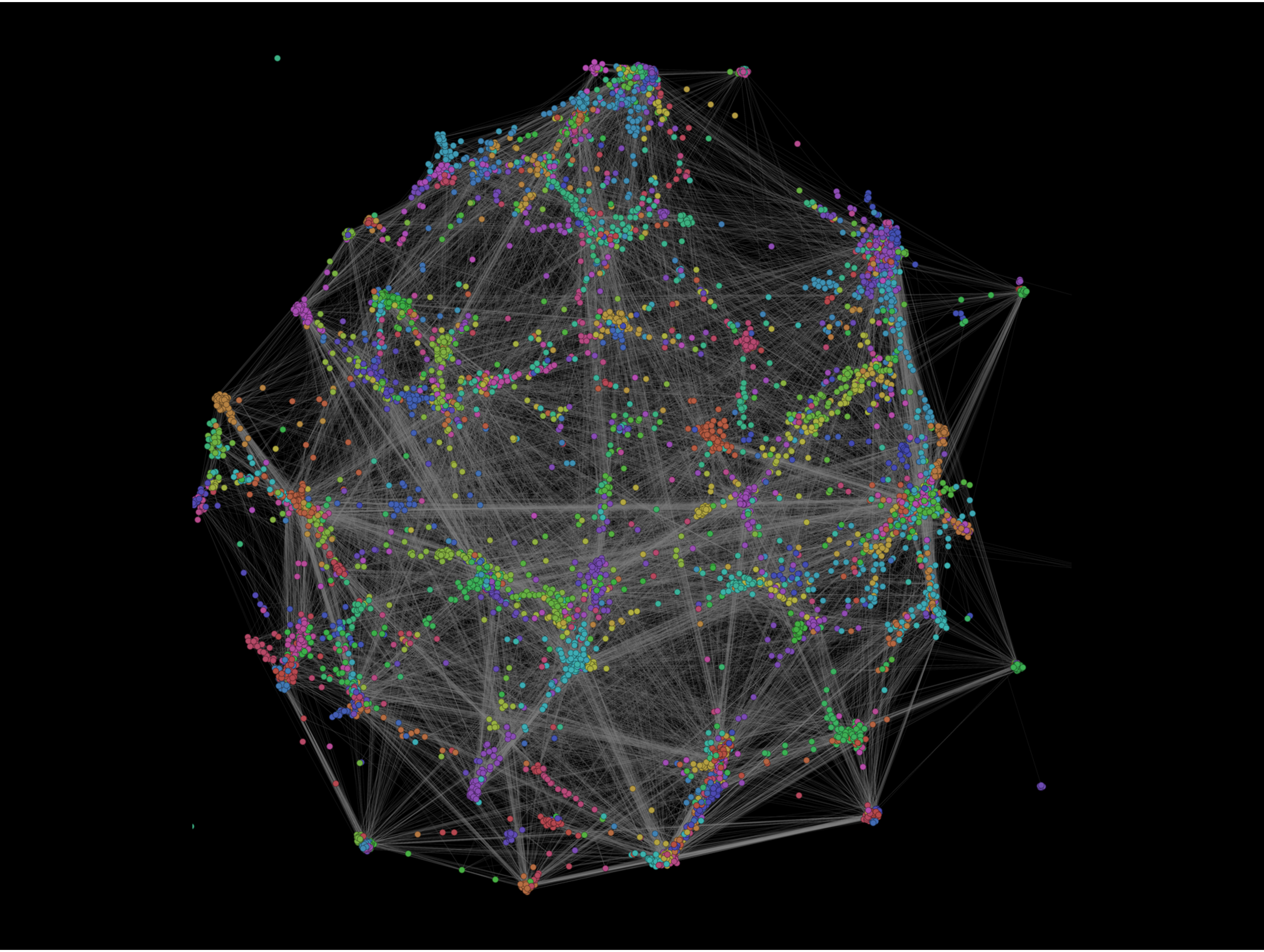
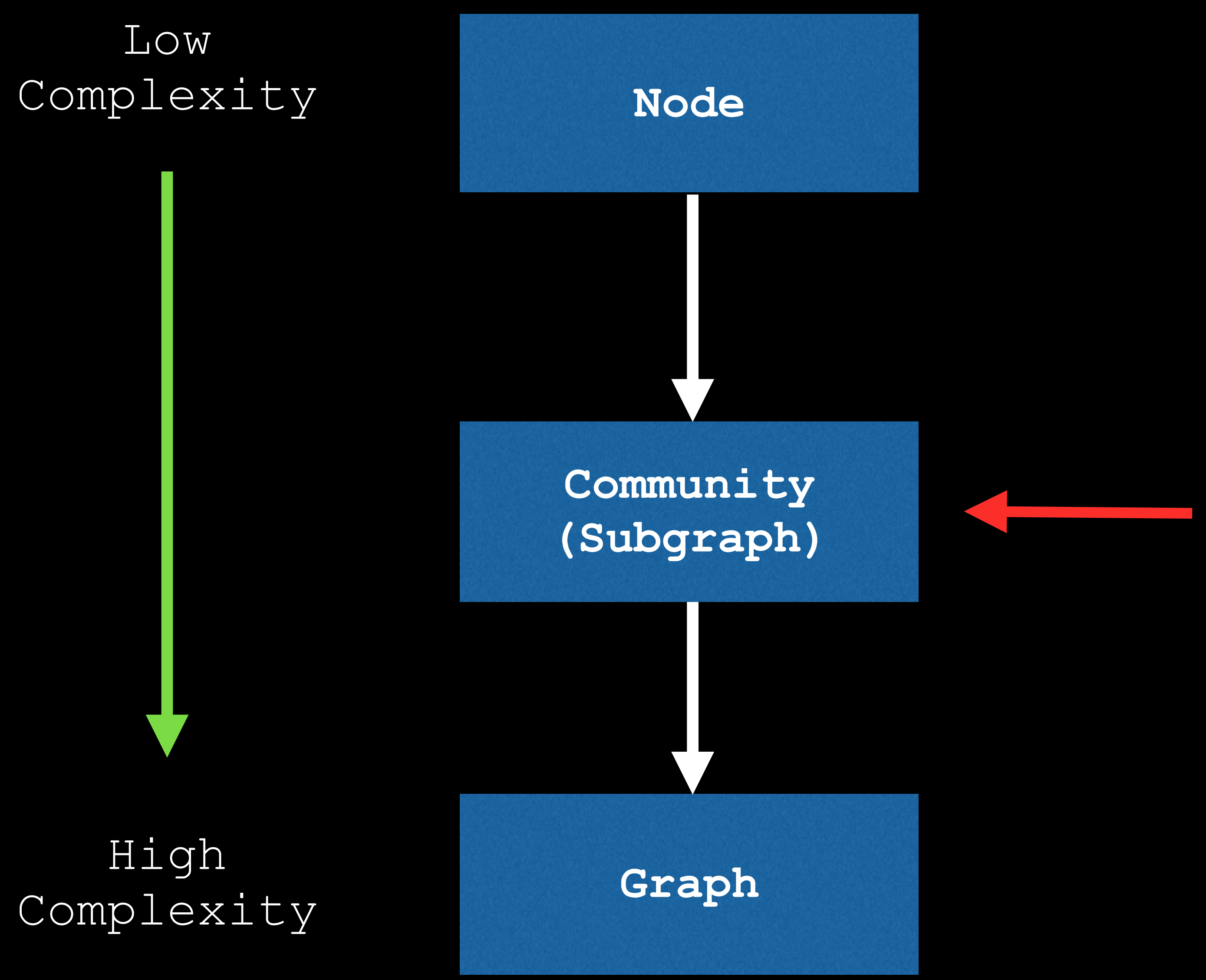
Introduction to Graphs



- **Morning**
  - ⋆ General understanding of graphs
  - ⋆ Importance of an individual node
- **Afternoon**
  - ⋆ Defining communities in graphs
  - ⋆ Finding communities



Low
Complexity
Node
Community
(Subgraph)
Graph
High
Complexity

Goals

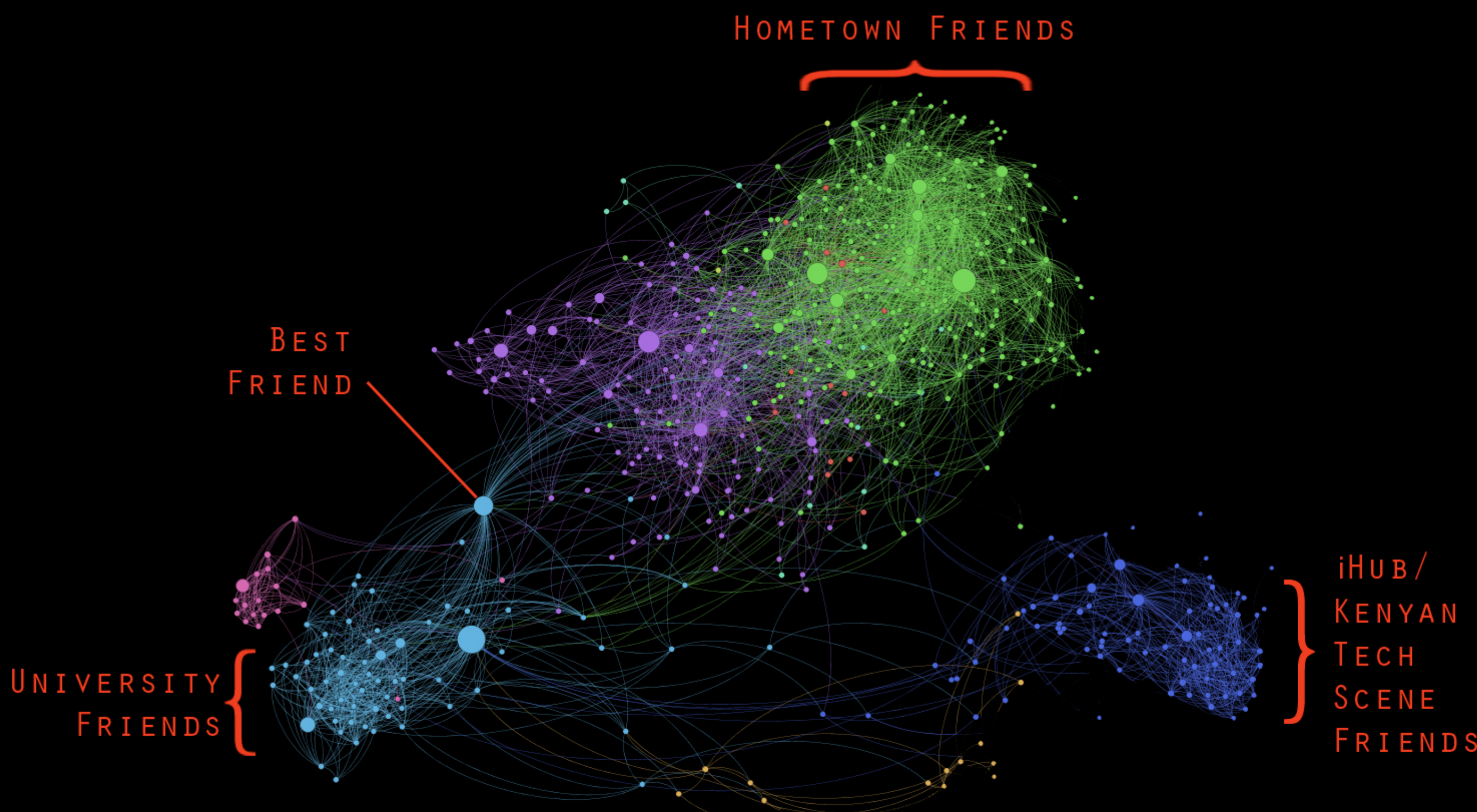
- Applications
- Power Law
- Graph Basics
- Centrality
- Graph Search

Goals

- Applications
- Power Law
- Graph Basics
- Centrality
- Graph Search

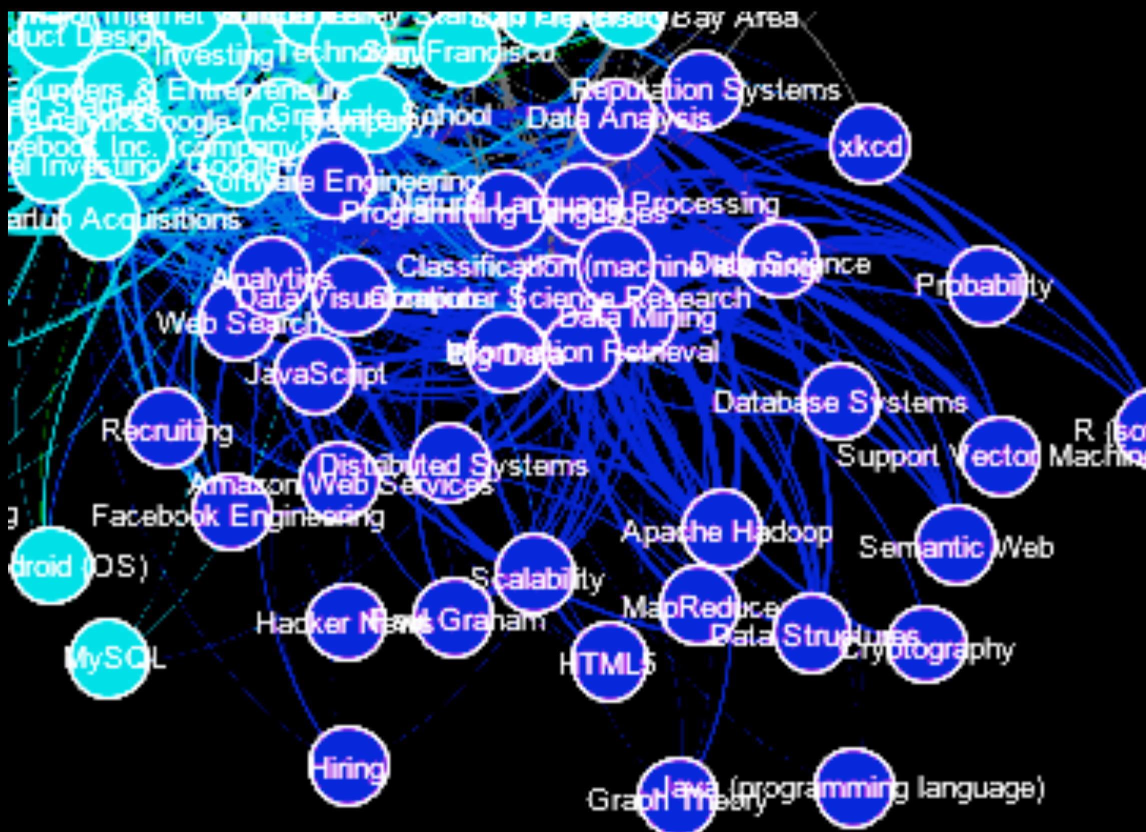
Application 1: Measure Connectedness

- Distance between diff. groups of friends
- Find socially important individuals



Application 2: Measure Co-occurrence

- Largest connected co-occurrence sub-graph
- At difference co-occurrence threshold



Application 3: Measure Propagation

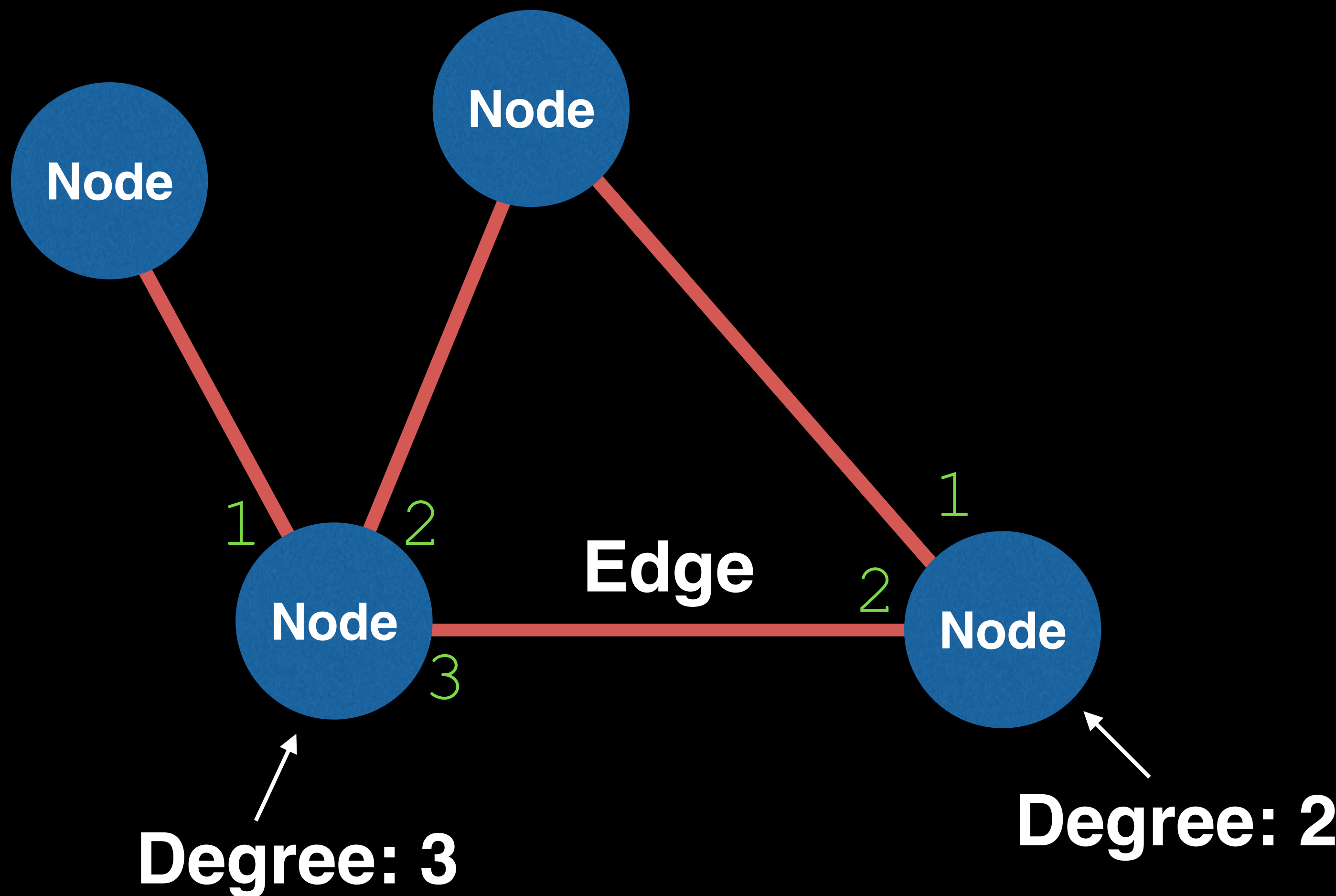
- Based on flow between the airports model the number of planes at a port at a given time

http://openflights.org/data.html

Goals

- Applications
- Power Law
- Graph Basics
- Centrality
- Graph Search

Basic Terminology



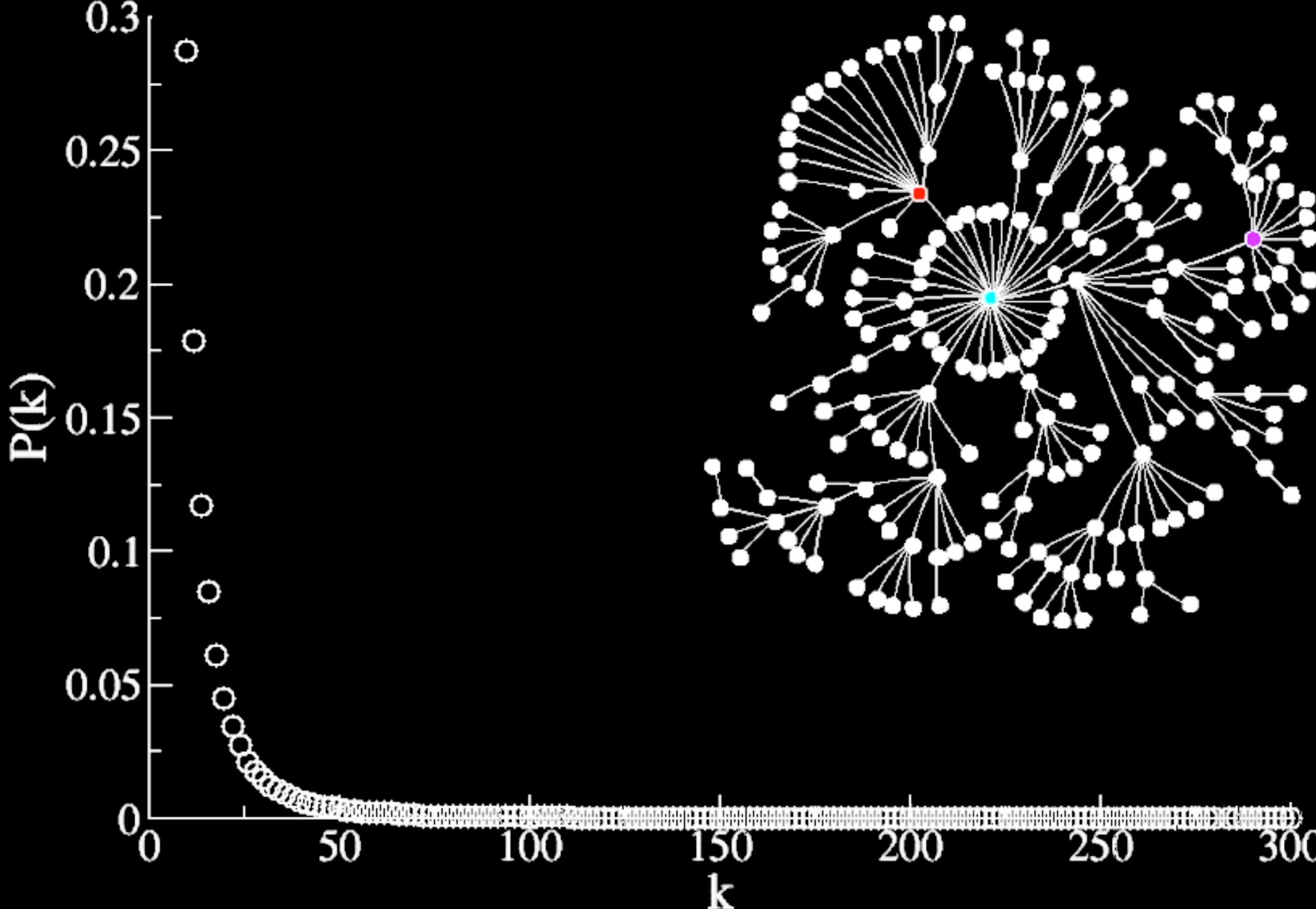
Power Law

$$P(k) \sim k^{-\gamma}$$

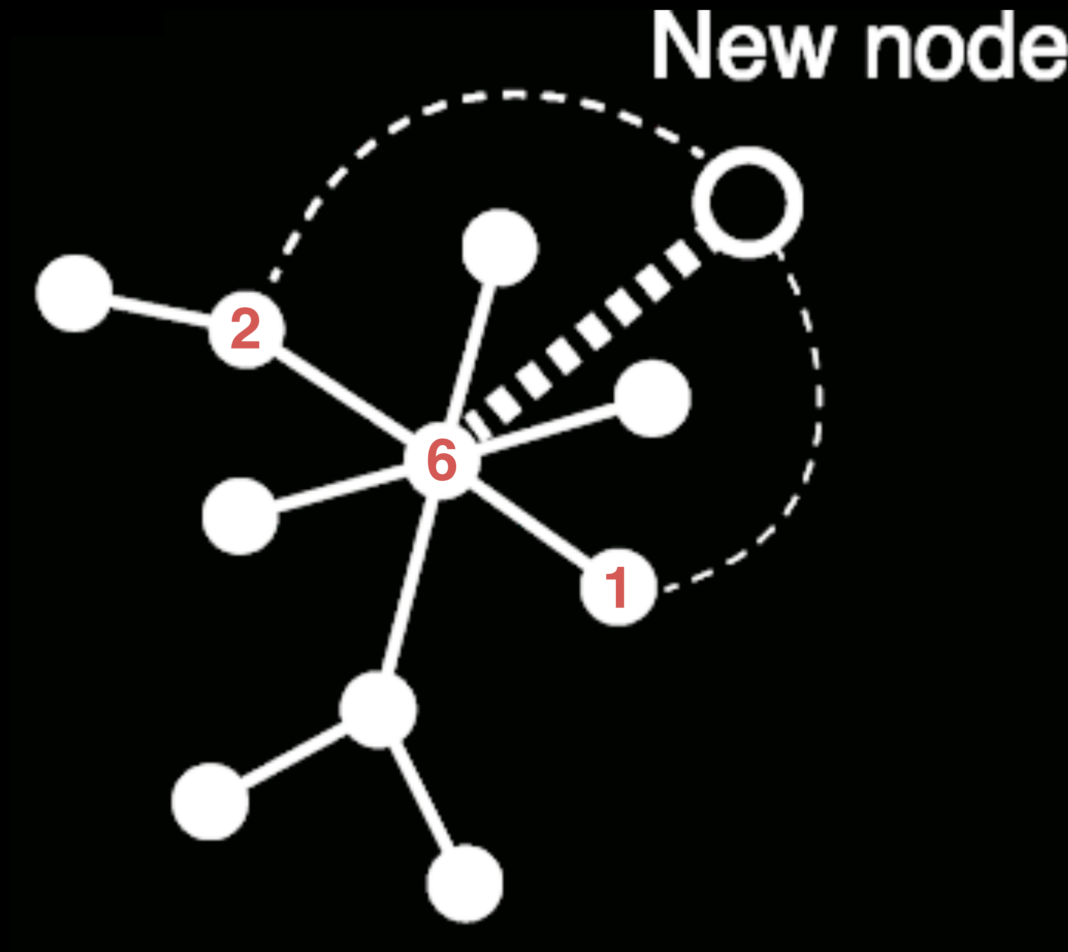
where

- **k** is the degree of a node
- **2 ≤** γ **≤ 3**
- $P(k)$ is the probability of a node being degree **k**

Degree Distribution



Preferential Attachment

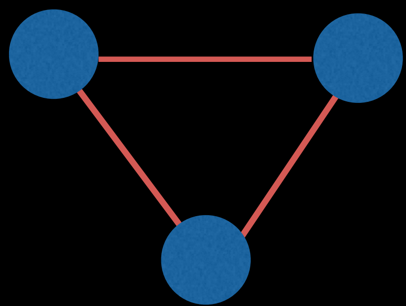


Goals

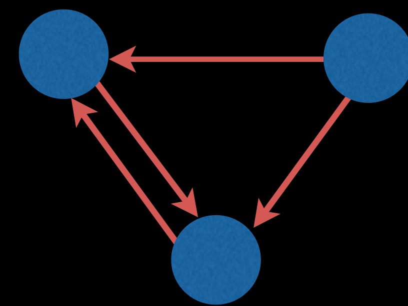
- Applications
- Power Law
- Graph Basics
- Centrality
- Graph Search

Types of Graphs

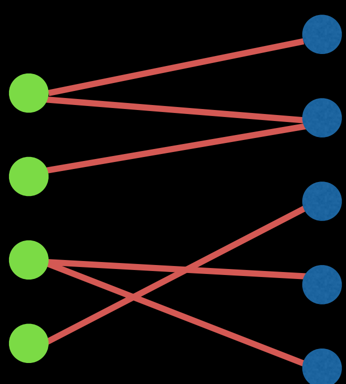
Undirected



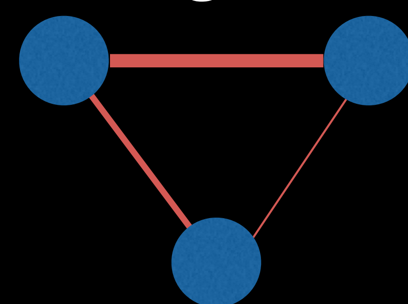
Directed



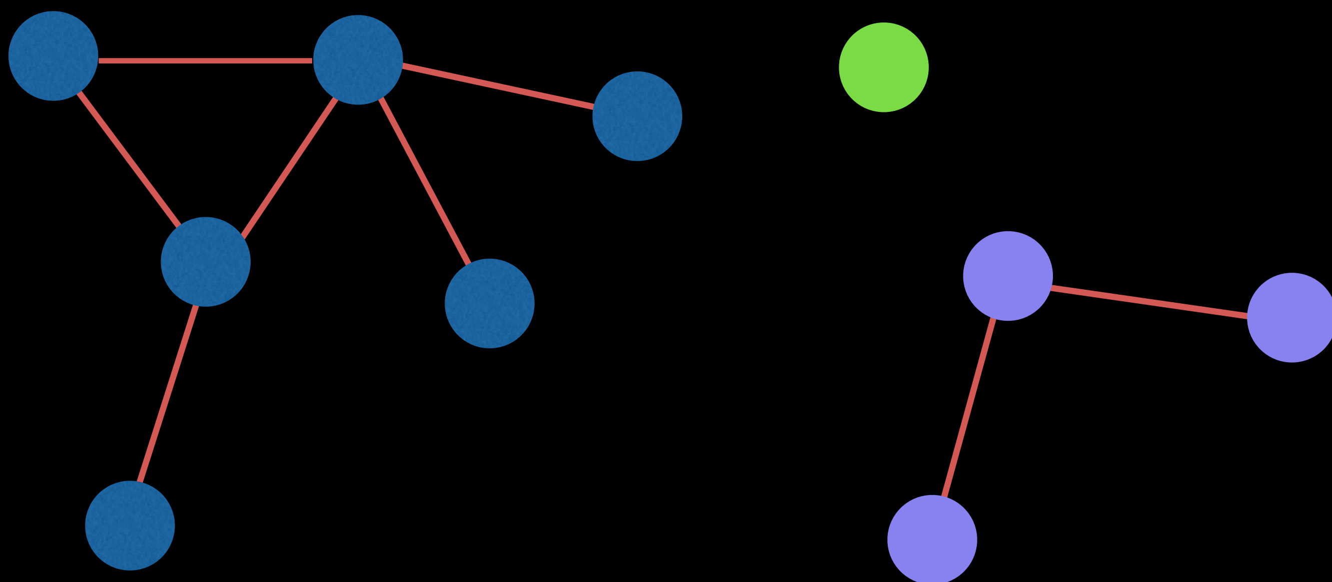
Bipartite



Weighted



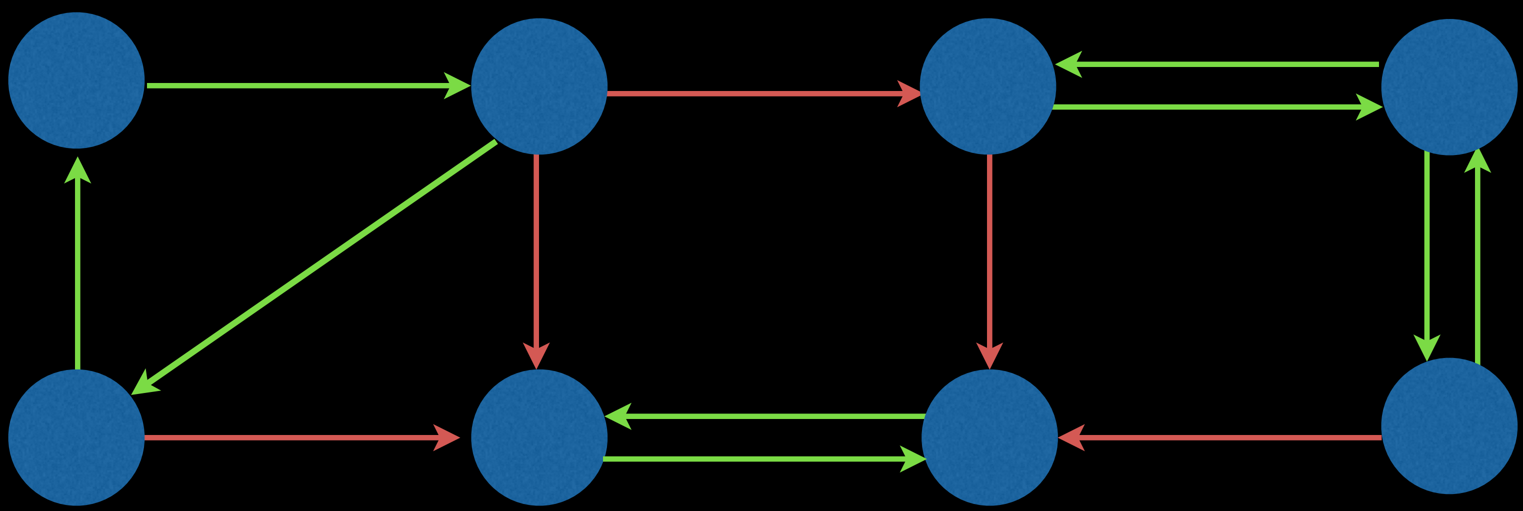
Connected Components



Subgraph where any two vertices are connected to each other by an edge(s)

(Directed / Undirected Graphs)

Strongly Connected Component



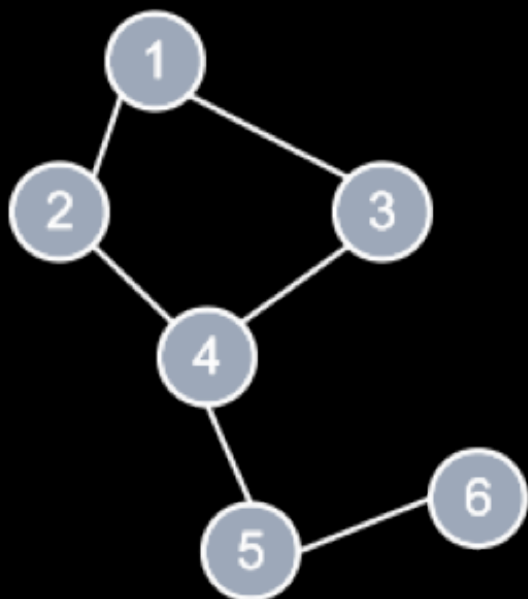
Subgraph where every node is reachable by another node

(Directed Graphs)

Data Structure

- Adjacency List
- Adjacency Matrix

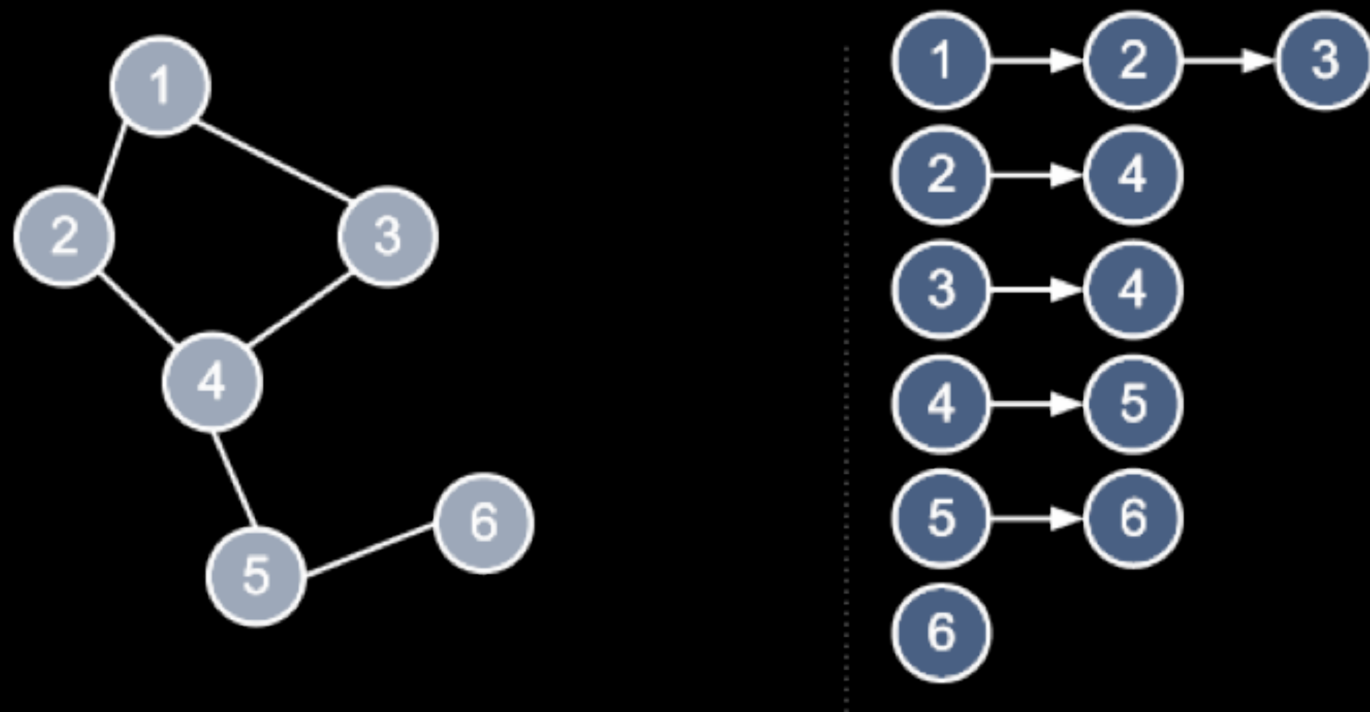
Adjacency Matrix



	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

- Take more space if sparse
- Faster to look up

Adjacency List



- Takes up less space
- Slower to look up

Goals

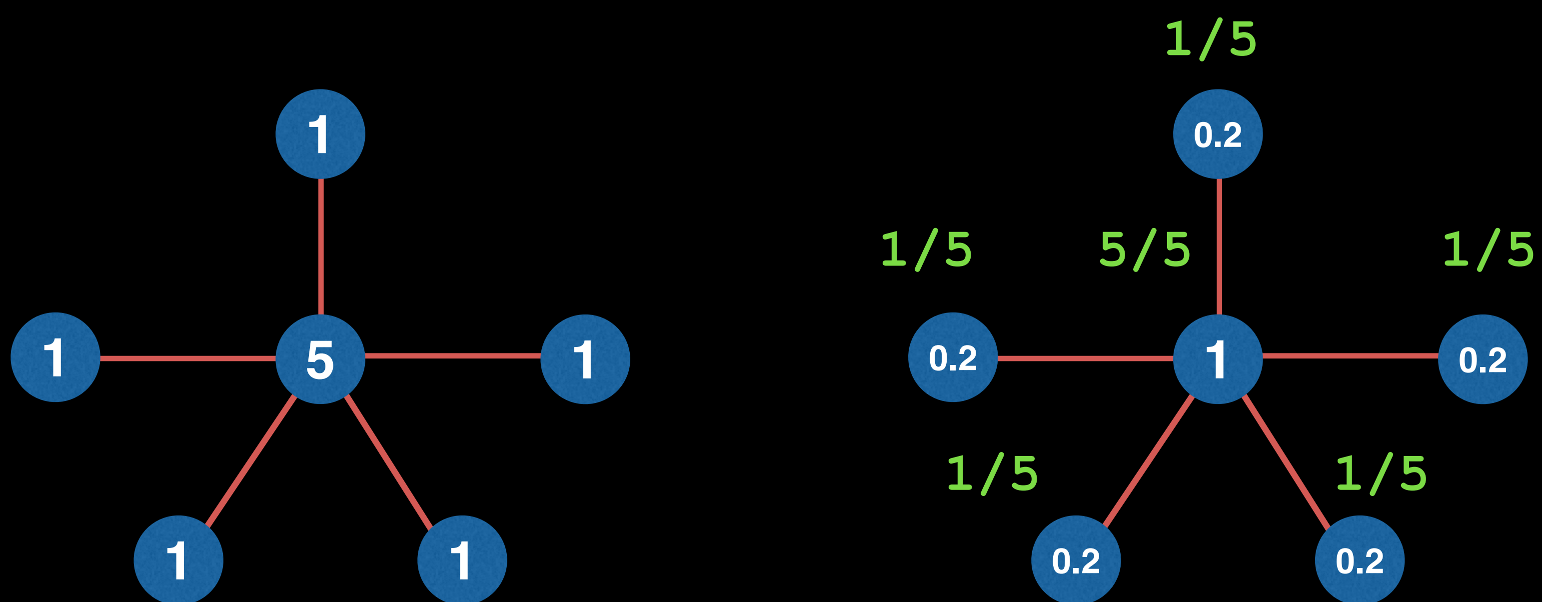
- Applications
- Power Law
- Graph Basics
- Centrality
- Graph Search

Centrality

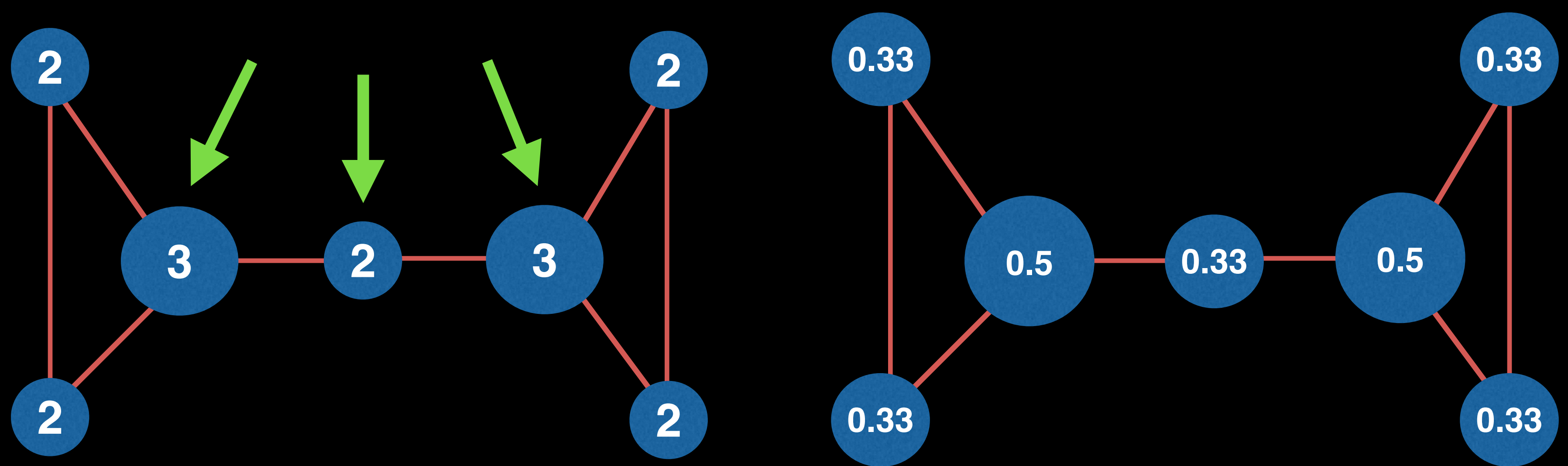
- **Importance of a node**
- Different definitions of importance
- Normalized to range from 0 to 1
- **Only relevant to the context of a particular graph**

Degree Centrality

- An important node is connected to a large number of other nodes
- Normalized by dividing
  `(total number of nodes - 1)`

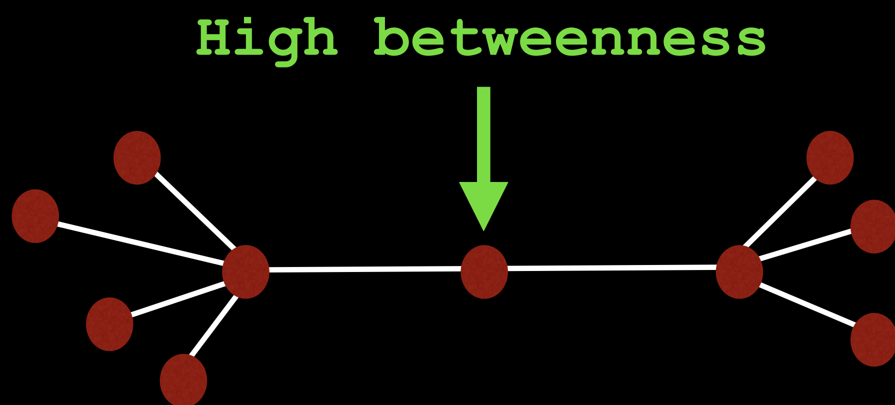


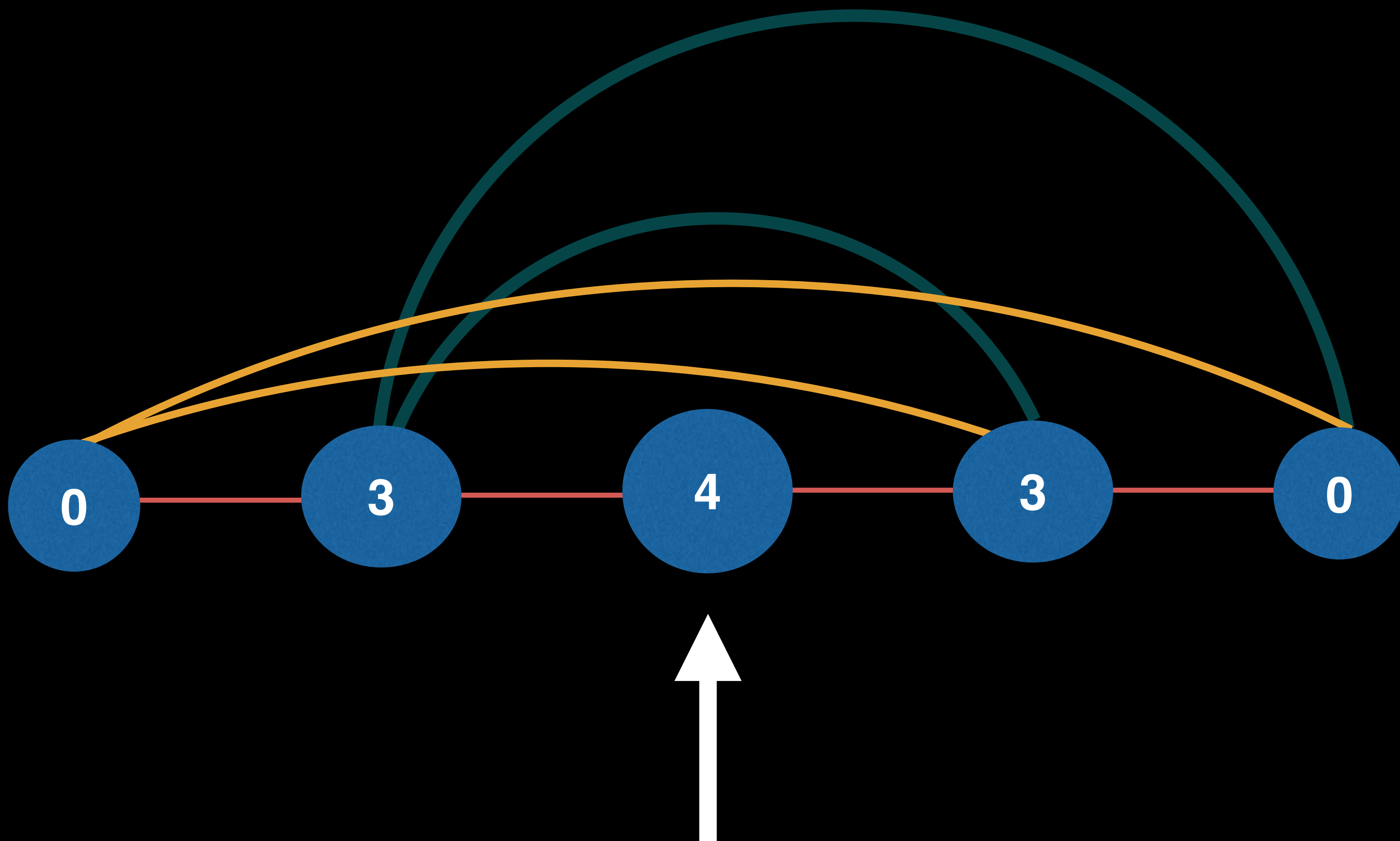
Degree Centrality does not always capture the most “important” nodes



Betweenness Centrality

- An important node controls the passage from one node to the other




0
3
4
3
0

Sum of fractions of shortest paths that pass *v*

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

\# of shortest path between *s* and *t* that pass *v*

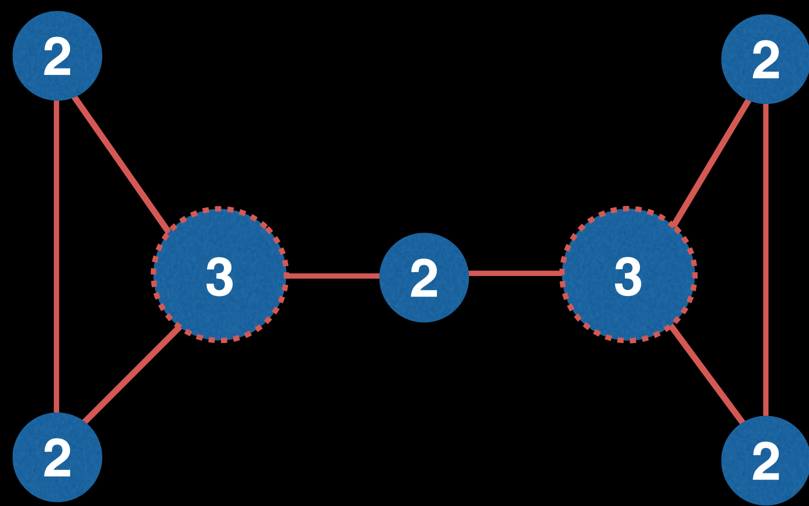
\# of shortest path between *s* and *t*

Betweenness Centrality Normalization

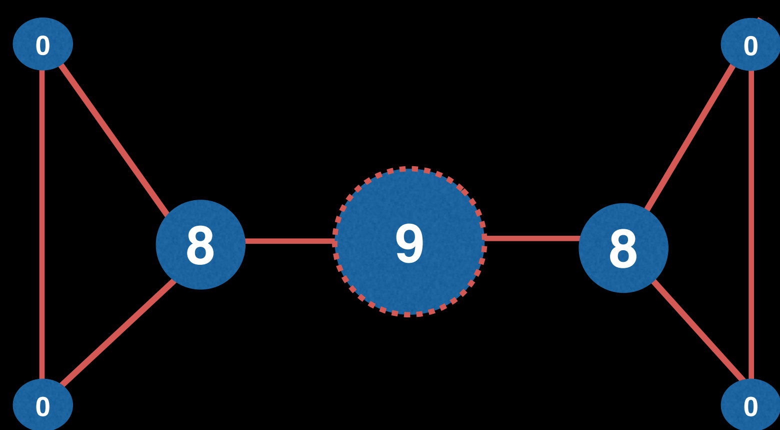
$$normal(C_b(v)) = \frac{C_b(v)}{(N-1)(N-2)}$$

`where N is the number of nodes in the graph`

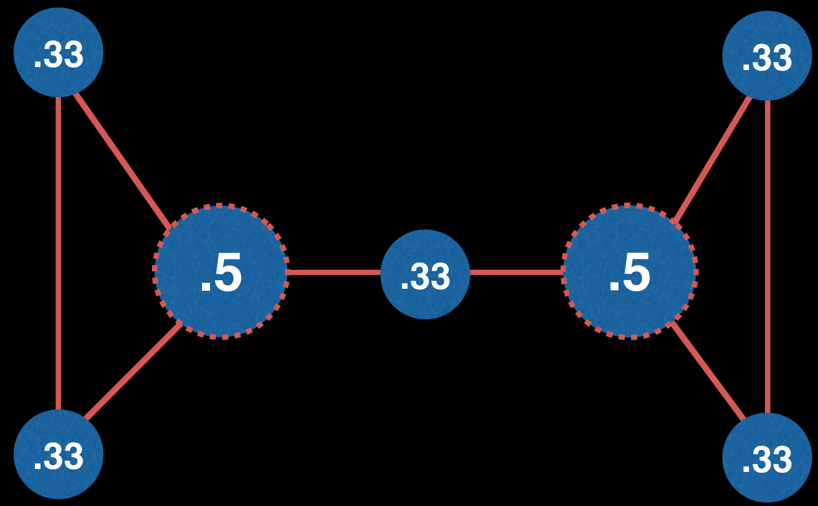
Degree Centrality



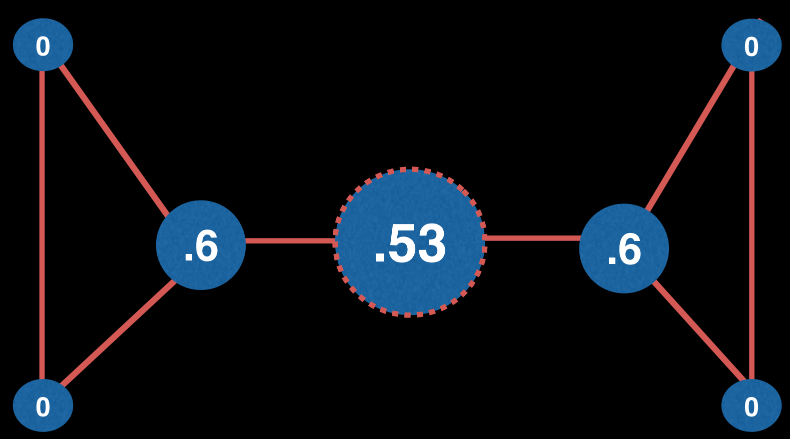
Betweenness Centrality



Normalized
Degree Centrality

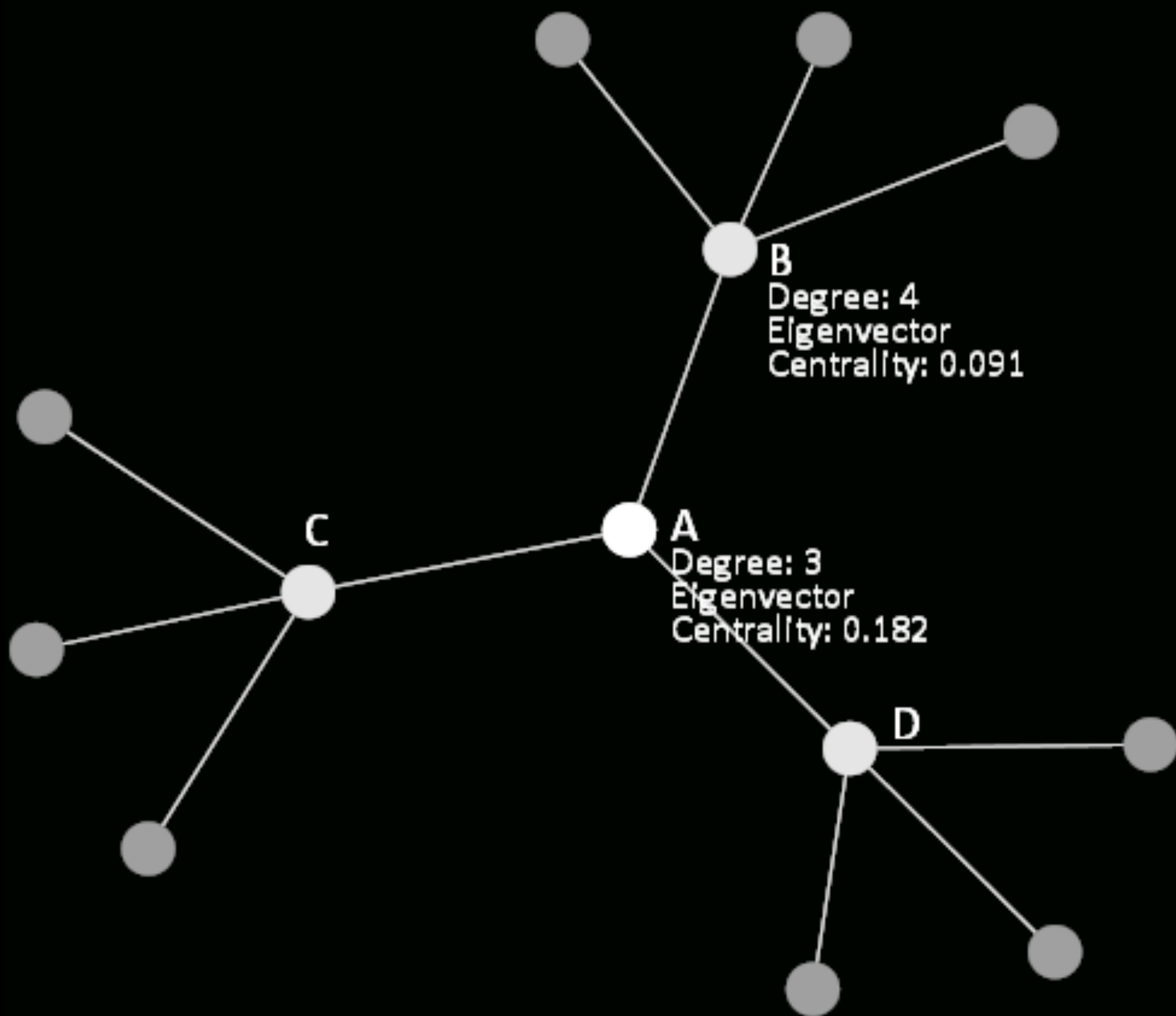


Normalized
Betweenness Centrality



Eigenvector Centrality

- **Important nodes**
  - ★ Have important neighbors
  - ★ Connected to nodes with high degree
  - ★ Themselves do not necessarily have high degree


B
Degree: 4
Eigenvector
Centrality: 0.091
C
A
Degree: 3
Eigenvector
Centrality: 0.182
D

C_E of node i

neighbors? (1/0)

C_E of node j

$$\boxed{x_i} = \frac{1}{\boxed{\lambda}} \sum_{j \epsilon G} \boxed{A_{ij}} \cdot \boxed{x_j}$$

eigenvalue

$$A\boxed{x} = \boxed{\lambda}\boxed{x}$$

Eigenvector with C_E of all nodes

C_E = Eigenvector Centrality

Goals

- Applications
- Power Law
- Graph Basics
- Centrality
- Graph Traversal

Graph Traversal

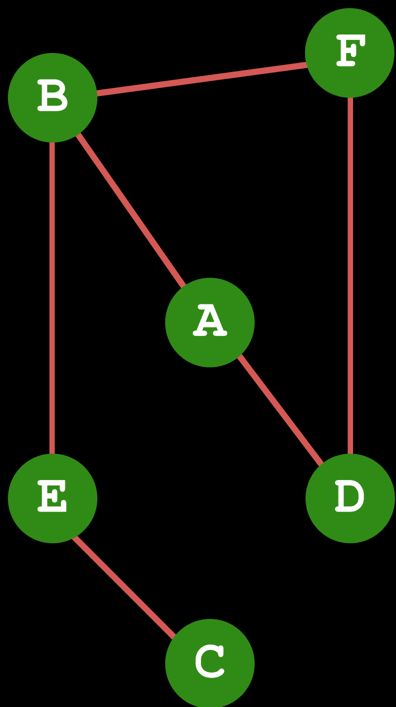
- In order perform graph related operations
  `(e.g. connected components, shortest path …)`
- Must systematically visit all the nodes
- Most popular graph traversal algorithm:

Breath First Search

Breath First Search

- Visit all the neighbors of a node before visiting the neighbors of those neighbors
- Uses a **First In First Out (FIFO)** queue

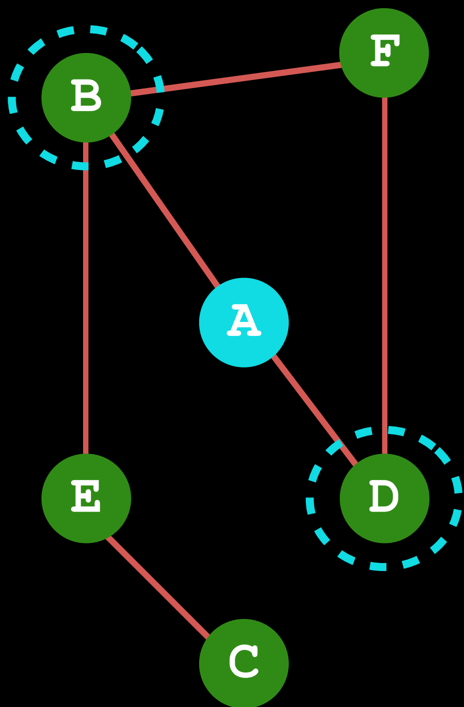
Breath First Search



NODE	VISITED	VISITED BY
A	F	–
B	F	–
C	F	–
D	F	–
E	F	–
F	F	–

Queue	
Visited	

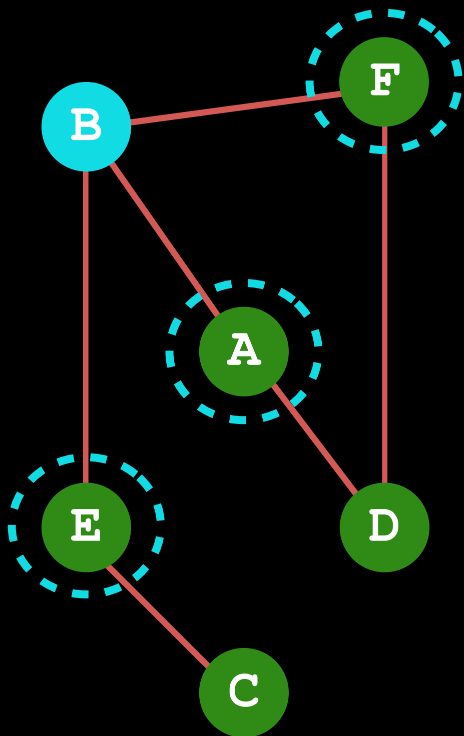
Breath First Search



NODE	VISITED	VISITED BY
A	**T**	**–**
B	**T**	**A**
C	F	–
D	**T**	**A**
E	F	–
F	F	–

Queue	B, D
Visited	A

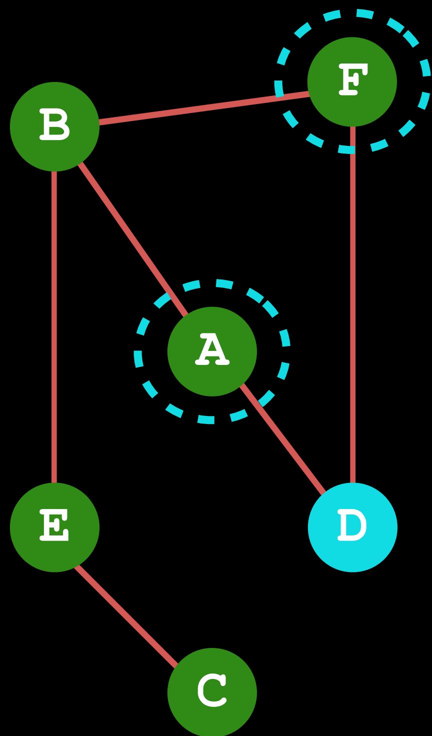
Breath First Search



NODE	VISITED	VISITED BY
A	T	–
B	T	A
C	F	–
D	T	A
E	**T**	**B**
F	**T**	**B**

Queue	D, E, F
Visited	A, B

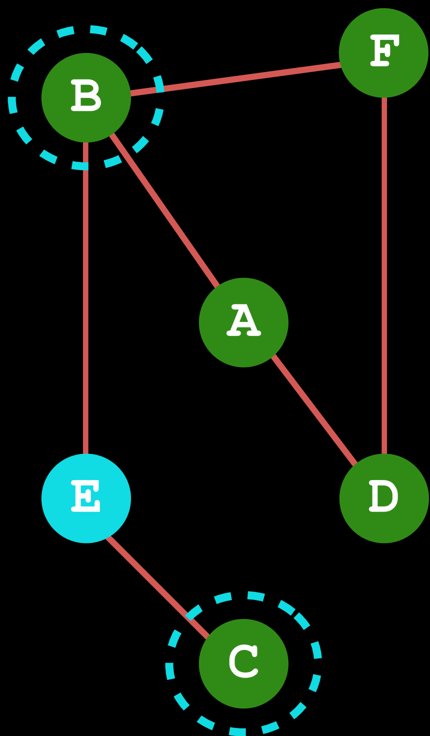
Breath First Search



NODE	VISITED	VISITED BY
A	T	-
B	T	A
C	F	-
D	T	A
E	T	B
F	T	B

Queue	E, F
Visited	A, B, D

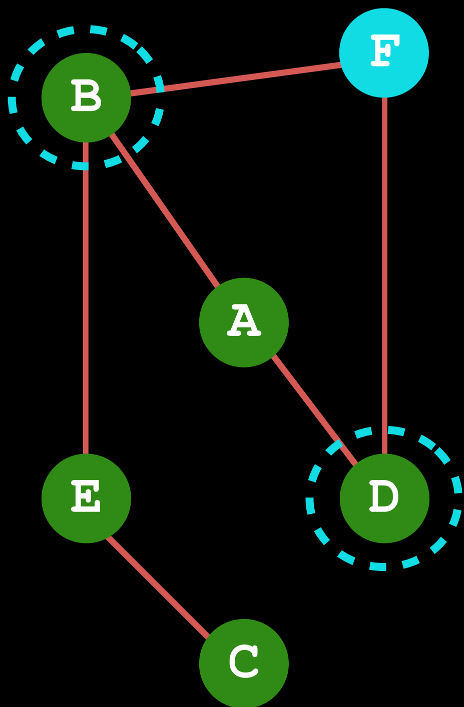
Breath First Search



NODE	VISITED	VISITED BY
A	T	-
B	T	A
C	**T**	**E**
D	T	A
E	T	B
F	T	B

Queue	F, C
Visited	A, B, D, E

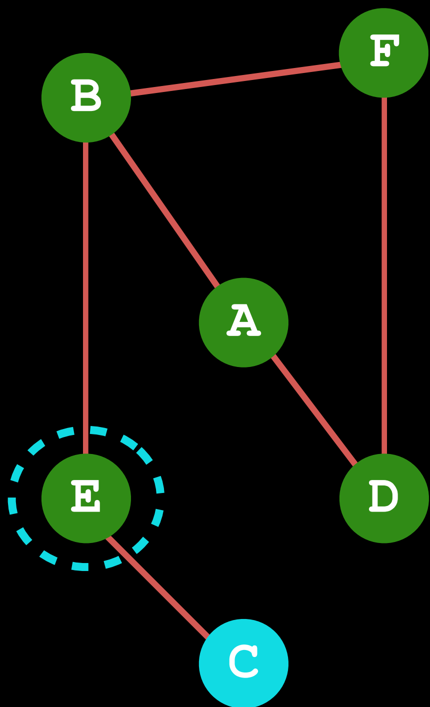
Breath First Search



NODE	VISITED	VISITED BY
A	T	–
B	T	A
C	T	E
D	T	A
E	T	B
F	T	B

Queue	C
Visited	A, B, D, E, F

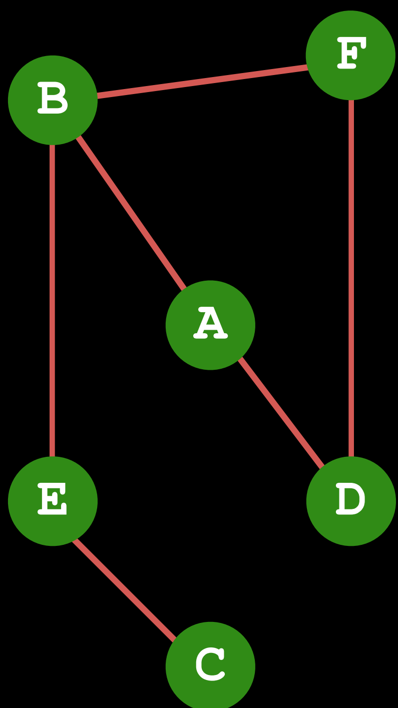
Breath First Search



NODE	VISITED	VISITED BY
A	T	-
B	T	A
C	T	E
D	T	A
E	T	B
F	T	B

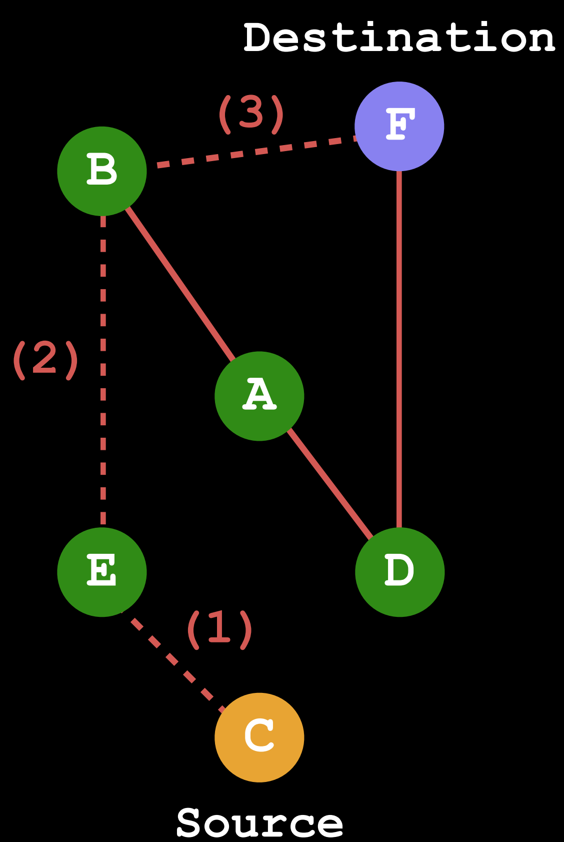
Queue	
Visited	A, B, D, E, F, C

Find Connected Components



Queue	
Visited	A, B, D, E, F, C

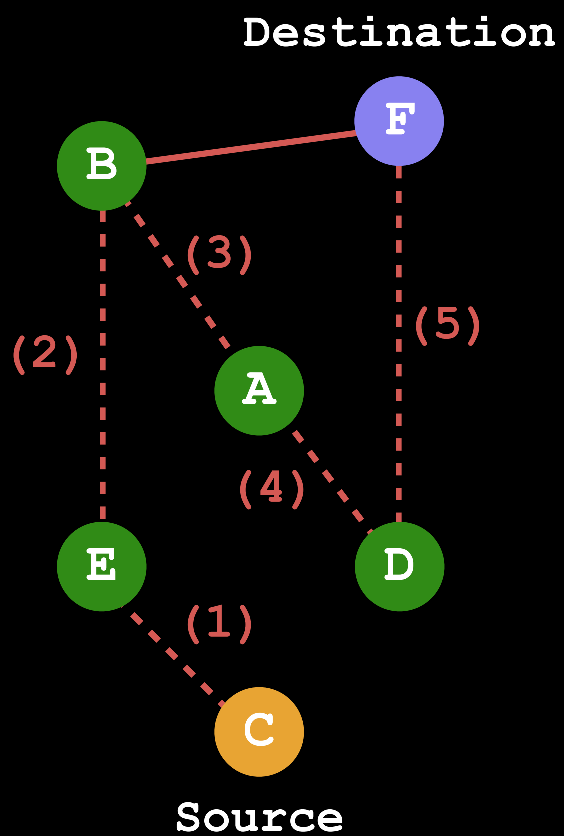
Find Shortest Path



NODE	VISITED BY
A	-
B	A
C	E (1)
D	A
E (1)	B (2)
F (3)	B (2)

C —> F : 3 steps

Find Shortest Path



NODE	VISITED BY
A	–
B (2)	A (3)
C	E (1)
D (4)	A (3)
E (1)	B (2)
F (5)	B

C —> F : 5 steps

Summary

- Different Uses of Graphs
  - ⋆ Connections / Co-occurrence / Propagation
- Power law in social networks
- Directed / Undirected / Bipartite
- (Strongly) Connected Components
- Degree / Betweenness / Eigenvector Centrality
- Breath First Search