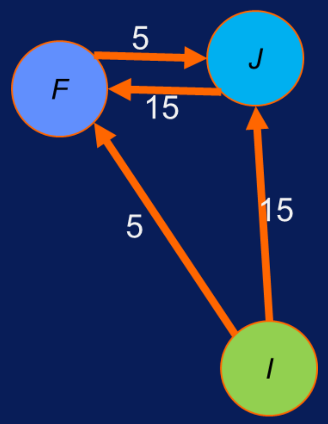


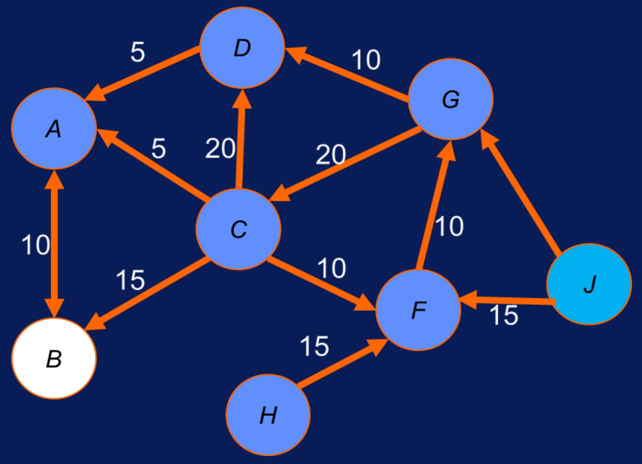
$$w'(u,v) = w(u,v) - \lambda(u) + \lambda(v)$$

- How to choose a potential function?
  - If the graph is geometric (e.g., road network)
    - Use landmark nodes
      - What if B is a landmark node?
        - Dist(B,G) = 80
        - Dist(B,F) = 20
        - Dist (B,E) = 15
      - Modified weights
        - wt(F→G)=10-20+80 = 70
        - wt(F→E)=20-20+15 = 15

Connectivity Analytics

Questions We are Trying to Address

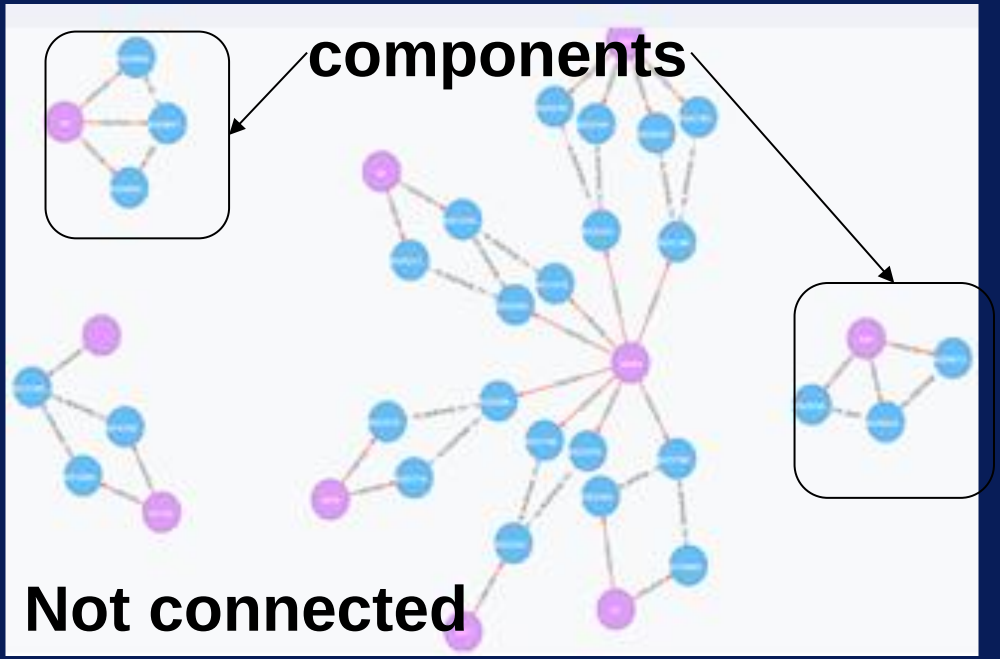


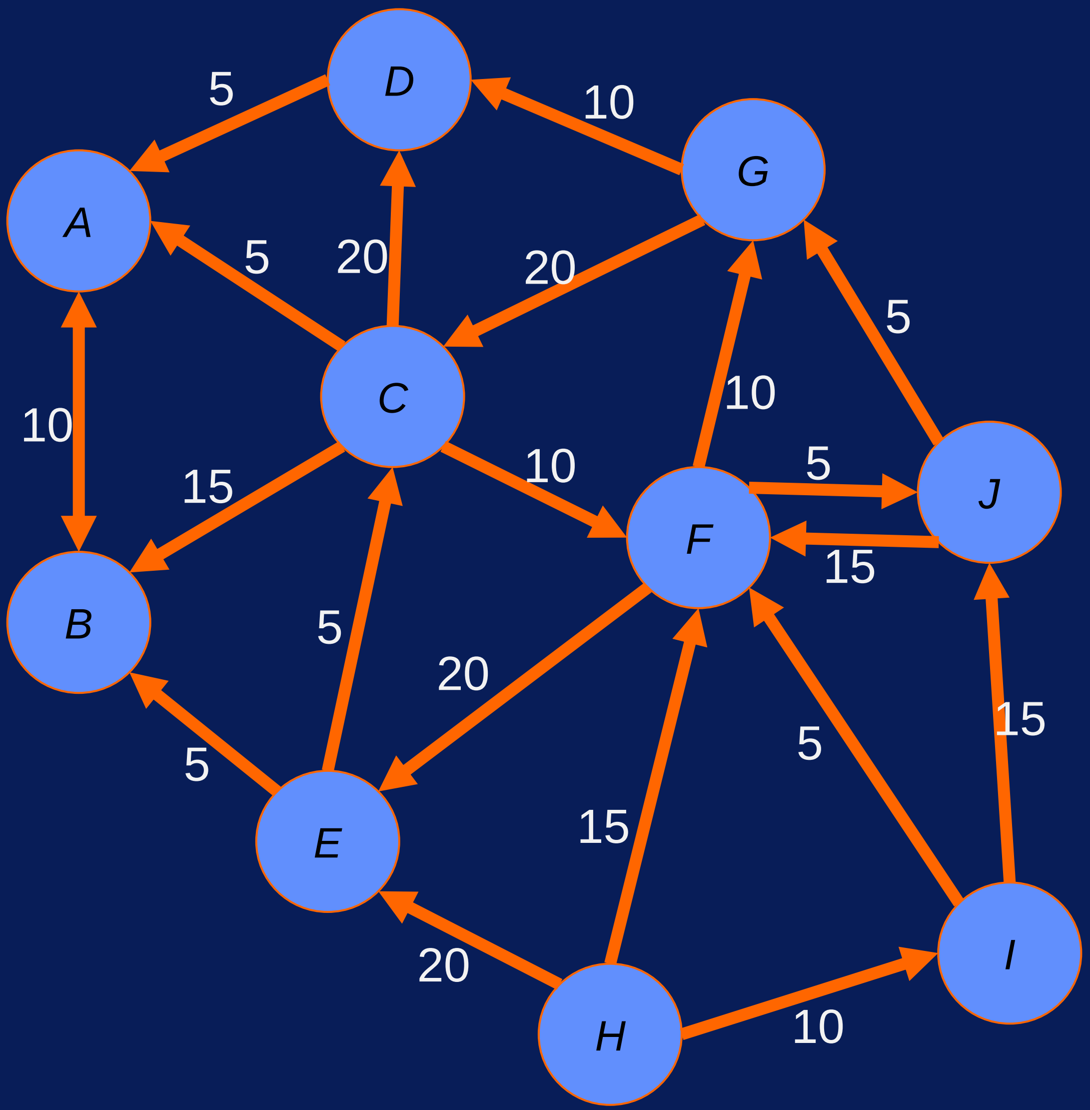


- **How “robust” is the graph?**
  - How easy is it to “break” the graph by removing a few nodes/edges?
- **How similar is the “structure” of graph G1 to graph G2?**
  - What are some computable “features” that can describe the structure of a graph?
  - How can two graphs be compared based on these features?

Connectedness

- A graph is connected if it contains a directed path from u to v or a directed path from v to u for every vertex-pair (u, v)



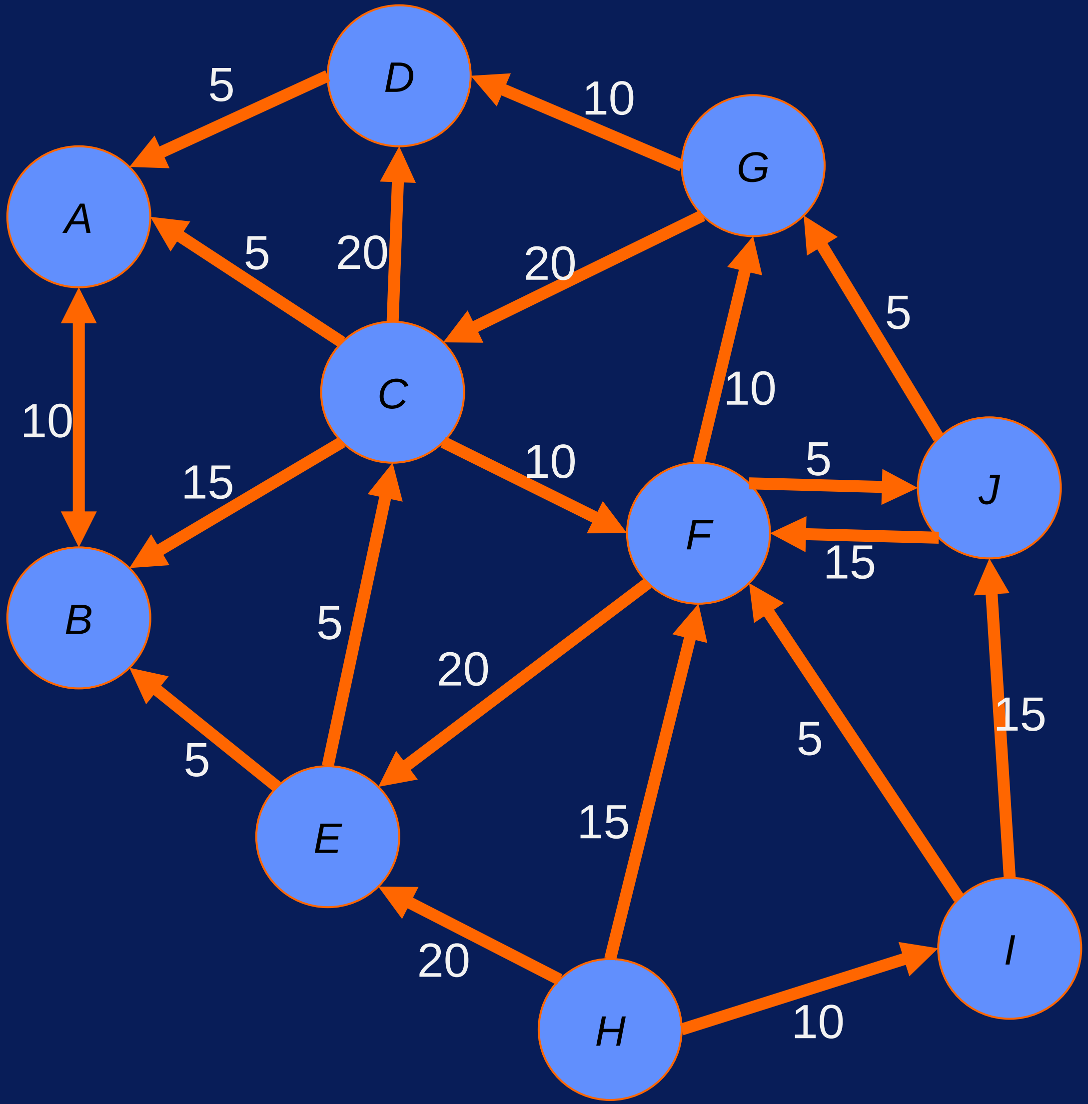


Strongly connected → a directed path from every u to every v

Weakly connected → connected after converting a directed graph to an undirected graph

Is this graph strongly or weakly connected?

In-Video Quiz



Is this graph strongly or weakly connected?

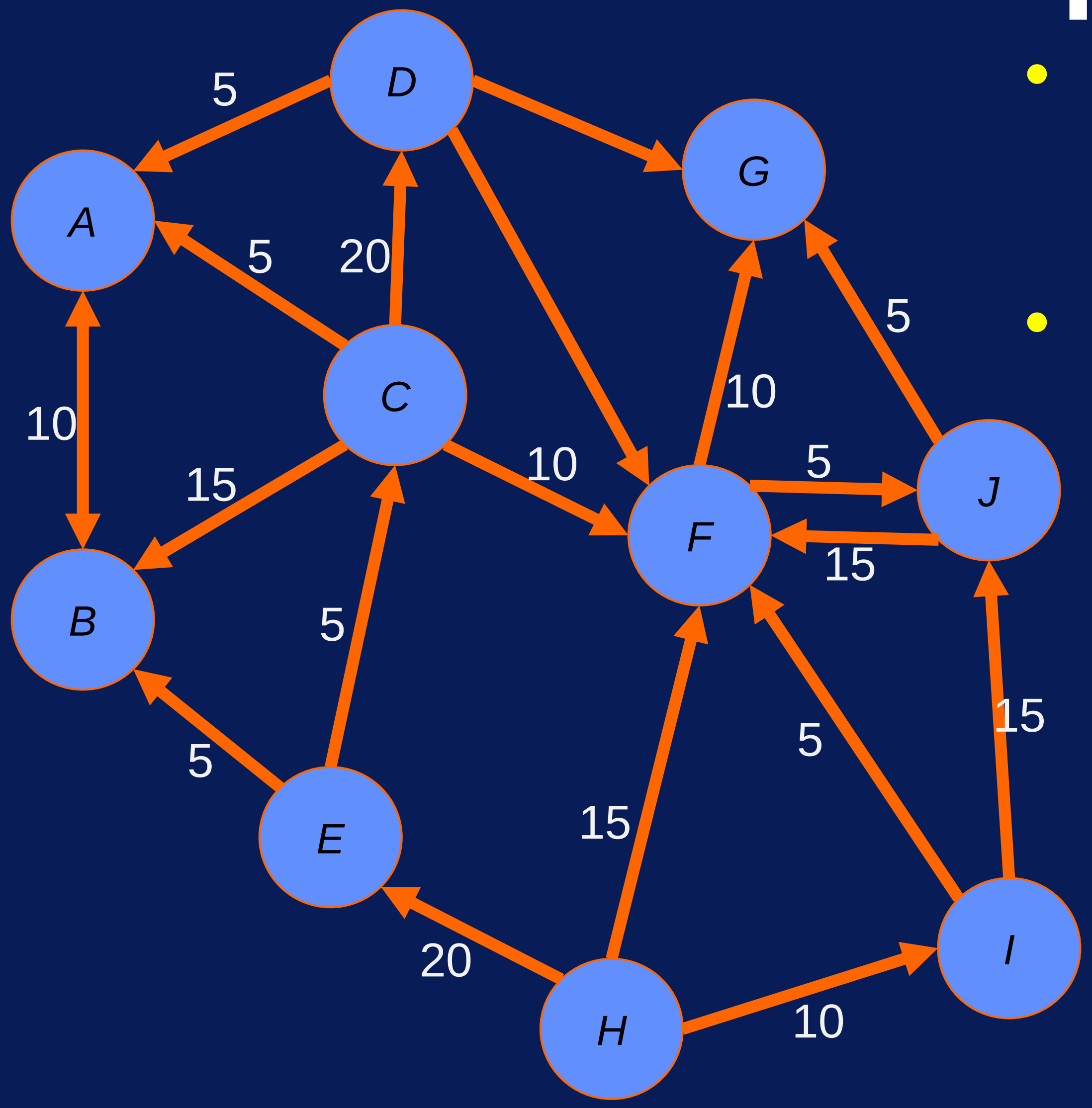
This graph is weakly connected

Some Computing Problems

- **Scalable solution for finding**
  - Connected components of a graph
  - Strongly connected fragment(s) of a graph

Will discuss in Module 4

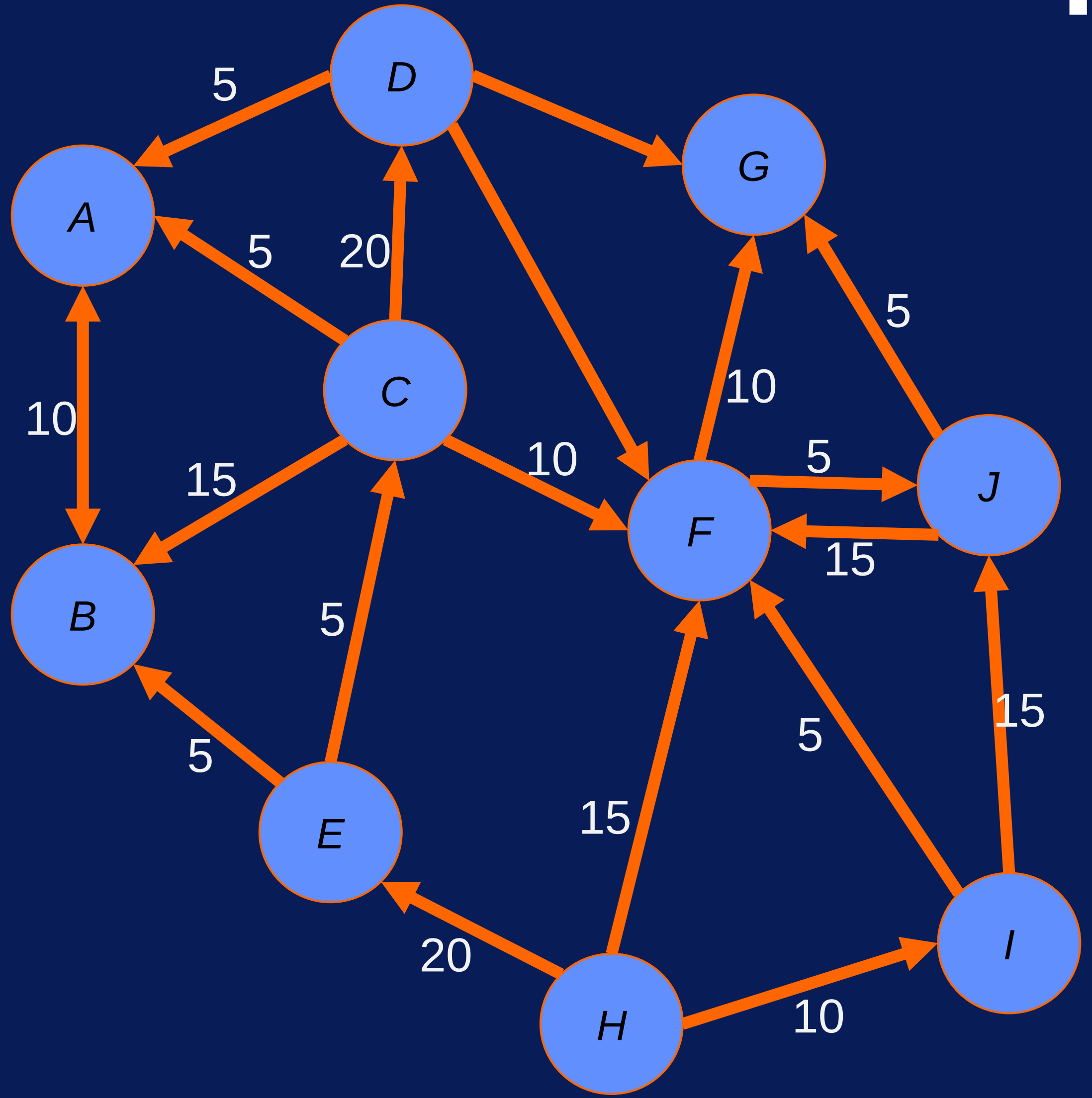
Disconnecting a Connected Graph



Node Based

- *Separating Set*: S ⊂ V such that V/S has more than one component
- *Connectivity of graph* G, $\kappa(G)$: minimum size of S

Disconnecting a Connected Graph



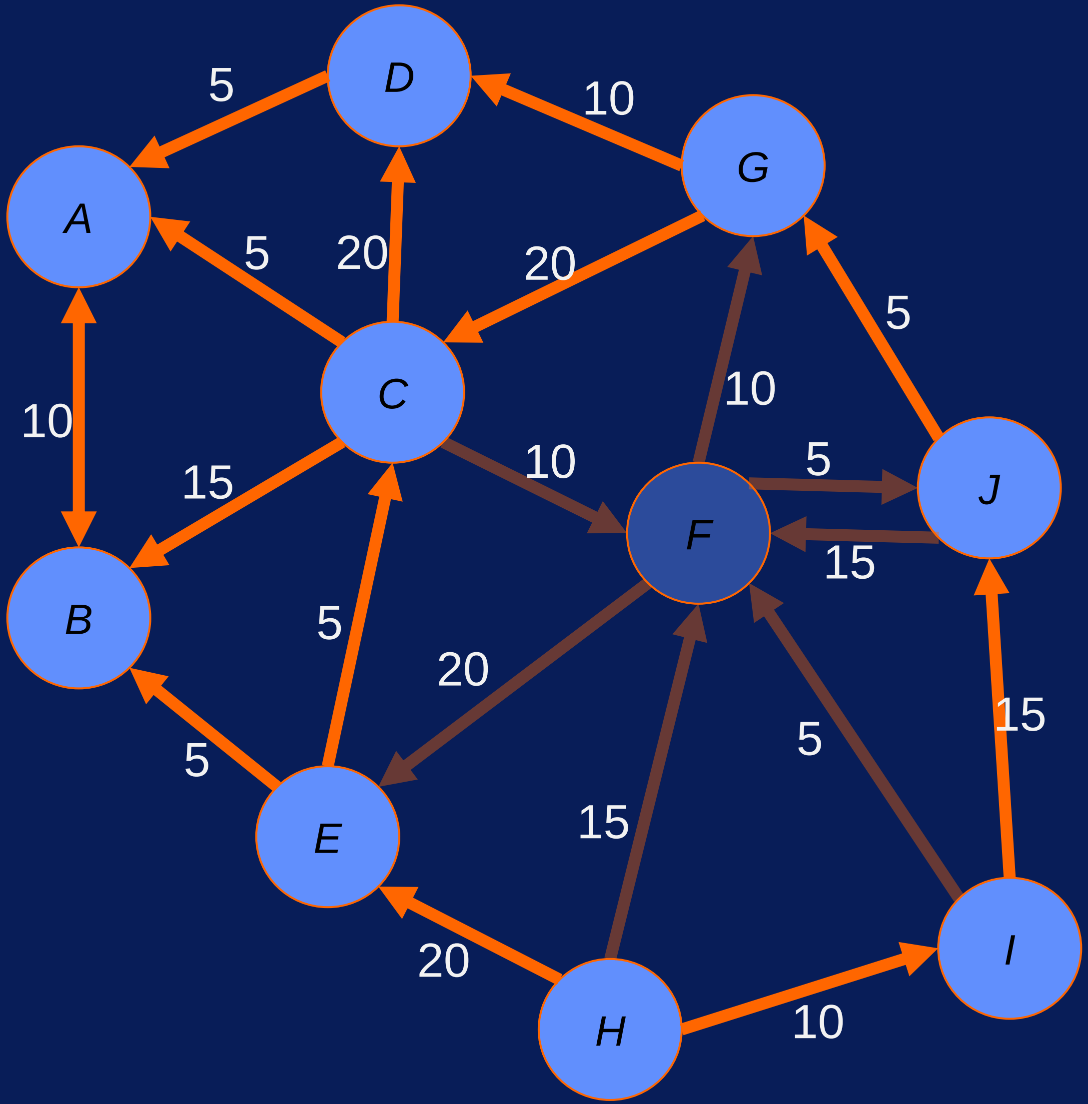
Edge Based

- ***Disconnecting set (of edges)***: H ⊂ E such that G – H has multiple components
- ***Edge-connectivity of graph G***: minimum size of H

Network Robustness

- **If node v is reachable from node v originally, it should remain reachable even if the network is "attacked"**
- **Attacked**
  - Node/edge removed

Is this network robust?



Robustness – assessed with respect to attacks
Remove F (why F?)

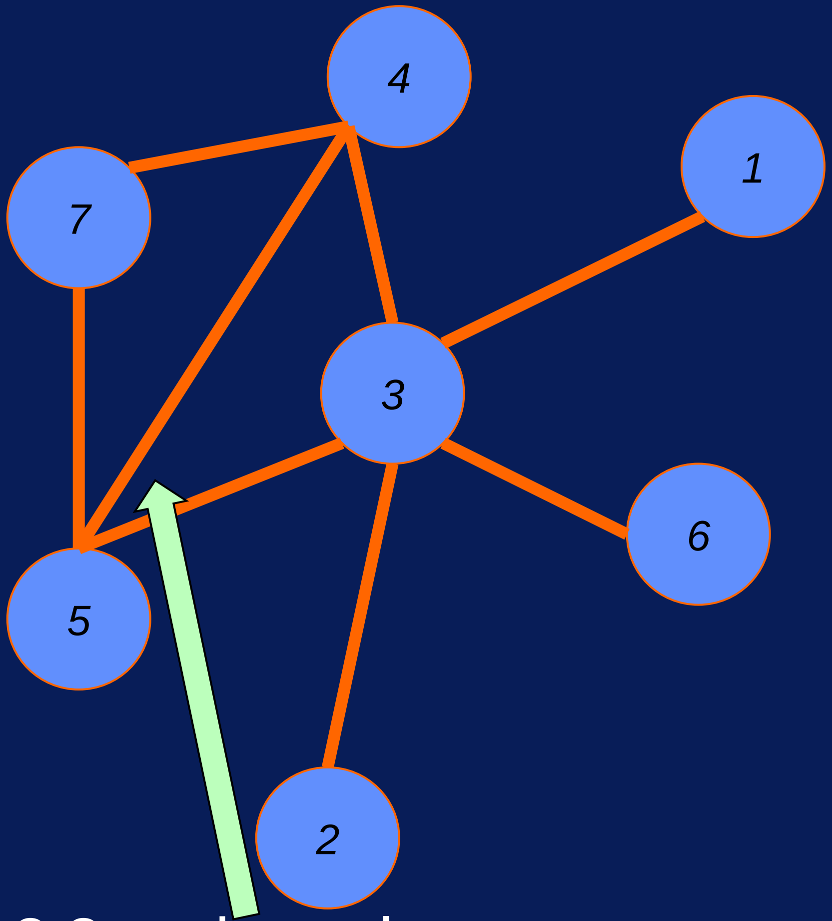
Paths C→G, C→J, C→E, J→E, I→E disrupted

Which node should the attacker target next?

C

pause

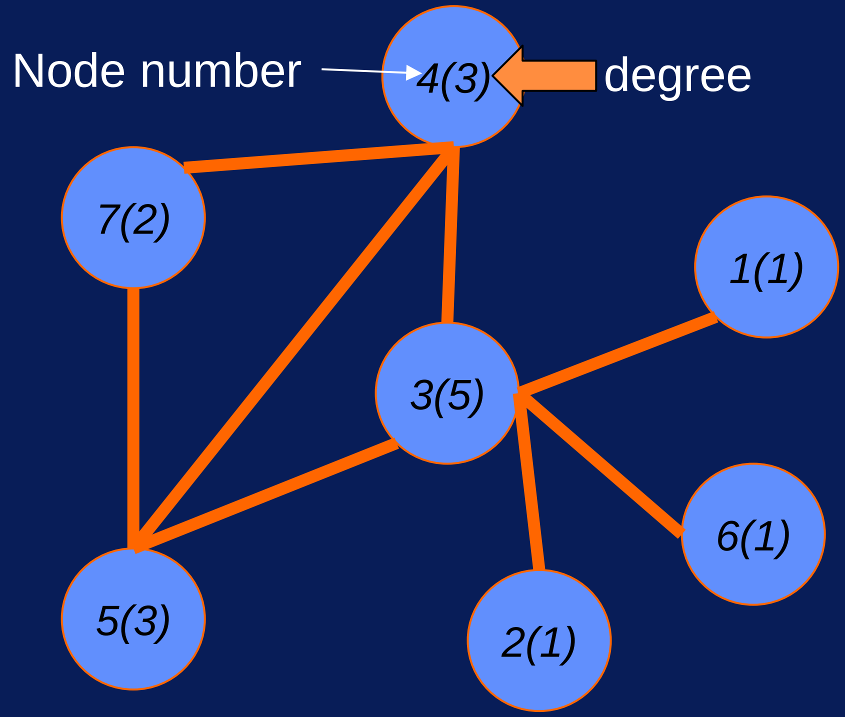
Measuring Network Robustness



2 3-node cycles
1 4-node cycles

- *Estimate the connectedness of a network*
  - *Algebraic Connectivity*
- Evaluate how much a structure is affected by an attack
  - *Weighted Spectral Distribution (WSD)*

Weighted Spectral Distribution



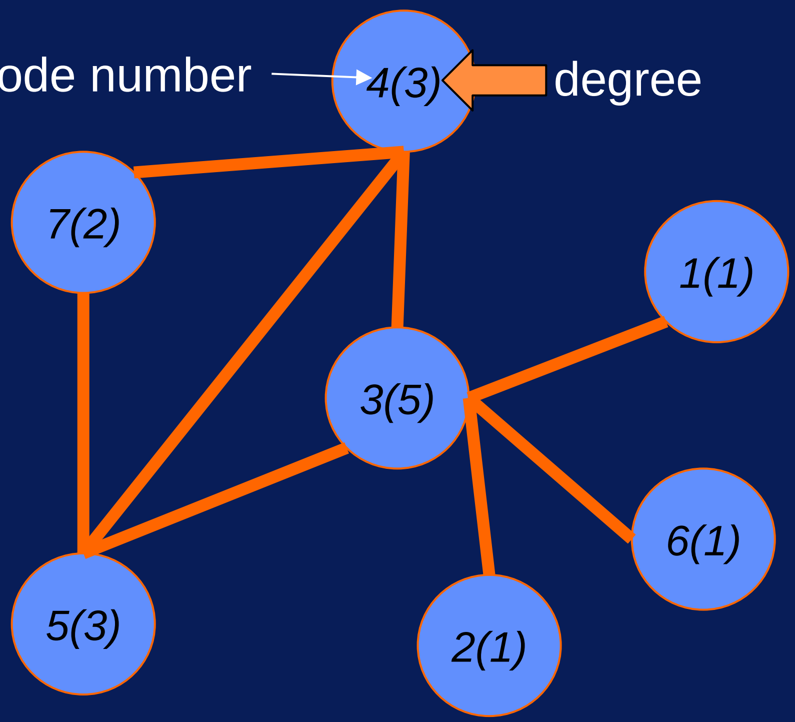
$$L(u,v)=\begin{cases}1 & if\ u=v\ ,d(v)\neq 0\\ -\frac{1}{\sqrt{d(u).d(v)}} & if\ e(u,v)\ exists\\ 0 & otherwise\end{cases}$$

Normalized Laplacian Matrix L – symmetric

	1	2	3	4	5	6	7
1							
2							
3							
4							
5							
6							
7							

Weighted Spectral Distribution

Node number

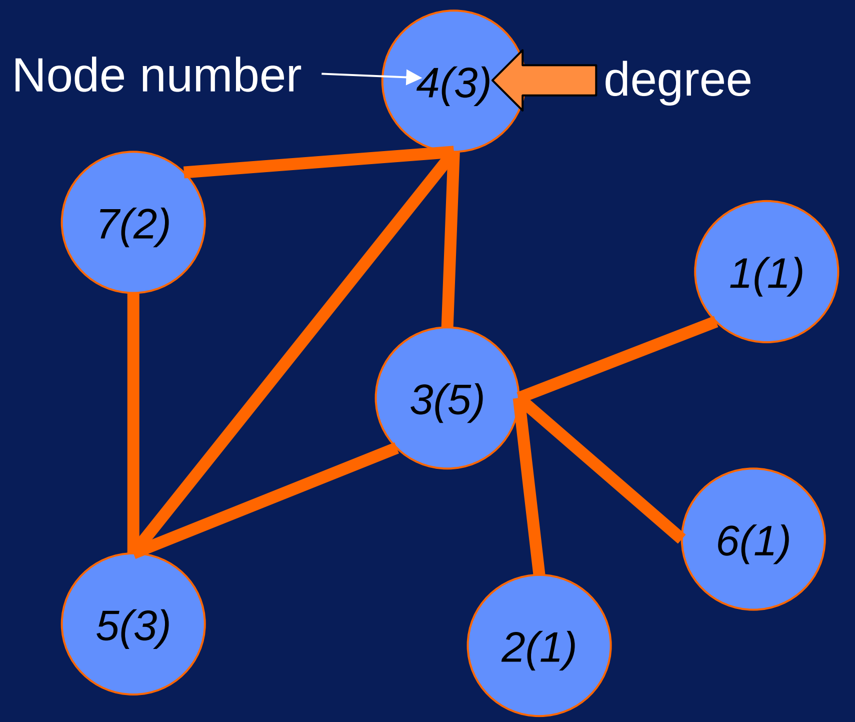


$$L(u,v) = \begin{cases} 1 & if\ u = v\ , d(v) \neq 0 \\ -\frac{1}{\sqrt{d(u).d(v)}} & if\ e(u,v)\ exists \\ 0 & otherwise \end{cases}$$

Normalized Laplacian Matrix L – symmetric

	1	2	3	4	5	6	7
1	1						
2		1					
3			1				
4				1			
5					1		
6						1	
7							1

Weighted Spectral Distribution

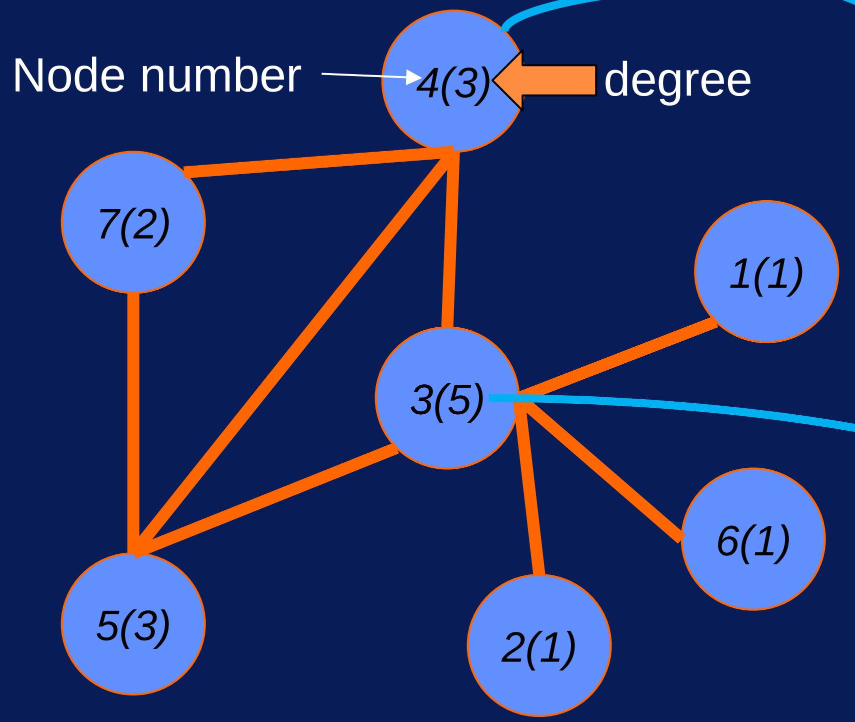


$$L(u,v) = \begin{cases} 1 & if\ u = v\ , d(v) \neq 0 \\ -\frac{1}{\sqrt{d(u).d(v)}} & if\ e(u,v)\ exists \\ 0 & otherwise \end{cases}$$

Normalized Laplacian Matrix L – symmetric

	1	2	3	4	5	6	7
1	1	0		0	0	0	0
2	0	1		0	0	0	0
3			1				0
4	0	0		1		0	
5	0	0			1	0	
6	0	0		0	0	1	0
7	0	0	0			0	1

Weighted Spectral Distribution



$$L(u,v) = \begin{cases} 1 & if\ u = v\,, d(v) \neq 0 \\ -\frac{1}{\sqrt{d(u).d(v)}} & if\ e(u,v)\ exists \\ 0 & otherwise \end{cases}$$

Normalized Laplacian Matrix L – symmetric

	1	2	3	4	5	6	7
1	1	0	$\frac{-1}{\sqrt{1\times 5}}$	0	0	0	0
2	0	1	$\frac{-1}{\sqrt{1\times 5}}$	0	0	0	0
3	$\frac{-1}{\sqrt{1\times 5}}$	$\frac{-1}{\sqrt{1\times 5}}$	1	$\frac{-1}{\sqrt{3\times 5}}$	$\frac{-1}{\sqrt{3\times 5}}$	$\frac{-1}{\sqrt{1\times 5}}$	0
4	0	0	$\boldsymbol{\frac{-1}{\sqrt{3\times 5}}}$	1	$\frac{-1}{\sqrt{3\times 3}}$	0	$\frac{-1}{\sqrt{3\times 2}}$
5	0	0	$\frac{-1}{\sqrt{3\times 5}}$	$\frac{-1}{\sqrt{3\times 3}}$	1	0	$\frac{-1}{\sqrt{3\times 2}}$
6	0	0	$\frac{-1}{\sqrt{1\times 5}}$	0	0	1	0
7	0	0	0	$\frac{-1}{\sqrt{3\times 2}}$	$\frac{-1}{\sqrt{3\times 2}}$	0	1

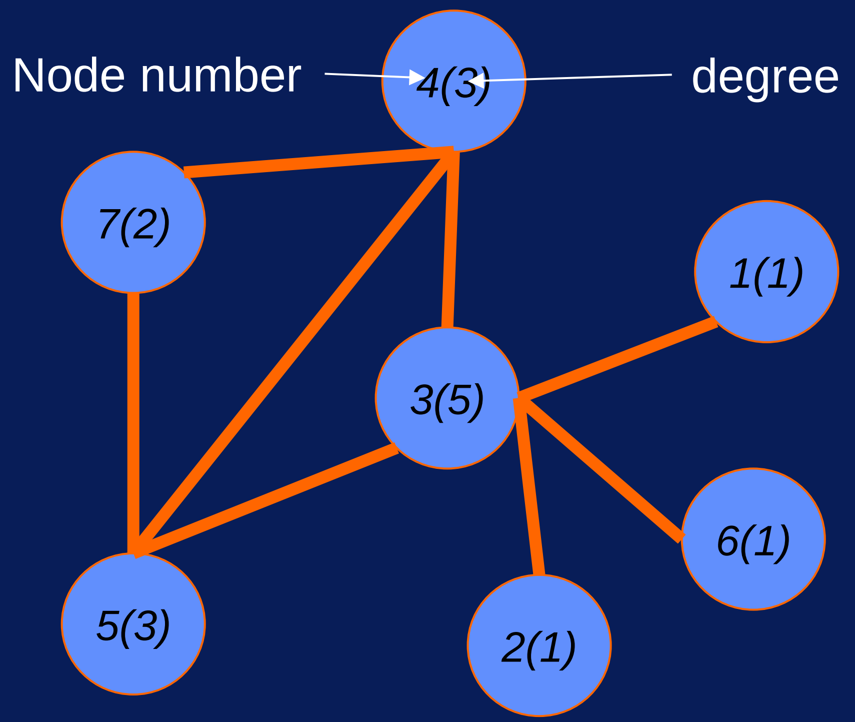
In the next slide we will refer to a matrix operation called eigenvalue computation.

There are many software libraries that perform these matrix computations in parallel. So it is fine for you to think of it as a black box.

If you want to know more details, here are two excellent YouTube videos explaining the notion behind eigenvalues and how to compute them.

- https://www.youtube.com/watch?v=0UbkMlTu1vo
- https://www.youtube.com/watch?v=BbvCa87U15M

Weighted Spectral Distribution



$$L(u,v) = \begin{cases} 1 & if\ u = v\ , d(v) \neq 0 \\ -\frac{1}{\sqrt{d(u).d(v)}} & if\ e(u,v)\ exists \\ 0 & otherwise \end{cases}$$

Find λ, the eigenvalues of L by solving $Lv = \lambda v$

Eigen-Vector	λ	$1-\lambda$	$(1-\lambda)^3$
1	1.8615	-.8615	-0.6394
2	1.3942	-.3942	-.0612
3	1.3333	-.3333	-.0370
4	1.0000	0	0
5	1.0000	0	0
6	0.4110	.5890	.2043
7	0	1	1

Interpreting Eigenvalues

Eigen-Vector	λ	$1-\lambda$	$(1-\lambda)^3$
1	1.8615	-.8615	-0.6394
2	1.3942	-.3942	-.0612
3	1.3333	-.3333	-.0370
4	1.0000	0	0
5	1.0000	0	0
6	**0.4110**	.5890	.2043
7	0	1	1
WSD: $\sum_{i=1}^{7}(1-\lambda)^3$			**0.4667**

Algebraic Connectivity

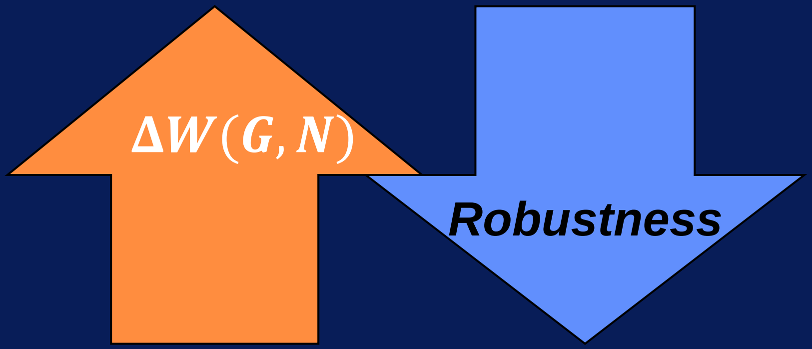
# of 0 eigenvalues = # of components

Why $(1-\lambda)^3$? There are 2 3-cycles in G

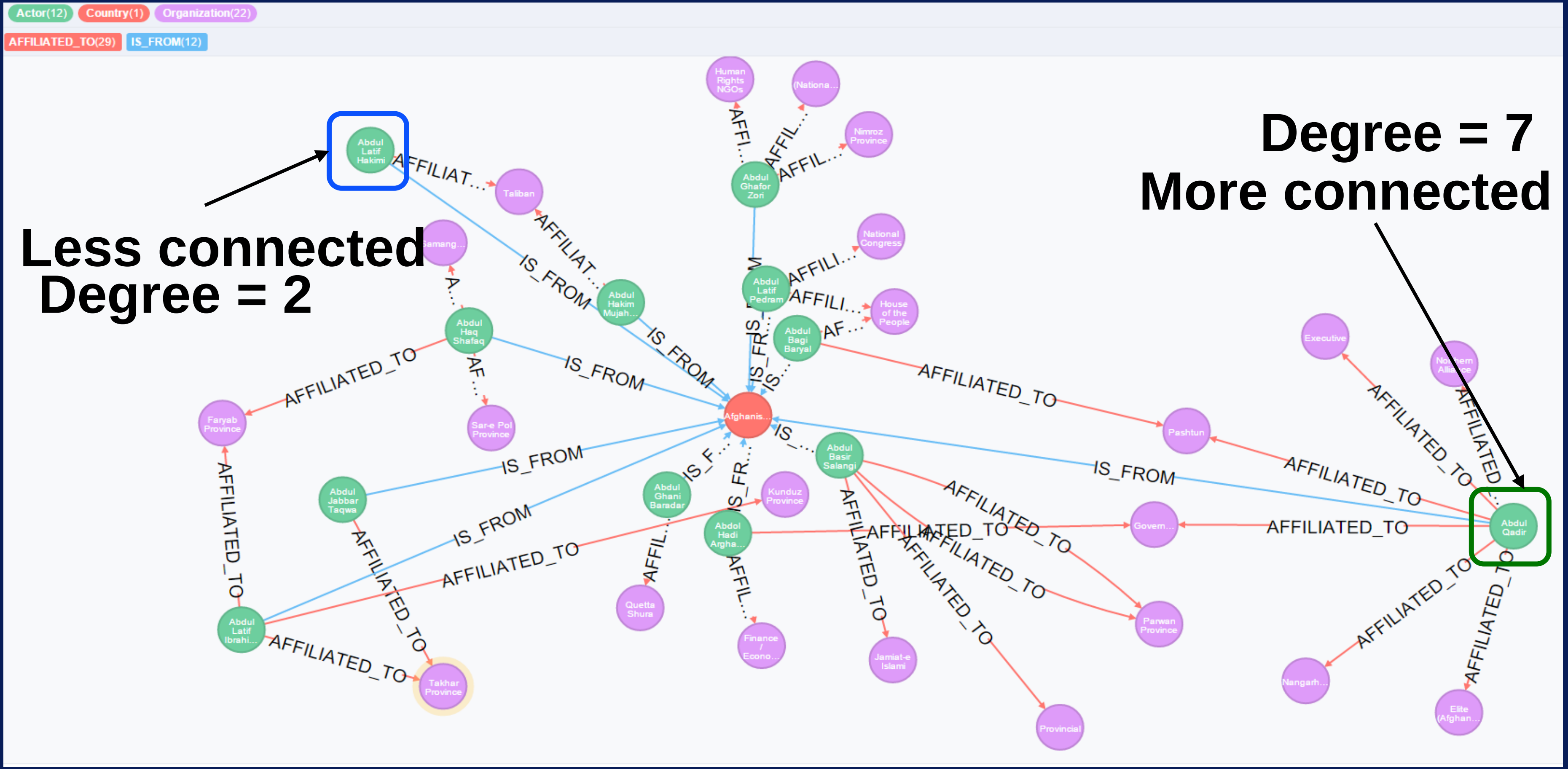
$$W(G,N) = \sum_{i=1}^{7}(1-\lambda)^N$$

$\Delta W(G,N)$ is the difference in W before and after an attack.

Use different values of N



pause

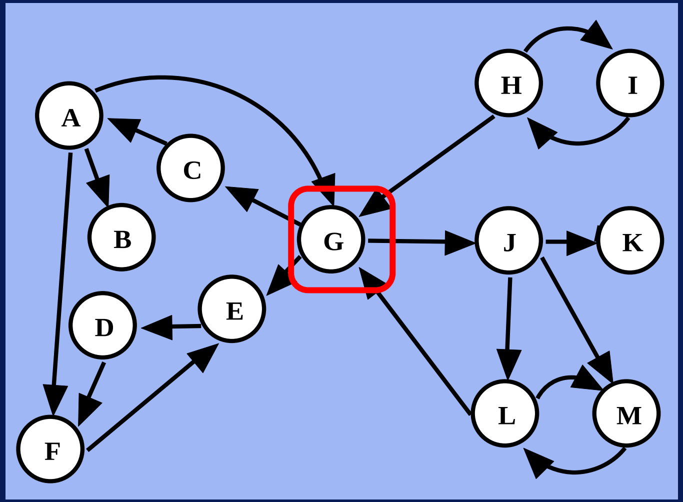


Degree of a node: number of edges connected to it

Some Nodes are More Connected

Indegree and Outdegree

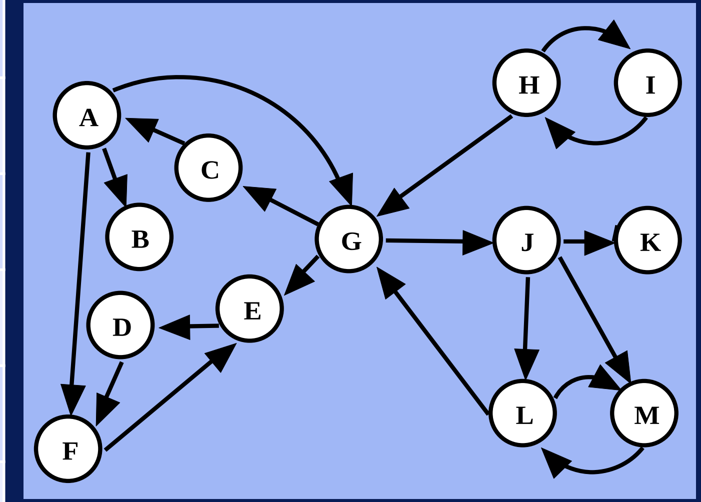
- Degrees
  - ***Indegree*** of a node: number of incident edges
  - ***Outdegree*** of a node: number of edges emanating from it
  - ***Degree*** of a node: indegree + outdegree



Indegree of G = 3
Outdegree of G = 3
Degree of G = 6

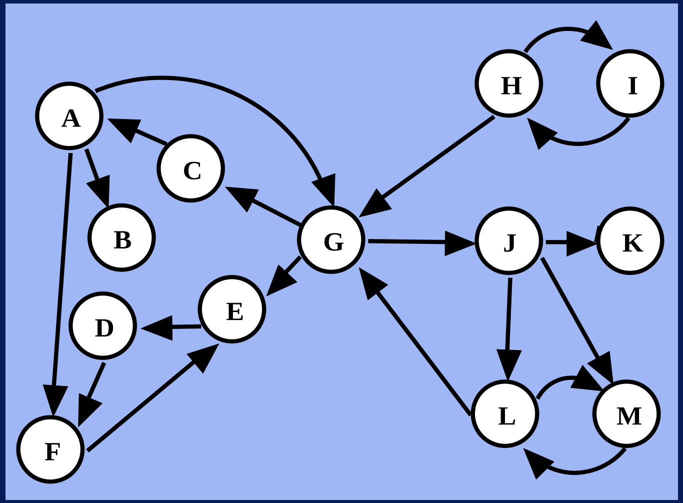
Count the Degree of each Node

Node	Degree
A	4
B	1
C	2
D	2
E	3
F	3
G	6
H	3
I	2
J	4
K	1
L	4
M	2



Can you create the histogram for this data?

Degree Histogram



Degree	Count
0	0
1	2
2	4
3	3
4	3
5	0
6	1

- Graph similarity
  - Vector distance functions
    - Euclidian distance
  - Many other distance functions
  - Statistical difference measures of two distributions

Comparing Graph Structures

- Graph similarity
  - Vector distance functions
    - Euclidian distance

$$D_{L2} = \sqrt{\sum_i \left(h_1(i) - h_2(i)\right)^2}$$

  - Many other distance functions
    - Statistical difference measures of two distributions

G1

Degree	Count
0	0
1	2
2	4
3	3
4	3
5	0
6	1

G2

Degree	Count
0	0
1	3
2	4
3	3
4	2
5	1
6	1

Euclidian distance = $\sqrt{3}$ Quite similar

Here is a quick list of some histogram distance functions.

- http://stats.stackexchange.com/questions/7400/how-to-assess-the-similarity-of-two-histograms

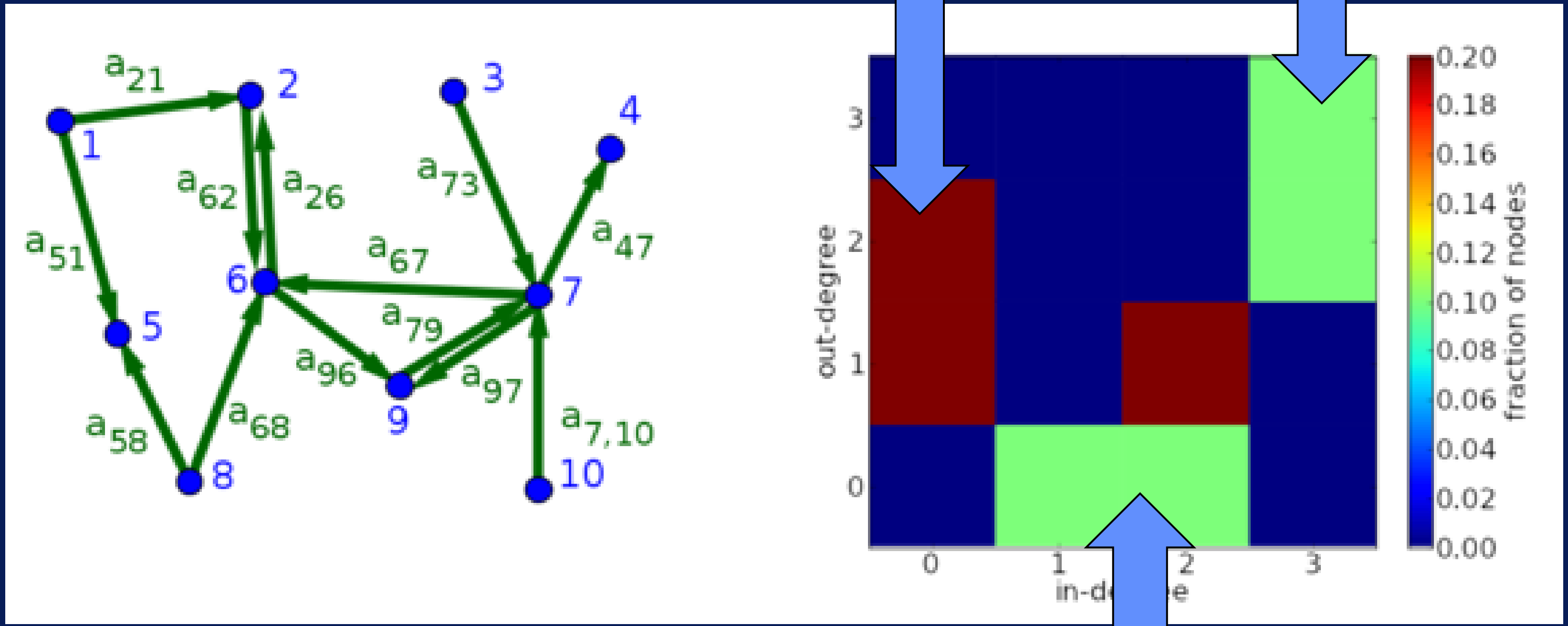
More detailed information about these functions is available online.

Joint Degree Histograms

One can also compute the in-degree and out-degree histograms of a graph

talkers

communicators



listeners

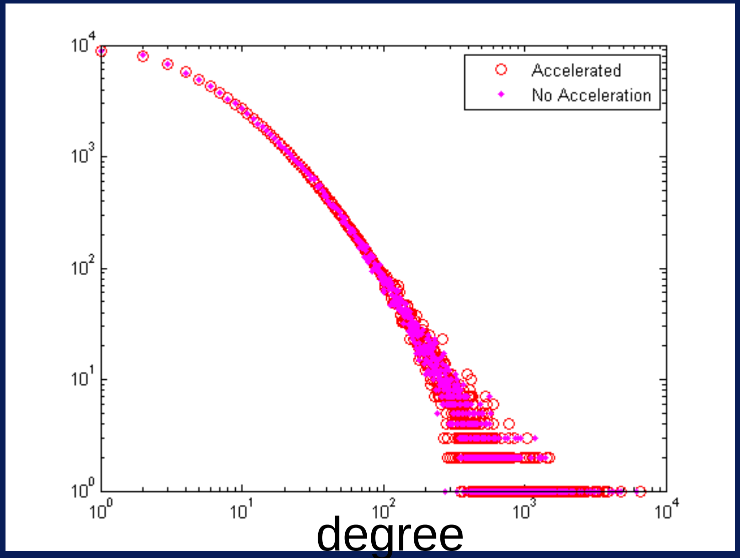
We have an optional video that you can go through.

This video discusses the statistical distribution of node degrees and a well-known model family called the Power Law

Optional 3

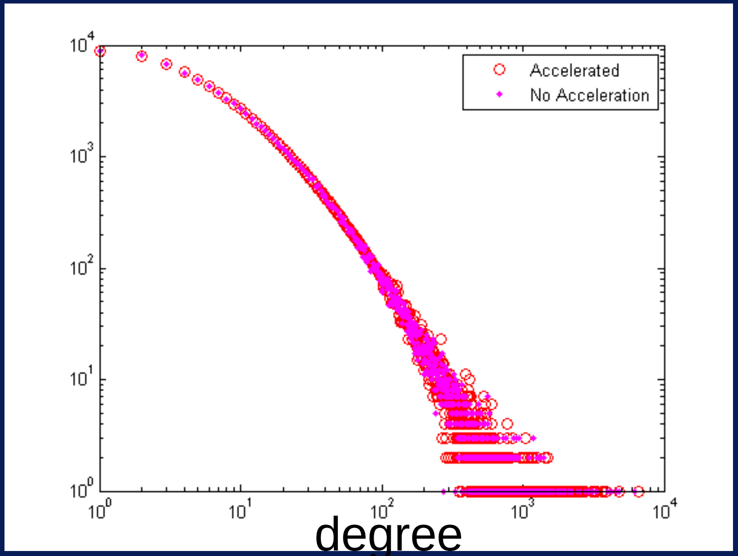
Degree Distribution

- The probability $P(k)$ that a vertex v will have k neighbors
- Some interesting cases
  - Scale Free (Power Law) Graph
    - $P(k) \sim k^{-\alpha}$
  - Log-Normal Graph
    - $P(k) = \frac{1}{\sqrt{2\pi}\sigma k} e^{-(\ln k - \mu)2/2\sigma^2}$



Power Law Graphs

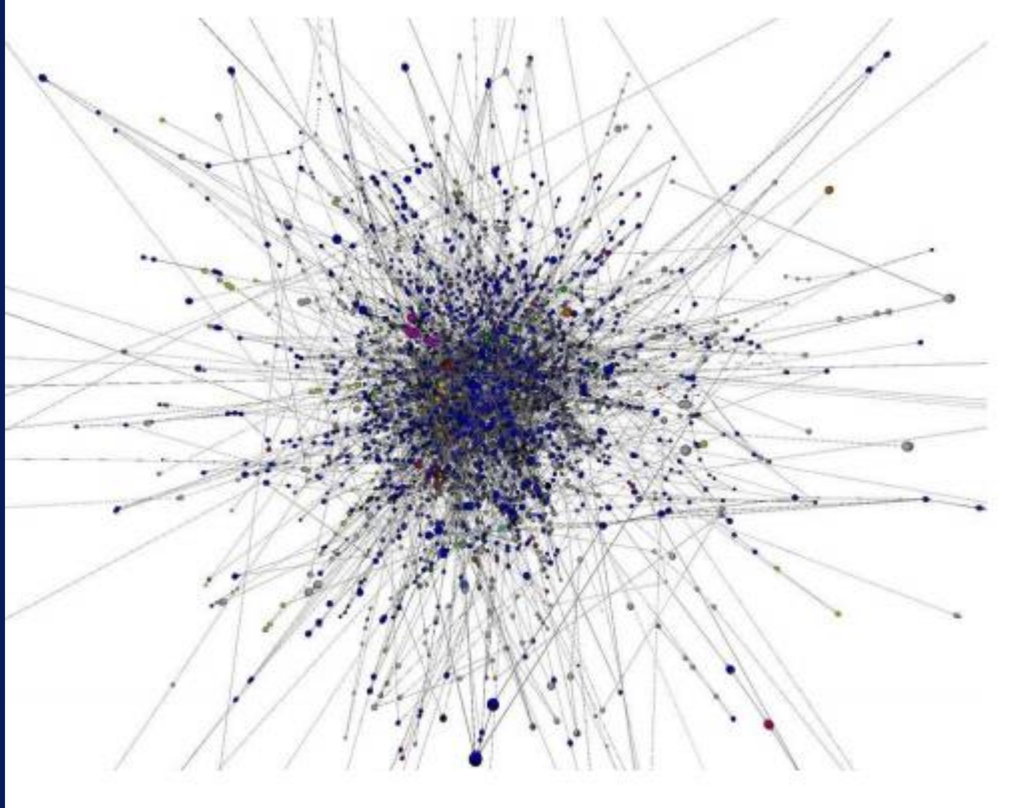
- Very common in nature
  - Internet
  - Social Networks
  - Protein-protein interactions
  - Airline Networks
  - Financial Networks
- Has a few (often 1) hub nodes
  - A large number of nodes connected directly or indirectly to hubs
  - High-density sub-region around hub



Interesting computing implications (module 4)

Features of Power Law Graphs

- Has a few (often 1) high-degree (hub) nodes
  - A large number of nodes connected directly or indirectly to hubs
  - High-density, low-degree sub-regions around hub



Power-Law Graphs are Scale-Free

- The scale-free property strongly correlates with the network's robustness to failure. The major hubs are closely followed by smaller ones, followed by other nodes with an even smaller degree and so on. This hierarchy allows for a fault tolerant behavior. If failures occur at random and the vast majority of nodes are those with small degree, the likelihood that a hub would be affected is almost negligible. Even if a hub-failure occurs, the network will generally not lose its connectedness, due to the remaining hubs.

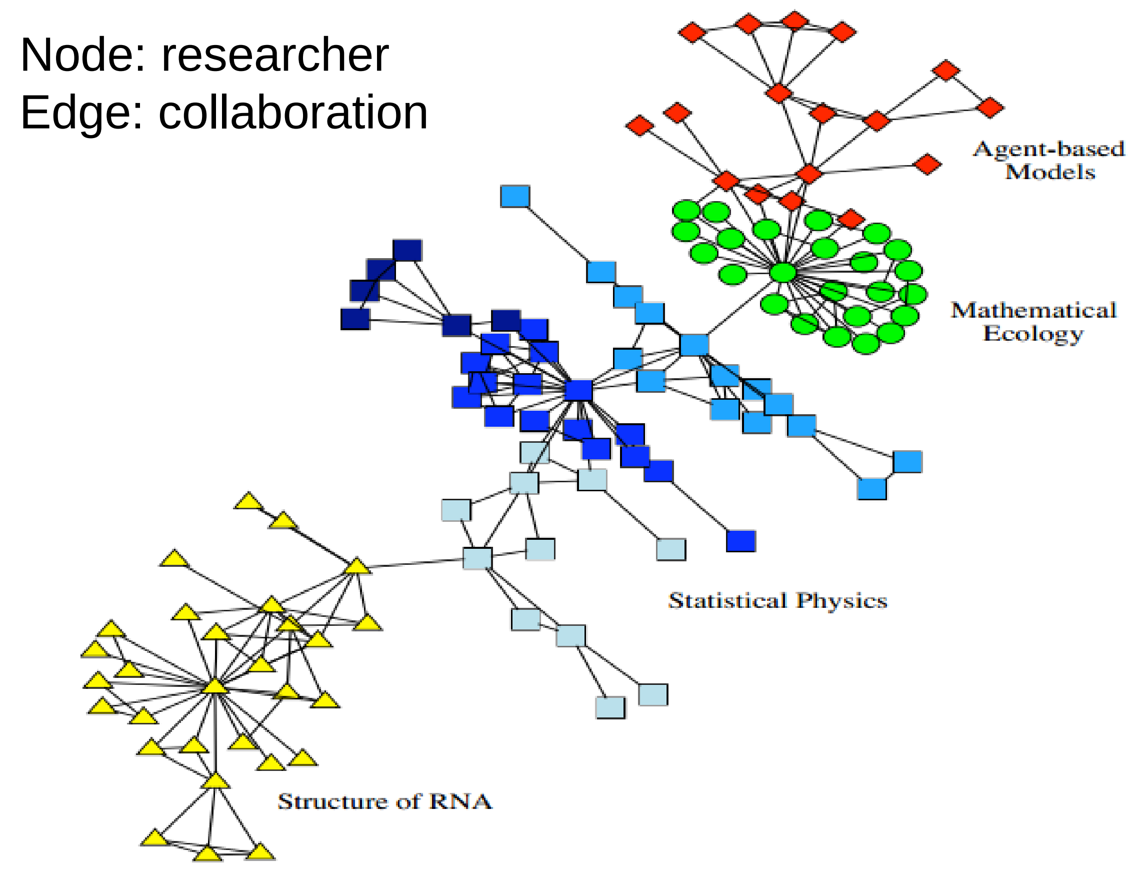
- On the other hand, if we choose a few major hubs and take them out of the network, the network is turned into a set of rather isolated graphs. Thus, hubs are both a strength and a weakness of scale-free networks.

Community Analytics

What is a Community?

- Entities often interact within groups
- Interactions form ***clusters***
- ***Community***
  - ***a dense subgraph (cluster) within a graph whose nodes are more connected within the cluster than to nodes outside the cluster***

Node: researcher
Edge: collaboration

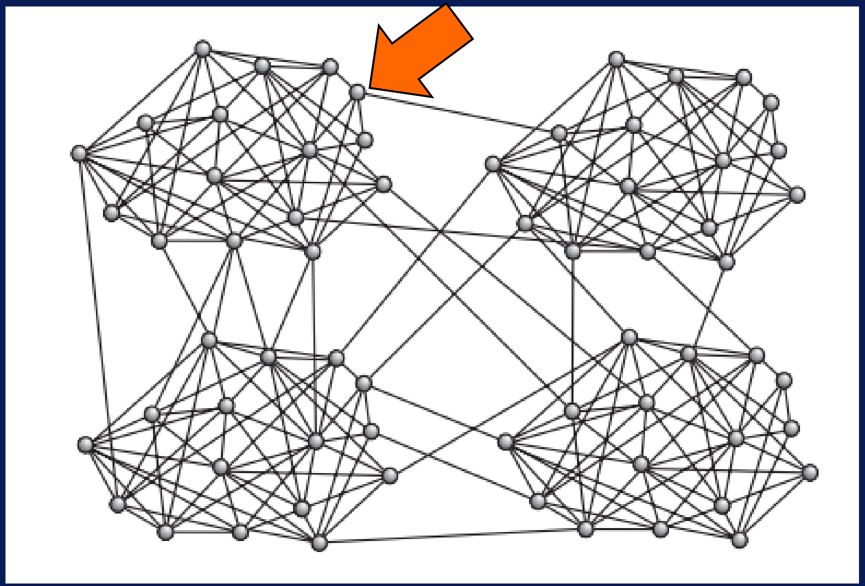


Which researchers collaborate?

Some Analytics Questions

- "Static" Analyses
  - What are the communities at time T?
  - Who belong to a community?
  - How closely knit is this community?
- Temporal/Evolution Analyses
  - How did this community form?
  - Which communities are stable?
  - Find strong *transient* communities – why did they form or dissolve?
- Predictive Analyses
  - Is this community likely to grow?
  - Will these nodes continue as a community in future?
  - Are dominant roles emerging in this community?

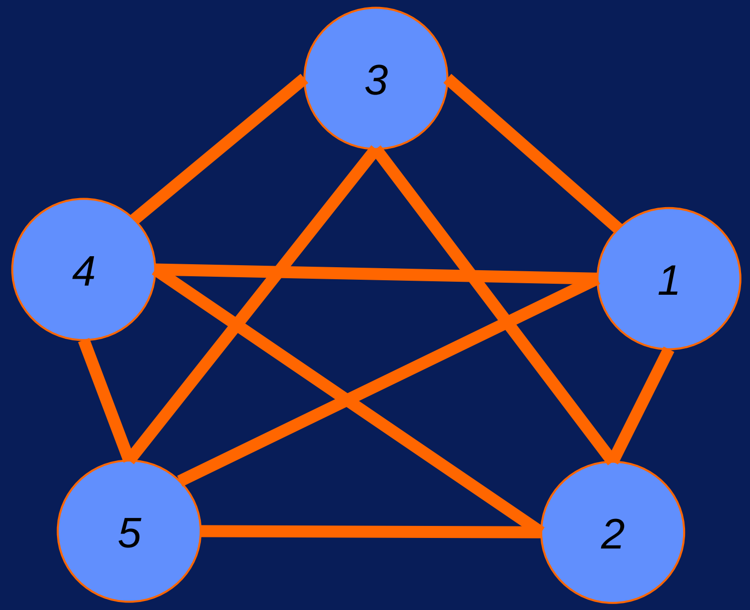
Detecting a Community



- C – **connected subgraph of graph** G
- **We can compute**
  - The internal and external degree of a vertex
    - Internal – within C
    - External – outside C
  - The internal and external degree of the cluster C
    - Sum of the internal/external degree of the vertices of C
    - Intra-cluster density -- $\delta_{int} = \frac{\#\ of\ internal\ edges\ in\ C}{n_c(nC-1)/2}$
    - Inter-cluster density -- $\delta_{ext} = \frac{\#\ of\ inter\ cluster\ edges\ of\ C}{n_c(n-nC)}$
  - For C to be a community
    - $\boldsymbol{\delta_{int}}$ ***should be high and*** $\boldsymbol{\delta_{ext}}$ ***should be low***

Local Properties

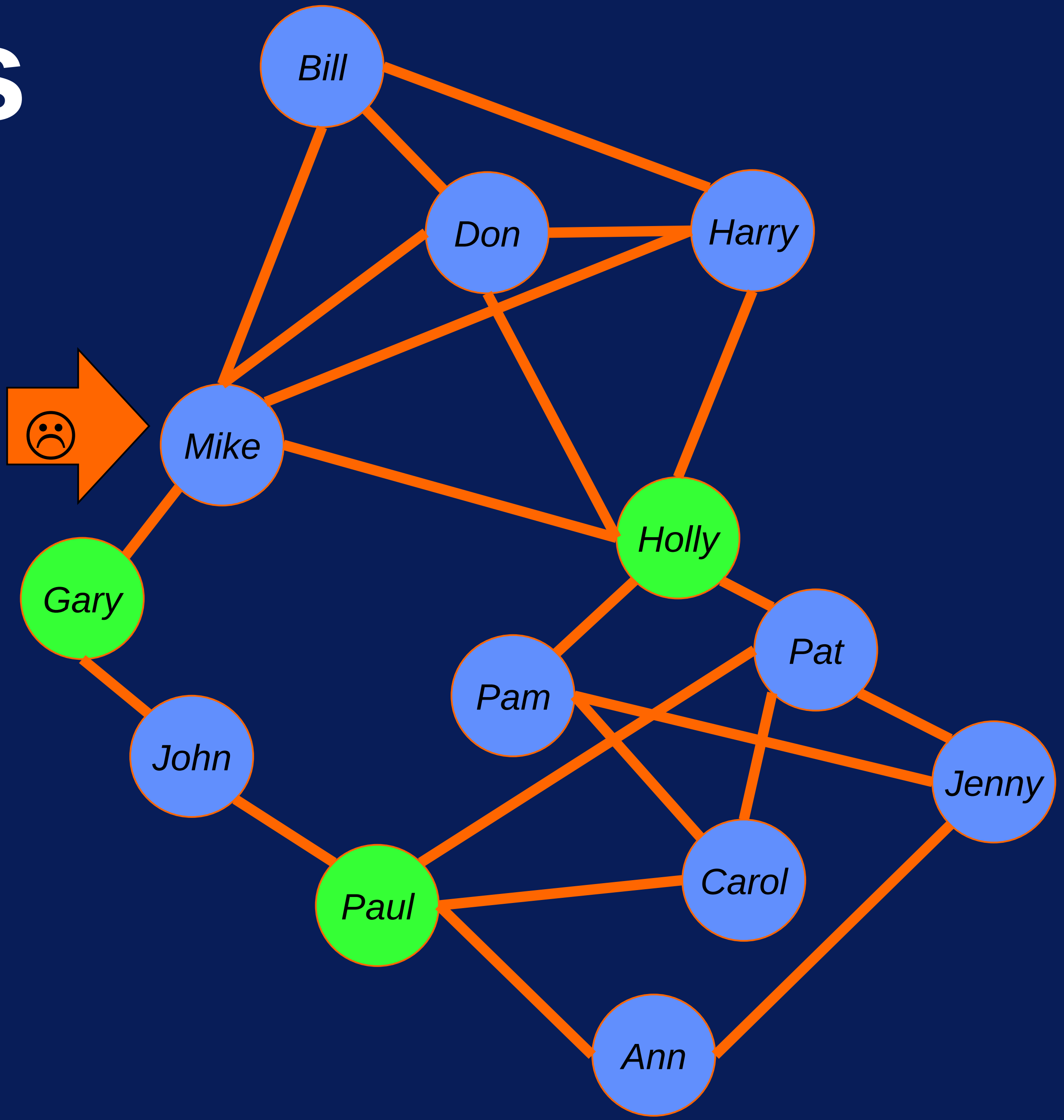
Properties of a subgraph and its neighborhood



- Clique
  - The perfect community
    - every two distinct vertices in the clique are adjacent
  - Finding the largest clique within a graph
    - Computationally hard problem
    - Simpler to find cliques of size k

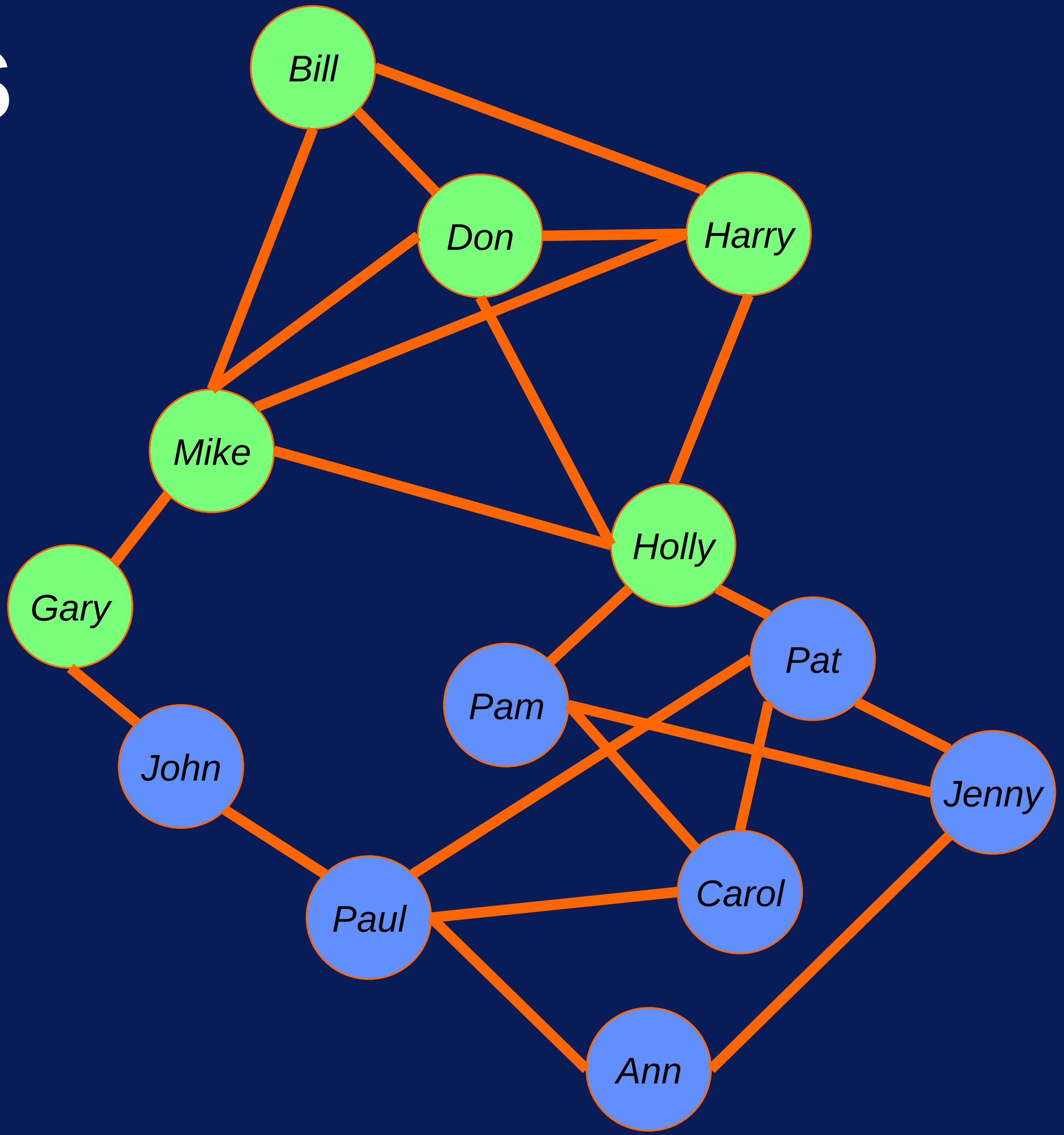
Near Cliques

- n-**clique**
  - Maximal subgraph such that the distance of each pair of its vertices is not larger than n
    - $n = 1$ for a clique

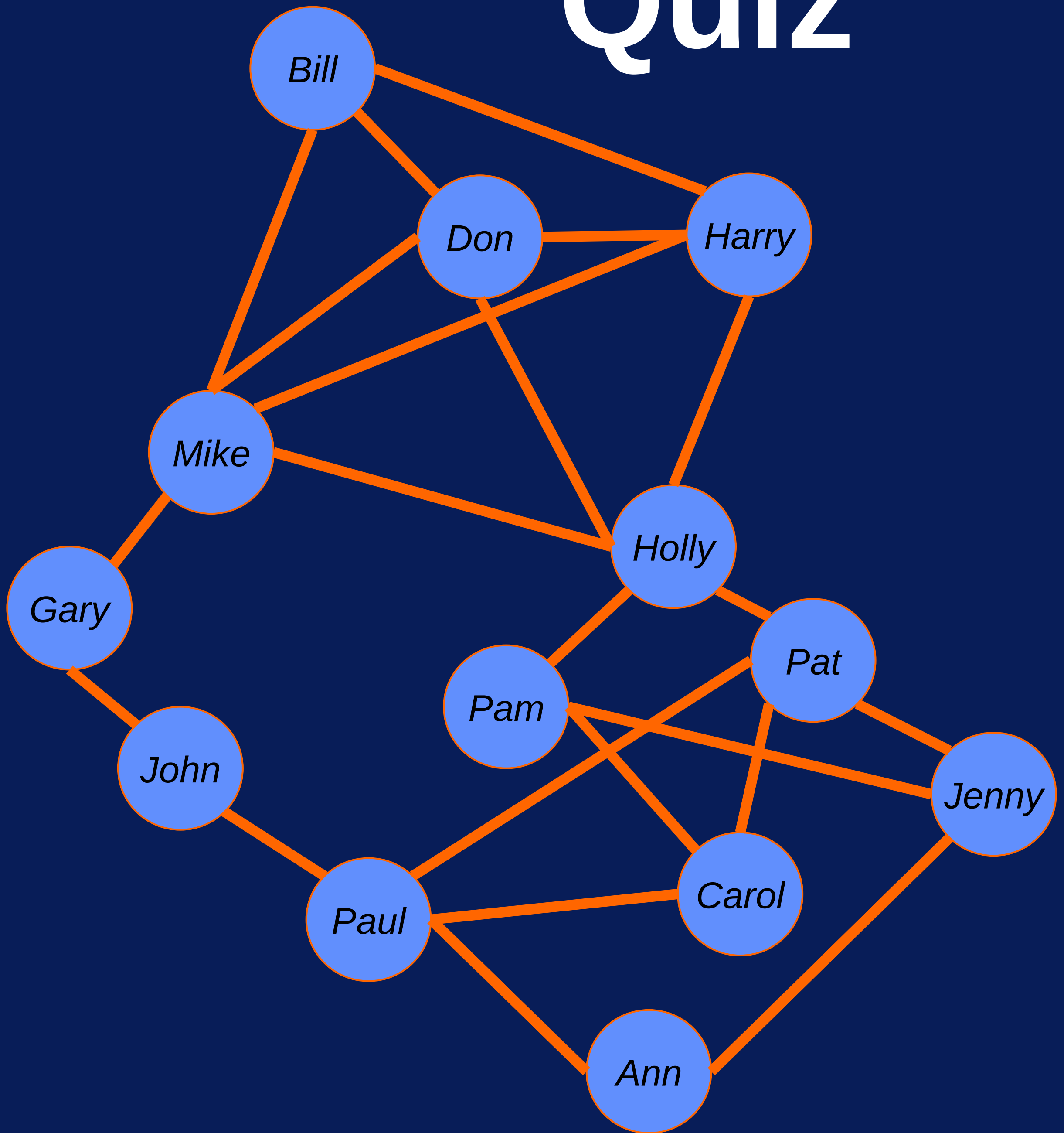


Near Cliques

- n-**clan**
  - An n -clique in which *geodesic distance* between nodes in the subgraph is no greater than n



Quiz

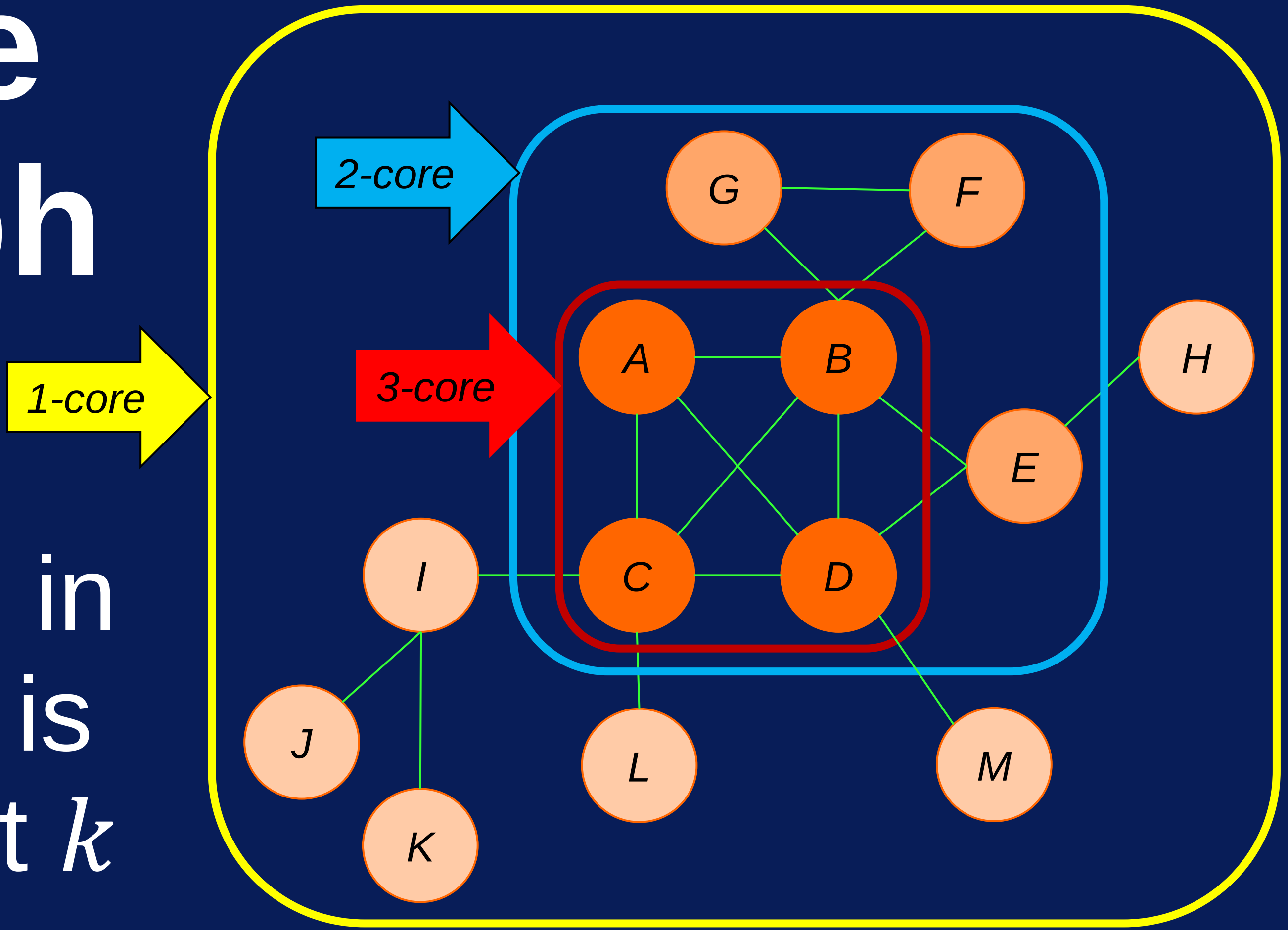


Which of the following groups are 2-clans?

(a) John Gary Mike Jenny Holly
(b) Pam Pat Harry Mike Don Holly
(c) Holly Pam Pat Carol Paul Jenny Ann
(d) Don Mike Gary John Paul

Finding dense parts of a graph

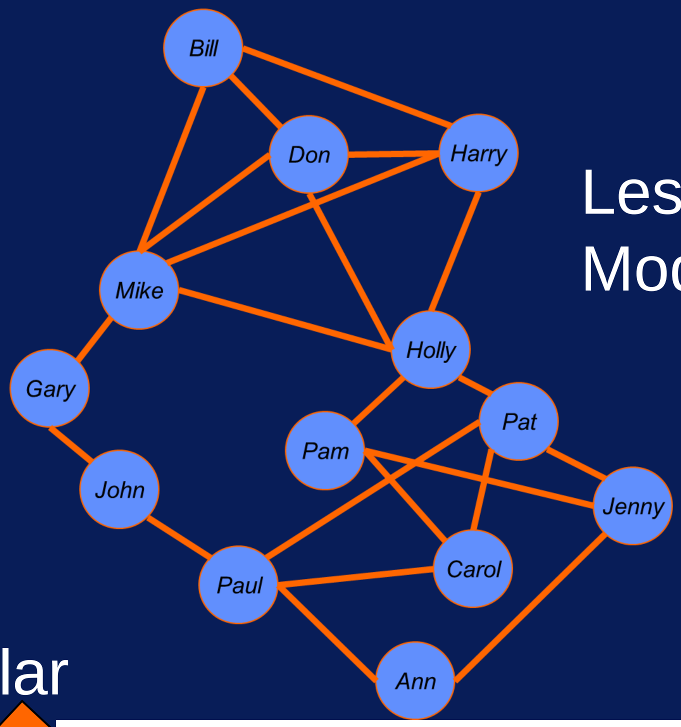
- k-core
  - Maximal subgraph in which each vertex is adjacent to at least k other vertices of the subgraph



pause

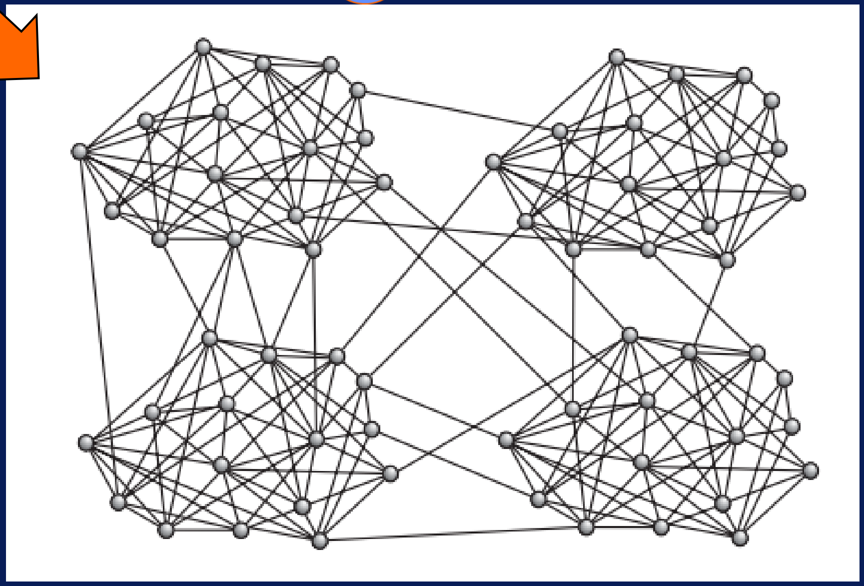
Modularity

- **A *global measure* of cluster quality**
  - fraction of the edges within the given groups minus the expected such fraction if edges were distributed at random.
  - value of the modularity lies in the range [−1/2,1).



Less Modular

More Modular



Modularity

- A *global measure* of cluster quality
- $Q = \frac{1}{2m} \sum_{ij} (A_{ij} - Pij) . \delta(Ci, Cj)$
- m: number of edges
- A: adjacency matrix
- P: expected value of the probability of nodes i, j to be connected under some probability model
- C: clusters
- δ:1 if i and j are on the same cluster, 0 otherwise

A vertex could be attached to any other vertex of the graph and the probability that vertices i and j, with degrees k_i and k_j are connected ➔ $p_{ij} = \frac{k_i.kj}{4m^2}$ ➔ $P_{ij} = \frac{k_i.kj}{2m}$

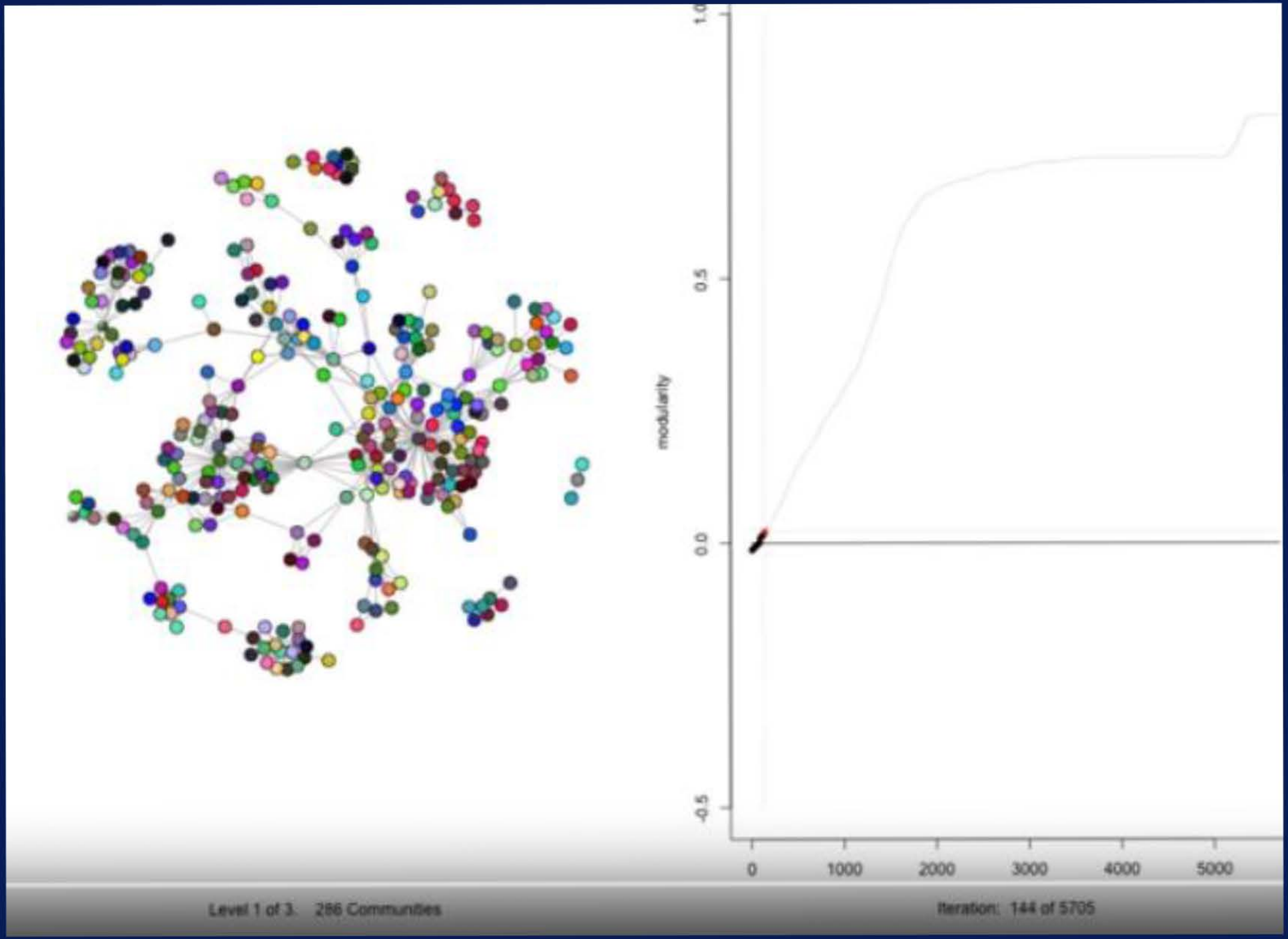
- Ideally, a community finding algorithm will find clusters with maximal modularity
  - Too hard – approx. methods are used

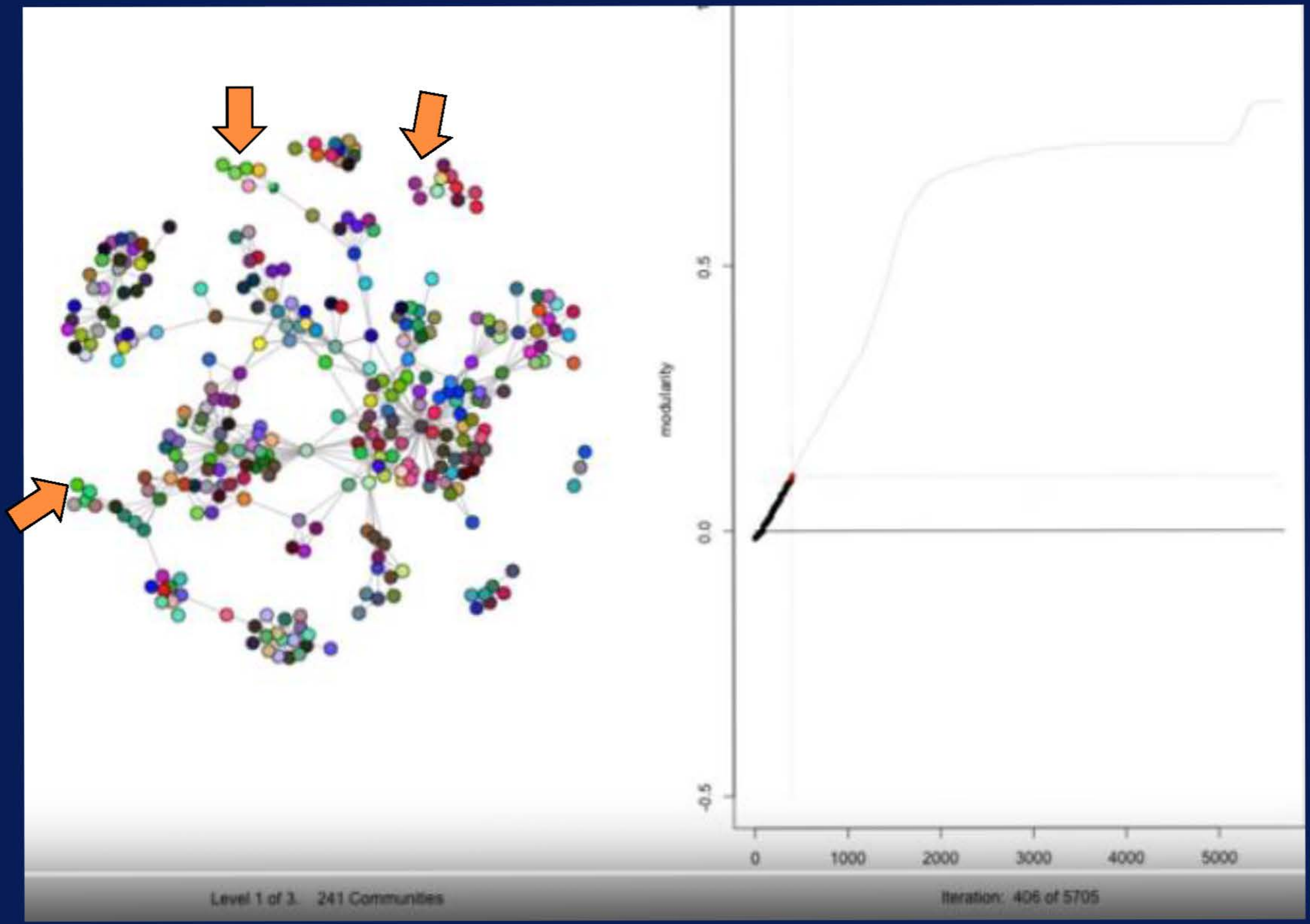
Now we will illustrate a very popular method of using this method of modularity called the Louvain method.

The best way to describe it to you may be to show you a YouTube video.

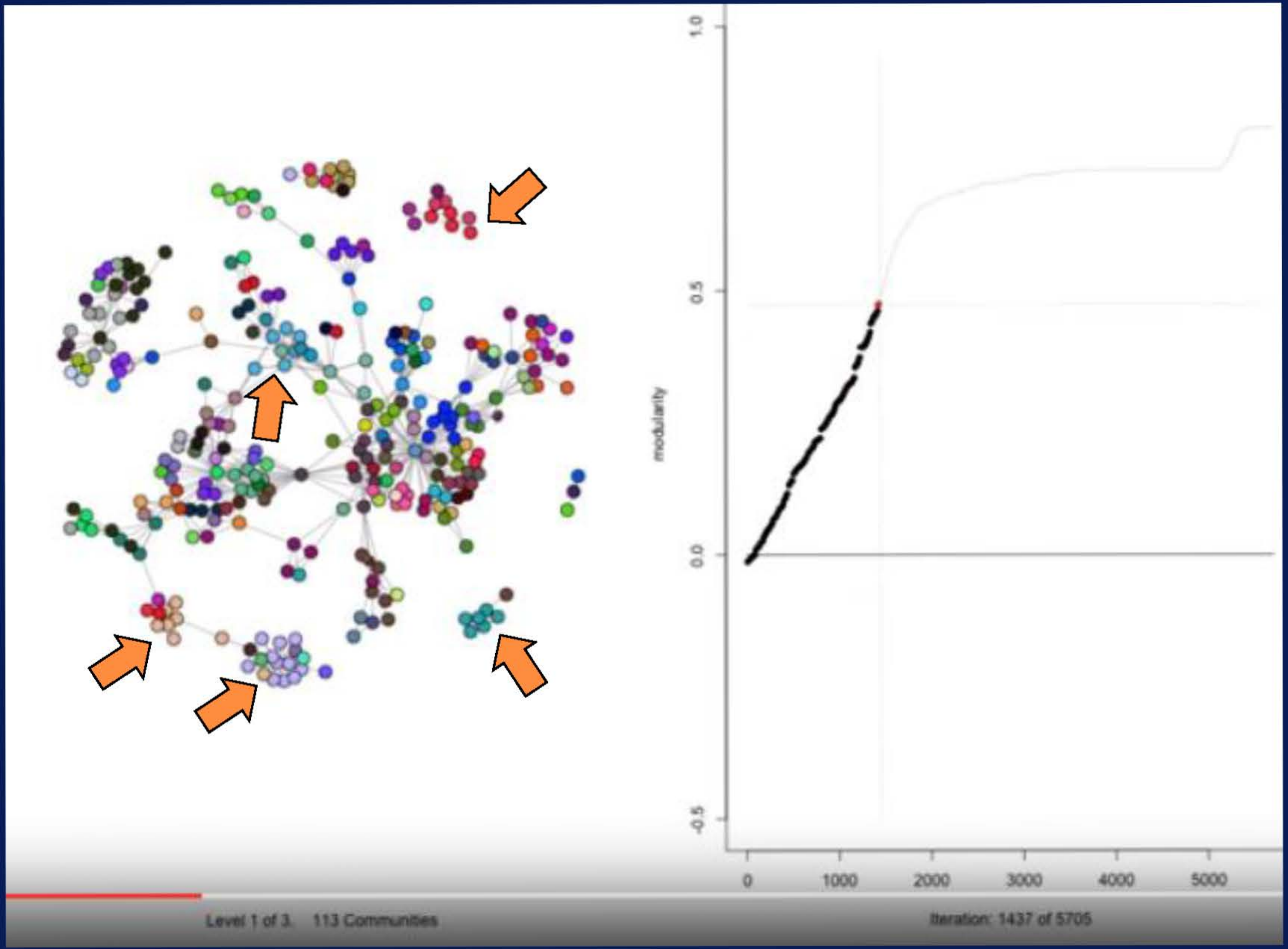
- https://www.youtube.com/watch?v=dGa-TXpoPz8

So I will leave this slide and the next one for you to read later, but I am going to skip them to show you the action as it happens.

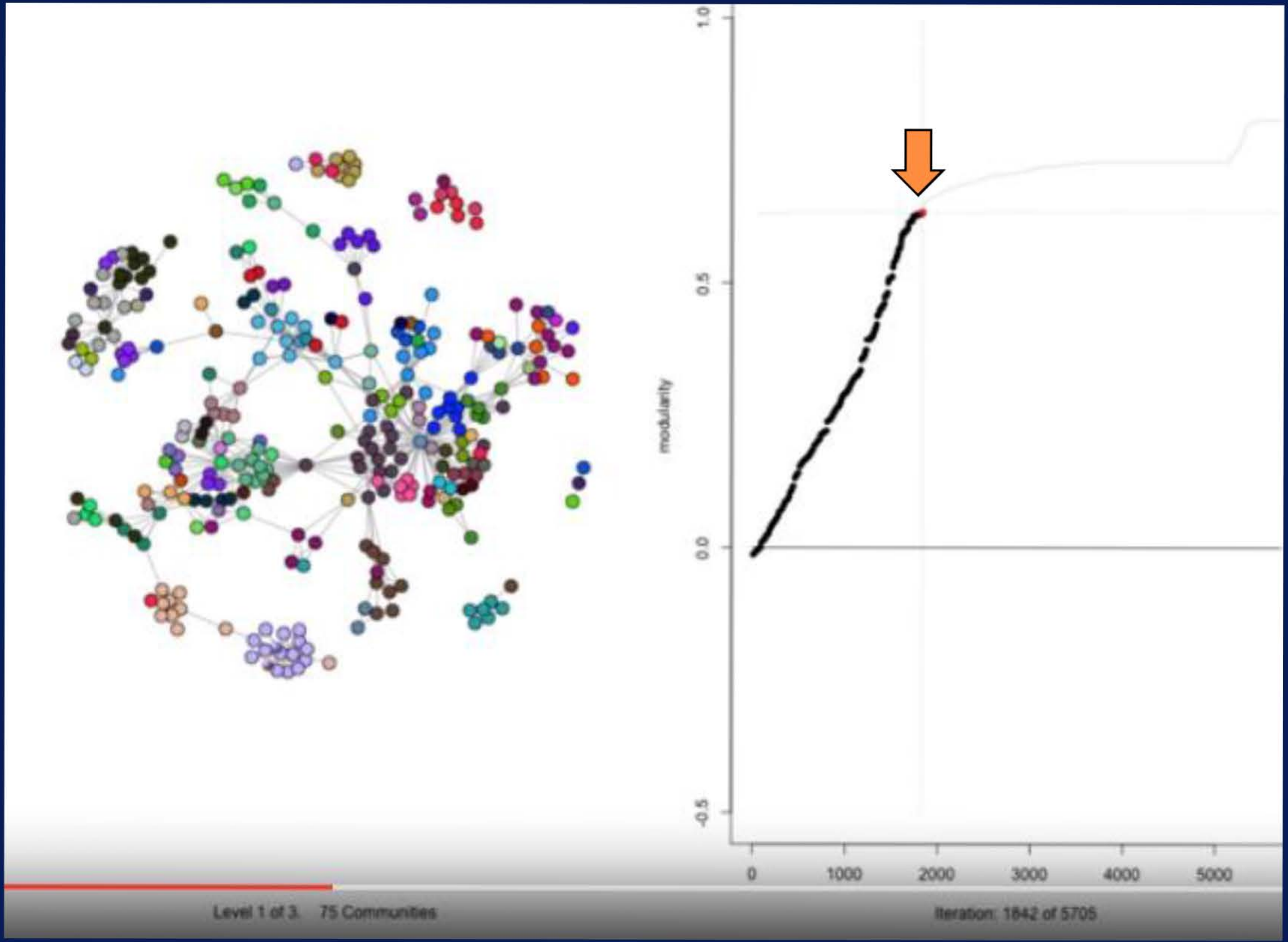

1.0
0.5
modularity
0.0
-0.5
0
1000
2000
3000
4000
5000
Level 1 of 3. 286 Communities
Iteration: 144 of 5705



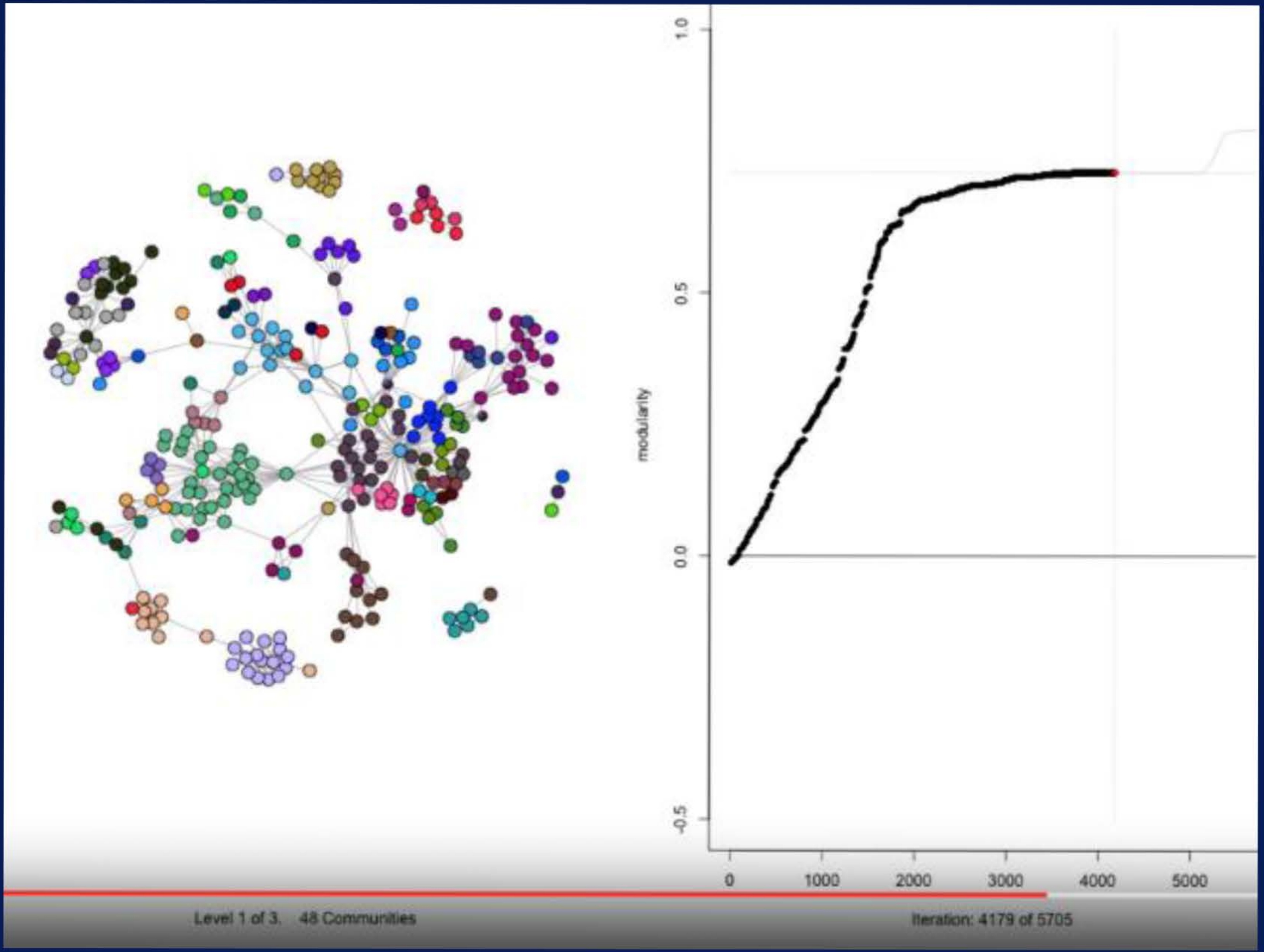
modularity
0.5
0.0
-0.5
0
1000
2000
3000
4000
5000
Level 1 of 3. 241 Communities
Iteration: 406 of 5705



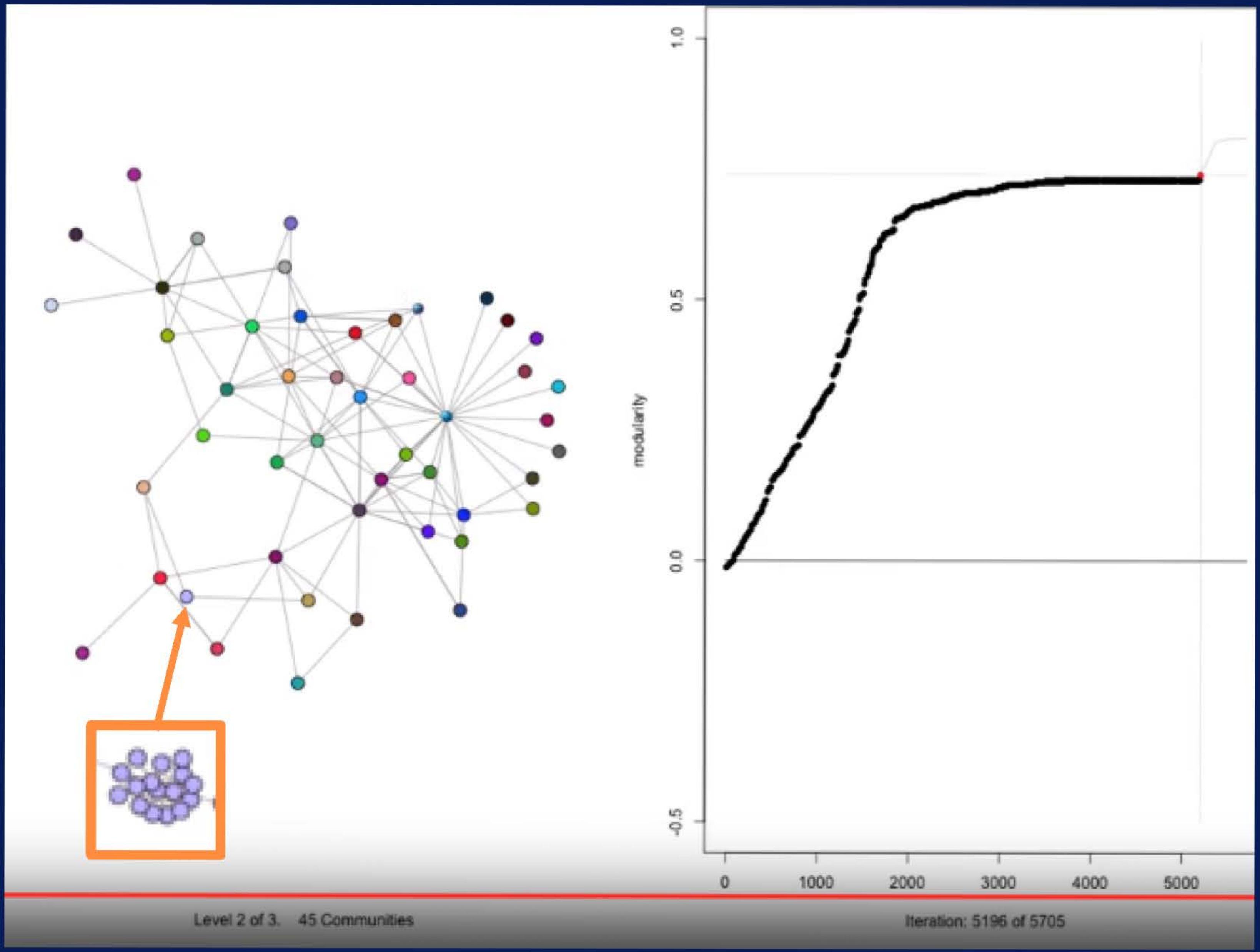
1.0
0.5
0.0
-0.5
modularity
0
1000
2000
3000
4000
5000
Level 1 of 3.
113 Communities
Iteration: 1437 of 5705

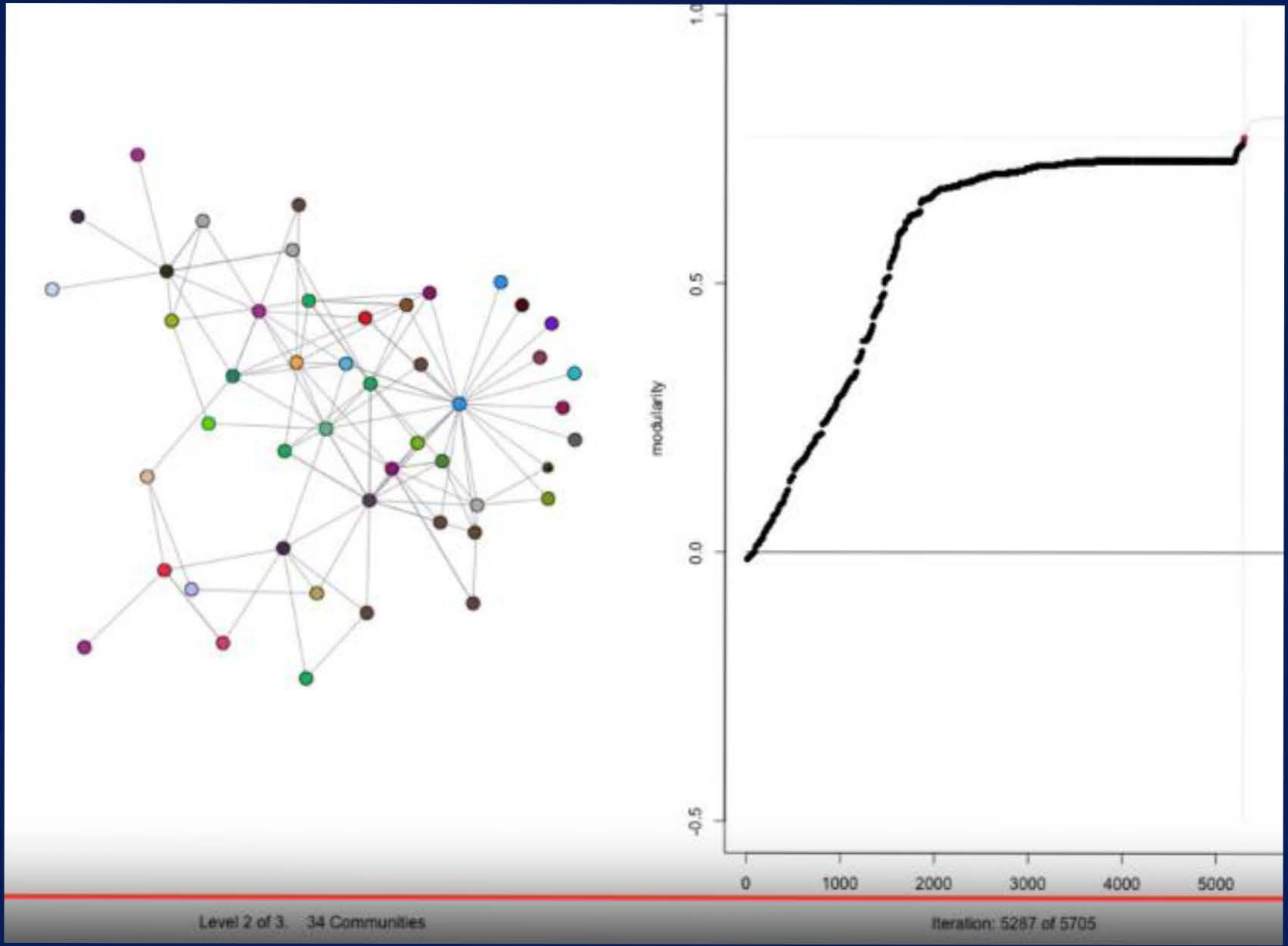


1.0
0.5
0.0
-0.5
modularity
0
1000
2000
3000
4000
5000
Level 1 of 3. 75 Communities
Iteration: 1842 of 5705

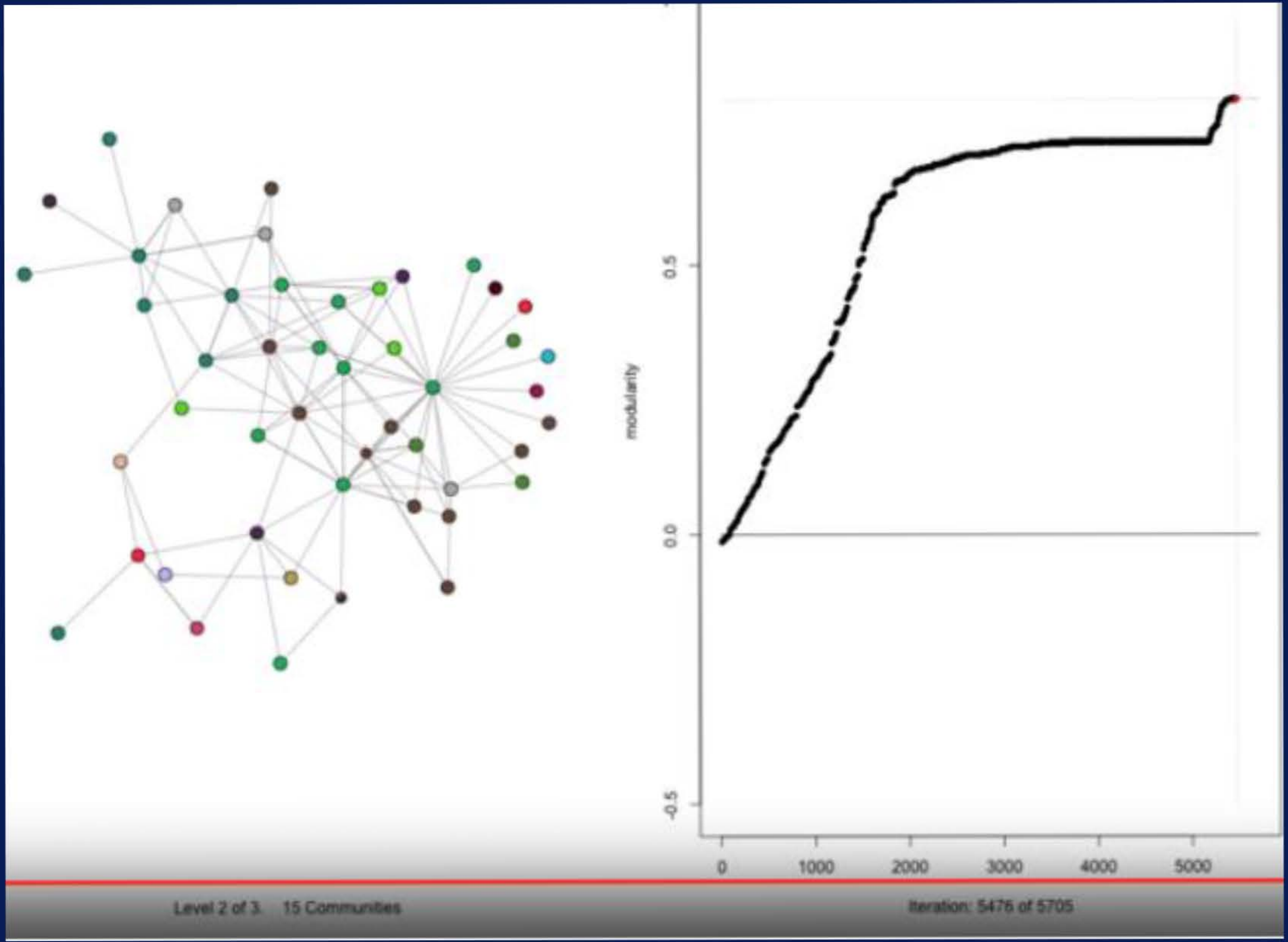


1.0
0.5
0.0
-0.5
modularity
0
1000
2000
3000
4000
5000
Level 1 of 3.   48 Communities
Iteration: 4179 of 5705

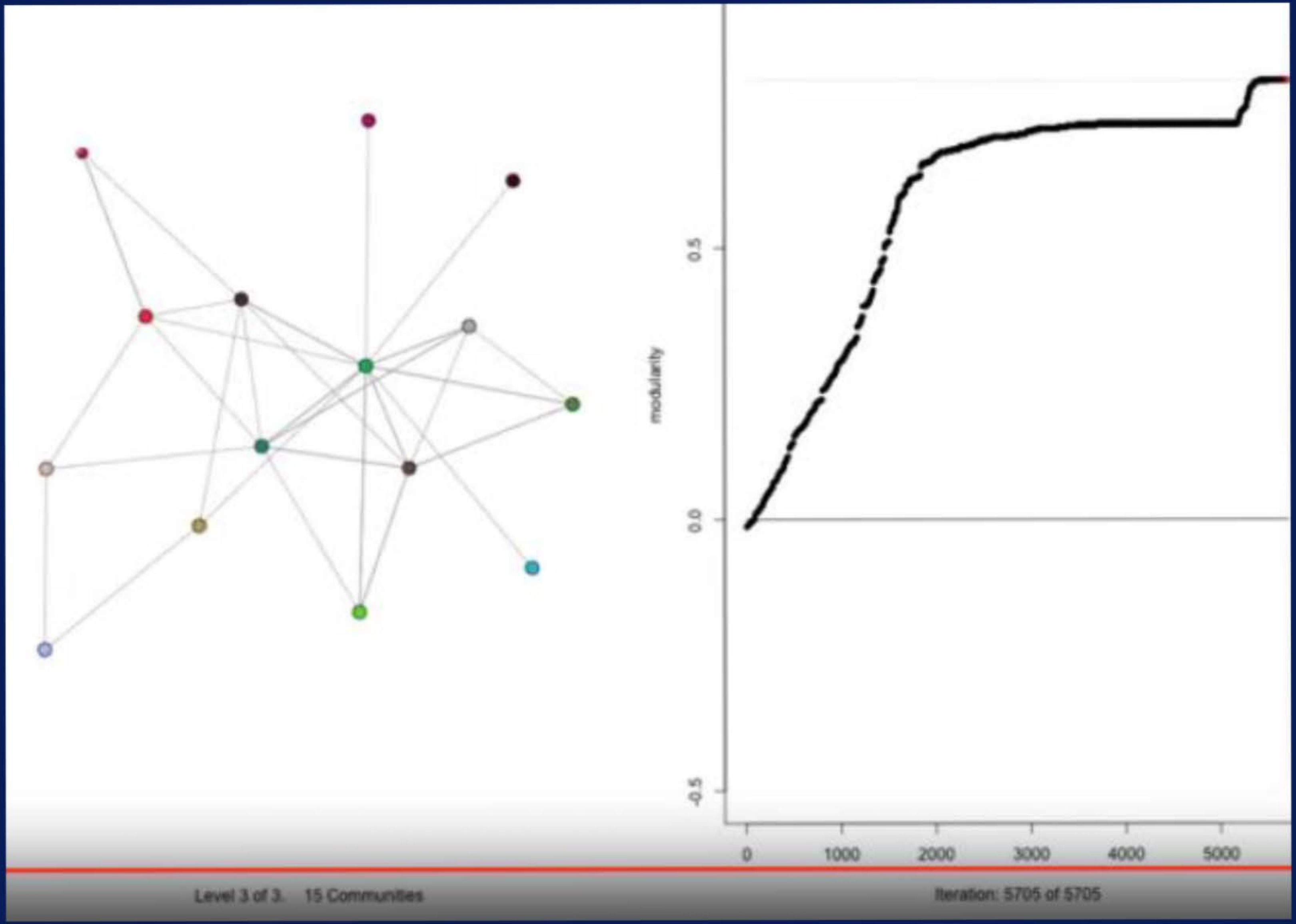

1.0
0.5
0.0
-0.5
modularity
0
1000
2000
3000
4000
5000
Level 2 of 3. 45 Communities
Iteration: 5196 of 5705



1.0
0.5
0.0
-0.5
modularity
0
1000
2000
3000
4000
5000
Level 2 of 3. 34 Communities
Iteration: 5287 of 5705



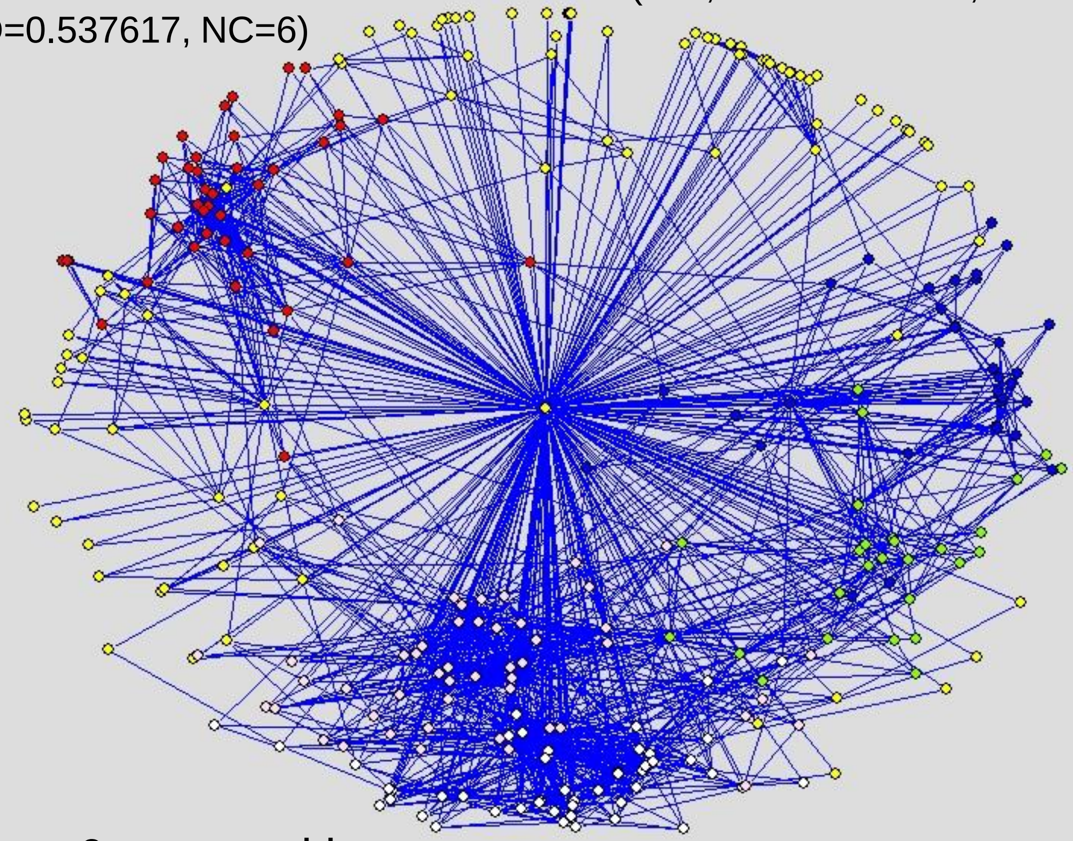
modularity
0.5
0.0
-0.5
0
1000
2000
3000
4000
5000
Level 2 of 3. 15 Communities
Iteration: 5476 of 5705



0.5
0.0
-0.5
modularity
0
1000
2000
3000
4000
5000
Level 3 of 3. 15 Communities
Iteration: 5705 of 5705

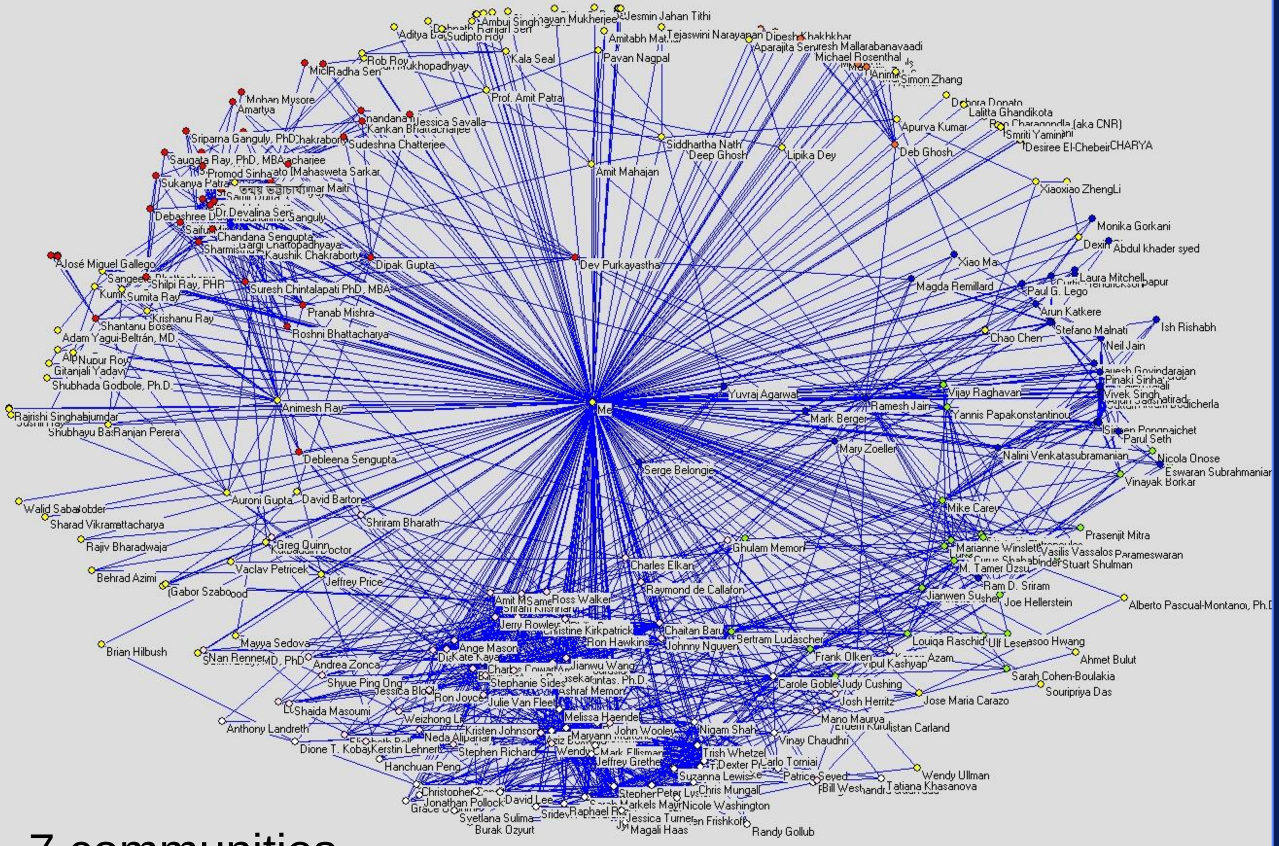
Louvain Method on AG's Linked-in Network

Multi-Level Louvain Communities in N3 (315, Res=1.000000, Q=0.537617, NC=6)



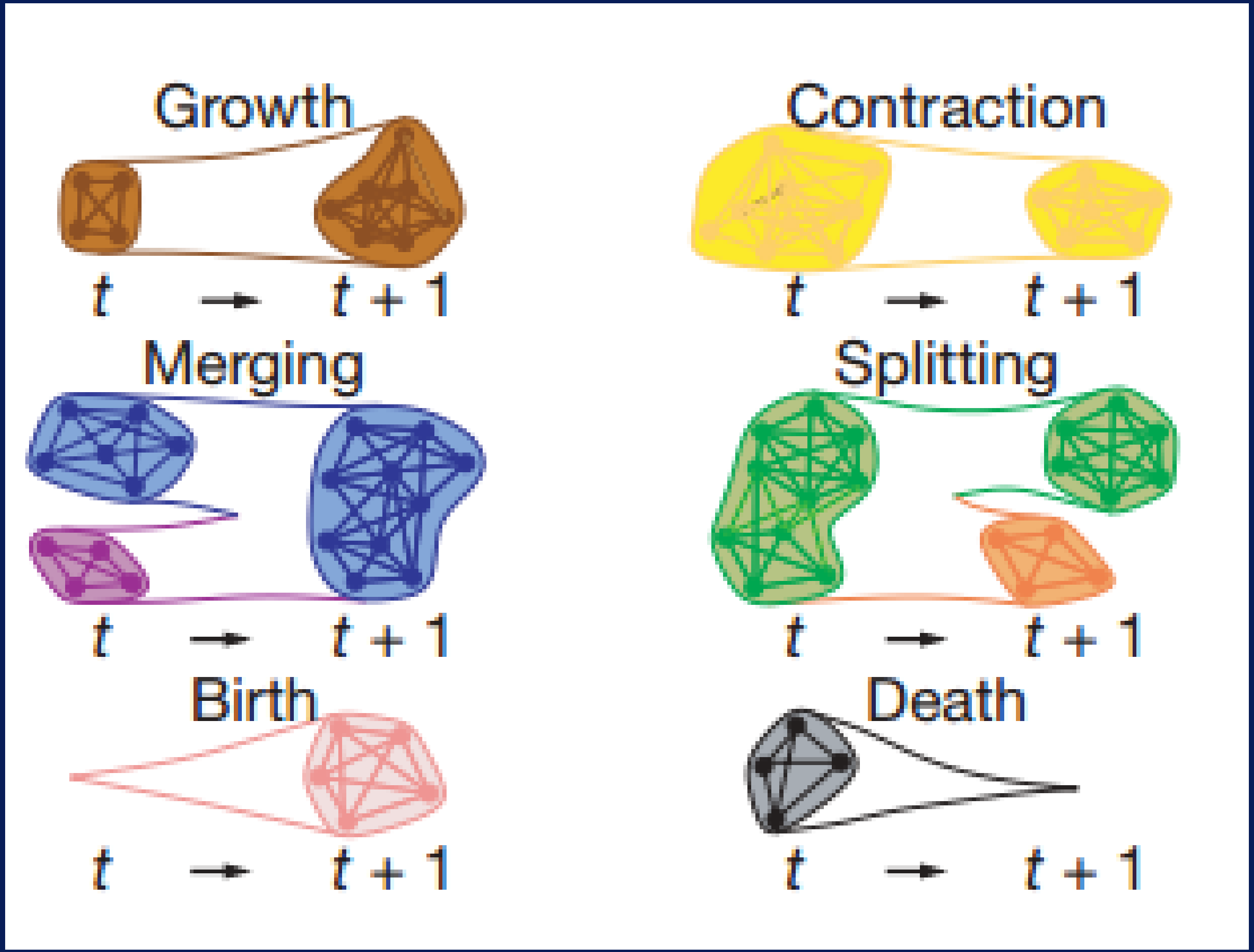
6 communities

Louvain Communities in N3 (315, Res=1.000000, Q=0.537802, NC=7)



7 communities

Evolving Communities



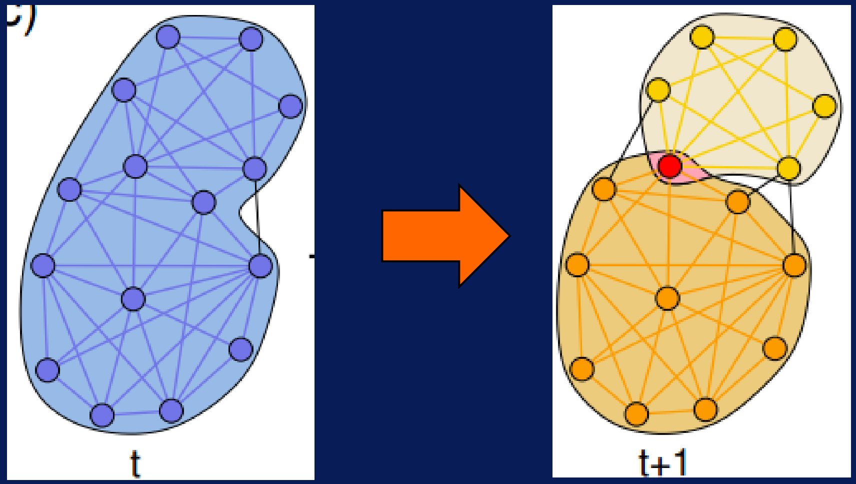
We have an optional video that you can go through.

This video discusses ways to measure and quantify the nature and extent of community evolution

Optional 4

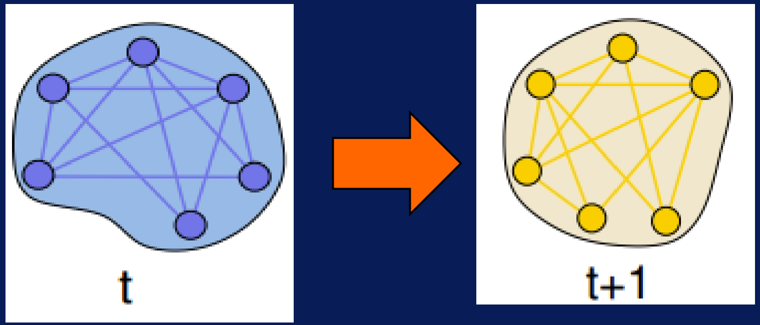
Measuring Evolution

Find a graph at time t_0 and then at t_1

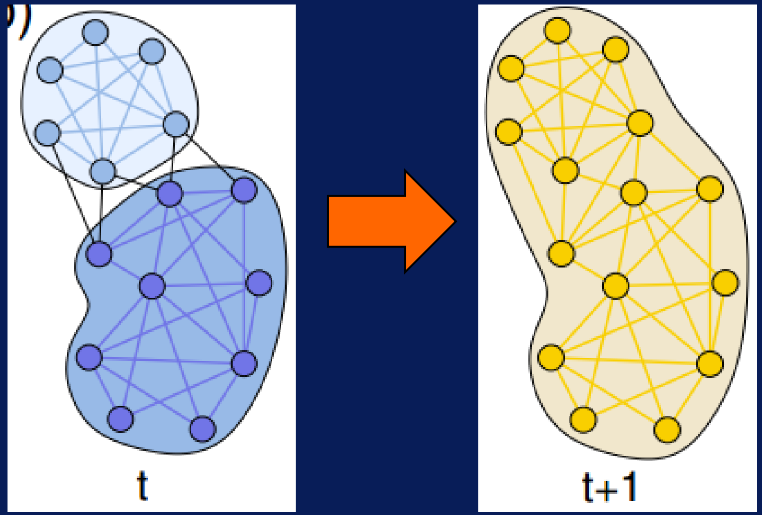


Case 3

Case 1



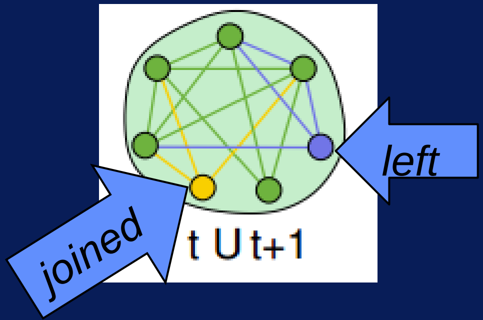
Case 2



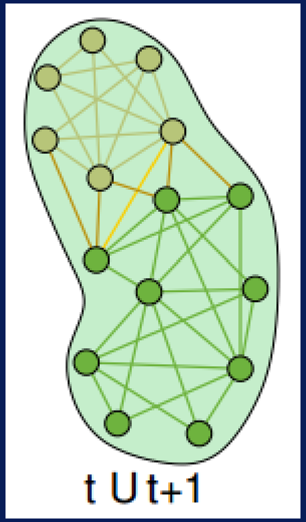
Measuring Evolution

Join the graphs

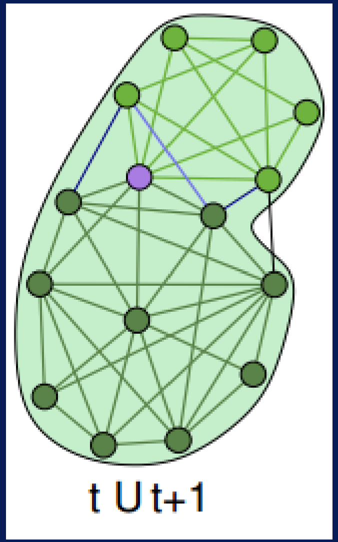
Case 1



Case 2



Case 3



Measuring Evolution

Compute

1. Autocorrelation (measures node overlap)

Count of overlapping nodes

$$C(t) \equiv \frac{|A(t_0) \cap A(t_0+t)|}{|A(t_0) \cup A(t_0+t)|}$$

Count of nodes in the joint graph

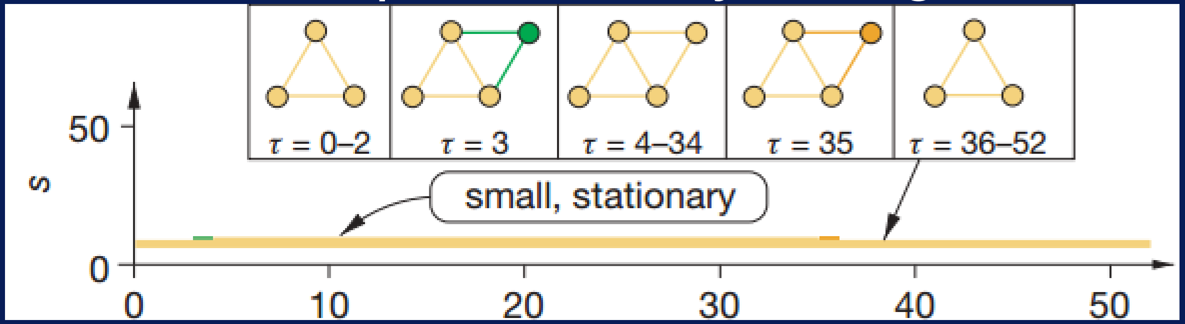
2. Stationarity (change in autocorrelation over a period)

$$\zeta \equiv \frac{\sum_{t=t_0}^{t_{\max}-1} C(t,t+1)}{t_{\max}-t_0-1}$$

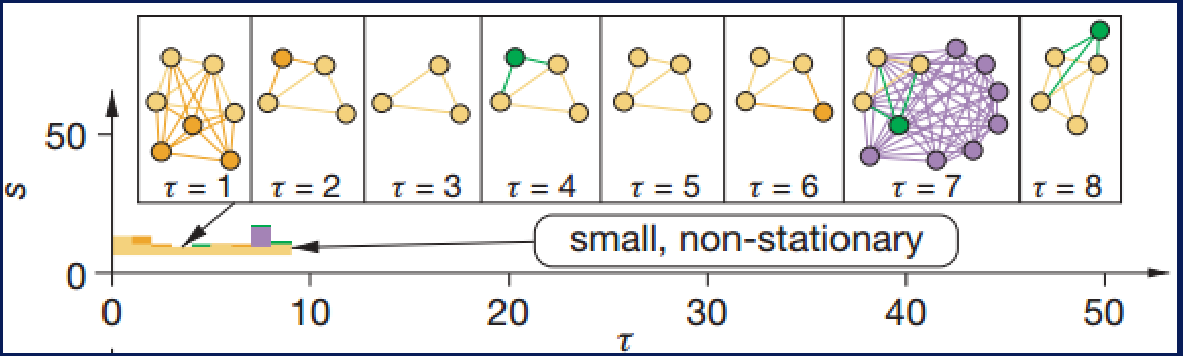
$1 - \zeta$ represents the average ratio of members changed in one step

Stationarity

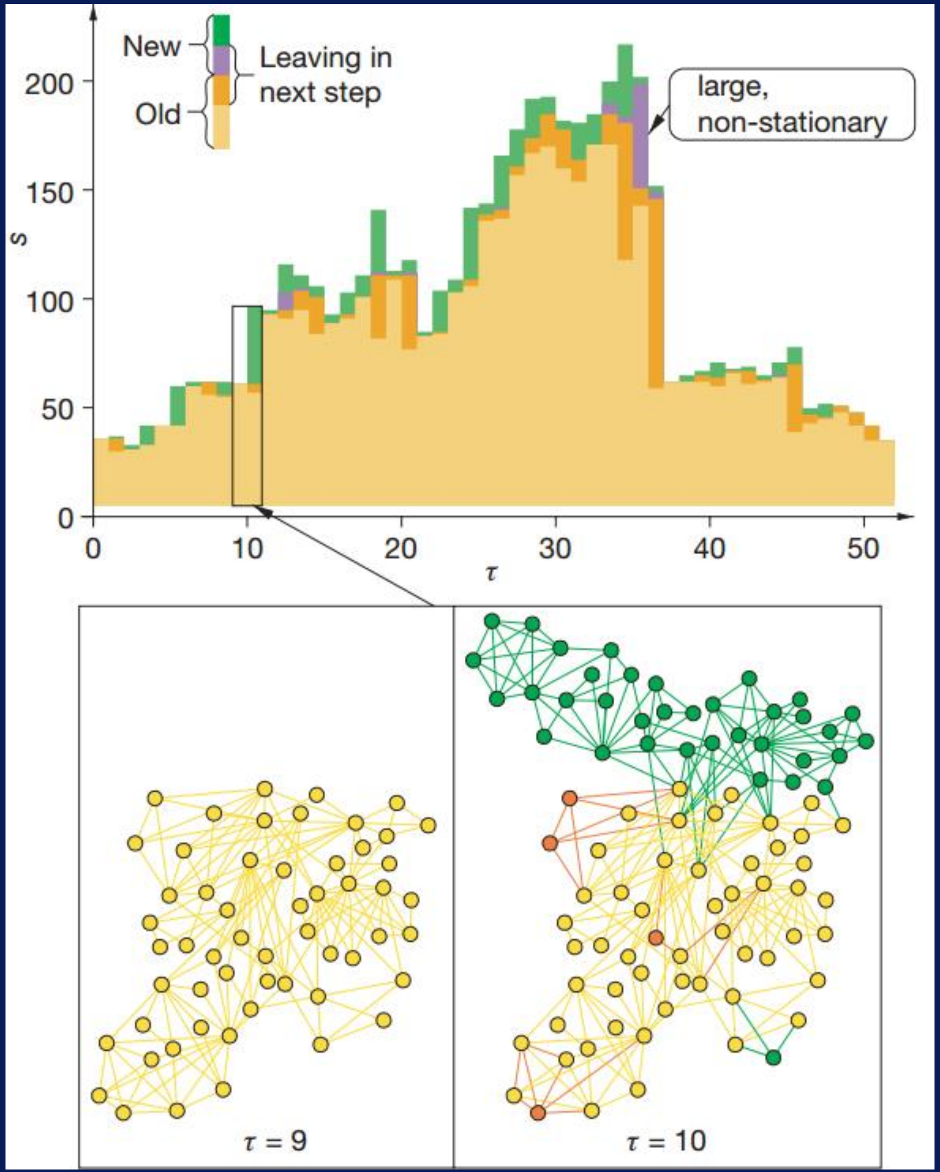
Small Graph, not many changes



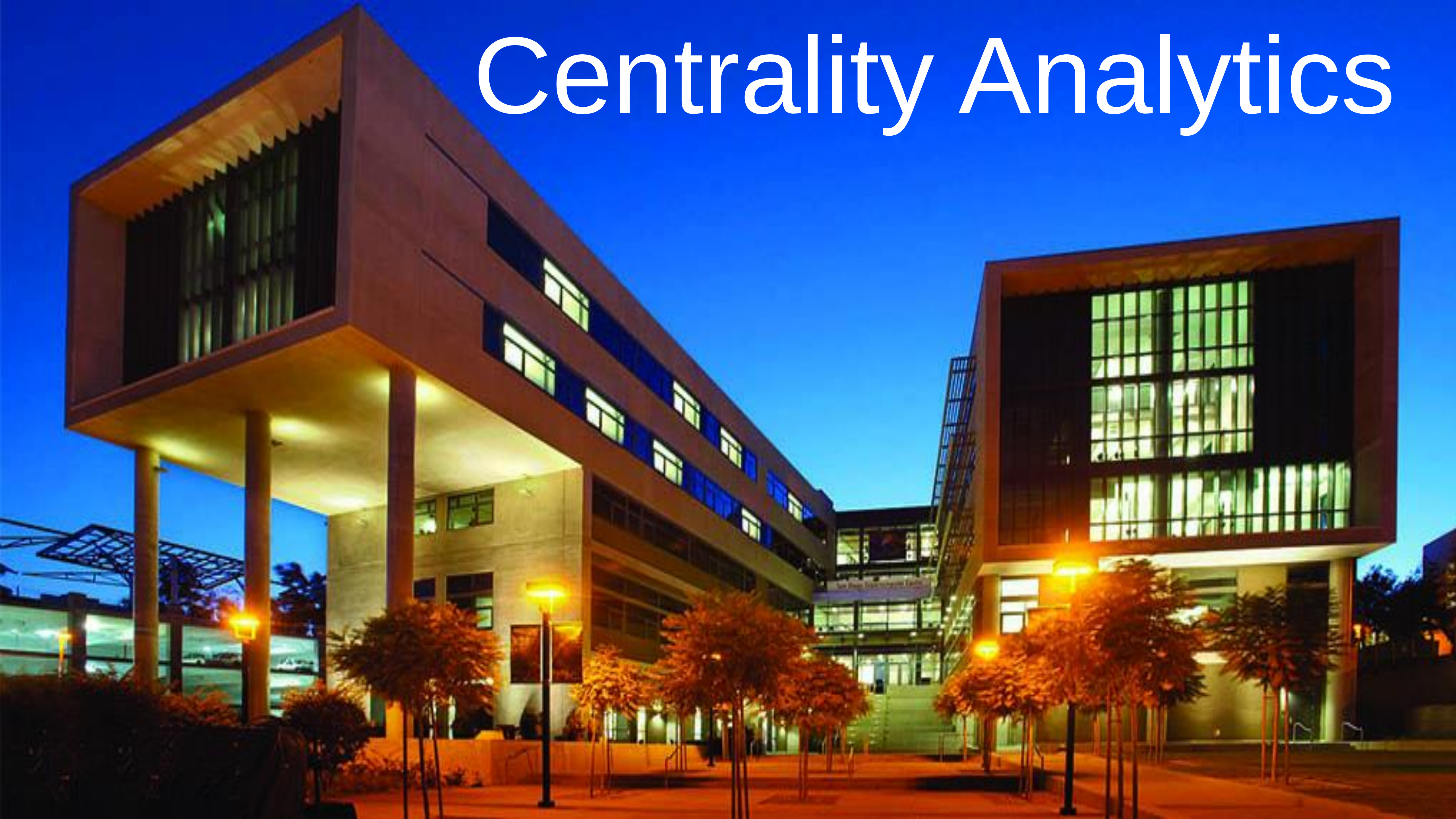
Small Graph, many changes



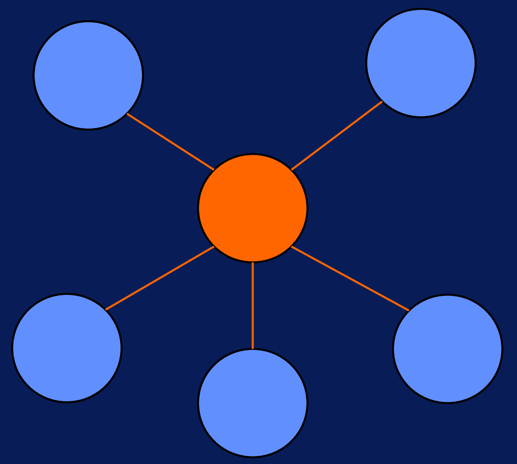
Large Graph, many changes



Centrality Analytics



Are all nodes equally important in a network?



What makes some nodes more important/valuable than others?

- Influencers in a social network
- A junction station in a transport network
- A house-keeping gene in a biological network
- A central server in a computer network

Key Player Problems

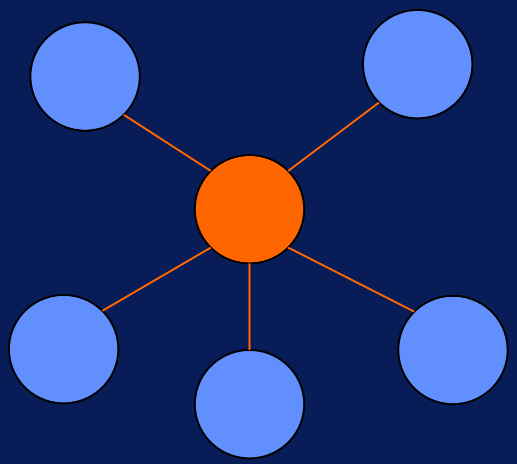
- Given a social network find a set of k nodes
  1. which, if removed, would maximally disrupt communication among the remaining nodes
     - Given an infectious disease in a city which subpopulation should be immunized so as to maximally hinder the spread of the infection?
  2. that is maximally connected to all other nodes
     - Given a community of a 1000 members find 5 members who would be able to convince others to vote for a candidate

Centrality and Centralization

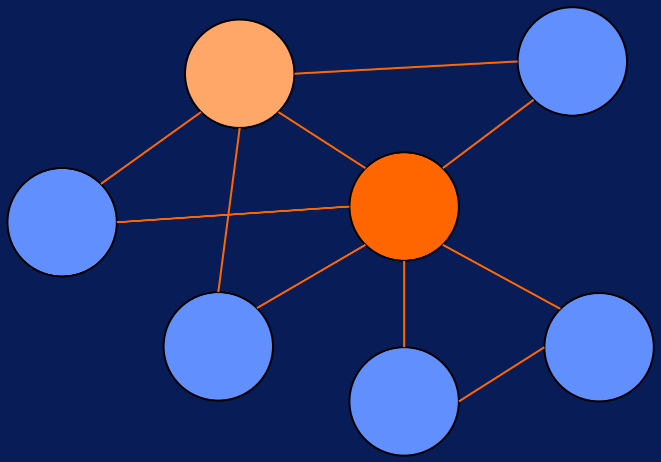
- Centrality
  - Measure of importance of a node (or edge) based on its position in the network
  - Different ways to measure centrality

- Centralization
  - Measure for a network and not a node
  - Degree of variation in the centrality scores among the nodes

$$\frac{\sum(c_{max} - c(v_i))}{c_{max}}$$



More centralized



Less centralized

Different Measures of Centrality

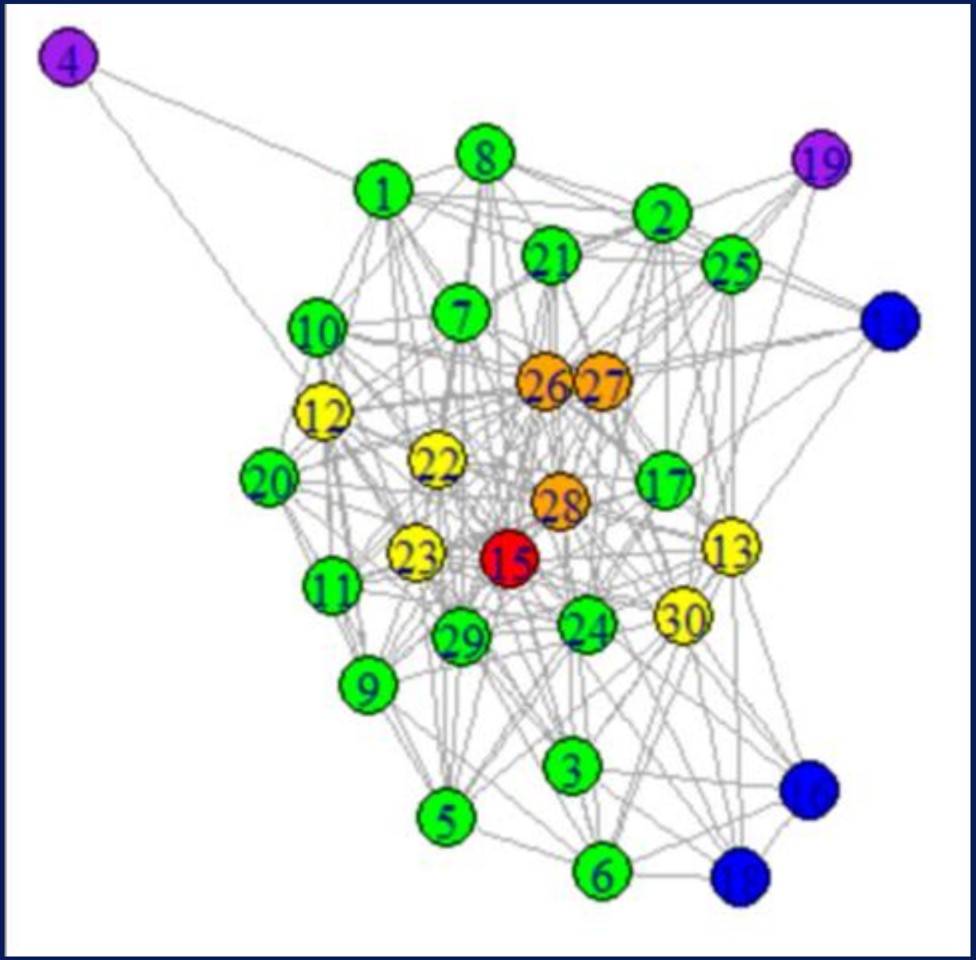
- Types
  - Degree Centrality
  - Closeness Centrality
  - Betweenness Centrality
  - Eigenvector Centrality
  - Katz Centrality
  - … and many more …
- Our goal
  - The principle behind some of these methods
  - A couple of cases

Degree Centrality

- Seen this already (sort of)!!
- Count of the number of edges incident upon a given node normalized by the possible number of edges

$$(\#\ of\ edges)/(N-1)$$

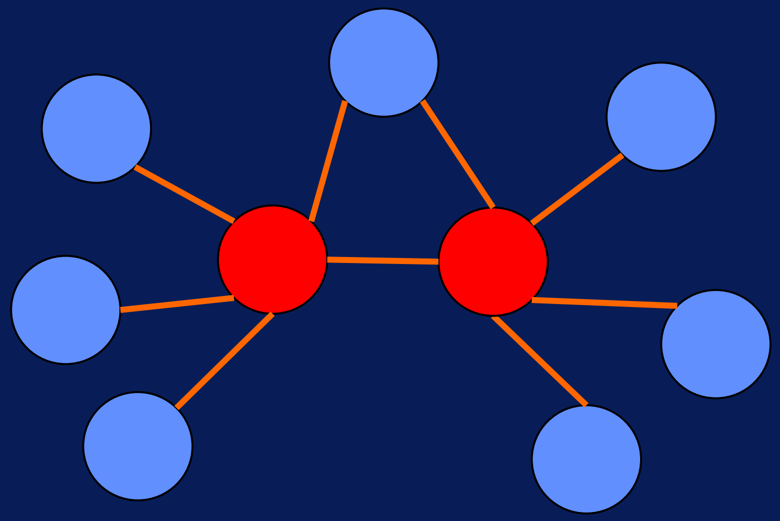
- “hub” (maximally-connected) nodes



Group Degree Centrality

- Consider a group as a single entity
- Count of the number of edges incident upon the group normalized by non-group members

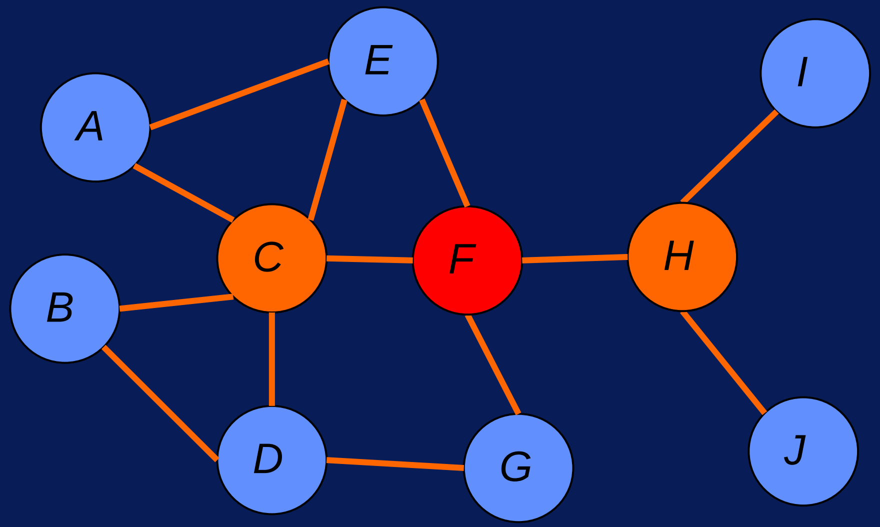
$$\frac{\#\ of\ edges\ into\ the\ group}{\#\ of\ non-group\ members}$$



Group centrality = 1

Closeness Centrality

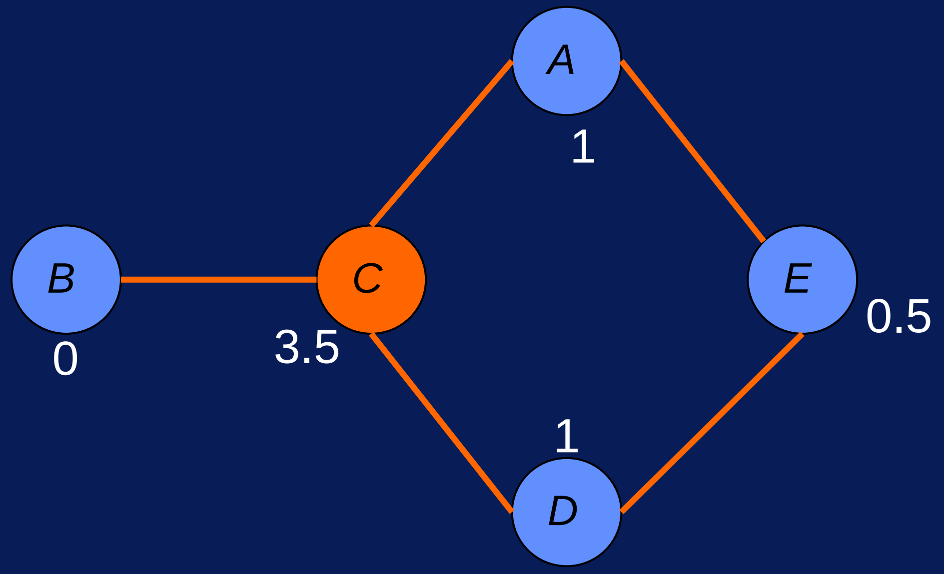
- Sum of shortest-path distances from all other nodes (normalized)
  - Low raw closeness means node has short distance from other nodes
- For an information flow network
  - Low closeness nodes receive information sooner than other nodes
    - Same for other flows *if the flow happens through shortest paths*
    - How about gossip?
  - A low closeness can influence many others, directly and indirectly



F has the highest CC score

Betweenness Centrality

- Ratio of pairwise shortest paths that flows through node i and count of all shortest paths in the graph
- Measures fraction of shortest-path commodity flow passing through a node
  - Not applicable to non-flow situations e.g., rumor, infection
  - Not applicable when shortest path routes are not taken



Paths: BCA, BCAE, BCD, BCDE, CAE, CDE, AED

We have two optional videos that you can go through.

The first video discusses another, a little more mathematical form of centrality called Eigenvector Centrality that is related to Pagerank , made famous by the Google Search Engine

The second video discusses takes a second look at the two key player problems and presents two new metrics which may be more appropriate for these two problems

Optional 5

Eigenvector Centrality

- A node is important if its neighbors are important

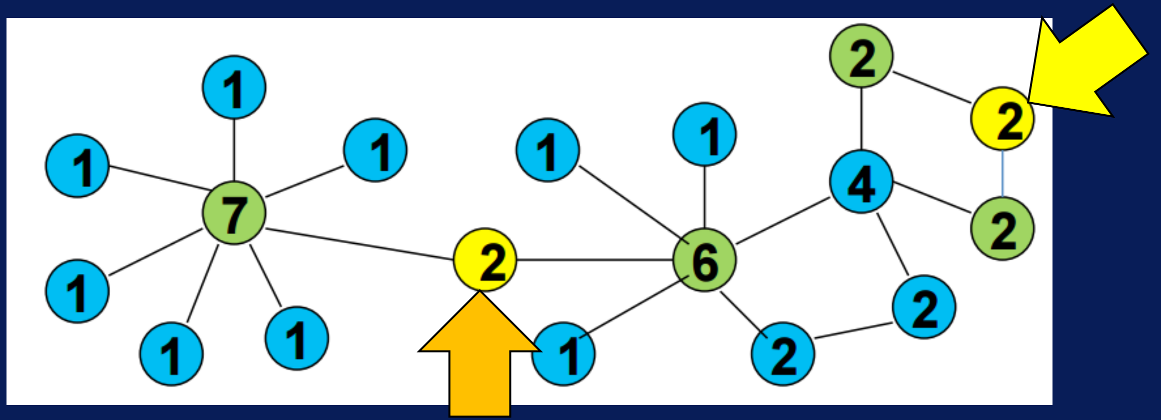
$$C_{E(v_i)} \propto \sum_{v_j \in Ni} A(i,j) C_E(vj)$$

Centrality of node v_i

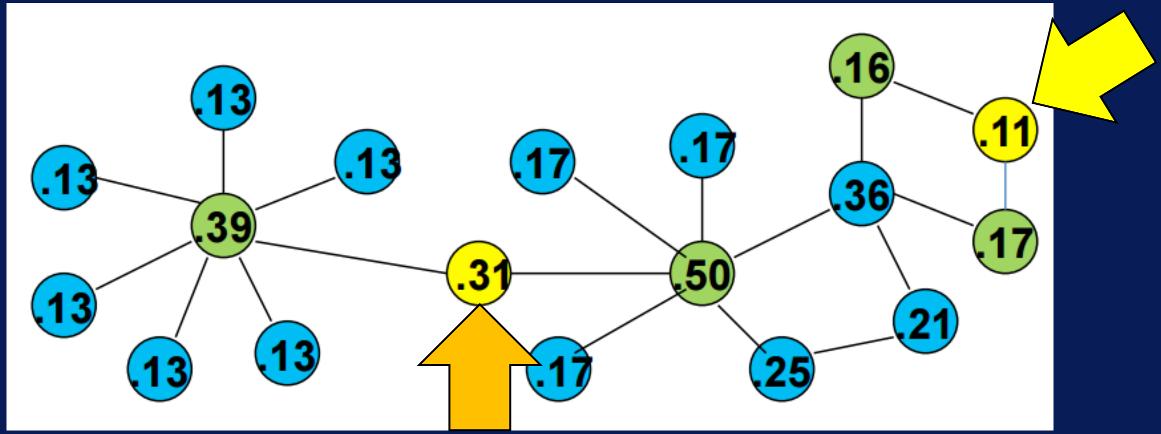
Neighbors of v_i

$$x \propto Ax$$
$$\lambda x = Ax$$

- The measure is “local”



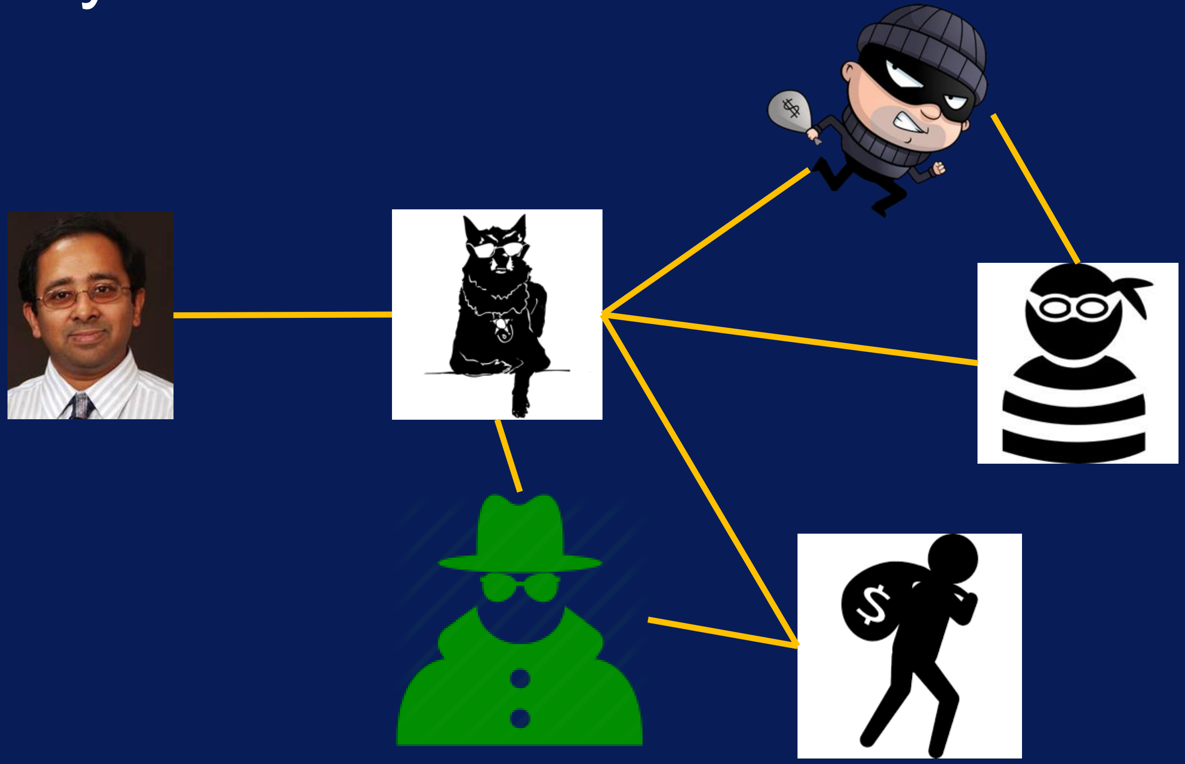
Degree centrality



Eigenvector centrality

Eigenvector Centrality

- “It’s not what you know but who you know”



Pagerank

- A variant of Eigenvector centrality

- **Random Surfer:** a surfer who, given a graph, starts from a node v, exits a random outbound link to the next node with probability α, but visits a completely new node with probability $1 - \alpha$

Pagerank

The Classical Graph Analytics Algorithm

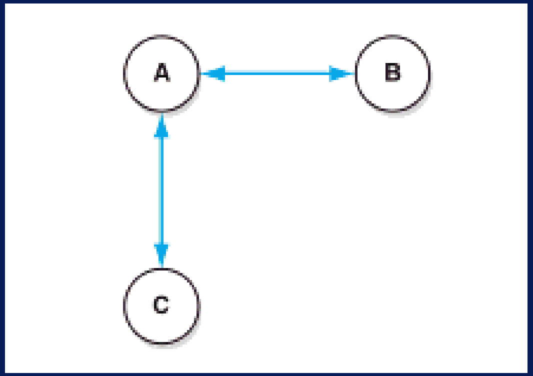
- **Idea (Page and Brin)**
  - A random surfer of a graph is most likely to find nodes that are highly “central” because a lot of nodes point to these nodes directly or indirectly
  - Pagerank
    - The *stationary probability distribution* of a random walk on the graph

http://www.cs.duke.edu/csed/principles/pagerank/

Scalable Computation of Pagerank

- **Power Iteration**
  - Assign an arbitrary pagerank value to every page
    - PR is an n-vector
  - $\boldsymbol{M}$: transition matrix
    - Implements the random surfer model
  - $PR(t+1) = \boldsymbol{M}.PR(t)$
  - Repeat till convergence
    - $|PR(t+1) - PR(t)| < \varepsilon$

Power Iteration Algorithm



Three nodes A,B And C

- P(A)=(1-α)+ α(pagerank(B)/1+pagerank(C)/1)
- P(B)=(1-α)+ α(pagerank(A)/2)
- P(C)=(1-α)+ α(pagerank(A)/2)

Begin with the initial value as 0 (bad choice?)

1st iteration:

P(A)=0.15+0.85*0=0.15

P(B)=0.15+0.85*(0.15/2)=0.21

P(c)=0.15+0.85*(0.15/2)=0.21

2nd iteration:

P(A)=0.15+0.85*(0.21*2)=0.51

P(B)=0.15+0.85*(0.51/2)=0.37

P(C)=0.15+0.85*(0.51/2)=0.37

3rd iteration:

P(A)=0.15+0.85*(0.37*2)=0.78

P(B)=0.15+0.85*(0.87/2)=0.48

P(C)=0.15+0.85*(0.87/2)=0.48

After 20 iterations

P(A)=1.46

P(B)=0.77

P(C)=0.77

Optional 6

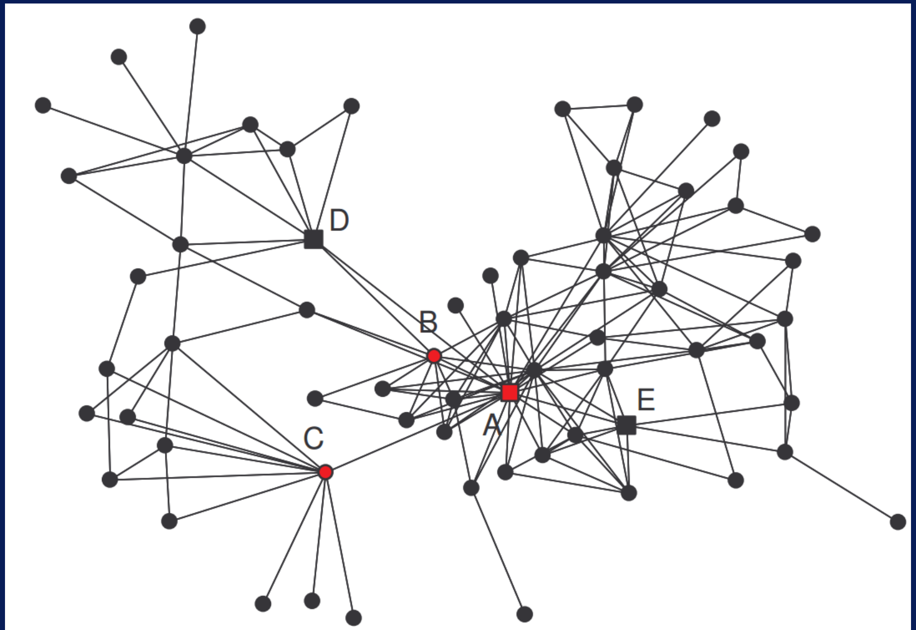
Key Player Problem 1

- Given an infectious disease in a city which subpopulation should be immunized so as to maximally hinder the spread of the infection?
- Heart of the problem
  - **Removal of the set will maximally disrupt**
- Suppose the removal fragments the graph into k components
- Fragmentation metric

$$F = 1 - \frac{2\sum_{i<j} 1/dij}{n(n-1)}$$

d_{ij}: distance between nodes i and j

Terrorist Network (Krebs 2002)



Removing A, B & C breaks up the graph into 7 components, F = 0.59

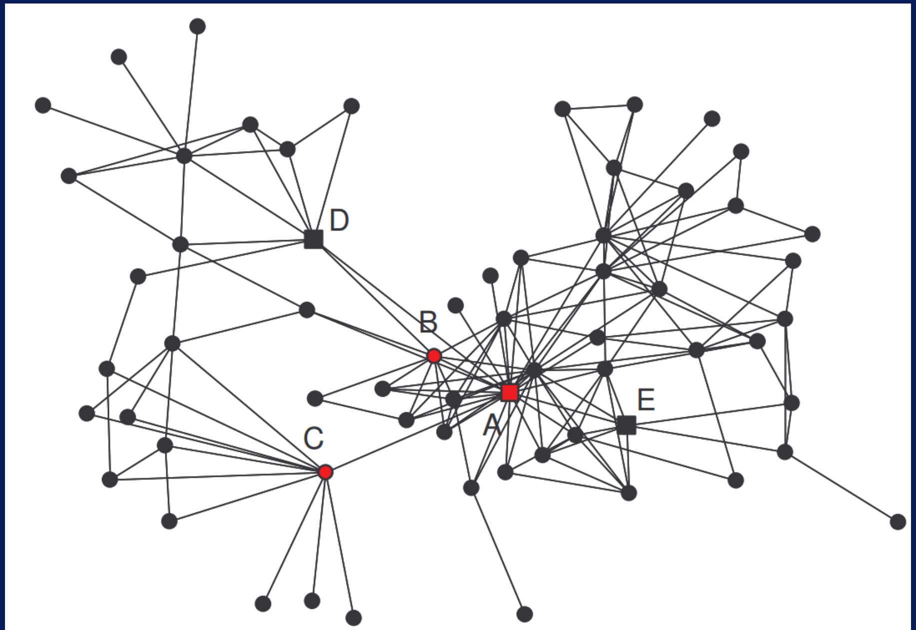
Key Player Problem 2

- Given a community of a 1000 members find 5 members who would be able to convince others to vote for a candidate
- Heart of the problem
  - **Reaching the maximum number of people in k hops (or less) from a small group of size S**
- Distance-weighted Reach

$$^{D}R = 1 - \frac{\sum_j \frac{1}{d_{Sj}}}{n}$$

d_{Sj}: distance of a node j from the group S

Terrorist Network (Krebs 2002)



Targeting A, C and D will reach 100% of the network