GRAPHS

Agenda

Morning

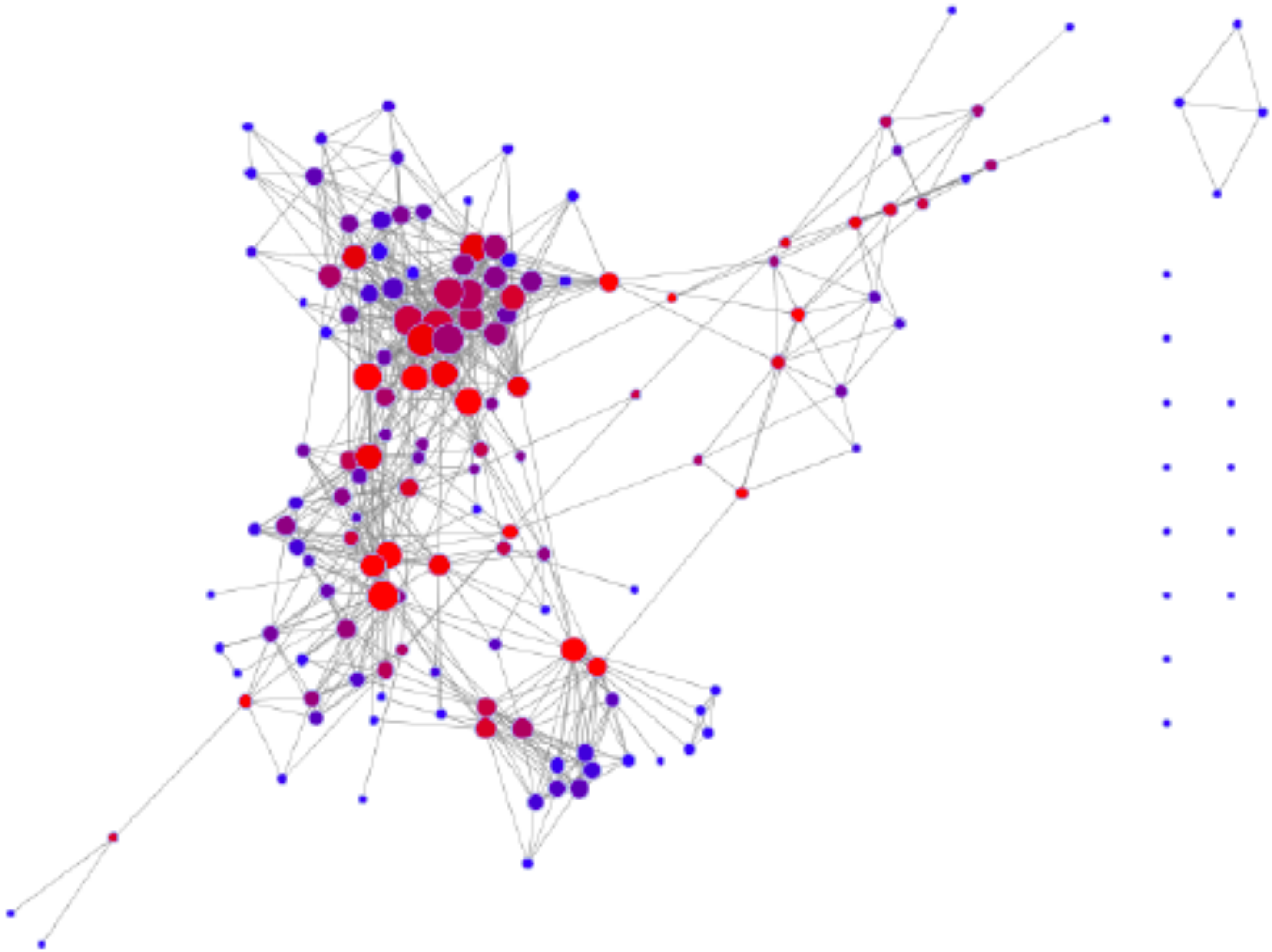
- General understanding of graphs
- Traversing a graph
- Importance of an node

Afternoon

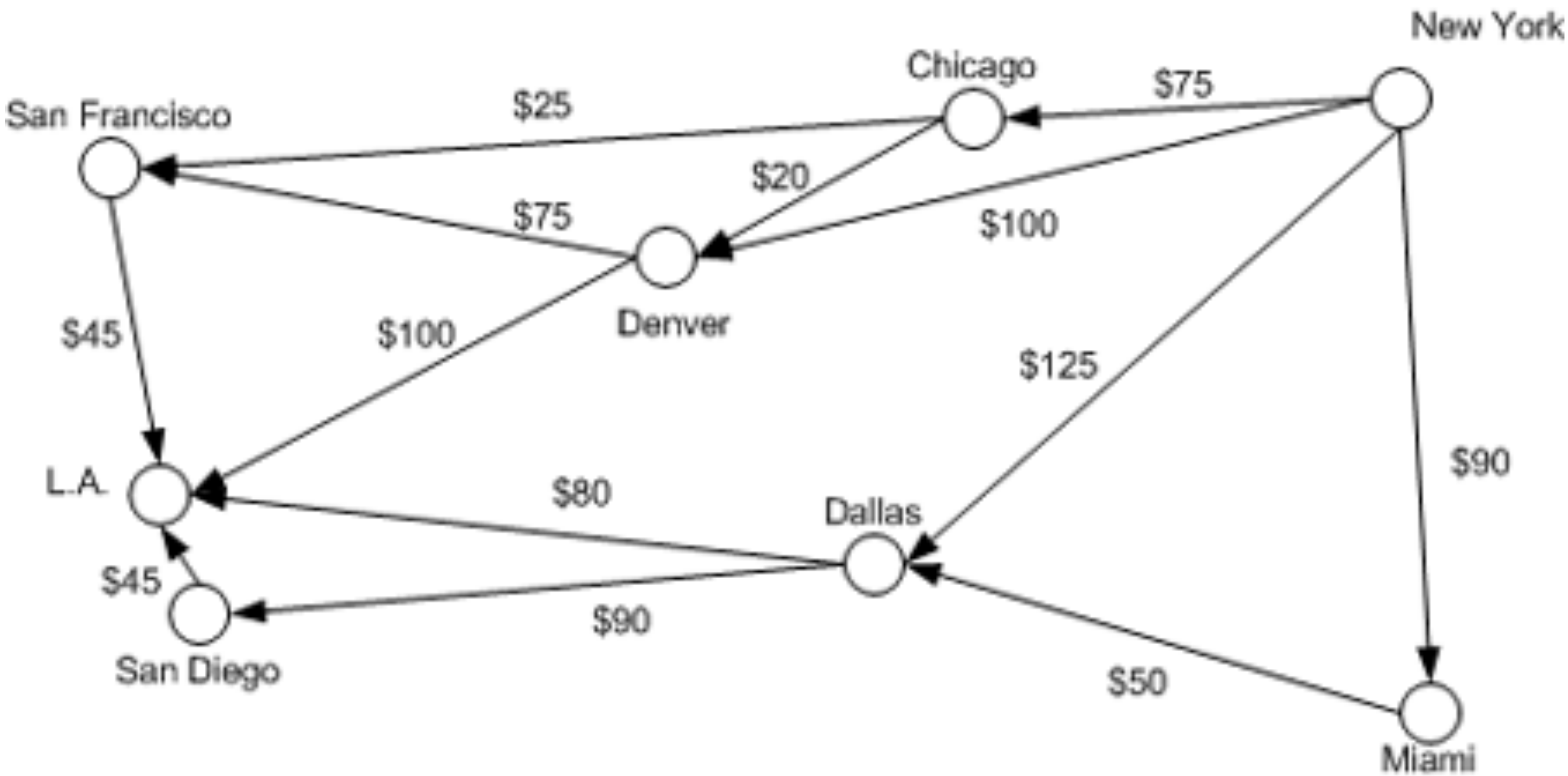
- Defining communities in graphs
- Finding communities

SESSION 1

General understanding of graphs
Traversing a graph
Importance of an node



Real world example



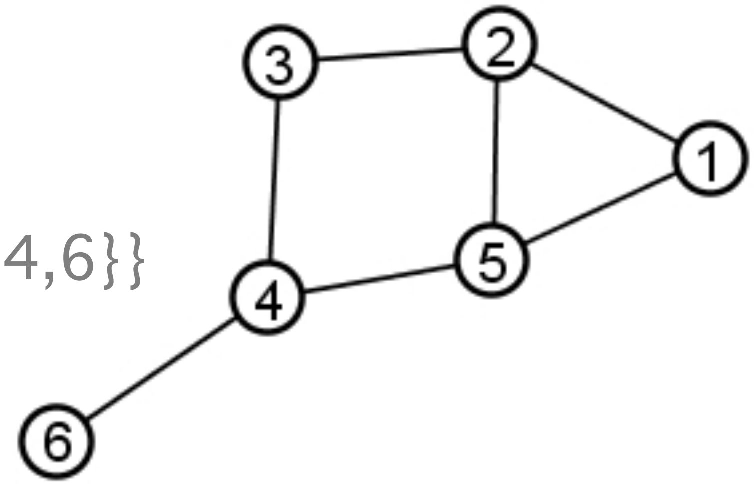
Definition of a graph

A graph is an ordered pair G = (V,E) such that:
- V is a set of vertices
- E is a set of relations

Each edge is 2-element subset {Vi,Vj} of V

V =
{1,2,3,4,5,6}
E =
{{1,2},{1,5},{2,3},{2,5},{3,4},{4,5},{4,6}}

G = (V,E)



A *node* is also known as a vertex
If there's an edge between two nodes, we say that those nodes are *connected*.

Graph Terminology

Neighbors: The *neighbors* of a node are the nodes that it is connected to.

Degree: The *degree* is the number of neighbors a node has.

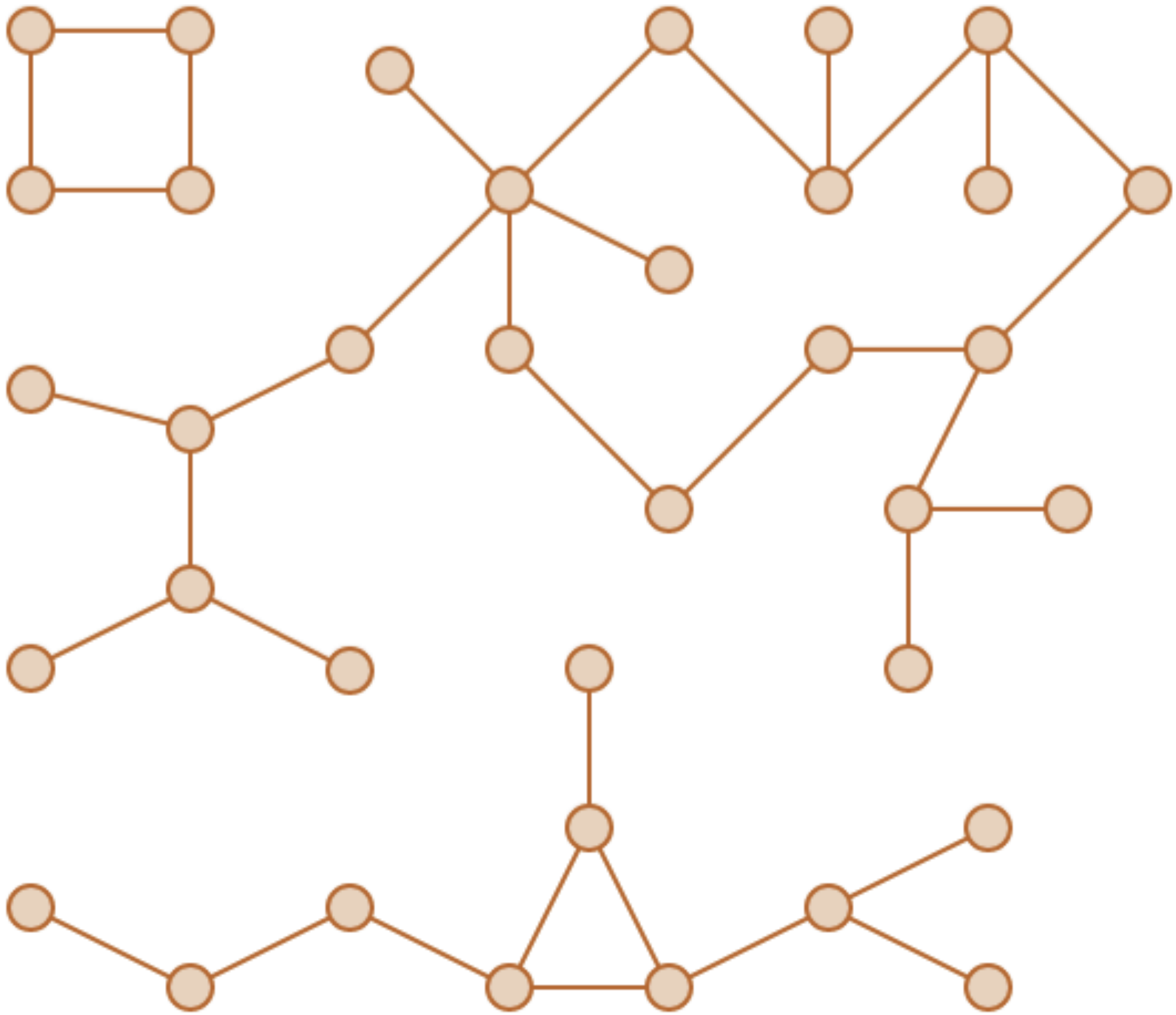
Path: A *path* is a series of nodes and the edges that connect them

Connected: A graph is *connected* if every pair of vertices is connected by some path.

Subgraph: A *subgraph* is a subset of the nodes of a graph and all the edges between them.

Connected Component: A *connected component* is a subgraph that is *connected* and which is connected to no additional vertices in the supergraph.

Graph Terminology – Pop Quiz



More Terminology – types of graphs

Directed Graph: A directed graph is a graph where edges only go in one direction.

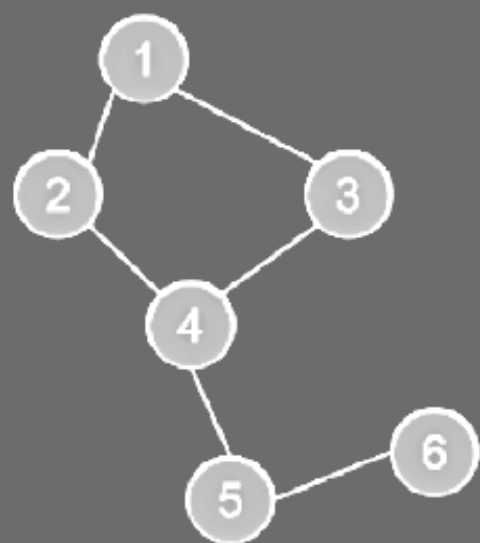
Undirected Graph: In an undirected graph, edges go both ways.

Weighted Graph: In a weighted graph, the edges have weights.

Unweighted Graph: In an unweighted graph, all edges are the same.

Graph Representations

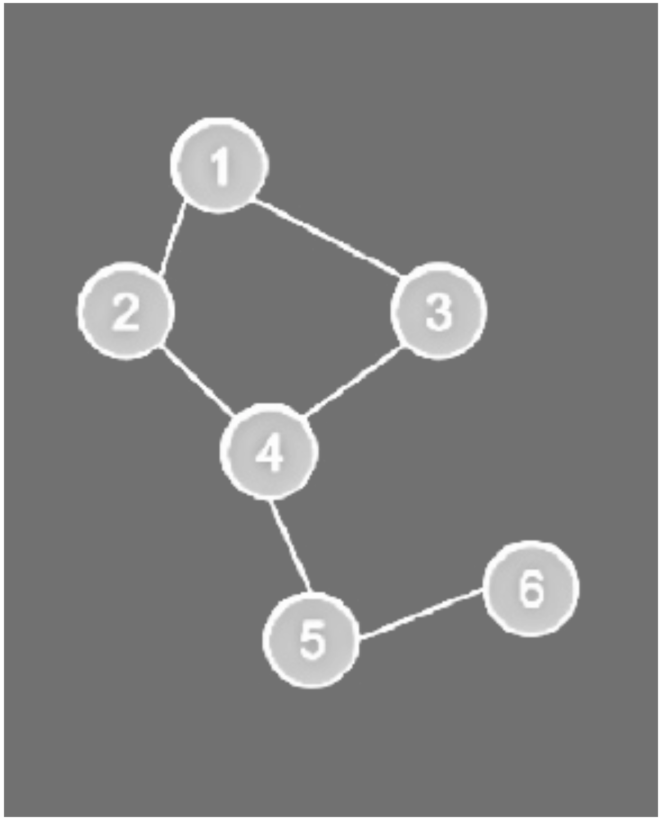
Adjacency Matrix



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

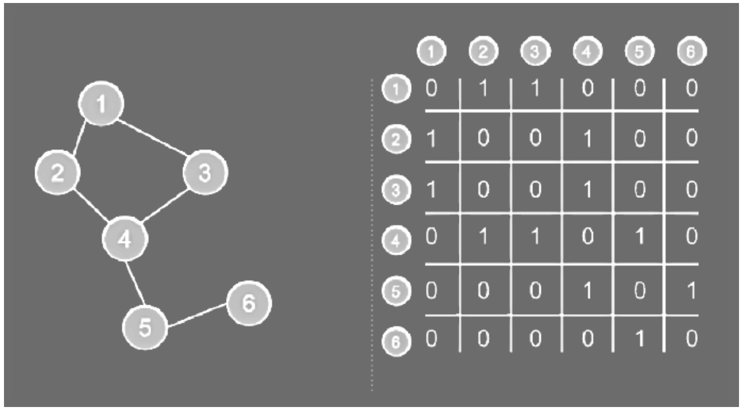
Graph Representations

Adjacency List



{1}: {2, 3}
{2}: {1, 4}
{3}: {1, 4}
{4}: {2,3,5}
{5}: {4,6}
{6}: {5}

Graph Representation



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

{1}: {2, 3}
{2}: {1, 4}
{3}: {1, 4}
{4}: {2,3,5}
{5}: {4,6}
{6}: {5}

Storage:

- Adjacency matrix: $O(|V|^2)$
- Adjacency list: $(O|V|+O|E|)$

Lookup, Finding edge between two vertices:

- Adjacency matrix: O(1), constant time
- Adjacency list: $(O|V|)$ or $(O|1|)$ depending on implementation

Graph Representation, Pop Quiz

Which representation will give you an answer faster if you wanted to check if two nodes were connected?

Which representation will give you an answer faster if you wanted to find the neighbors of a node?

What if your graph had a ton of edges and maybe even loops?

Graph Search Algorithms

- Systematically visit/traverse all nodes
- Useful to perform all sorts of graph related operations

"Breadth First Search" (BFS) is a popular graph search algorithm

Breadth First Search (BFS)

Animation:
https://upload.wikimedia.org/wikipedia/commons/4/46/Animated_BFS.gif

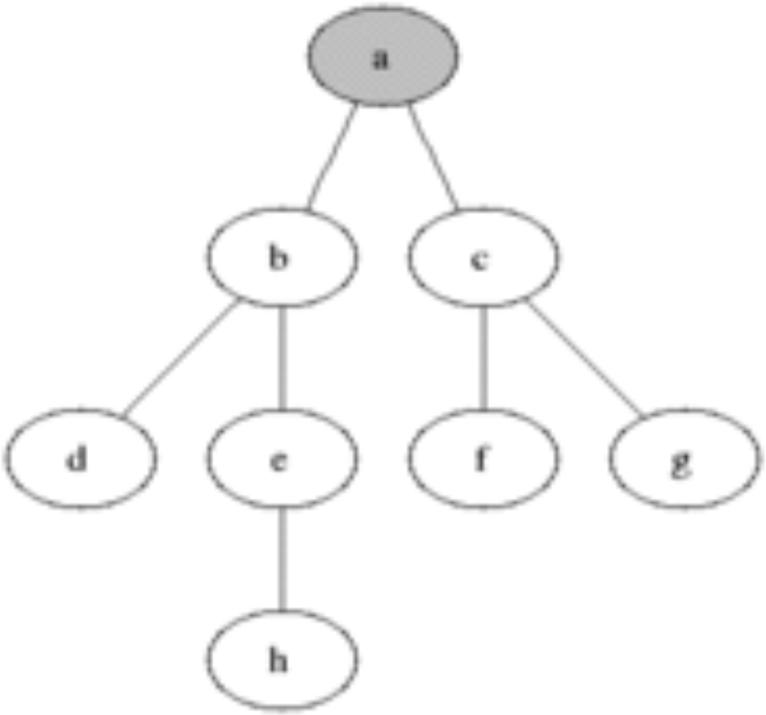
Starting with a given node, find all of that node's neighbors. Then, find all of those neighbors' neighbors, and so on....

Visit ALL neighbors of a node BEFORE visiting neighbors of those neighbors...

BFS, pseudo code

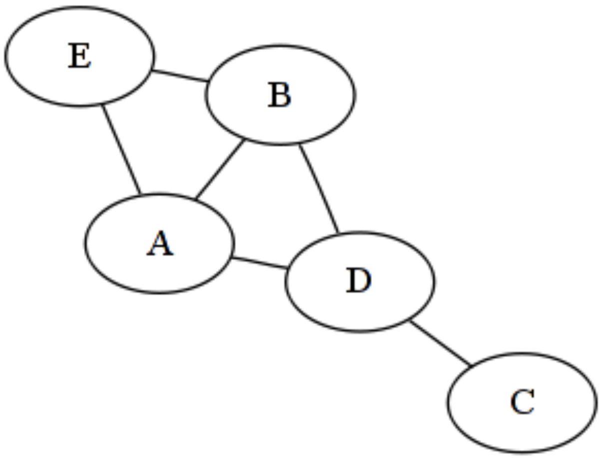
```
function BFS(graph, starting_node):
    create an empty queue Q
    initialize empty set V (set of visited nodes)
    add A to Q
    while Q is not empty:
        take node off Q, call it N
        if N isn't already visited:
            add N to the visited set
            add every neighbor of N to Q
```

Sprint : Modify for shortest path



<table>
<tr><th></th><th>Current Node (N)</th><th>Queue (Q)</th><th>Visited (V)</th><th>explanation</th></tr>
<tr><td></td><td></td><td>A,0</td><td></td><td>initialization</td></tr>
<tr><td>1</td><td>A,0</td><td></td><td></td><td>take first node off queue</td></tr>
<tr><td></td><td>A,0</td><td>E,1|B,1|D,1</td><td>A</td><td>add A's neighbors to queue</td></tr>
<tr><td>2</td><td>E,1</td><td>B,1|D,1</td><td>A</td><td>take E off queue</td></tr>
<tr><td></td><td>E,1</td><td>B,1|D,1|A,2|B,2</td><td>A,E</td><td>add E's neighbors to queue</td></tr>
<tr><td>3</td><td>B,1</td><td>D,1|A,2|B,2</td><td>A,E</td><td>take B off queue</td></tr>
<tr><td></td><td>B,1</td><td>D,1|A,2|B,2|E,2|D,2</td><td>A,E,B</td><td>add B's neighbors to queue</td></tr>
<tr><td>4</td><td>D,1</td><td>A,2|B,2|E,2|D,2</td><td>A,E,B</td><td>take D off queue</td></tr>
<tr><td></td><td>D,1</td><td>A,2|B,2|E,2|D,2|C,2|B,2</td><td>A,E,B,D</td><td>add D's neighbors to queue</td></tr>
<tr><td>5</td><td>A,2</td><td>B,2|E,2|D,2|C,2|B,2</td><td>A,E,B,D</td><td>take A off queue</td></tr>
<tr><td></td><td>A,2</td><td>B,2|E,2|D,2|C,2|B,2</td><td>A,E,B,D</td><td>skip A since already visited</td></tr>
<tr><td>6</td><td>B,2</td><td>E,2|D,2|C,2|B,2</td><td>A,E,B,D</td><td>take B off queue</td></tr>
<tr><td></td><td>B,2</td><td>E,2|D,2|C,2|B,2</td><td>A,E,B,D</td><td>skip B since already visited</td></tr>
<tr><td>7</td><td>E,2</td><td>D,2|C,2|B,2</td><td>A,E,B,D</td><td>take E off queue</td></tr>
<tr><td></td><td>E,2</td><td>D,2|C,2|B,2</td><td>A,E,B,D</td><td>skip E since already visited</td></tr>
<tr><td>8</td><td>D,2</td><td>C,2|B,2</td><td>A,E,B,D</td><td>take D off queue</td></tr>
<tr><td></td><td>D,2</td><td>C,2|B,2</td><td>A,E,B,D</td><td>skip D since already visited</td></tr>
<tr><td>9</td><td>C,2</td><td>B,2</td><td>A,E,B,D</td><td>take C off queue</td></tr>
<tr><td></td><td>C,2</td><td>B,2</td><td>A,E,B,D</td><td>We're done!! :tada:</td></tr>
</table>

Modify for
shortest path from A to C



BFS, pop quiz

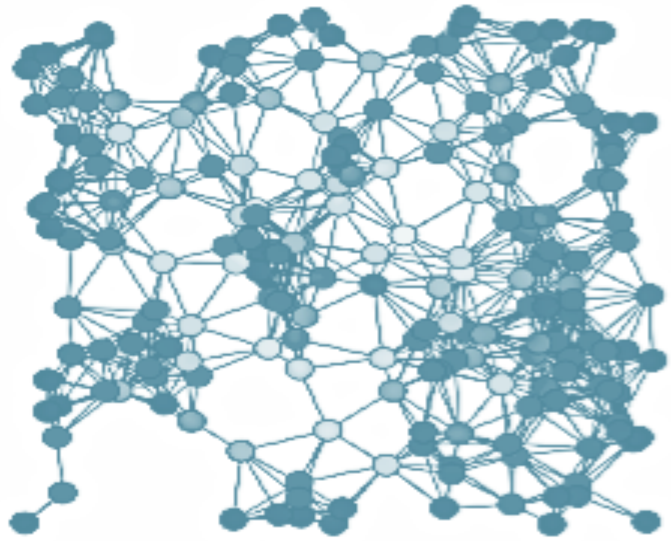
BFS implies that there is also a depth first search(DFS). Any guesses on what that would look like?

BFS is much more widely used than DFS. Why would that be?

What would be a good use case for depth first search?

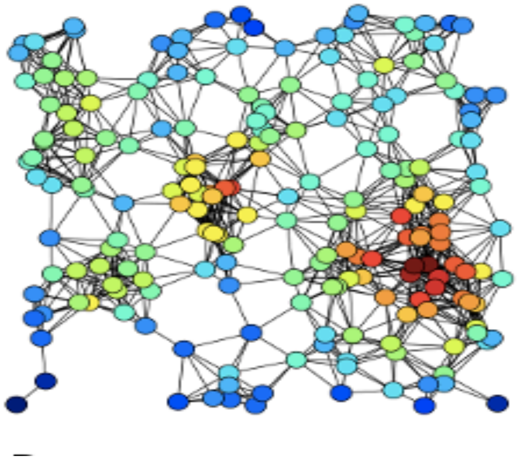
FYI, BFS pseudo-code is a popular interview question.

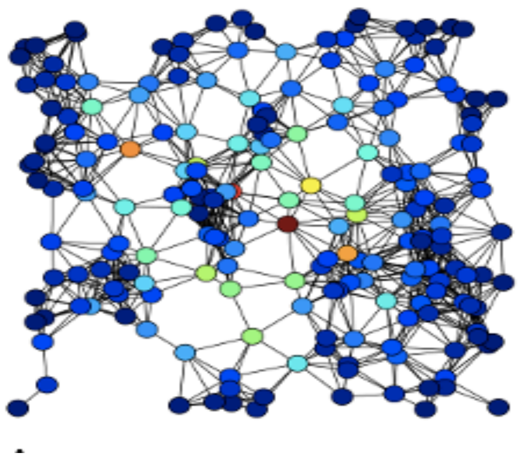
“Important” nodes?

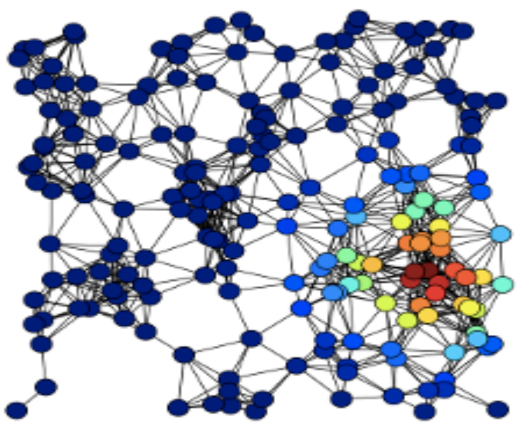


What makes a node important or central?

Centrality of a node







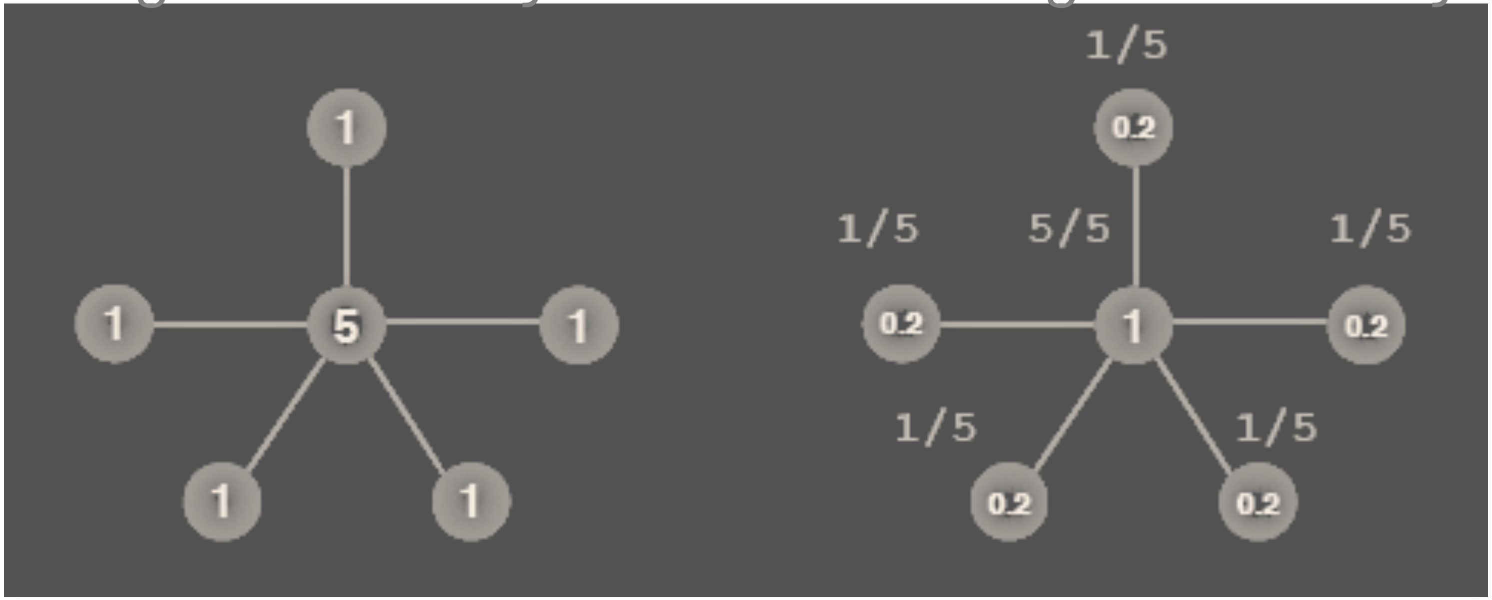
Different definitions of importance different measures of centrality

https://en.wikipedia.org/wiki/File:6_centrality_measures.png

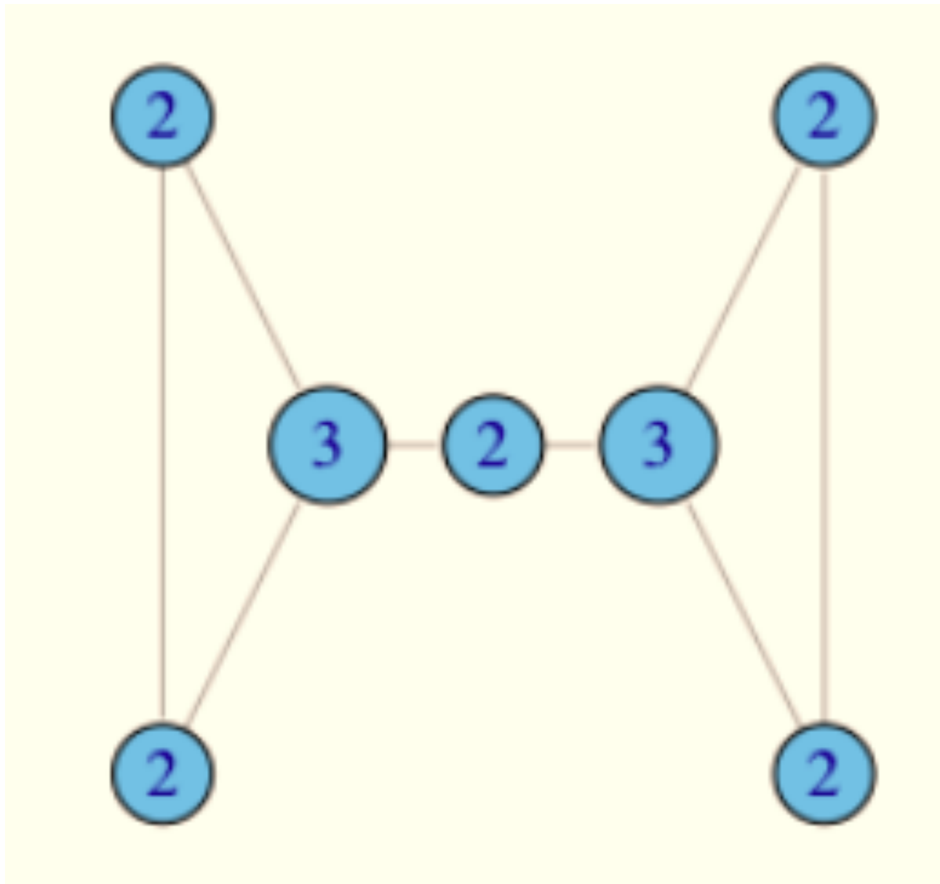
Degree Centrality

```
                              d(n)              degree of n
degree centrality(n) = --------- = --------------------------
                            |V| - 1      number of nodes in G - 1
```

Degree Centrality & Normalized Degree Centrality



In what way does degree fail to capture centrality here?



Betweeness Centrality

Betweeness centrality can be represented as:

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

σ_{st} is total number of shortest paths from node s to node t

$\sigma_{st}(v)$ is the number of those paths that pass through v.

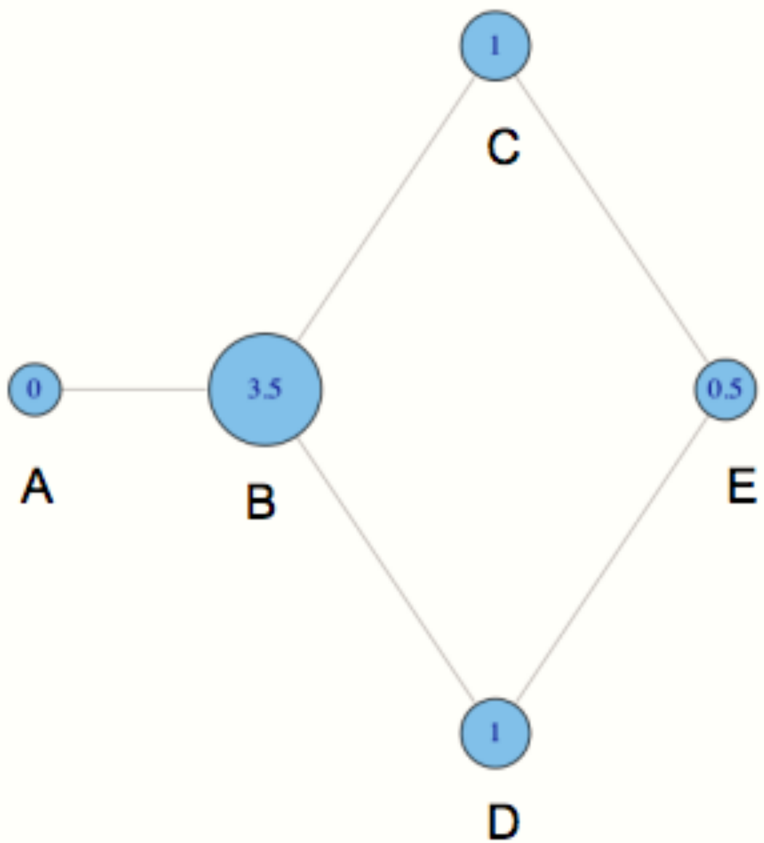
Normalized Betweeness Centrality (undirected graph)

$$C'_B(i) = C_B(i) / \boxed{[(n-1)(n-2)/2]}$$

number of pairs of vertices
excluding the vertex itself

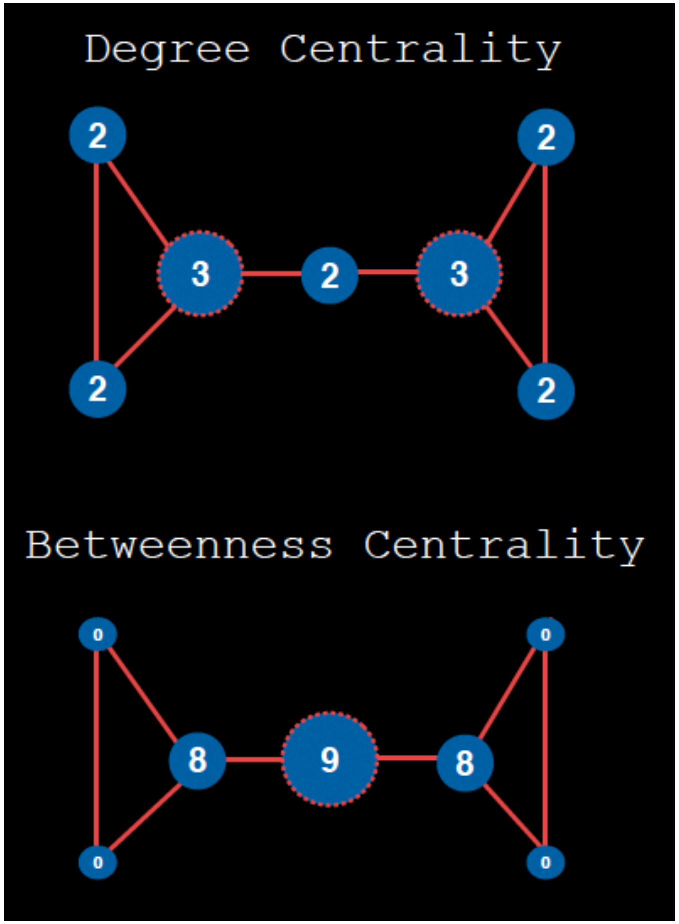
Betweeness Centrality, toy example

- non-normalized version:

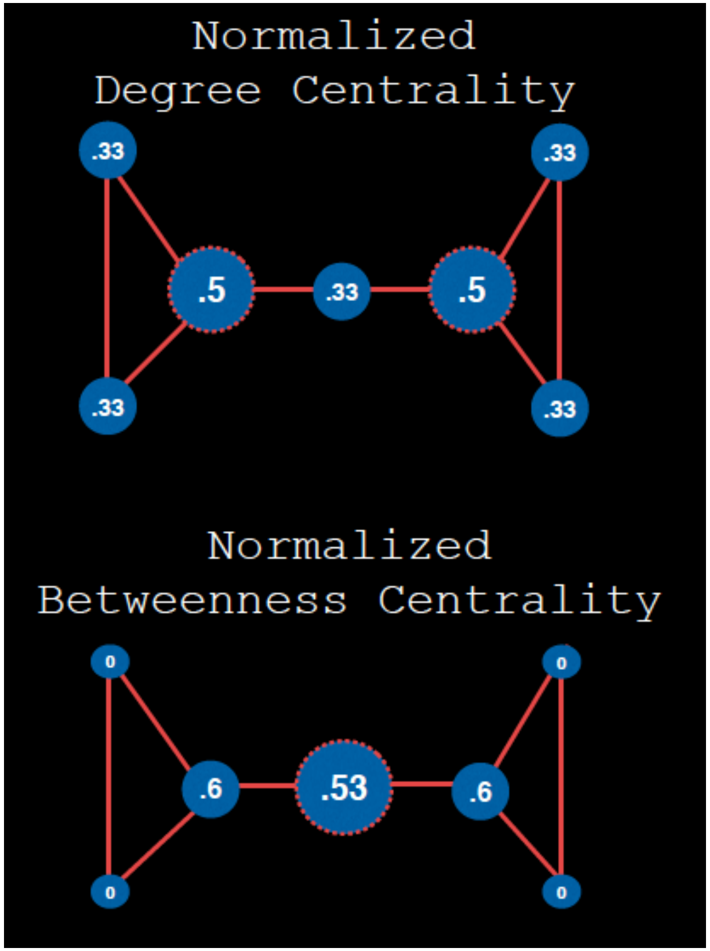


- why do C and D each have betweenness 1?
- They are both on shortest paths for pairs (A,E), and (B,E), and so must share credit:
  - ½+½ = 1
- Can you figure out why B has betweenness 3.5 while E has betweenness 0.5?

In review:



Degree Centrality
2
2
3
2
3
2
2
Betweenness Centrality
0
0
8
9
8
0
0



Normalized
Degree Centrality
.33
.33
.5
.33
.5
.33
.33
Normalized
Betweenness Centrality
0
0
.6
.53
.6
0
0

Eigenvector Centrality

Important nodes

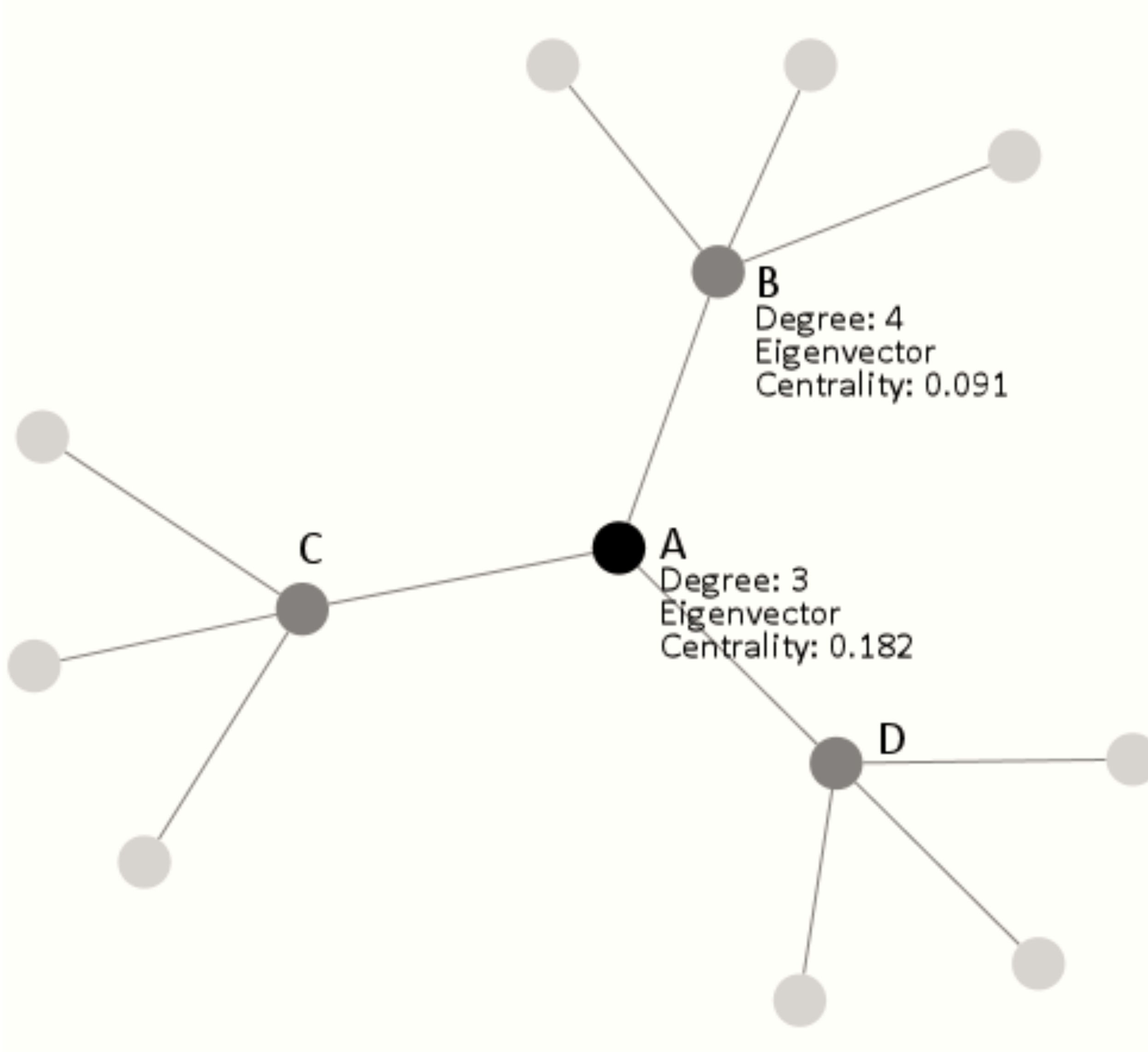
- Have important neighbors
- Connected to node with high degrees
- Themselves do not necessarily have high degree

Tied to the idea of influence. The centrality of a given node has is equal to sum of the centrality of my neighbors.

Page rank, a variation on this, was used to rank websites in a search result.

NOTE: We have seen this in an earlier individual sprint with stochastic matrices! An implementation of page rank was extra credit.

Eigenvector Centrality



Eigenvector Centrality

neighbors 1/0
A is adjacency matrix

$$x_i = \frac{1}{\lambda} \sum_{j \in G} A_{ij} . x_j$$

C_E of node i

eigenvalue

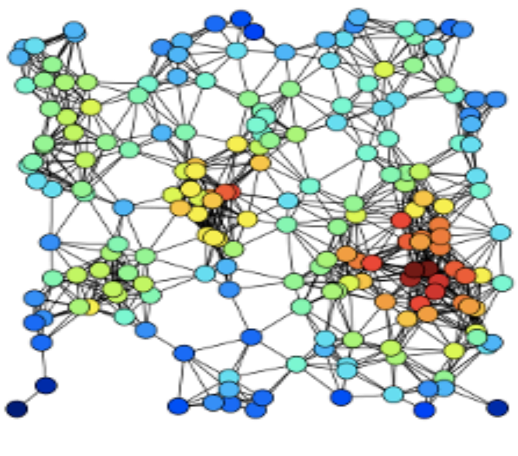
C_E of node j

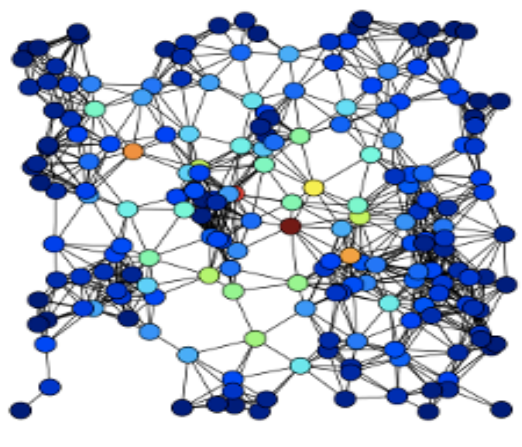
$$\mathrm{Ax} = \lambda \, \mathrm{x}$$

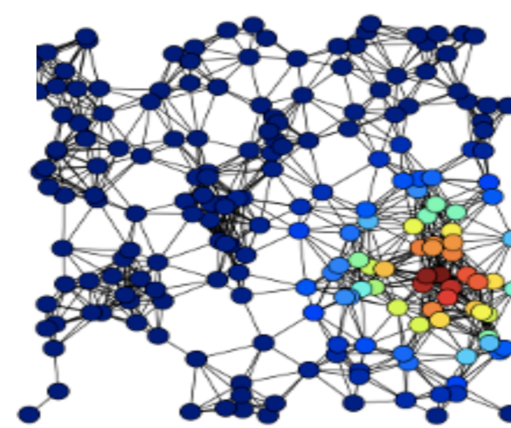
Eigenvector with
C_E of all nodes

“Centrality” pop quiz

Match the graph to the centrality depicted







Session Summary and SPRINT

- Formal definition of graphs and applications
- Traversing a graph with Breadth First Search
- Importance of an node

SPRINT

- Explore the IMBD dataset of movies and actors
  - Write code to find the shortest path between two actors using BFS. Kevin Bacon Number, anyone?