

Network Flow and Metrics

CS 579 Online Social Network Analysis

Dr. Cindy Hood
9/9/25

Homework Assignments

- ▶ HW #2 assigned yesterday
 - ▶ Due by midnight Friday 9/19
- ▶ Please contact TAs with questions on hw grading

Exams and Final Project Poster Presentation

- ▶ Exam 1 - Oct 9 in class
- ▶ Exam 2 - Dec 2 in class
- ▶ Final Project Poster Session - Dec 4 in class
- ▶ Online students (sections 2 and 3) will have remote options

Teaching Assistants

- ▶ **Siva Krishna Golla**
 - ▶ sgolla2@hawk.illinoistech.edu
 - ▶ Mondays 2-3pm on zoom
- ▶ **Khush Dhiren Patel**
 - ▶ kpatel210@hawk.illinoistech.edu
 - ▶ Wednesdays 11-12 online
- ▶ **Aswith Sama**
 - ▶ asama@hawk.illinoistech.edu
 - ▶ Thursdays 3-4pm on zoom
- ▶ **Not yet officially working, waiting for authorization (US govt)**

HW #2 due by midnight 9/19

- ▶ In this assignment you will create networks/graph models from 2 different datasets. You may use any tool/platform/language that you like. I have attached a few pages from the Elements of Network Science Book that illustrate basic use of Stata, R and Python. [Section 2.3 ElementsofNetworkScience.pdf Download Section 2.3 ElementsofNetworkScience.pdf](#)
- ▶ (1) The first dataset is Chicago Community Areas https://en.wikipedia.org/wiki/Community_areas_in_Chicago
- ▶ [Links to an external site.](#)
 - Nodes = Community areas
 - Edges = Shared physical boundary (i.e. adjacency) with other community area. Note that you may have to make some assumptions here since you are determining boundaries from the image of the map on the page cited above. State your assumptions.
- ▶ You will then create a labelled visualization of this graph and plot the degree distribution of the nodes. You will submit
 - ▶ (1a) Input file with graph representation,
 - ▶ (1b) Labelled visualization of network created.
 - ▶ (1c) Plot of degree distribution.

HW #2 con't

- ▶ (2) The second dataset is the CS 579 Class Participant Data [Social Network Data collection.xlsx](#)
- ▶ [Links to an external site.](#)
 - Nodes = Class Participants, entities in common
 - Edges = Shared entity
- ▶ You will create a bipartite graph. Some data cleaning will be necessary. State and justify any assumptions you make during the data cleaning. You will then create a unimodal graph that is a projection of the bipartite graph.
- ▶ You will create labelled visualizations of both the bipartite and unimodal graphs and plot the degree distribution of the unimodal graph. You will submit
 - ▶ (2a) Input file for the bipartite graph.
 - ▶ (2b) Labelled visualization of bipartite graph.
 - ▶ (2c) Description of method for projecting bipartite graph to unimodal graph including code.
 - ▶ (2d) Labelled visualization of unimodal graph.
 - ▶ (2e) Plot of degree distribution of unimodal graph.

HW #2 - con't

- ▶ (3) Compare the degree distributions of the graphs from the two different datasets. What is similar? What is different? Is this what you expected? Why or why not?
- ▶ (4) Provide the details of how you did this assignment. What tools did you use to complete the assignment? Why did you choose the tool? Provide citations and links to references and code used. If AI (e.g. ChatGPT, etc.) was used, please include a transcript of the exchange.
- ▶ The above can be submitted in a zipped folder that includes
 - input files labelled as Input_file1, Input_file2
 - pdf report of everything else

Chicago



A. Burnside
B. Oakland
C. Montclare



HW #2 Data cleaning

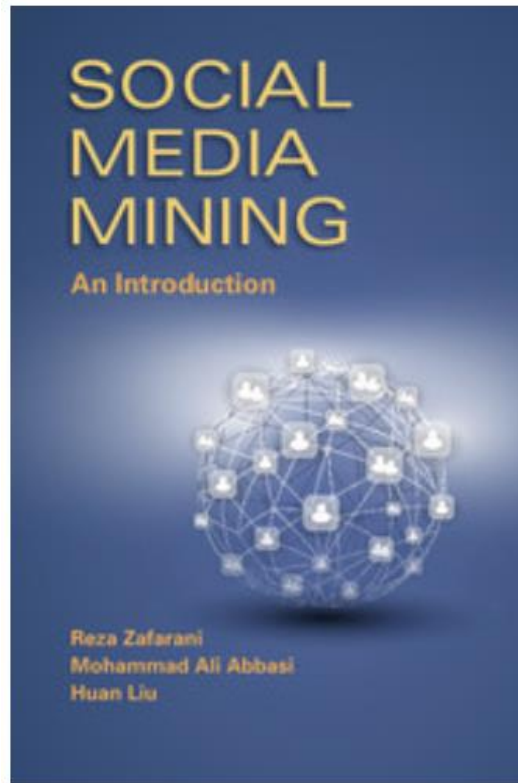
Data cleaning for HW #2

Updated 32 seconds ago by Cindy Hood

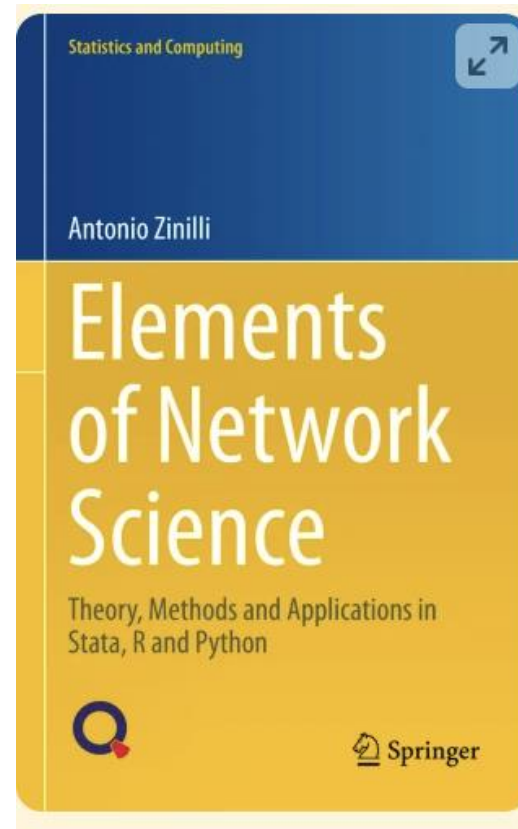
Please feel free to start threads to work on cleaning of student data for hw#2. Substantive contributions will be given extra credit.

hw2

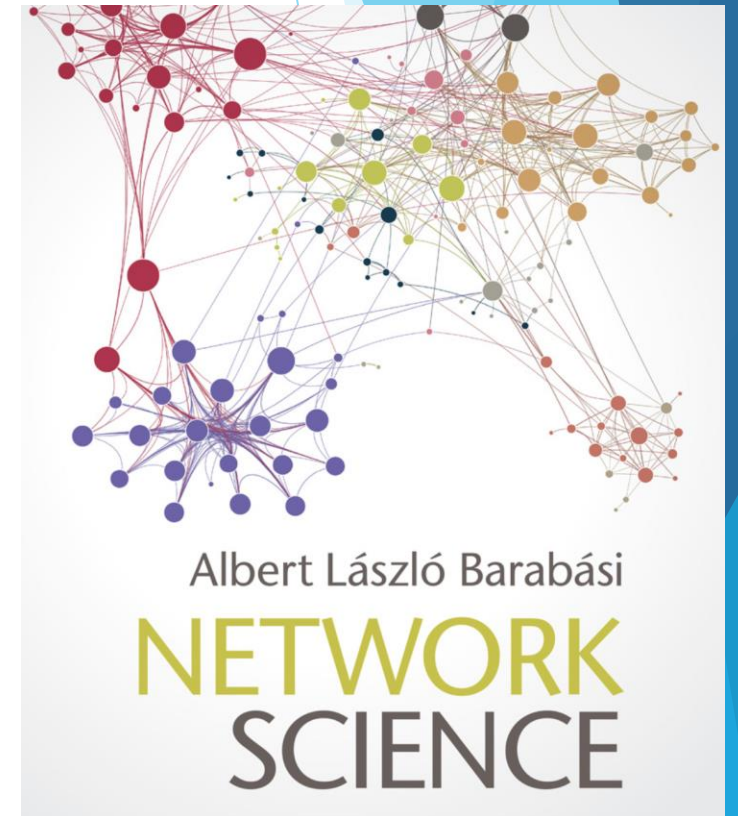
References



<http://www.socialmediamining.info>



<https://link.springer.com/book/10.1007/978-3-031-84712-7>



<http://networksciencebook.com>

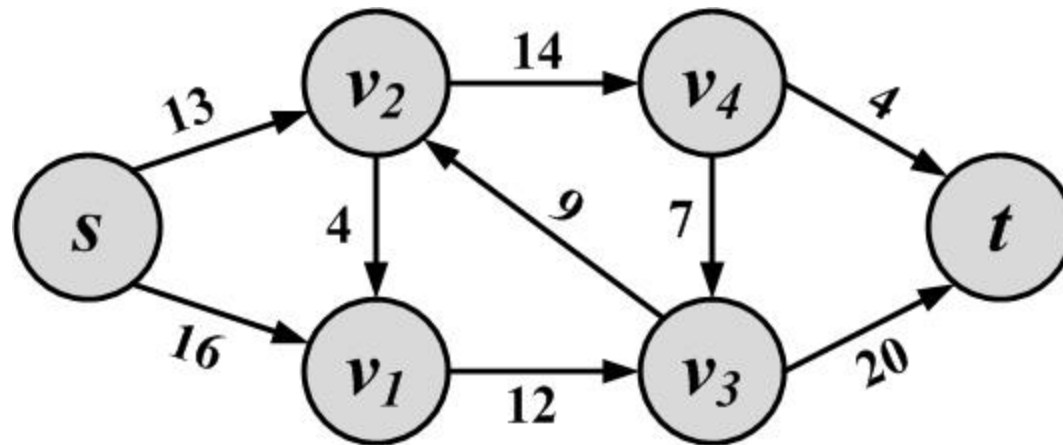
Network Flow

Network Flow

- ▶ Consider a network of pipes that connects an infinite water source to a water sink.
 - ▶ Given the capacity of these pipes, what is the maximum flow that can be sent from the source to the sink?
- ▶ Parallel in Social Media:
 - ▶ Users have daily cognitive/time limits (the capacity, here) of sending messages (the flow) to others,
 - ▶ What is the maximum number of messages the network should be prepared to handle at any time?

Flow Network

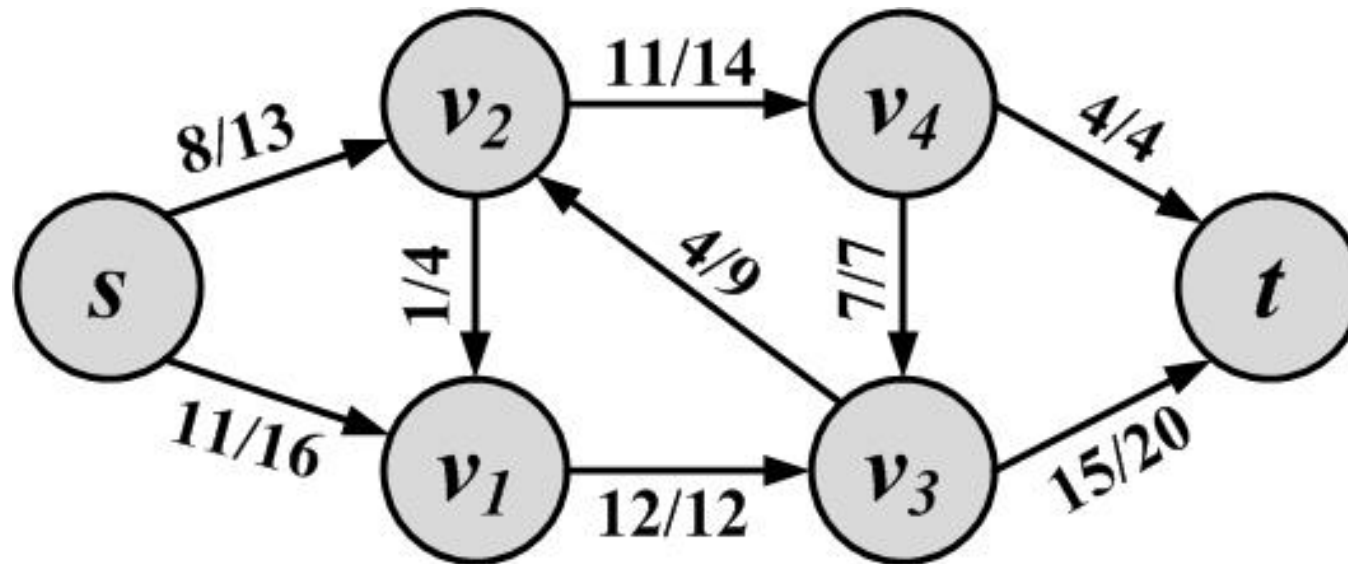
- ▶ A Flow network $G(V,E,C)$ is a directed weighted graph, where we have the following:
 - ▶ $\forall (u,v) \in E, c(u,v) \geq 0$ defines the edge capacity.
 - ▶ When $(u,v) \in E, (v,u) \notin E$ (opposite flow is impossible)
 - ▶ s defines the source node and t defines the sink node. An infinite supply of flow is connected to the source.



- ▶ Given edges with certain capacities, we can fill these edges with the flow up to their capacities (*capacity constraint*)
- ▶ The flow that enters any node other than source s and sink t is equal to the flow that exits it so that no flow is lost (*flow conservation constraint*)
- ▶ $\forall (u, v) \in E, f(u, v) \geq 0$ defines the flow passing through the edge.
- ▶ $\forall (u, v) \in E, 0 \leq f(u, v) \leq c(u, v)$ (**capacity constraint**)
- ▶ $\forall v \in V - \{s, t\}, \sum_{k:(k,v) \in E} f(k, v) = \sum_{l:(v,l) \in E} f(v, l)$
(**flow conservation constraint**)

A Sample Flow Network

- Commonly, to visualize an edge with capacity c and flow f , we use the notation f/c .

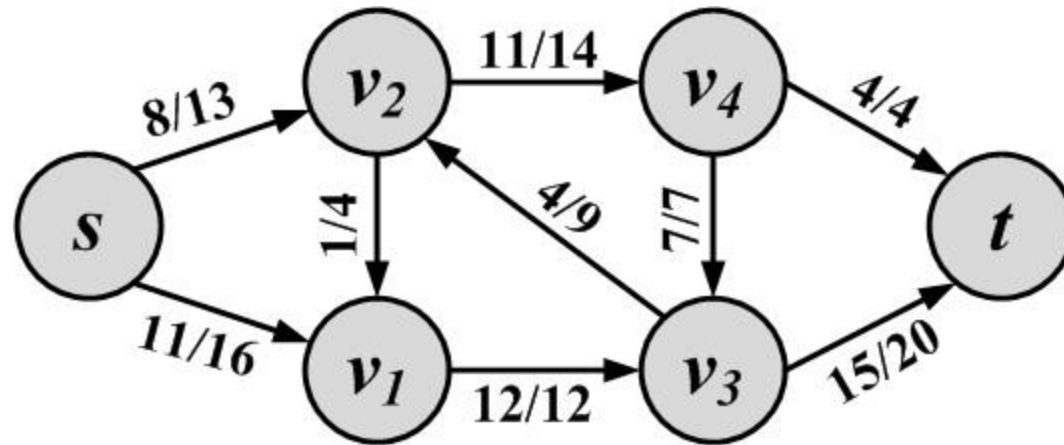


Flow Quantity

- ▶ The flow quantity (or value of the flow) in any network is the amount of
 - ▶ Outgoing flow from the source minus the incoming flow to the source.
 - ▶ Alternatively, one can compute this value by subtracting the outgoing flow from the sink from its incoming value

$$flow = \sum_v f(s, v) - \sum_v f(v, s) = \sum_v f(v, t) - \sum_v f(t, v)$$

What is the flow value?



► **19**

► **11+8** from **s**, or

► **4+15** to **t**

Ford-Fulkerson Algorithm

- ▶ Find a path from source to sink such that there is unused capacity for all edges in the path.
- ▶ Use that capacity (the minimum capacity unused among all edges on the path) to increase the flow.
- ▶ Iterate until no other path is available.

Residual Network

- ▶ Given a flow network $G(V, E, C)$, we define another network $G(V, E_R, C_R)$
- ▶ This network defines how much capacity remains in the original network.
- ▶ The residual network has an edge between nodes u and v if and only if either (u, v) or (v, u) exists in the original graph.
 - ▶ If one of these two exists in the original network, we would have **two** edges in the residual network: one from (u, v) and one from (v, u) .

Intuition

- ▶ When there is no flow going through an edge in the original network, a flow of as much as the capacity of the edge remains in the residual.
- ▶ In the residual network, one has the ability to send flow in the opposite direction to cancel some amount of flow in the original network.

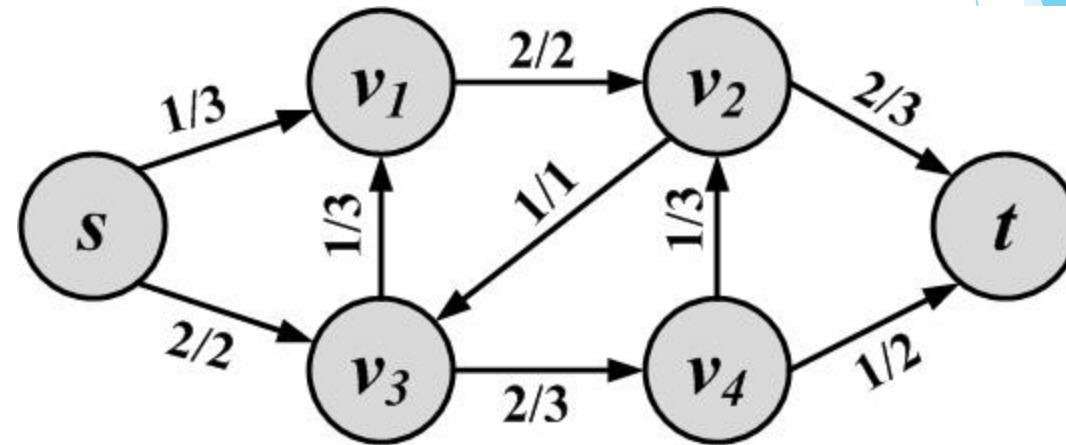
$$c_R(u, v) = \begin{cases} c(u, v) - f(u, v) & \text{if } (u, v) \in E \\ f(v, u) & \text{if } (u, v) \notin E \end{cases}$$

Residual Network (Example)

- ▶ Edges that have zero capacity in the residual are not shown

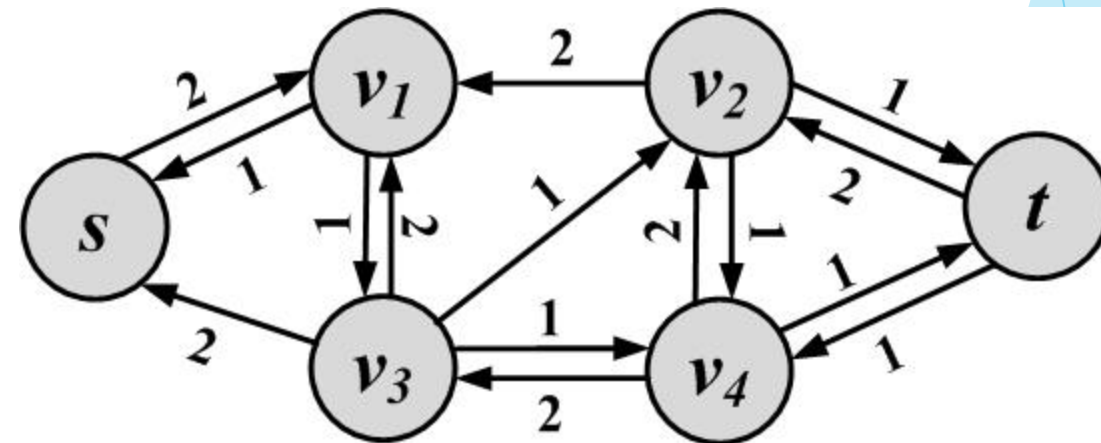
Original Flow Network

Edge labels = flow/capacity



Residual Network

- Edges labeled with available capacity
 - Capacity in the original direction (capacity-flow)
 - Capacity in the reverse direction (flow)
- When capacity=0, no edge



Augmentation / Augmenting Paths

1. In the residual graph, when edges are in the same direction as the original graph,
 - ▶ Their capacity shows how much **more** flow can be pushed along that edge in the **original** graph.
 2. When edges are in the opposite direction,
 - ▶ their capacities show how much flow can be **pushed back** on the **original graph edge**.
- ▶ By finding a flow in the residual, we can **augment** the flow in the original graph.

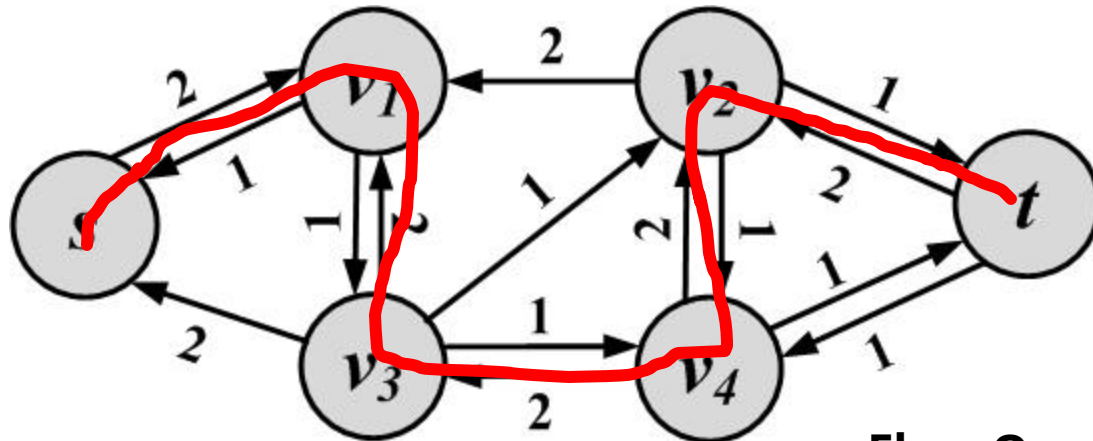
Augmentation / Augmenting Paths

- ▶ Any simple path from s to t in the residual graph is an *augmenting path*.
- ▶ **All capacities in the residual are positive,**
 - ▶ These paths can augment flows in the original, thus increasing the flow.
- ▶ The amount of flow that can be pushed along this path is equal to the **minimum capacity** along the path
 - ▶ The edge with the minimum capacity limits the amount of flow being pushed
 - ▶ We call the edge the **Weak link**

How do we augment?

- ▶ Given flow $f(u, v)$ in the original graph and flow $f_R(u, v)$ and $f_R(v, u)$ in the residual graph, we can augment the flow as follows:

$$f_{augmented}(u, v) = f(u, v) + f_R(u, v) - f_R(v, u)$$

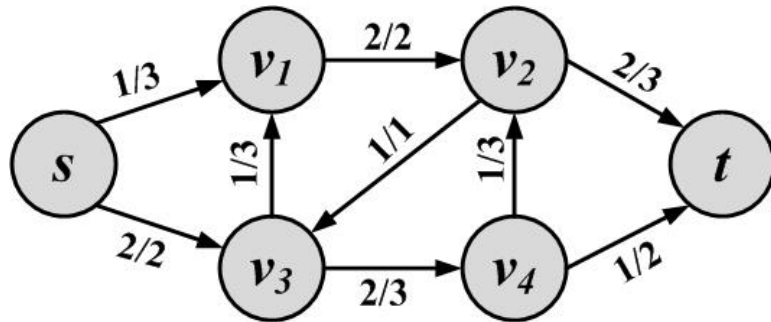


Note that when just looking at the residual network alone, you can't discern between flow $f_R(u, v)$ and reverse $f_R(v, u)$ edge direction

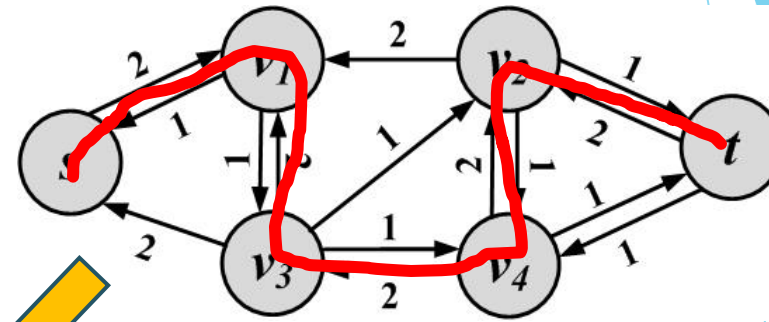
Flow Quantity: 1

Augmenting

Flow Network

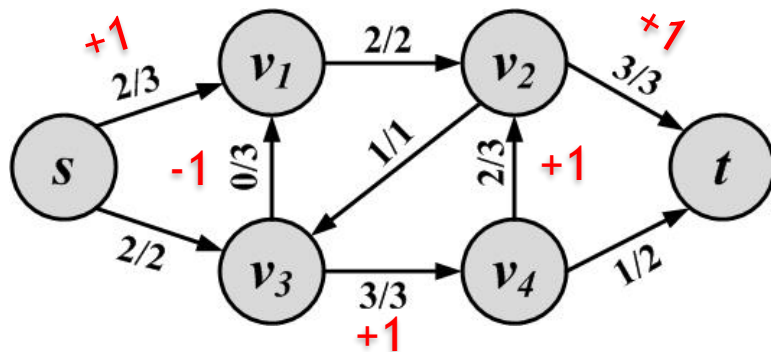


Residual Network



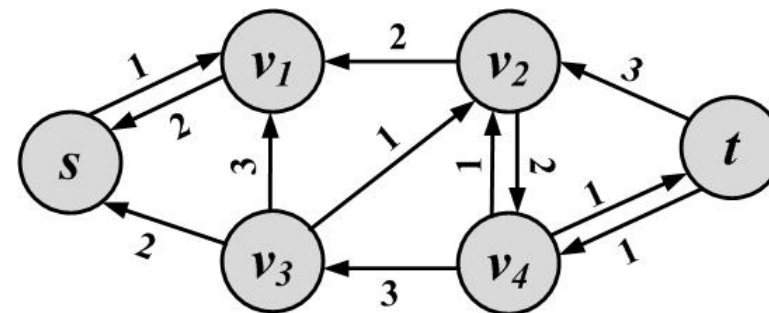
Augmenting Path Flow = 1

New Flow Network



New Flow Network = Old Flow Network + Augmenting Path Flow (1 in this case)

New Residual Network



No augmenting path

The Ford-Fulkerson Algorithm

Algorithm 2.6 Ford-Fulkerson Algorithm

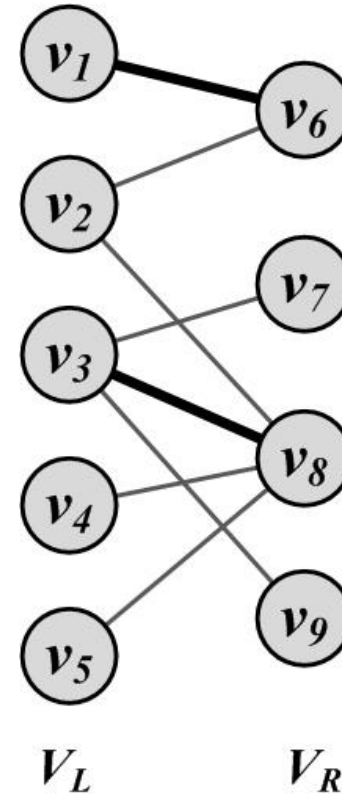
Require: Connected weighted graph $G(V, E, W)$, Source s , Sink t

- 1: **return** A Maximum flow graph
 - 2: $\forall (u, v) \in E, f(u, v) = 0$
 - 3: **while** there exists an augmenting path p in the residual graph G_R **do**
 - 4: Augment flows by p
 - 5: **end while**
 - 6: Return flow value and flow graph;
-

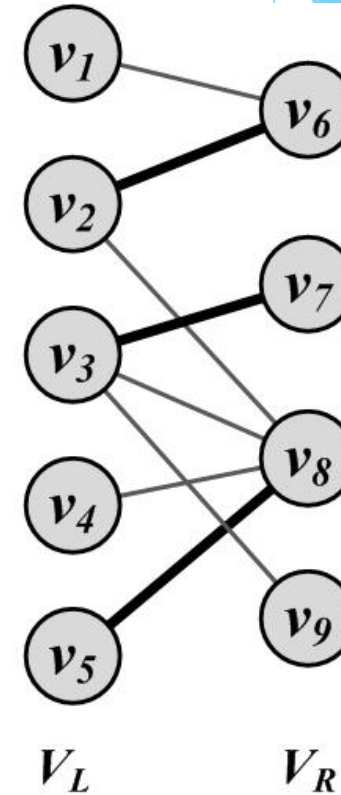
Maximum Bipartite Matching

Example

- ▶ Given n products and m users
 - ▶ Some users are only interested in certain products
 - ▶ We have only one copy of each product.
 - ▶ Can be represented as a bipartite graph
 - ▶ Find the maximum number of products that can be bought by users
 - ▶ No two edges selected share a node



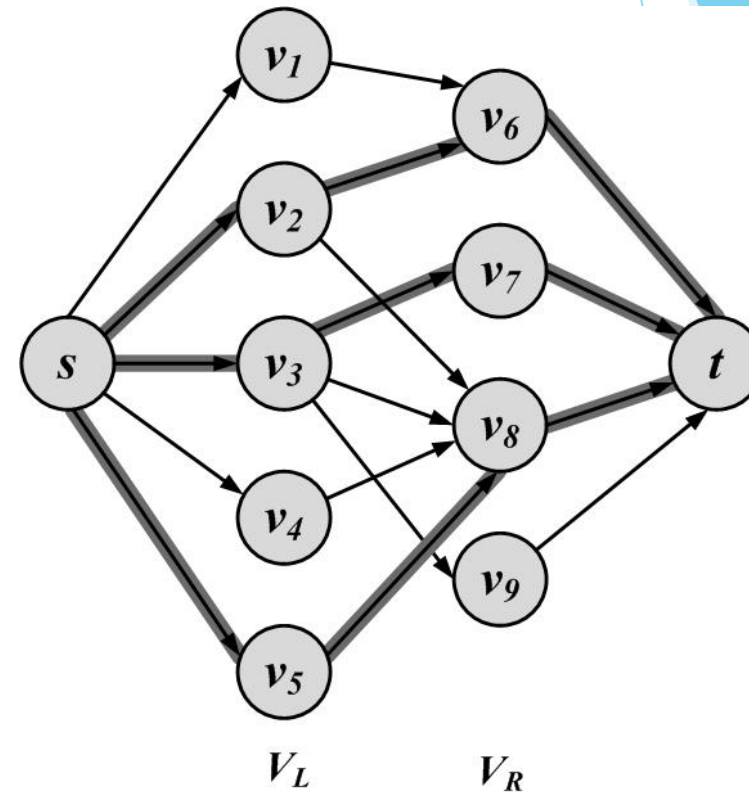
Matching



Maximum Matching

Matching Solved with Max-Flow

- ▶ Create a flow graph $G(V', E', C)$ from our bipartite graph $G(V, E)$
 1. Set $V' = V \cup \{s\} \cup \{t\}$
 2. Connect all nodes in V_L to s and all nodes in V_R to t
 3. Set $c(u, v) = 1$, for all edges in E'



Ch 3 (in Social Media Mining) - Network Measures/Metrics

Recall

- ▶ A ***graph*** is a way of specifying relationships amongst a collection of items
- ▶ A graph consists of a set of objects called ***nodes***
- ▶ Nodes may be connected by links called ***edges***
 - ▶ ***Undirected edges*** indicate a symmetric relationship
 - ▶ ***Directed edges*** indicate the direction of the relationship between the nodes

Ego Networks

- ▶ Ego networks focus on an *individual and his/her connections*
- ▶ An egocentric network contains only those people directly connected to an individual

Degree of Egocentric Networks

- ▶ Egocentric networks can extend out any number of degrees from ego
 - ▶ 1-degree network consists of ego and the direct connections or alters
 - ▶ 1.5-degree network also includes the connections between the alters
 - ▶ 2-degree network includes all of the alters own alters or connections
 - ▶ Friends of friends
 - ▶ Etc.

Complete Network

- ▶ A complete or full network includes all the people of interest and the connections amongst them
 - ▶ All egos are treated equally
- ▶ In all but very limited small contexts
 - ▶ Complete network information may not be available or easily collected
- ▶ Even if data is available, may not be feasible to analyze due to size


How to analyze?

- ▶ Typically analyze part of network
 - ▶ Random sample
 - ▶ Particular slice

Unimodal networks

- ▶ If all of the vertices in a given network are the same kind of entity, it is a *unimodal* network

Multimodal Networks

- ▶ Networks that include different types of vertices are *multimodal* networks
- ▶ For example
 - ▶ Network may connect students to classes, classes to projects and students to projects
- ▶ Most social media networks are complex multimodal networks
 - ▶ For example
 -  ▶ Facebook friends, posts, pictures, tags, likes, etc.
- ▶ **Typically must simplify or transform into a unimodal network to analyze**
 - ▶ Most network measures are designed for unimodal networks

Network Measures

- ▶ Once network is defined, many standard network metrics that can be used to *quantify* networks
 - ▶ Compare networks
 - ▶ Track changes over time
 - ▶ Determine relative position of individuals
 - ▶ Determine relative position of clusters
- ▶ Provide valuable information ***focused on relationships***
- ▶ Use as a feature(s) for data mining/ML

Types of Metrics

- ▶ Aggregate
- ▶ Vertex-specific
- ▶ Clustering and Community Detection
- ▶ Structures, Network Motifs (subgraphs or patterns) and Social Roles

Aggregate Network Metrics

- ▶ Calculated over entire network
- ▶ In multi-component networks, aggregate network metrics may be calculated over component

Density

- ▶ Characterizes the interconnectedness of the network
 - ▶ $\text{Number of edges} / \text{Total number of possible edges}$

Centralization

- ▶ Characterizes the amount to which a network is centered around a few important nodes
 - ▶ Centralized
 - ▶ Decentralized

Some other aggregate metrics

- ▶ # Vertices
- ▶ # Unique edges
- ▶ # Edges with duplicates
- ▶ Total # edges
- ▶ # Self-loops
- ▶ # Connected components
- ▶ # Single vertex components

More aggregate network metrics

- ▶ Maximum vertices in a connected component
- ▶ Maximum edges in a connected component
- ▶ Maximum geodesic distance (diameter)
- ▶ Average geodesic distance
 - ▶ How close are network members?

Vertex-Specific Metrics

- ▶ Used to identify individual's position in a network
- ▶ Centrality key concept
 - ▶ How important is a vertex within a network?

Degree Centrality

- ▶ Total number of connections a vertex has
 - ▶ Degree of vertex
 - ▶ Total number of edges connected to a vertex
- ▶ Directed network
 - ▶ In-degree
 - ▶ Out-degree
- ▶ Can be considered a popularity measure
 - ▶ Is it a good popularity measure?

Betweenness Centrality

- ▶ How important is a node in the communication of others in the network?
- ▶ Geodesic distance
 - ▶ Shortest path between two nodes
- ▶ Betweenness centrality is a measure of how often a given vertex lies on the shortest path between two nodes
- ▶ Ranges from 0-1
 - ▶ 0 means vertex not on any shortest path
 - ▶ 1 means vertex on all shortest paths

Betweenness Centrality

Another way of looking at centrality is by considering how important nodes are in connecting other nodes



Linton Freeman

$$C_b(v_i) = \sum_{s \neq t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}}$$

σ_{st} The number of shortest paths from vertex s to t – a.k.a. **information pathways**

$\sigma_{st}(v_i)$ The number of **shortest paths** from s to t that pass through v_i

Normalizing Betweenness Centrality

- ▶ In the best case, node v_i is on all shortest paths from s to t , hence,

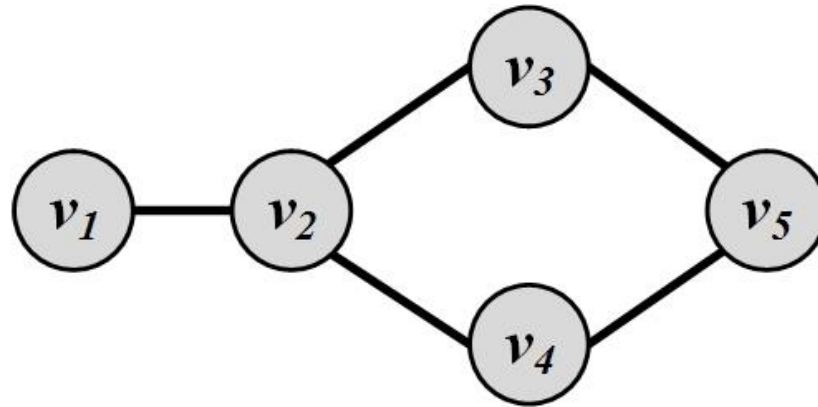
$$\frac{\sigma_{st}(v_i)}{\sigma_{st}} = 1$$

$$\begin{aligned} C_b(v_i) &= \sum_{s \neq t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}} \\ &= \sum_{s \neq t \neq v_i} 1 = 2 \binom{n-1}{2} = (n-1)(n-2) \end{aligned}$$

Therefore, the maximum value is $(n-1)(n-2)$

Betweenness centrality: $C_b^{\text{norm}}(v_i) = \frac{C_b(v_i)}{2 \binom{n-1}{2}}$

Betweenness Centrality: Example 1



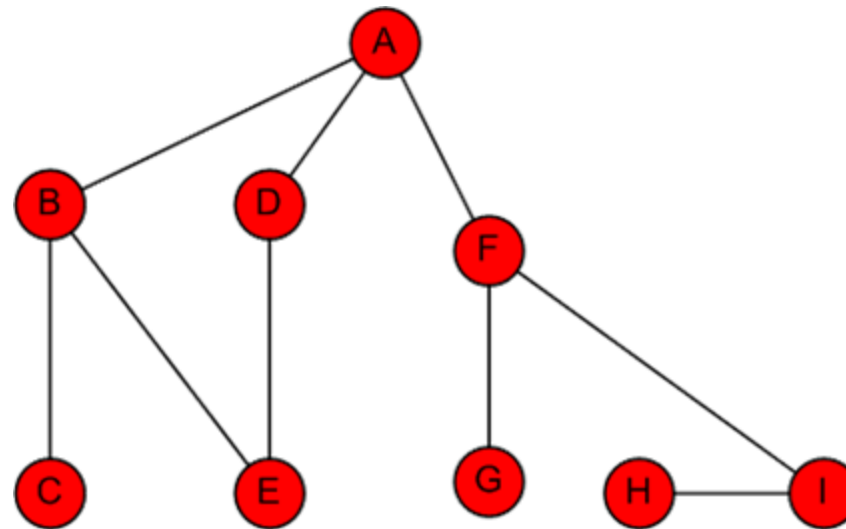
$$C_b(v_2) = 2 \times \left(\underbrace{(1/1)}_{s=v_1, t=v_3} + \underbrace{(1/1)}_{s=v_1, t=v_4} + \underbrace{(2/2)}_{s=v_1, t=v_5} + \underbrace{(1/2)}_{s=v_3, t=v_4} + \underbrace{0}_{s=v_3, t=v_5} + \underbrace{0}_{s=v_4, t=v_5} \right) \\ = 2 \times 3.5 = 7,$$

$$C_b(v_3) = 2 \times \left(\underbrace{0}_{s=v_1, t=v_2} + \underbrace{0}_{s=v_1, t=v_4} + \underbrace{(1/2)}_{s=v_1, t=v_5} + \underbrace{0}_{s=v_2, t=v_4} + \underbrace{(1/2)}_{s=v_2, t=v_5} + \underbrace{0}_{s=v_4, t=v_5} \right) \\ = 2 \times 1.0 = 2,$$

$$C_b(v_4) = C_b(v_3) = 2 \times 1.0 = 2,$$

$$C_b(v_5) = 2 \times \left(\underbrace{0}_{s=v_1, t=v_2} + \underbrace{0}_{s=v_1, t=v_3} + \underbrace{0}_{s=v_1, t=v_4} + \underbrace{0}_{s=v_2, t=v_3} + \underbrace{0}_{s=v_2, t=v_4} + \underbrace{(1/2)}_{s=v_3, t=v_4} \right) \\ = 2 \times 0.5 = 1,$$

Betweenness Centrality: Example 2



Node	Betweenness Centrality	Rank
A	$16 + 1/2 + 1/2$	1
B	$7 + 5/2$	3
C	0	7
D	$5/2$	5
E	$1/2 + 1/2$	6
F	$15 + 2$	1
G	0	7
H	0	7
I	7	4

Computing Betweenness

- ▶ In betweenness centrality, we compute shortest paths between all pairs of nodes to compute the betweenness value.
- ▶ **Trivial Solution:**
 - ▶ Use Dijkstra and run it $O(n)$ times
 - ▶ We get an $O(n^3)$ solution
- ▶ **Better Solution:**
 - ▶ Brandes Algorithm:
 - ▶ $O(nm)$ for unweighted graphs
 - ▶ $O(nm + n^2 \log n)$ for weighted graphs

Brandes Algorithm [2001]

$$C_b(v_i) = \sum_{s \neq t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}} = \sum_{s \neq v_i} \delta_s(v_i)$$

$$\delta_s(v_i) = \sum_{t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}}$$

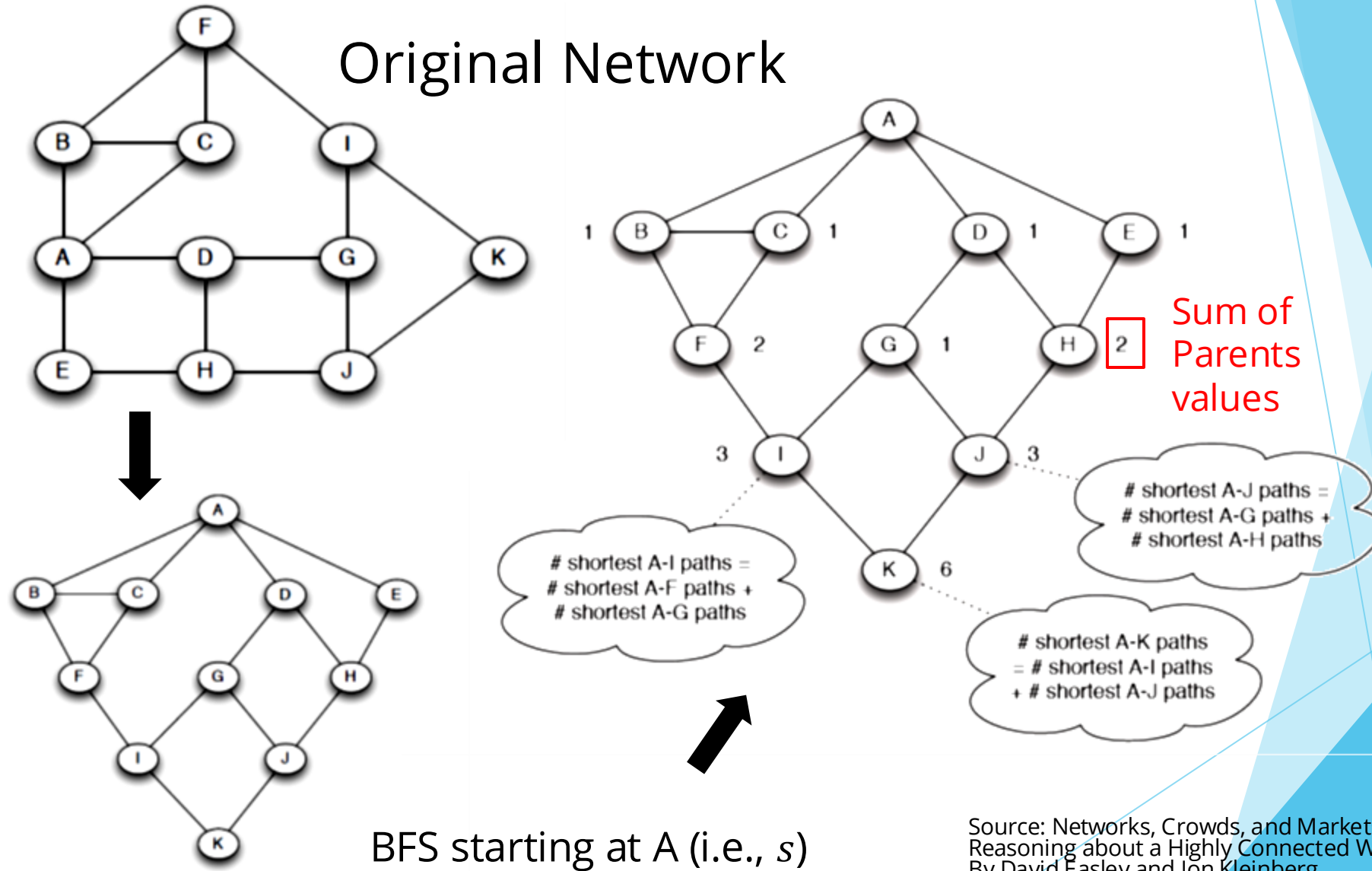
There exists a recurrence equation that can help us determine $\delta_s(v_i)$

$$\delta_s(v_i) = \sum_{w: v_i \in \text{pred}(s, w)} \frac{\sigma_{sv_i}}{\sigma_{sw}} (1 + \delta_s(w))$$

$\text{pred}(s, w)$ is the set of predecessors of w in the shortest paths from s to w .

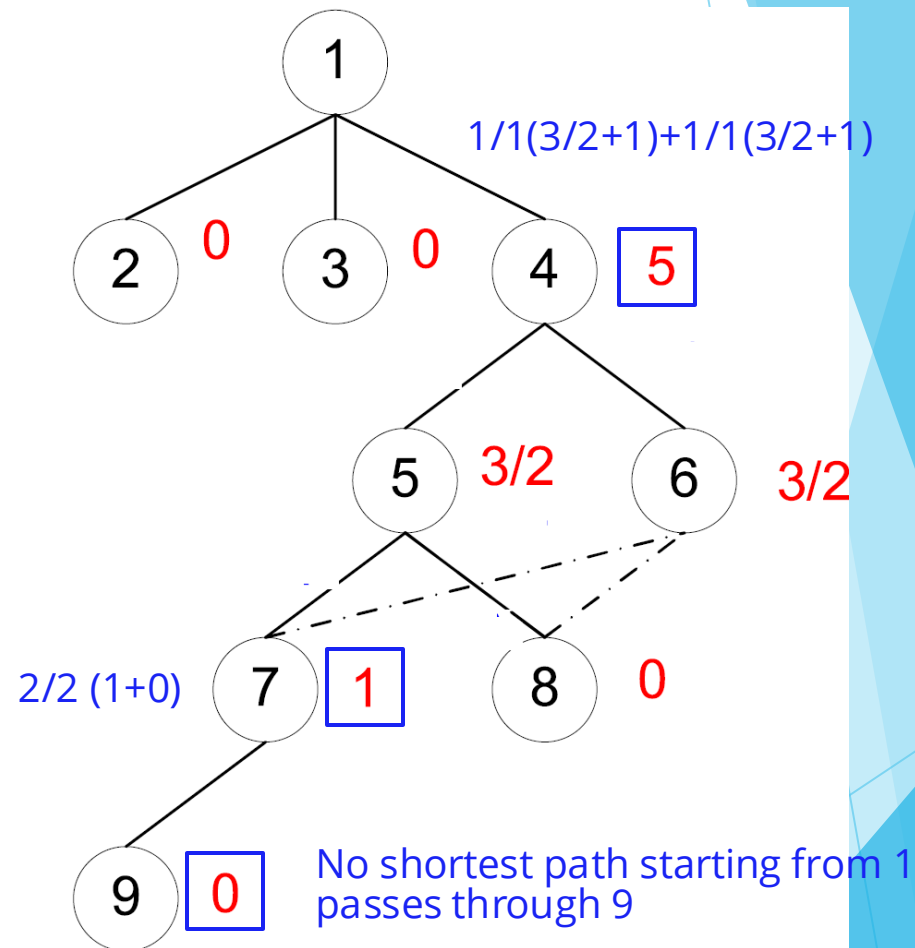
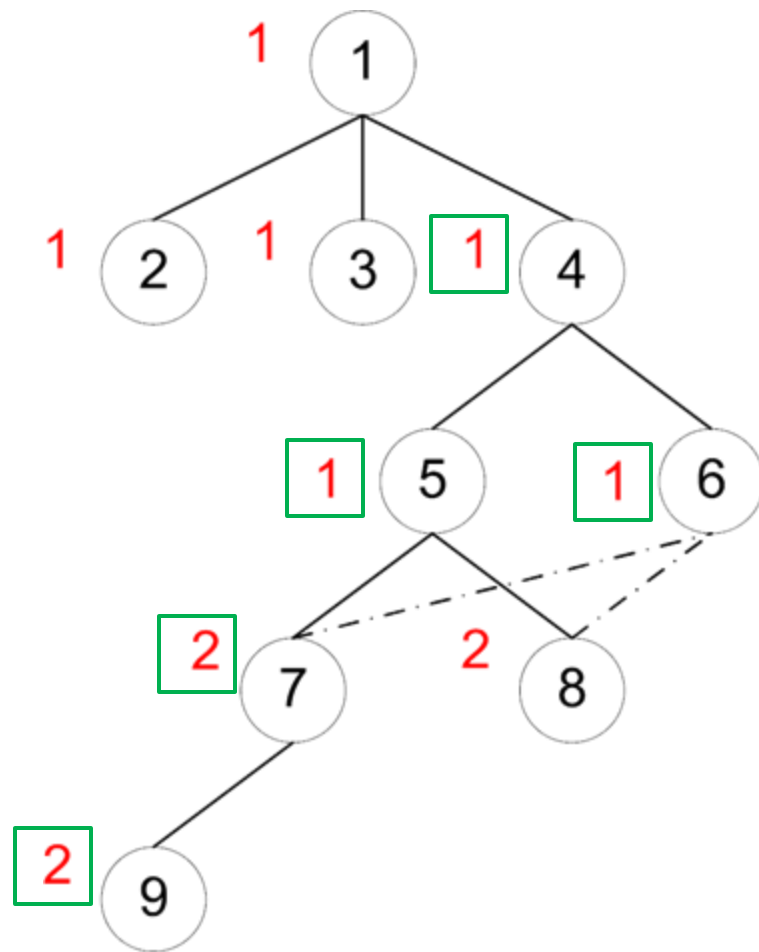
- In the most basic scenario, w is the immediate child of v_i

How to compute σ_{st}



Source: Networks, Crowds, and Markets:
Reasoning about a Highly Connected World.
By David Easley and Jon Kleinberg

How do you compute $\delta_s(v_i)$



$$\delta_s(v_i) = \sum_{w: v_i \in \text{pred}(s, w)} \frac{\sigma_{sv_i}}{\sigma_{sw}} (1 + \delta_s(w))$$