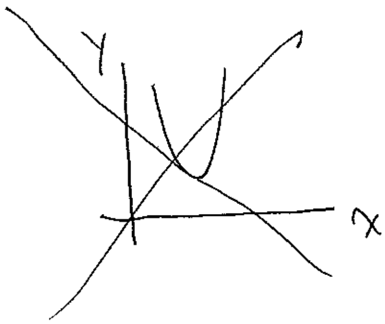
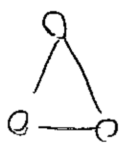
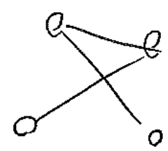
What is a graph?

Notes: if more than 2 subsets, it's called a hypergraph.

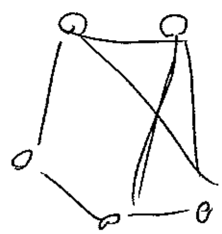




order 3
size 3

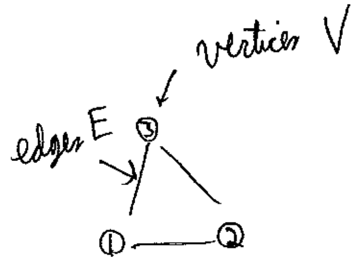


order 4
size 3



A graph G is an ordered pair $G = (V, E)$ where V is a (finite) set of elements and E is a set of 2 subsets of V
pairs

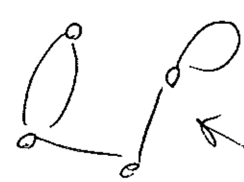
V is the set of vertices
E is the set of edges



G

$$V = \{1, 2, 3\} \qquad E = \{\{1,2\}, \{2,3\}, \{3,1\}\}$$

Simple Graphs: no loops
no multiple edges

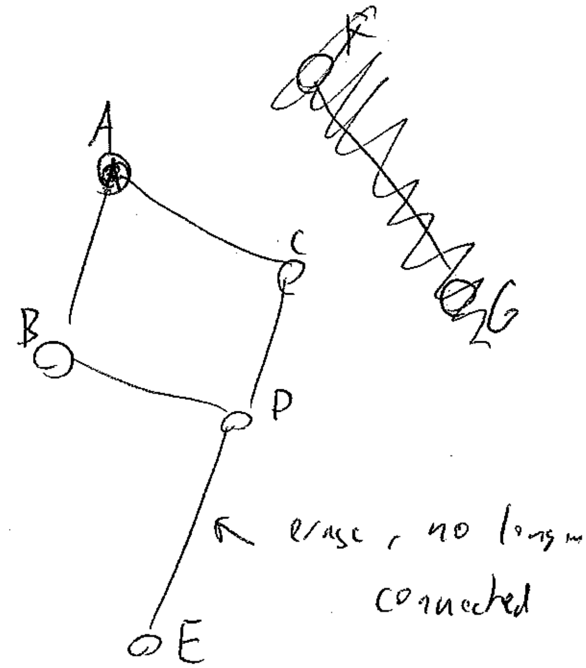


not simple

$|V| = $ order

$|E| = $ size

Terminology → write on board first before lecture.

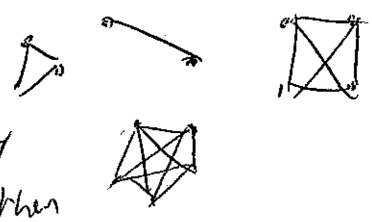


Neighbor – nodes connected to A

Degree – # of neighbors
in directed
indegrees & outdegrees

Path – a series of nodes and the edges that connect them, no repeating nodes

Complete – edge from every node to every other node



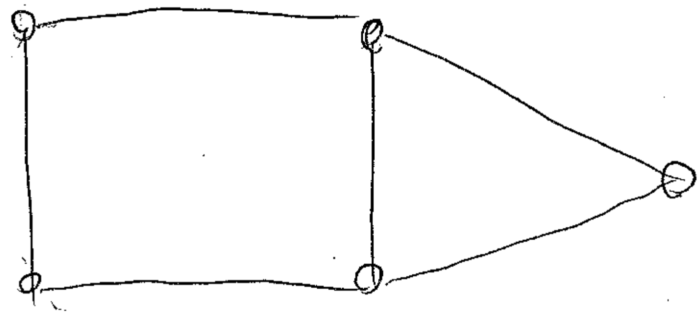
Connected – path from every node to every other node

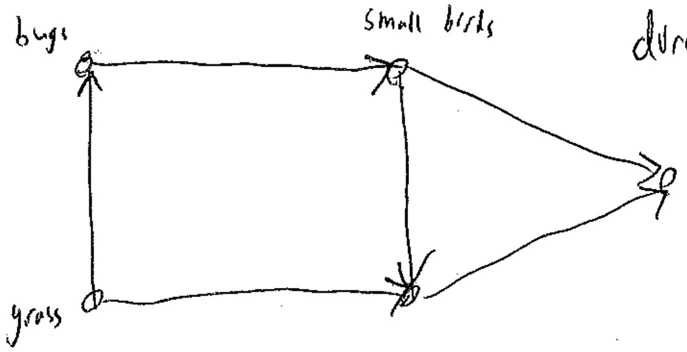
Conn. Component – subgraph that is connected

Subgraph – subset of nodes & their edges

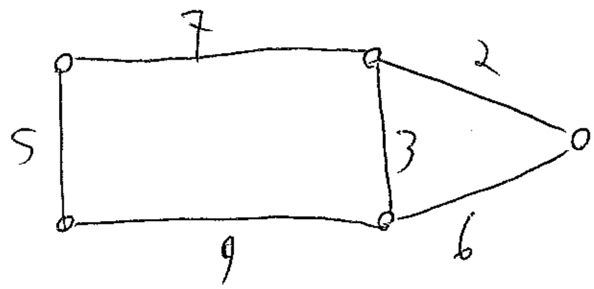
Types of Graphs

undirected - eg. social networks, facebooks, linkedin
if you are friends w/ someone
they're friends w/ you.





directed - direction matters
example - food webs, twitter
followers, email
phone calls
- flights



weighted - cost to the edges

ex. road networks - A, GHG, time

directed weighted graph → who owes who money?

more abstract weights → social pressure

Where did graph theory come from?..

- Explain Bridges of Koenigsberg 7 bridges, wanted only once

Questions we can answer:

shortest path b/w 2 nodes

~~shortest~~

How to find missing edges? → who might be your Fb friend you didn't add yet

social influencers

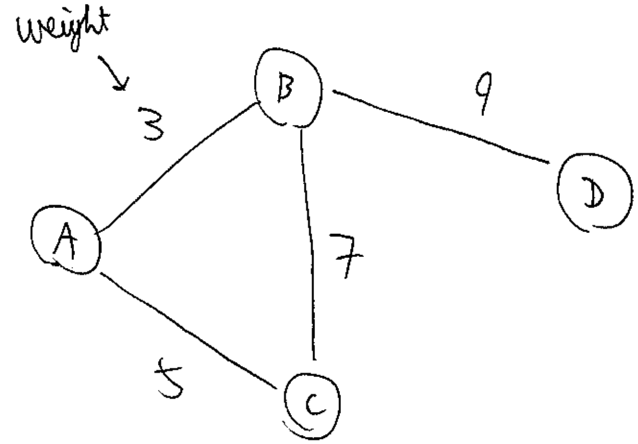
Searching - ways to traverse the nodes on a graph

- Finding shortest path → bike route
  collaboration
  clicks from websites

- find an extended network
  - interview w/ Yahoo CEO

Edge List

(A, B, 3)
(A, C, 5)
(B, C, 7)
(B, D, 9)



Adjacency List – list of lists corresponding to a vertex & containing a list of edges

A: (B, 3)(C, 5)
B: (A, 3)(C, 7)(D, 9)
C: (A, 5)(B, 7)
D: (B, 9)

Adjacency Matrix – a matrix of bits where each element corresponds to ...

{Before filling in ∞'s. explain why with Bay bridge example}

	A	B	C	D
A	0	1	1	0
B	1	0	1	1
C	1	1	0	0
D	0	1	0	0

	A	B	C	D
A	0	3	5	∞
B	3	0	7	9
C	5	7	0	∞
D	∞	9	∞	0

undirected is symmetric

Travel time OAK to SF

Travel time Berkeley to SF, building a bridge that goes straight there → would take a long time → ∞

Space / Time Complexity of Adj Lists / Matrices

again $G = (V, E)$ ✱ Big O notation to come in future lectures

Discuss Sparse v. Dense

Adj List: Space = List of vertices + list of edges

$\Theta(|V| + |E|)$ $\approx V + E$ or $V \cdot E$

Adj. Matrix: Space = $|V| \times |V|$ matrix

$\Theta(|V|^2)$ $\approx V^2$

Adj. List: Lookup = Go to the node, then look through its neighbors. The max # of edges is $(V-1)$

$\Theta(|V|)$ $\approx V$

Adj. Matrix: Lookup = Go straight to the table & pick it out

$\Theta(1)$ ≈ 1

Adj List: Neighbors = go to point & count the neighbors

$\Theta(\text{\# of neighbors})$

Adj. Matrix: Neighbors = go across the whole list of vertices and see which are 1's $\Theta(V)$

Adj List: add vertex = add each edge $\Theta(e_n)$ e_n = new edges

Adj Matrix: add vertex = add 1 column & 1 row

$\Theta(\cancel{2}|V|)$

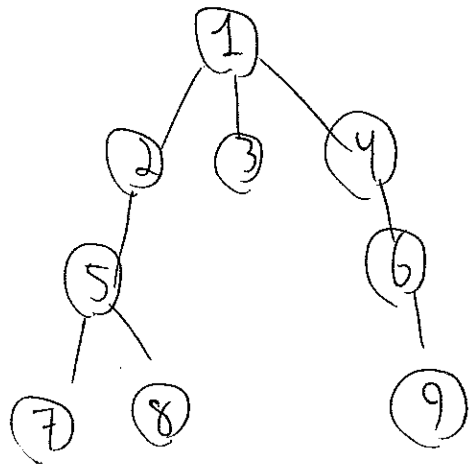
Queue data structure

Work as FIFO

popping from beginning of list is slow b/c it has to shift all the other elements by 1

BFS - Breadth First Search

- starting at a given node, find all the neighbors, find the neighbors of those neighbors & the neighbors of those neighbors ...
- works in a FIFO queue

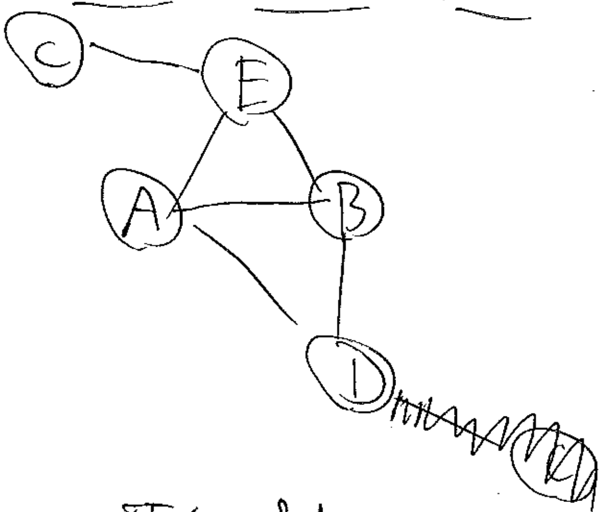


Uses for BFS: - find the shortest path from A to B
- find all friends of friends

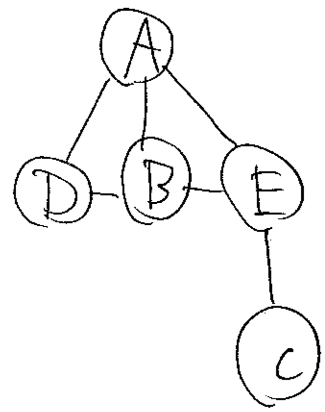
Basic BFS pseudocode

```
                  end
BFS(graph, start):
① Create an empty queue Q
② initialize empty set V (visited nodes set)
③ add A to Q  ⟶  add (A,0) to Q
④ While Q is not empty:
⑤     take node off Q, call it N ⟶ has a distance, d
⑥     If N not in V: (haven't visited already)
⑦         add N to V
⑧         add every neighbor of N to Q
⑨             ⟶ if N is desired end node:
⑩                    done
              ⟶ else:
                  add every neighbor of N to Q with
                                        distance d+1
```

BFS Shortest Path - Worked



define
isomorphic
→



Discuss BFS
Why BFS over DFS
When DFS

Visited	By
A	–
B	A
C	E
D	A
E	A

Find Shortest Path from A → C

Node (N)	Queue (Q)	Visited (V)	current distance, d	
			0	init
–	(A,0)	–		
A,0	–	–	0	take a off queue
A,0	(D,1)(B,1)(E,1)	A	1	add neighbors to Q
(D,1)	(B,1)(E,1)	A	1	
start cold calc (D,1)	(B,1)(E,1)(B,2)(A,2)	A, D	2	
(B,1)	(E,1)(B,2)(A,2)	A, D	2	
(B,1)	(E,1)(B,2)(A,2)(D,2)(A,2)(E,2)	A, D, B	2	
(E,1)	(B,2)(A,2)(D,2)(A,2)(E,2)	A, D, B, ~~E~~	2	
(E,1)	(B,2)(A,2)(D,2)(A,2)(E,2)(C,2)(B,2)(A,2)	A, D, B, E	2	
end cold call (B,2)	(A,2)(D,2)(A,2)(E,2)(C,2)(B,2)(A,2)	ADBE	3	
(A,2)	(D,2)(A,2)(E,2)(C,2)(B,2)(A,2)	ADBE	3	
(D,2)	(A,2)(E,2)(C,2)(B,2)(A,2)	ADBE	3	
(A,2)	(E,2)(C,2)(B,2)(A,2)	ADBE		
(E,2)	(C,2)(B,2)(A,2)	ADBE		
(C,2) → done		ADBEC		

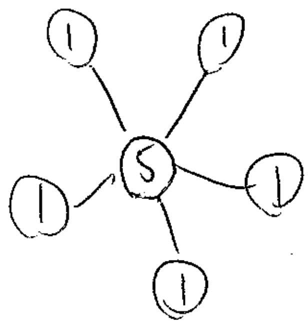
How important is a given node?

Centrality - find the ~~center of~~ most influential people in a social network

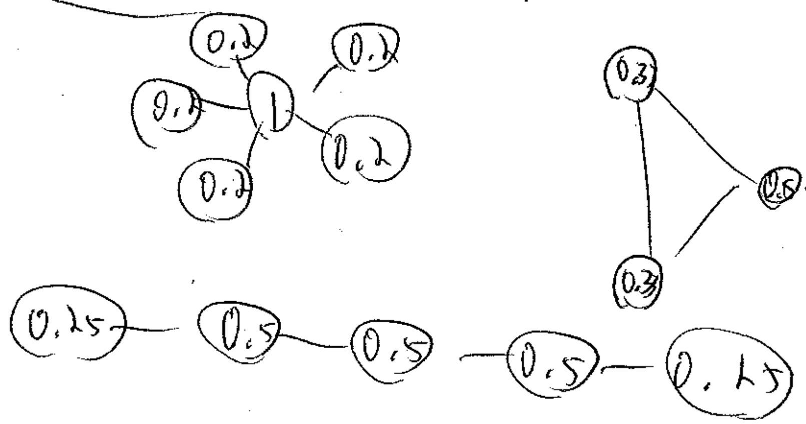
- understand how to help disseminate info
- stop epidemics
- protect the network infrastructure

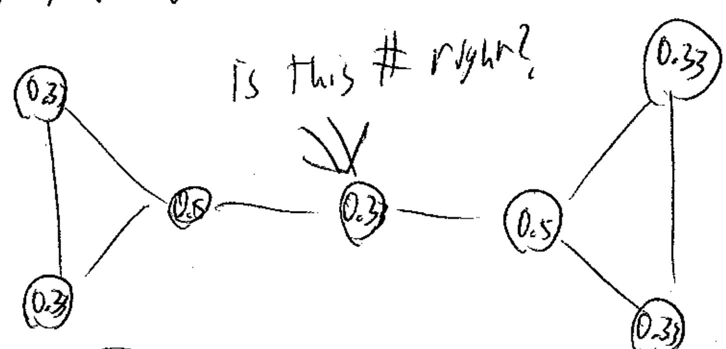
Degree Centrality - # of ~~conn~~ connections

- good for people to have a beer with
- close friends to help you move



Normalizing - divide by max # of nodes (N-1)





when is degree nonideal?

brokering b/w groups

likelihood of info spreading through the network

Betweenness Centrality

How many pairs would you need to go through to reach another in a min. # of hops

$$C_B(i) = \sum_{i \neq j \neq k} g_{jk}(i) / g_{jk}$$

$g_{jk}(i)$ = # of shortest paths connecting jk passing through i

g_{jk} = total # of shortest paths

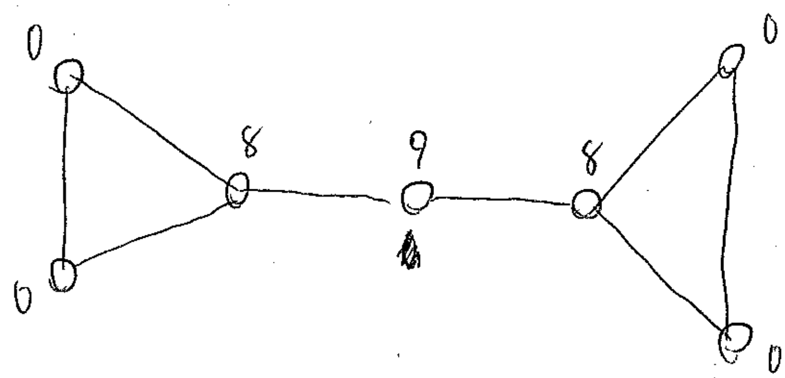
Normalized - $C'_B(i) = C_B(i) / [(n-1)(n-2)/2]$

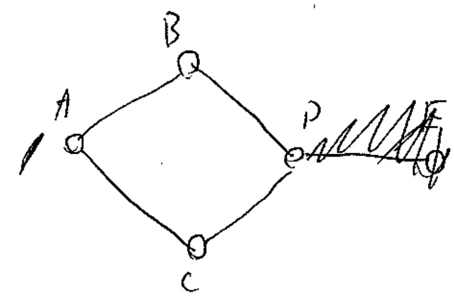
# of pairs of vertices excluding the vertex itself

the /2 isn't needed in directed graphs

~~Jeff Tay example incorrect?~~

Work on example below





$$\frac{(7-1)(7-2)}{2} = 15$$

↑ degree ↓ b/w redundant connections

↓ degree ↑ b/w few ties are crucial

Eigenvector Centrality

- Degree Centrality is focused on how many connections a node has. but are these connections connected?
- a central node should be connected to other central nodes

writes on board

- What makes a node central? Doesn't necessitate a high degree on that specific node.
- linkedin endorsement example - hire someone with 200 endorsements from random people or someone endorsed by Sergey Brin & Elon Musk?

- Eig Centrality is a more generalized form of degree centrality where we ~~...~~ account for ~~...~~ neighbors
- To keep track of these neighbors, we'll use an adj. matrix of Graph
- we want the centrality of v_i to be f(neighbors Ce) We say its proportional to the summation of their centralities

$$\underbrace{c_e(v_i)}_{\text{eig cent. of vertex } v_i} = \underbrace{\frac{1}{\lambda}}_{\text{Fixed constant}} \sum_{j=1}^{n} \underbrace{A_{j,i}}_{\text{adj. matrix}} \underbrace{Ce(v_j)}_{\text{neighbors Ce}}$$

$$\mathbb{C}_e = (Ce(v_1), Ce(v_2) \ldots Ce(v_n))^T$$

$$\lambda \mathbb{C}_e = A^T \mathbb{C}_e \Rightarrow \text{eig eq}^n \quad \lambda V = AV$$

So, ~~we call~~ $\mathbb{C}_e$ an eigenvector of matrix A & λ is the corresponding eigenvalue

Matrices have many eigenvectors, which one do we pick?

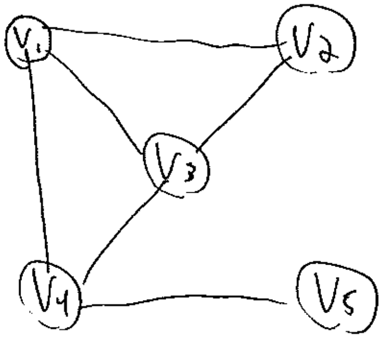
For comparison, we like + values.

Perron-Frobenius Theorem + $n \times n$ matrix, there is a λ_{max} > all others & an eigenvector associated w/ it when all the values are +

Local Vs. Global Centrality

local ↓ global

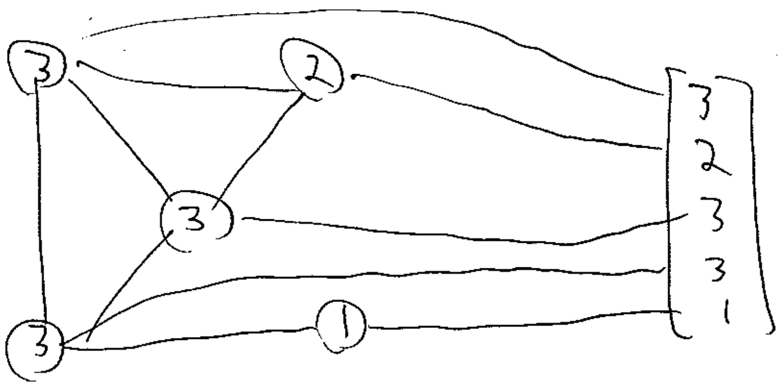
- Degree - super local
- Betweenness - somewhat local
- Eigenvector - ~~it~~ an



why don't we need to fill the other part?

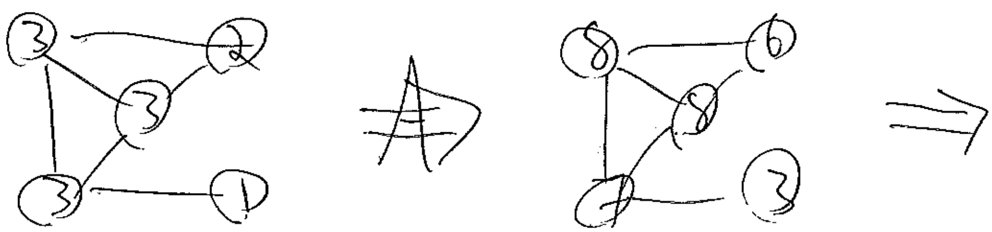
$$\begin{bmatrix} - & 1 & 1 & 1 & 0 \\ & - & 1 & 0 & 0 \\ & & - & 1 & 0 \\ & & & - & 1 \\ & & & & - \end{bmatrix}$$

$A \rightarrow 5\times 5$ adj Matrix

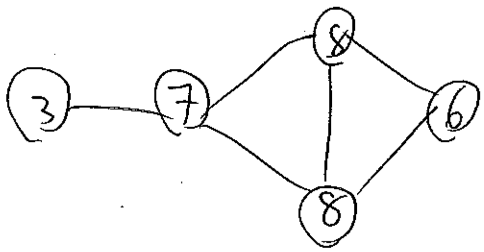


$$Ax = \begin{bmatrix} - & 1 & 1 & 1 & 0 \\ 1 & - & 1 & 0 & 0 \\ 1 & 1 & - & 1 & 0 \\ 1 & 0 & 1 & - & 1 \\ 0 & 0 & 0 & 1 & - \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 3 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0\times 3 + 1\times 2 + 1\times 3 + 1\times 3 + 0\cdot 1 \\ \\ \\ \\ \end{bmatrix} = \begin{bmatrix} 8 \\ 6 \\ 8 \\ 7 \\ 3 \end{bmatrix}$$

The 1's from the adj. Matrix effectively "pick up" the values of each vertex to which the first vertex is connected.
The resulting value is the sum of the values each vertex had.



We "spread out" our degree centrality. Redraw our graph to see better.



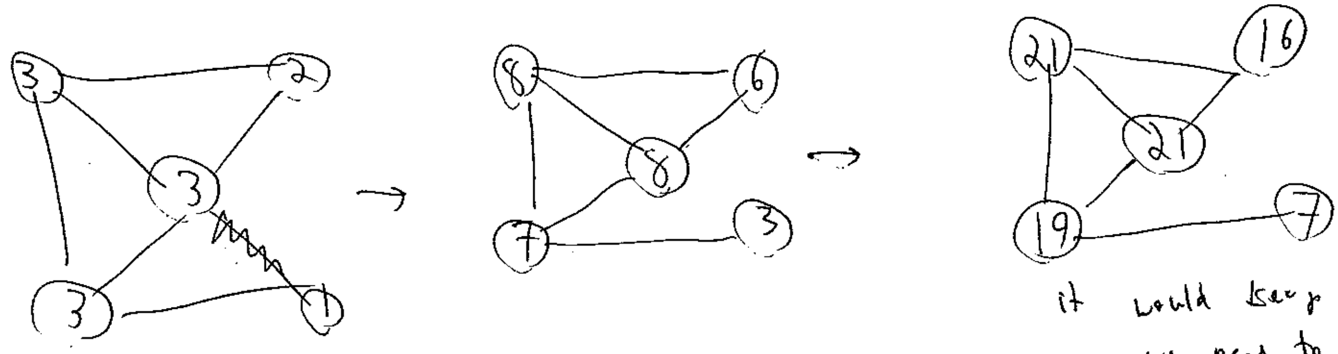
What if we multiplied the resulting vector by A again? What would this do? The spread in both directions (vertices give & get from their neighbors). We eventually reach an equilibrium when the amount coming into a vertex would balance to the amount going to its neighbors. The numbers will always keep increasing, but the share of each node would stabilize.

We can imagine the "centrality-ness" of the graph had equilibrated and the value of each node completely captured the centrality of the neighbors.

~~Let's instead think of our vector x as unknown.~~

$$\cancel{\begin{bmatrix} \quad & \quad \\ & \\ & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}}$$

Eig Centrality Intuition



it would keep increasing, so we need to normalize

Power Iteration

$$b_{k+1} = \frac{Ab_k}{\|Ab_k\|}$$

$$\frac{[21, 16, 21, 19, 7]}{\|[21, 16, 21, 19, 7]\|} = [0.53, 0.41, 0.53, 0.48, 0.18]$$

$$\mu_k = \frac{b_k A b_k}{b_k b_k} = [2.67, 2.62, 2.67, 2.58, 2.71]$$

actual eigenvalue = 2.64

vector = [0.54, 0.41, 0.54, 0.47, 0.18]

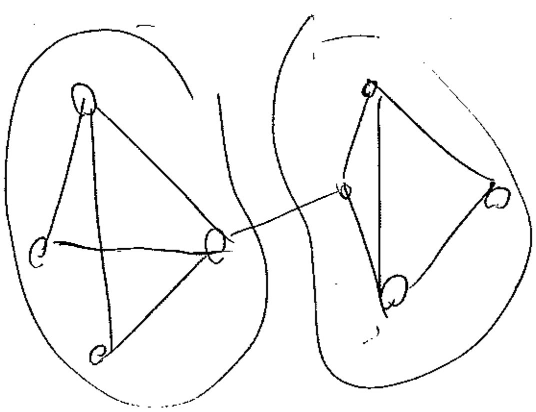
Communities → groups of vertices similar to each other.

What makes a community? (cohesive subgroup)

- Mutual Ties: everyone in the group has ties to one another
- Compactness: can reach other group members in a small # of steps
- Dense Edges: High frequency of edges within the group
- Separation from other Groups: comparing the frequency of edges within this group to people not in the group, it will be higher

Density = # of edges compared to total # of edges possible

Community Detection - assigning vertices to communities



	a	b	c	d	e	f
a	1	1	0	0	0	0
b	1	1	1	0	0	0
c	0	1	1	0	0	0
d	0	0	0	1	1	1
e	0	0	0	1	1	1
f	0	0	0	0	1	1

Modularity → Define it in words, math, then example

- Measure that defines how likely the community structure found is created ~~by~~ by random chance.
- If there is a community structure, it should be far from random.

→ within group connections more dense than b/w group connections

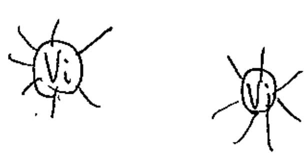
Consider an undirected graph $G(V,E)$

$|E| = m$ let's assume we know the degrees of each node, but not ~~where~~ where the edges connect

Consider 2 nodes V_i & V_j w/ degrees d_i & d_j

What is the expected # of edges b/w these 2 nodes?

consider V_i, for any edge randomly going out of V_i, the probability that the edge goes to V_j is



$$\frac{d_j}{\sum_i d_i} = \frac{d_j}{2m}$$

total ~~nodes~~ in the network degrees

the degree for V_i is d_i so we have d_i chances to hit one of the edges eminating ~~of~~ from V_j

Expected # of edges $= \frac{d_i d_j}{2m}$ → allowing loops & multi-edges

Now, we can calc the expected # of edges b/w any 2 nodes

Modularity $= Q = \sum \left(\begin{array}{c} \text{observed fraction of} \\ \text{links in the group} \end{array} - \begin{array}{c} \text{expected fraction of} \\ \text{links in the group} \end{array} \right)$

$$Q = \frac{1}{2m} \sum_{c \in C} \sum_{i,j \in c} A_{ij} - \frac{d_i d_j}{2m}$$

collection of communities

actual edges

expected edges

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ are neighbors} \\ 0 & \text{if theyre not} \end{cases}$$

$\frac{1}{2m}$ is a normalizing constant

The summation over all edges m and $A_{ij} = A_{ji}$ since all the edges are counted twice

alt Definition of communities

- groups of vertices such that vertices inside the group are connected with many more edges than b/w groups

• graph partitioning Problem

Graph Partitioning is combinatorial

n nodes into 2 groups = $\frac{n!}{n_1! n_2!}$ o o o o | o o o

$B_{20} \approx$ 5 trillion possible partition

⇒ Heuristic approach

~~First~~ Focus on edges that connect communities
- bridges

Edge Betweenness - # of shortest paths going through edge e
- extension of node betweenness

Girvan - Newman, 2004

Algorithm - edge betweenness

For all edges $e \in E$, compute edge betweenness $C_B(e)$:
remove the edge with highest $C_B(e_i)$

until all the edges are gone

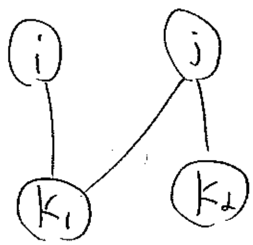
~~also gives a dendrogram~~

Node Similarity

- The number of neighbors that 2 nodes share
- Many shared neighbors → high similarity

$$n_{ij} = \sum_{k} A_{ik} A_{jk}$$

ex.



$A_{ik_1} = 1$ $\quad$ $A_{jk_1} = 1$

$A_{ik_2} = 0$ $\quad$ $A_{jk_2} = 2$

$$n_{ij} = (A_{ik_1} \cdot A_{jk_1}) + (A_{ik_2} \cdot A_{jk_2})$$

$$= (1 \cdot 1) + (0 \cdot 1) = 1$$

- Measure similarity w/ cos similarity
- dissimilarity w/ euclidean distance

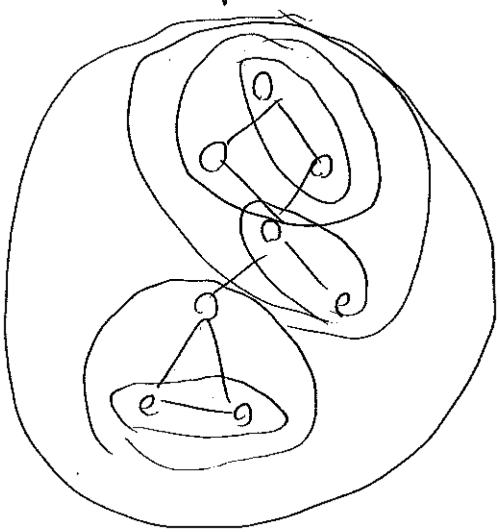
$$K = \frac{n(A \cap B)}{\sqrt{n(A) \times n(B)}}$$ $\quad$ n = number

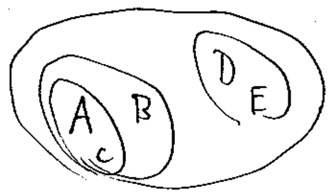
~~da~~

$$d_{ij} = \sum_{k} (A_{ik} - A_{jk})^2$$

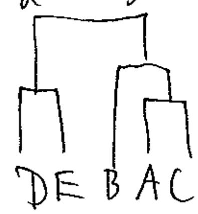
Hierarchical Clustering

- assign each vertex to a group of its own
- find 2 groups w/ highest similarity and join them into 1 group
- calculate similarity b/w groups:
  1) single-linkage (most similar in the group)
  2) complete linkage (least similar in the group)
  3) average-linkage clustering (mean similarity b/w groups)
- Repeat until all in one group





dendrogram



decide when