Graphs For Data Science

Bruno Gonçalves

graphs4sci.substack.com

https://github.com/DataForScience/G4DS

Question

https://github.com/DataForScience/G4DS

- What's your job title?
    - Data Scientist
    - Statistician
    - Data Engineer
    - Researcher
    - Business Analyst
    - Software Engineer
    - Other

Question

https://github.com/DataForScience/G4DS

- How experienced are you in Python?
    - Beginner (<1 year)
    - Intermediate (1-5 years)
    - Expert (5+ years)

Question

https://github.com/DataForScience/G4DS

- How did you hear about this webinar?
    - O'Reilly Platform
    - Newsletter
    - data4sci.com Website
    - Previous event
    - Other?

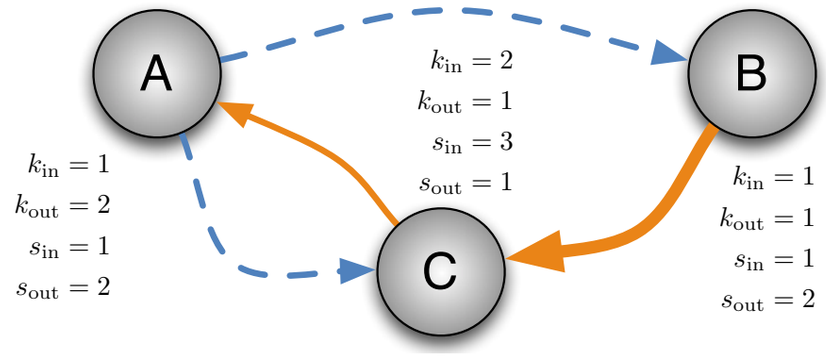
Table of Contents

1. Weighted Directed Graphs
   Airline Transportation Network
2. Navigating graphs
   OpenStreetMap
3. Patterns and Structure in graphs
   BitCoin Transaction Network
4. Social Structure
   Twitter
5. Recommendations and Link prediction
   MovieLens

1. Weighted Directed Graphs

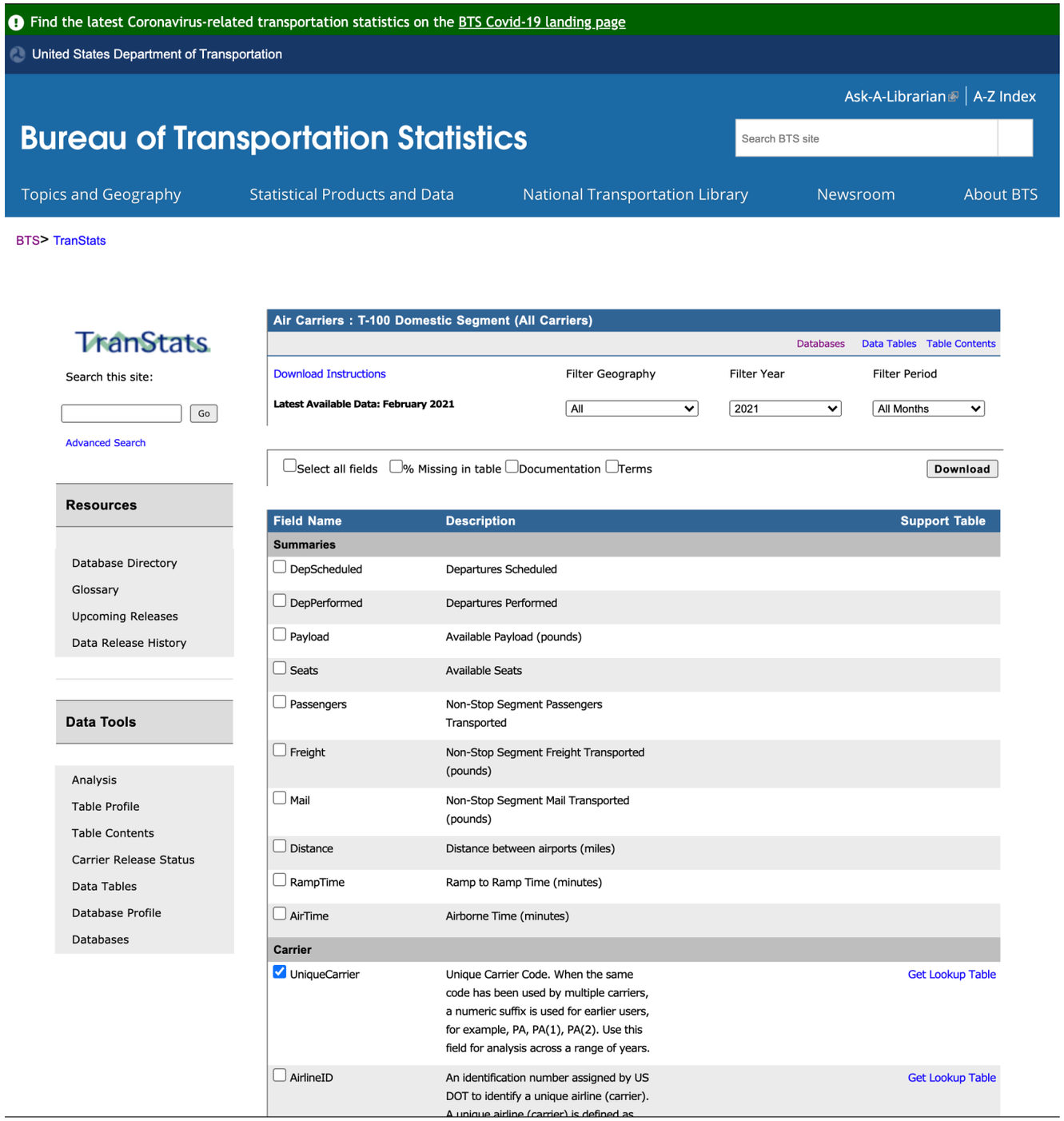
Mathematical Details

- Mathematical object, with set of nodes and edges
- Node - Individual Element
- Edge - Connection between element
- Degree - Number of edges connected to a node
- Weighted Edge - Edge with a weight associated
- Directed Edge - “One way street”



Airline Network

- The Bureau of Transportation Statistics provides a wealth of information about transportation in the US
- We'll look at the 'T-100 Domestic Segment table' that contains information on flights between any two airports serviced by a US Carrier



Airline Network

- Each row in this DataFrame corresponds to one edge in an 'edge list' format
- Airports correspond to individual nodes
- The database contains one row per individual flight so there are many repeated edges (which we can aggregate)
- Edges are directed as the number of people traveling from A to B is not the same that are traveling from B to A

	ORIGIN	DEST	PASSENGERS
9400	GCN	GCN	41012
2982	BLD	BLD	22881
19516	PGA	PGA	22466
14527	LKE	LKE	9876
15	1G4	1G4	3101
...	...	...	...
11877	ILI	ILI	0
11840	IGM	IGM	0
10628	HOU	HOU	0
10499	HNL	HNL	0
27230	YIP	YIP	0

Airline Network

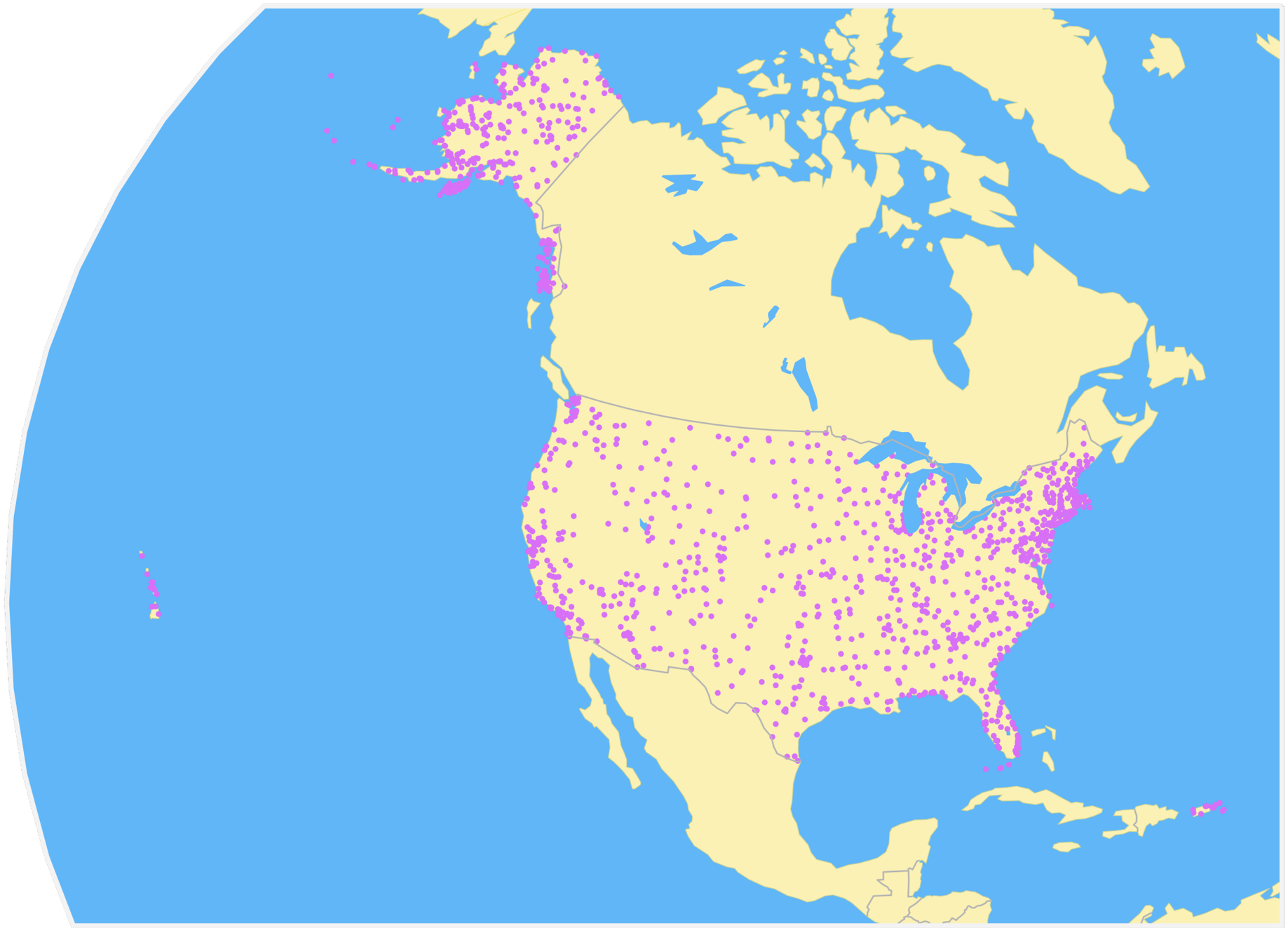
- Each row in this DataFrame corresponds to one edge in an 'edge list' format
- Airports correspond to individual nodes
- The database contains one row per individual flight so there are many repeated edges (which we can aggregate)
- Edges are directed as the number of people traveling from A to B is not the same that are traveling from B to A

	ORIGIN	DEST	PASSENGERS
9400	GCN	GCN	41012
2982	BLD	BLD	22881
19516	PGA	PGA	22466
14527	LKE	LKE	9876
		G4	3101
			...
11877	ILI	ILI	0
11840	IGM	IGM	0
10628	HOU	HOU	0
10499	HNL	HNL	0
27230	YIP	YIP	0

Weighted Directed Graph

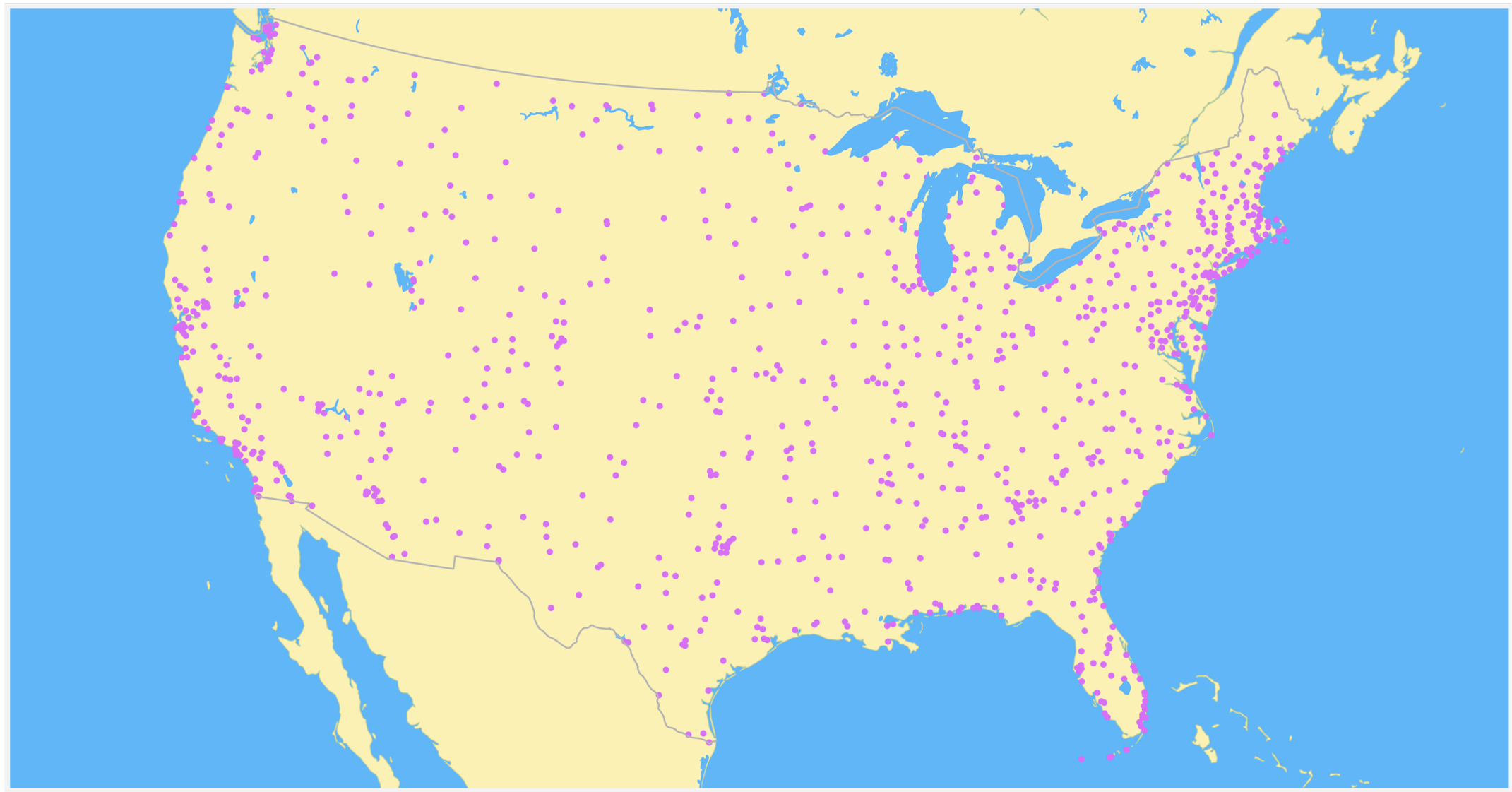
Nodes

- Nodes are airports



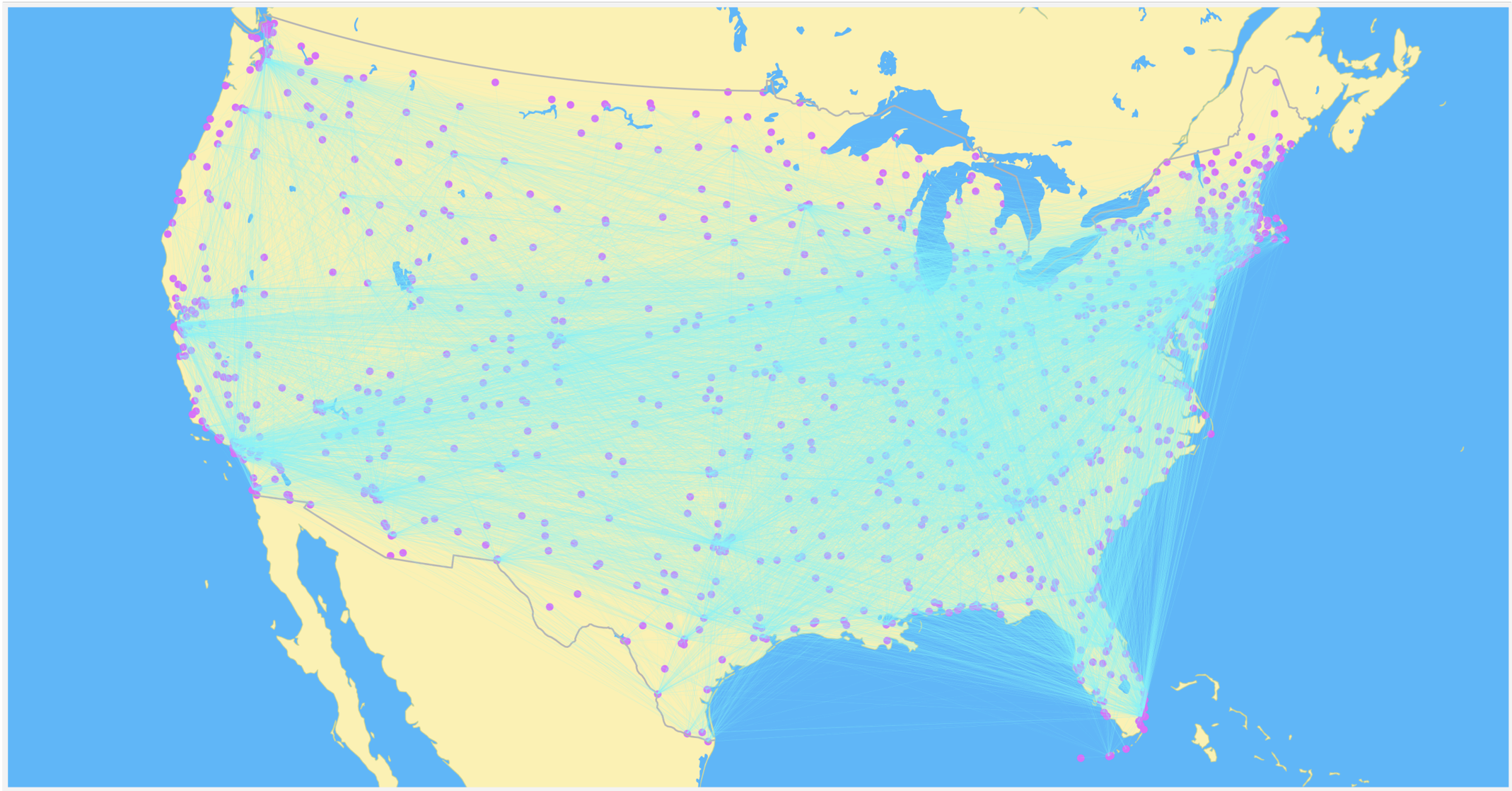
Nodes

- Nodes are airports
- We'll focus on the contiguous 48 states



Edges

- Edges are flights
- We aggregate all the flights going from airport A to airport B



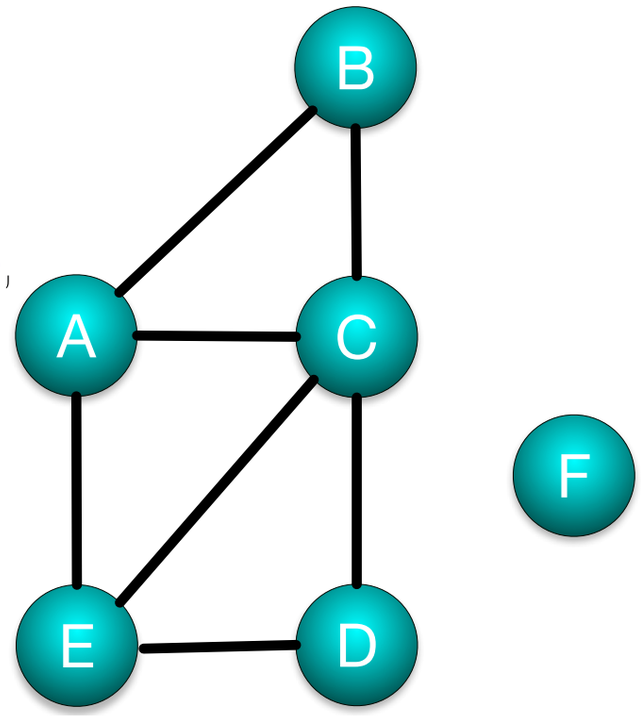
Degrees

- The most fundamental property of a node is its degree, k - the number of connections of a node

- In the case of **directed networks**, we distinguish between in-degree, k_{in} (number of incoming connections) and out-degree k_{out} (number of outgoing connections).

- Degree distribution - The relative frequency of each degree value:

$$P(k) = \frac{1}{N} \sum_{k_i \equiv k} 1$$

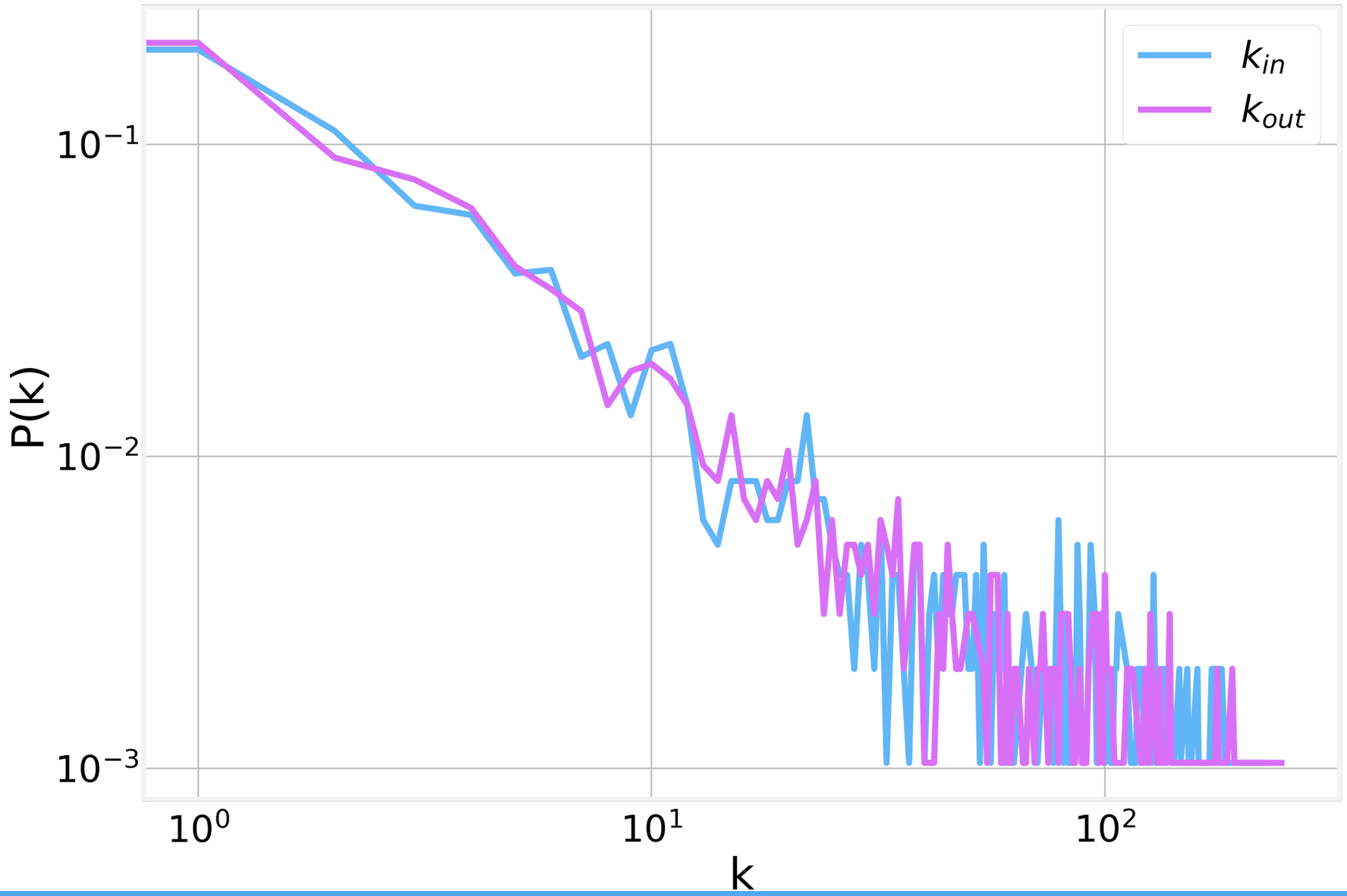


	Degree (k_i)
A	3
B	2
C	4
D	2
E	3
F	0

	P(k)
0	1/6
1	0
2	1/3
3	1/3
4	1/6

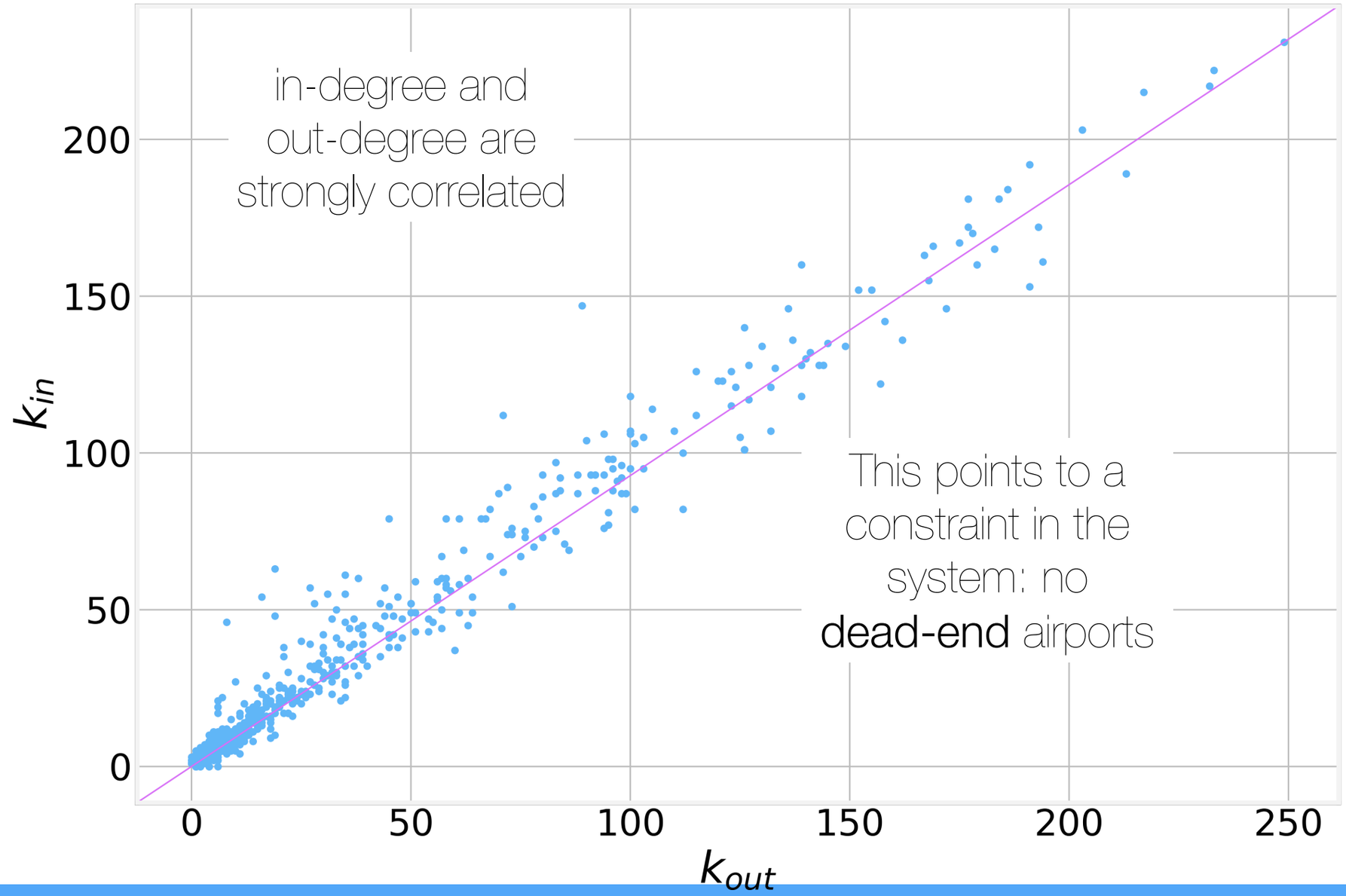
Degree Distribution

- Power-law distribution for both in- and out-degrees



in-out degree correlations

- Different networks will display different behaviors



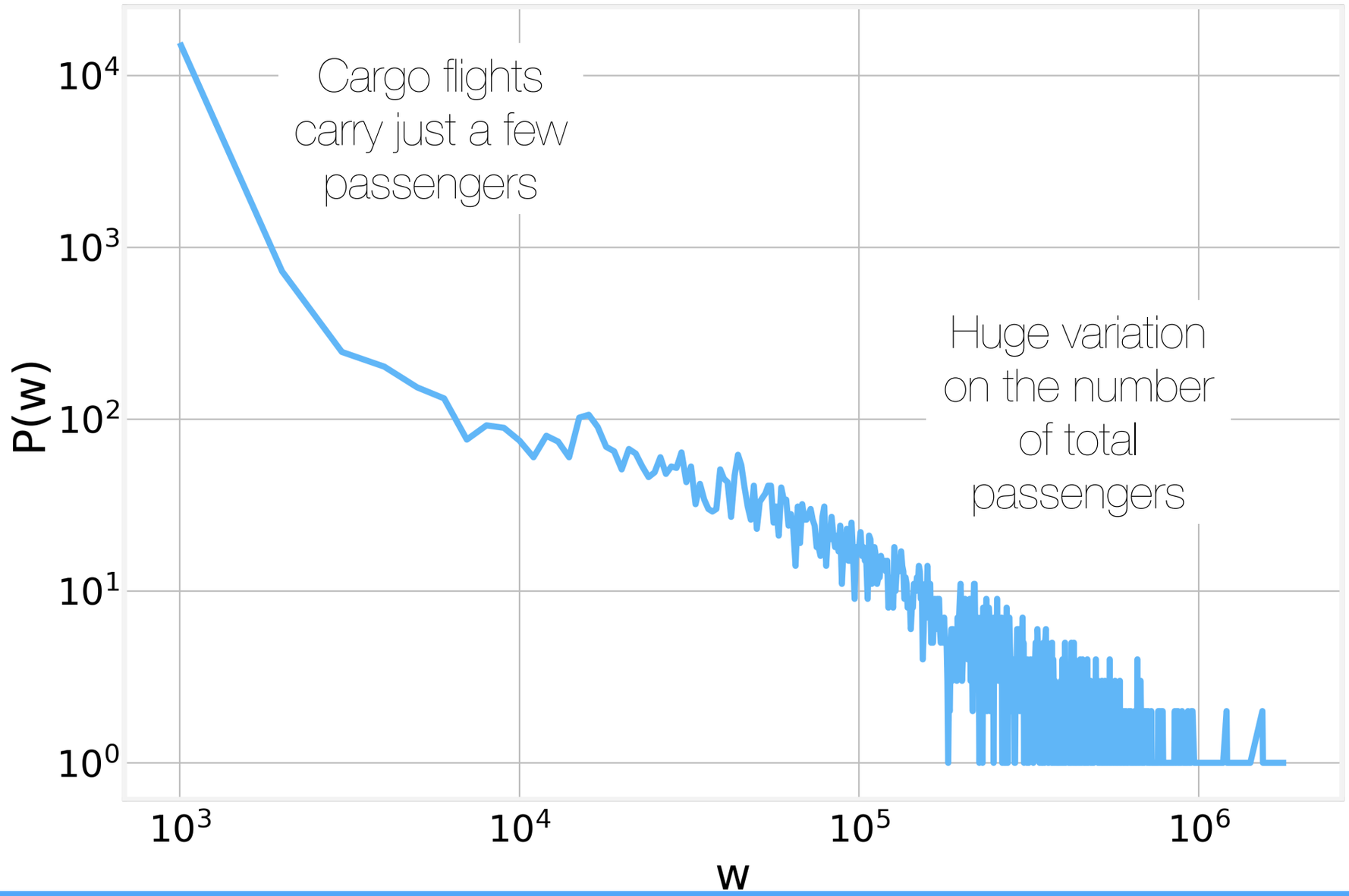
Edge Weight

- So far we have focused exclusively on the number of edges, but edges can also have weights

- Edge weights can represent:
  - connection strength (close friend vs acquaintance)
  - connection frequency (daily calls vs yearly meetings)
  - physical distance (flight time between cities)
  - etc…
- Node Strength:
  - The sum of the weights of all the nodes connected to the node
  - Weighted equivalent to node degree
- Weights are strongly related with network topology!

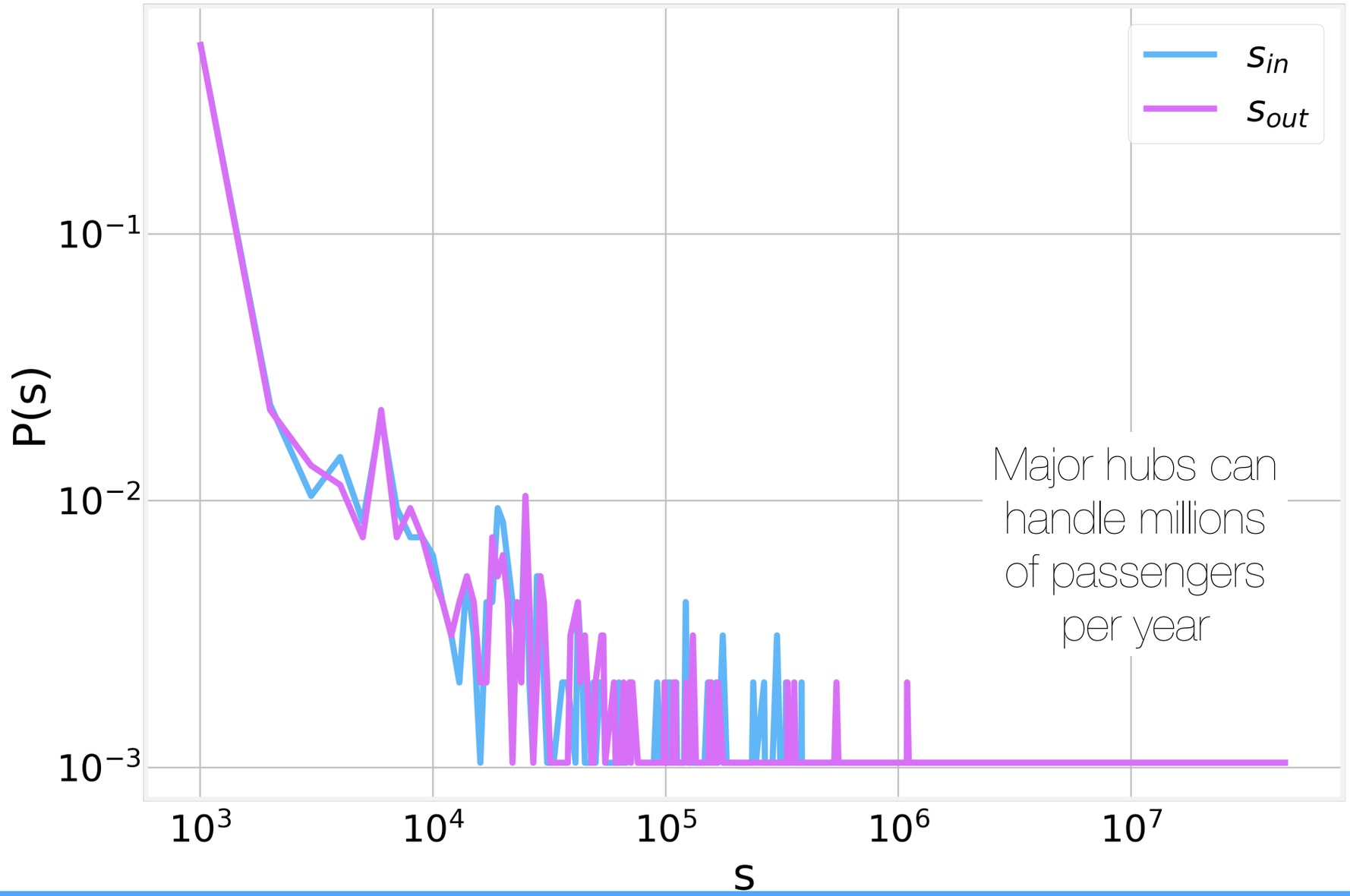
Edge Weight

- Weights are the number of passengers traveling from A to B
- Weight distribution is broad-tailed. Not necessarily power-law but allowing for large weights



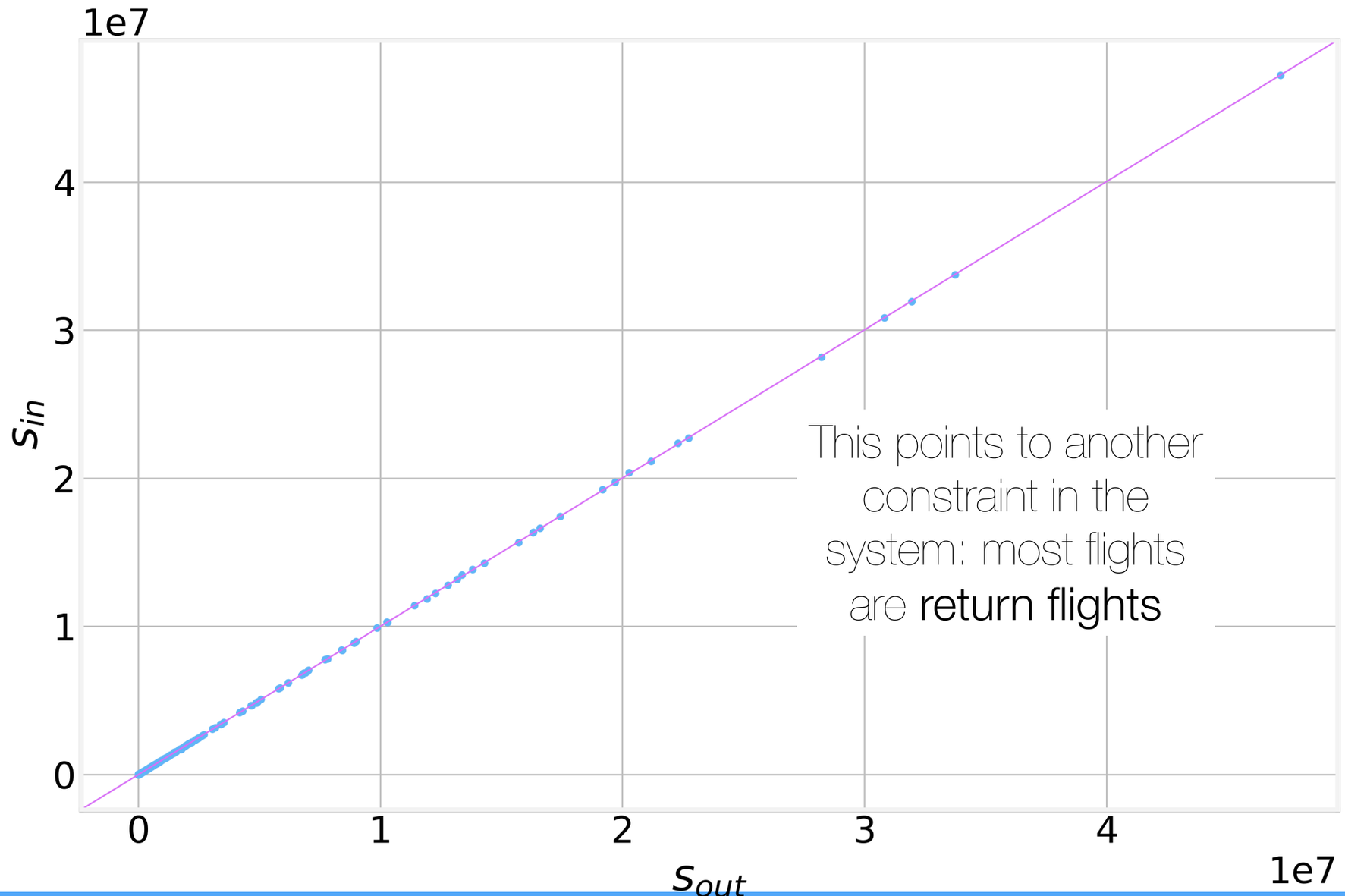
Strength

- Strength is the weighted equivalent of the degree
- We consider both in-strength and out-strength



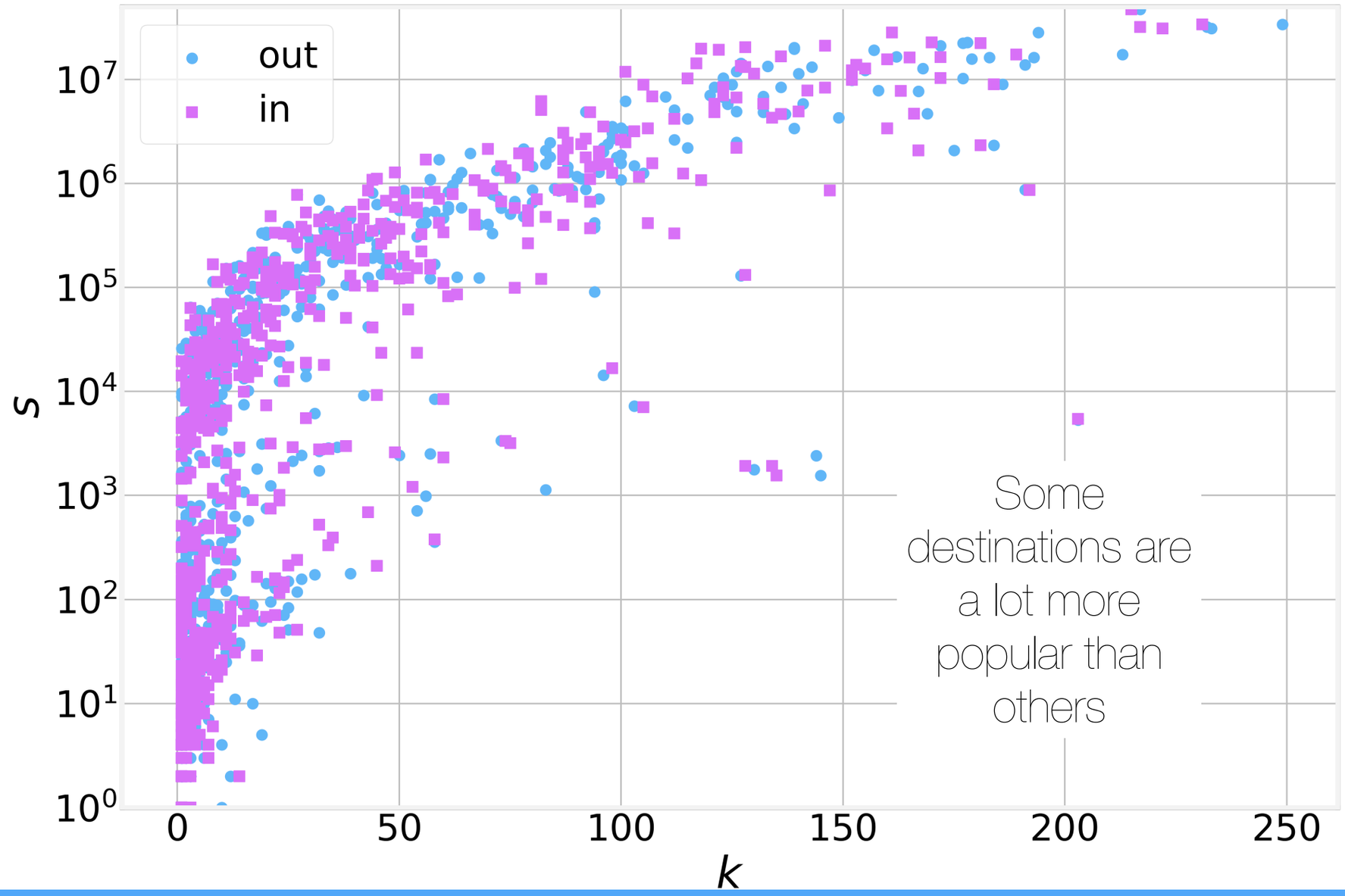
Strength

- Strong correlations between the in and out strength



Degree-Strength Correlations

- Airports with higher in- or out-degree also have higher in- out- strength but the dependency is not trivial



Code - Directed Weighted Graphs

https://github.com/DataForScience/G4DS

2. Navigating Graphs

OpenStreetMap

openstreetmap.org

- The Wikipedia of Maps
- Represents roads and intersections as edges and nodes, respectively

OpenStreetMap

From Wikipedia, the free encyclopedia

OpenStreetMap (**OSM**) is a collaborative project to create a free editable geographic database of the world. The geodata underlying the maps is considered the primary output of the project. The creation and growth of OSM has been motivated by restrictions on use or availability of map data across much of the world, and the advent of inexpensive portable satellite navigation devices.[5]

Created by Steve Coast in the UK in 2004, it was inspired by the success of Wikipedia and the predominance of proprietary map data in the UK and elsewhere.[6][7] Since then, it has grown to over two million registered users.[8] Users may collect data using manual survey, GPS devices, aerial photography, and other free sources, or use their own local knowledge of the area. This crowdsourced data is then made available under the Open Database License. The site is supported by the OpenStreetMap Foundation, a non-profit organisation registered in England and Wales.

The data from OSM can be used in various ways including production of paper maps and electronic maps, geocoding of address and place names, and route planning.[9] Prominent users include Facebook, Wikimedia Maps, Apple, Microsoft, Amazon Logistics, Uber, Craigslist, Snapchat, OsmAnd, Maps.me, MapQuest Open, JMP statistical software, and Foursquare. Many users of GPS devices use OSM data to replace the built-in map data on their devices.[10] OpenStreetMap data has been favourably compared with proprietary datasources,[11] although as of 2009 data quality varied across the world.[12][13][*needs update*]

Contents [hide]

1 History
2 Map production
  2.1 Software for editing maps
  2.2 Contributors
  2.3 Surveys and personal knowledge
  2.4 Street-level image data
  2.5 Government data
  2.6 Route planning
3 Usage
  3.1 Software for viewing maps
  3.2 Humanitarian aid
  3.3 Scientific research
4 "State of the Map" annual conference
5 Legal aspects
  5.1 Licensing terms
  5.2 Commercial data contributions
6 Operation
  6.1 Data format
  6.2 Data storage
  6.3 Popular services
7 See also
8 References
9 Further reading
10 External links

OpenStreetMap

OpenStreetMap's logo

Screenshot	[show]
Type of site	Collaborative mapping
Available in	UI: 96 languages and variants[1] Map data: Local languages
Owner	Community-owned; supported by OpenStreetMap Foundation[2]
Created by	Steve Coast (User page in OSM)
Products	Geographic data
URL	www.openstreetmap.org
Commercial	No
Registration	Required for contributors, not required for viewing
Users	7,450,000[3]
Launched	9 August 2004; 17 years ago[4]
Current status	Active (details)
Content license	Open Database License (ODbL)

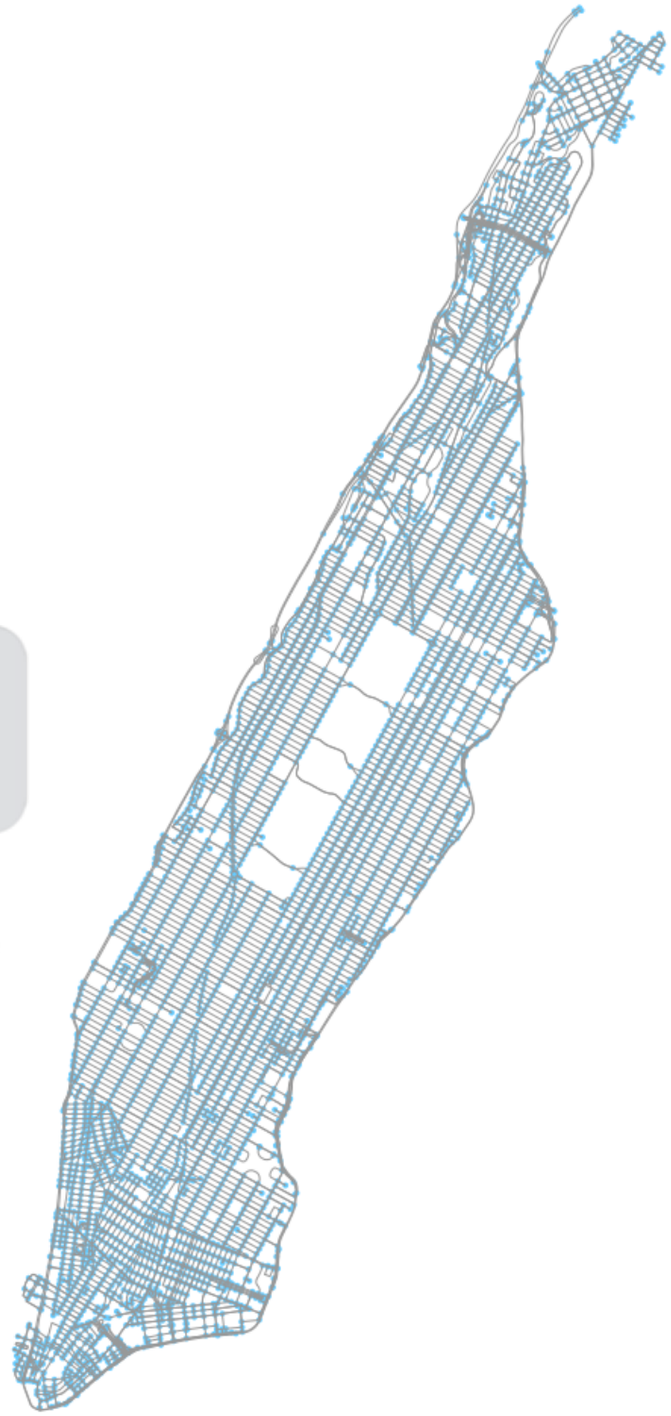
osmnx

https://github.com/gboeing/osmnx

- Python package to easily download OpenStreetMap data
- Builds on top of GeoPandas, NetworkX, and matplotlib
- Supports different modes of transportation (driving, walking, etc)
- We can obtain the driving network for a specific region with just a couple lines of code

```
import osmnx as ox

place = 'Manhattan, NY, USA'
G_roads = ox.graph_from_place(place, network_type='drive')
```

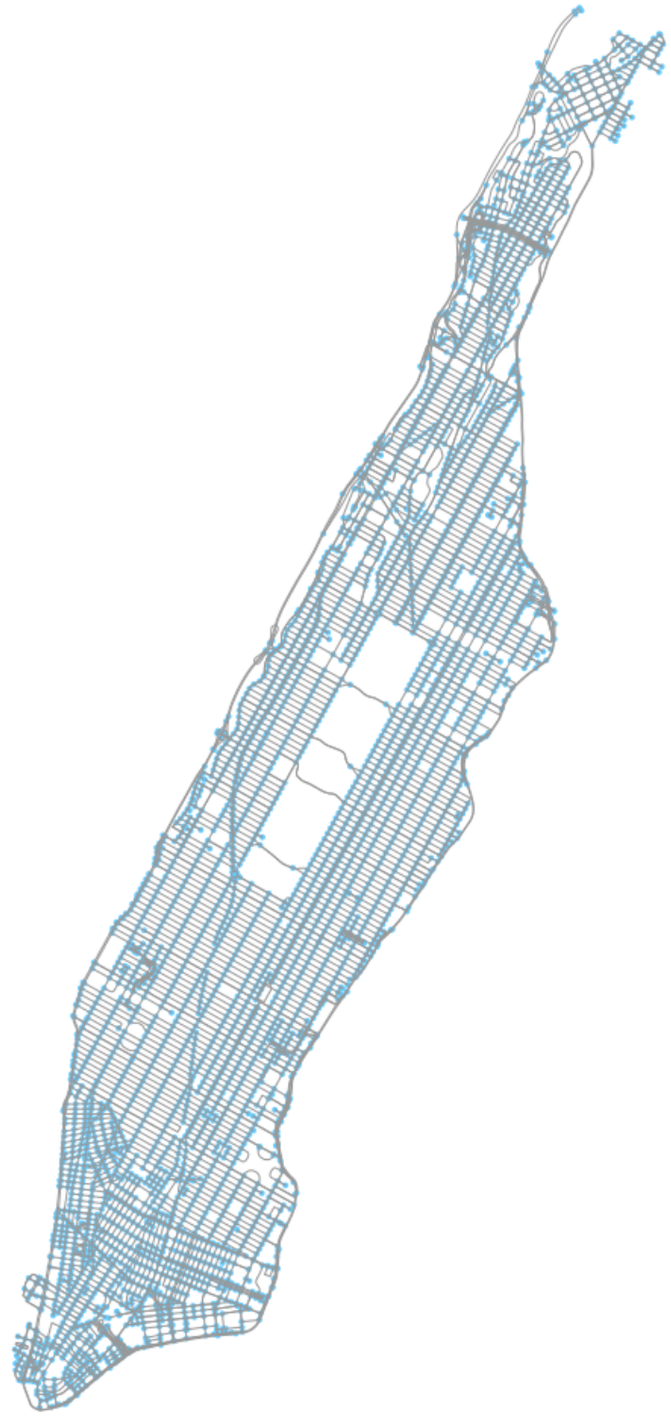
- Returns a NetworkX object that provides us with a simple interface to manipulate the graph



Shortest paths and Distances

https://github.com/gboeing/osmnx

- A common problem in graph analysis is to determine the shortest path between two given nodes:
  - Minimum number of hops on a computer network
  - Fastest drive path between two different locations on a city
  - etc
- There's a very rich literature on algorithms to identify the shortest paths and to measure the shortest distance between two nodes on a graph:
  - Breadth-First Search - Explore all the local neighbors first
  - Depth-First Search - Keep going until you hit a dead-end then backtrack
  - Dijkstra's Algorithm - Keep Extending the current shortest path

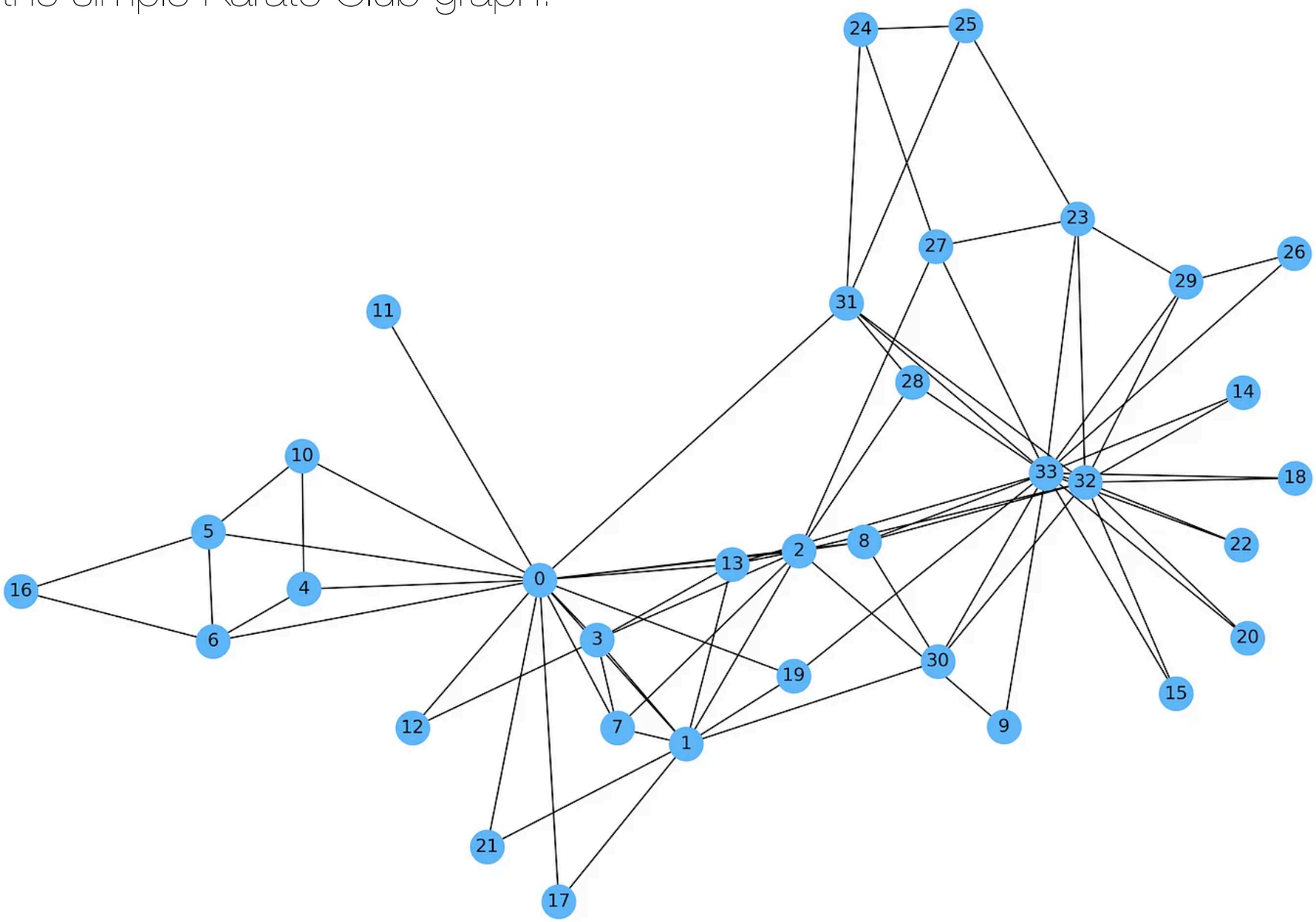


Depth First Search

- A simple idea:
    - Keep following edges until you hit a dead end
    - Backtrack until you find more edges to follow.
- Guaranteed to find a path between the root and every other node
- Paths tend to be unnecessarily long

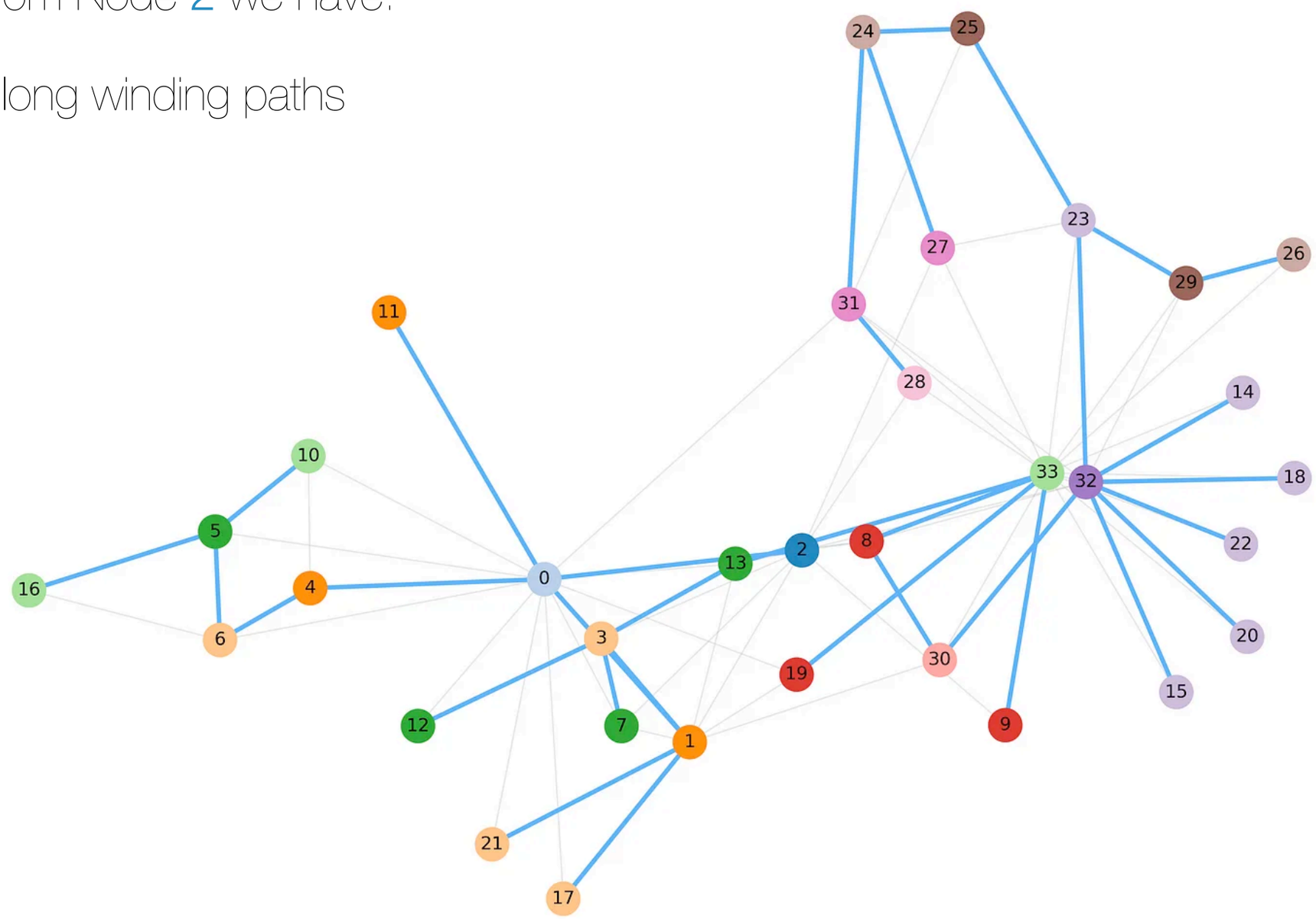
Depth First Search

- Consider the simple Karate Club graph:



Depth First Search

- Starting from Node 2 we have:
- Note the long winding paths



Breath First Search

- Improvement on the DFS algorithm by visiting all of a node's neighbors before moving on to the next node.

- Guaranteed that we find the shortest possible path between the root and every other node.

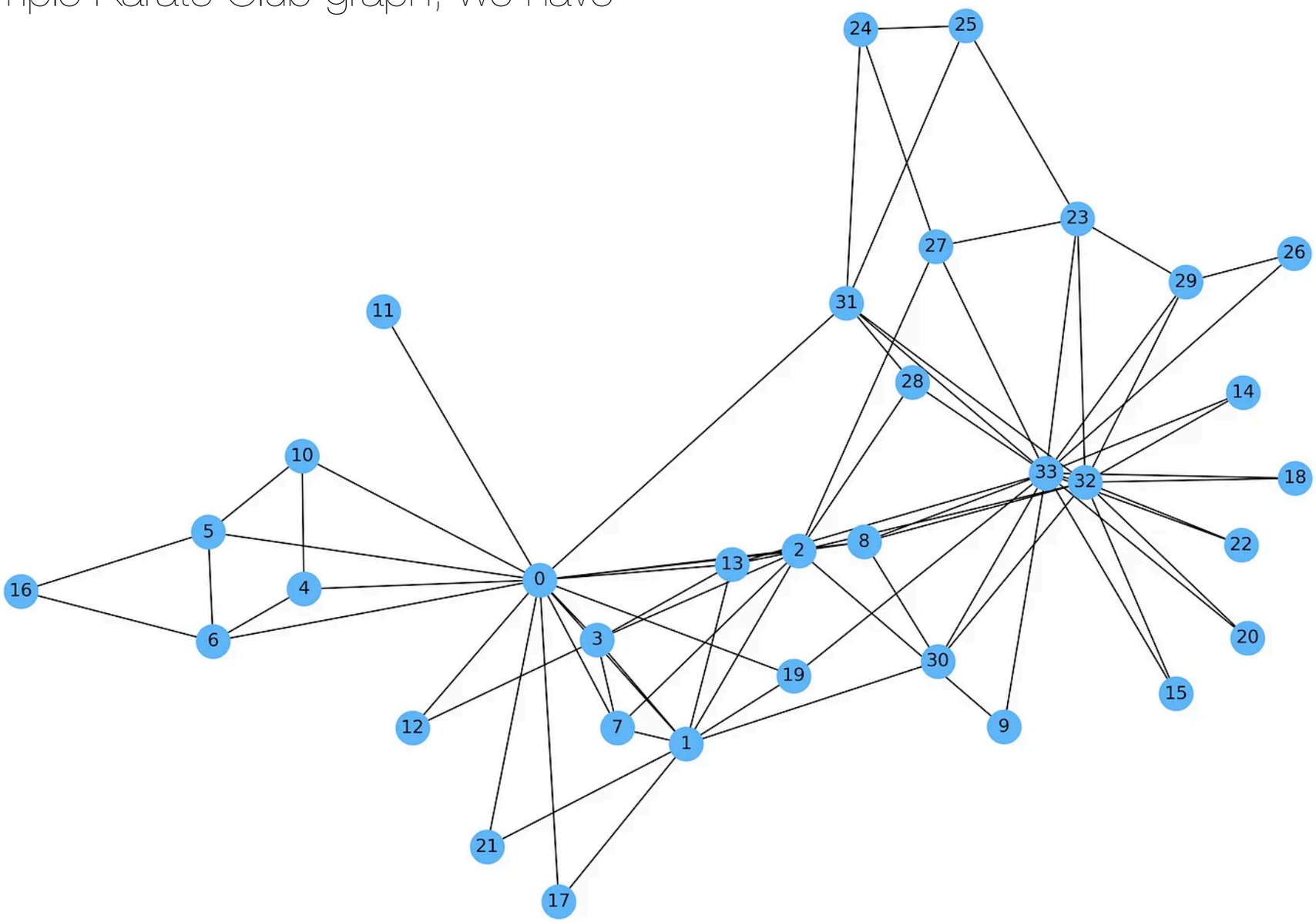
- Distance is defined purely as the number of hops between two nodes.

- Use a priority Queue to make sure we always visit the nodes in order

- BFS graphs as unweighted (every edge has the same length)

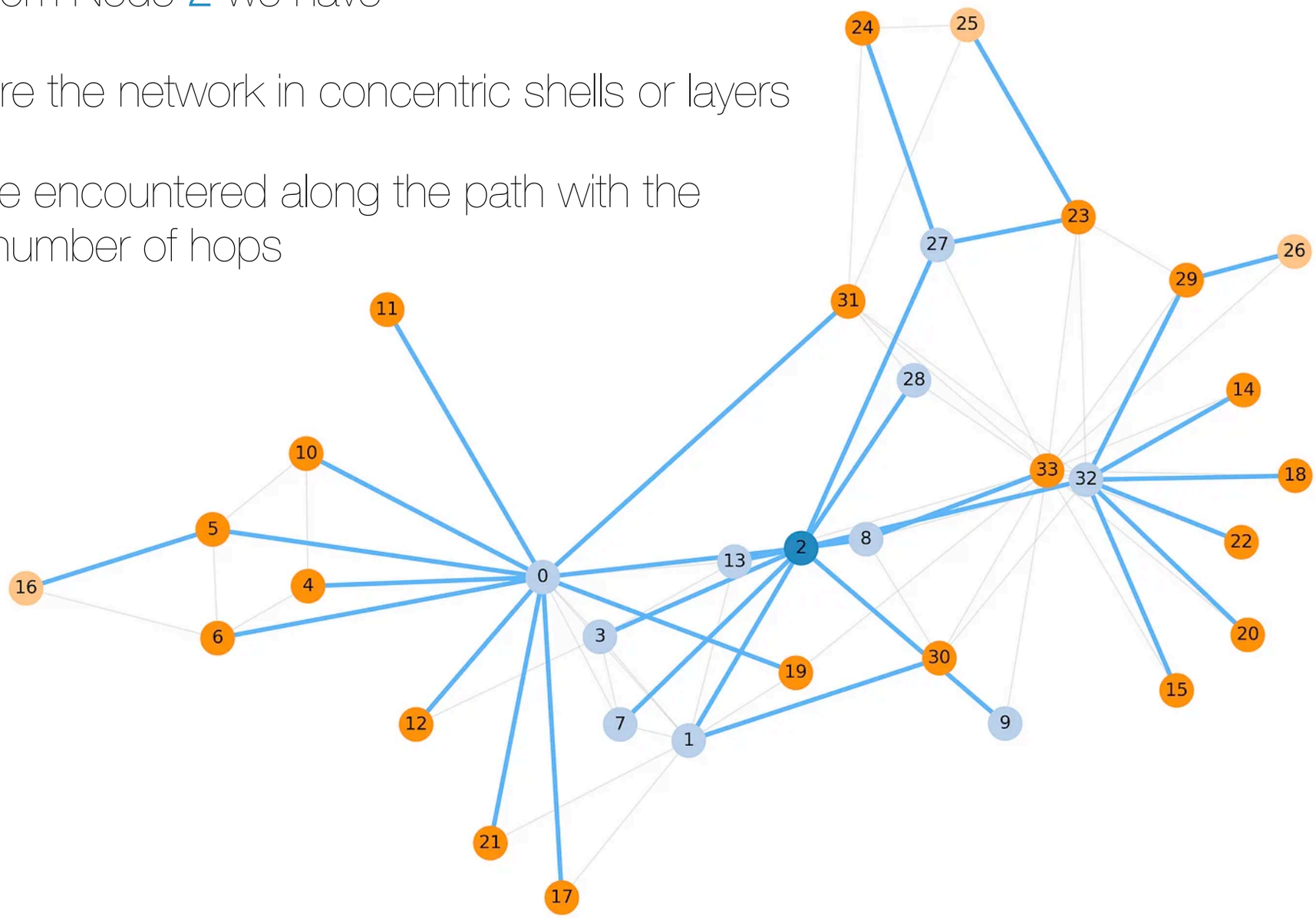
Breath First Search

- For the simple Karate Club graph, we have



Breath First Search

- Starting from Node 2 we have
- We explore the network in concentric shells or layers
- Nodes are encountered along the path with the shortest number of hops



Dijkstra's Algorithm

- Algorithm created by Edsger W. Dijkstra, a Dutch computer scientist, in 1956 to find the shortest paths between nodes on a weighted graph

- Most common formulation finds the shortest path between a source node and every other node on the graph

- Dijkstra's algorithm proceeds in an incremental way in which we continuously update our best estimate for the distance between the origin and every other node. Define each node to be at an unknown (or infinite) distance from the origin. The origin is defined to be at distance 0 from itself.

en.wikipedia.org/wiki/Dijkstra's_algorithm

Dijkstra's Algorithm

- At each step:
  - Find the node i with the smallest current distance to the origin
  - Update the distance to each of it's nearest neighbors, j

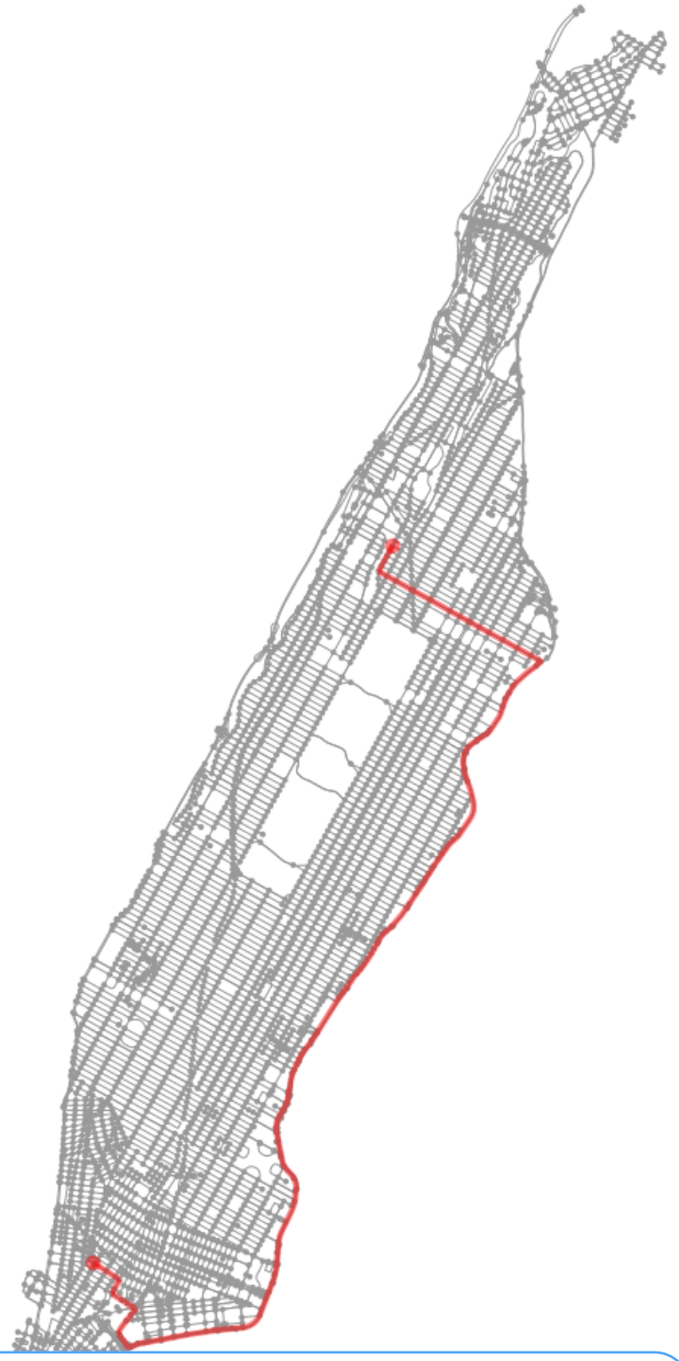
$$dist\left(node_j\right) = \min\left[dist\left(node_j\right), dist\left(node_i\right) + w_{ij}\right]$$

  - Continue until the destination has been reached or we have calculated the distance to every other node
  - The shortest path can be calculated as well by tracking which was the node that resulted in a distance update

en.wikipedia.org/wiki/Dijkstra's_algorithm

Dijkstra's Algorithm

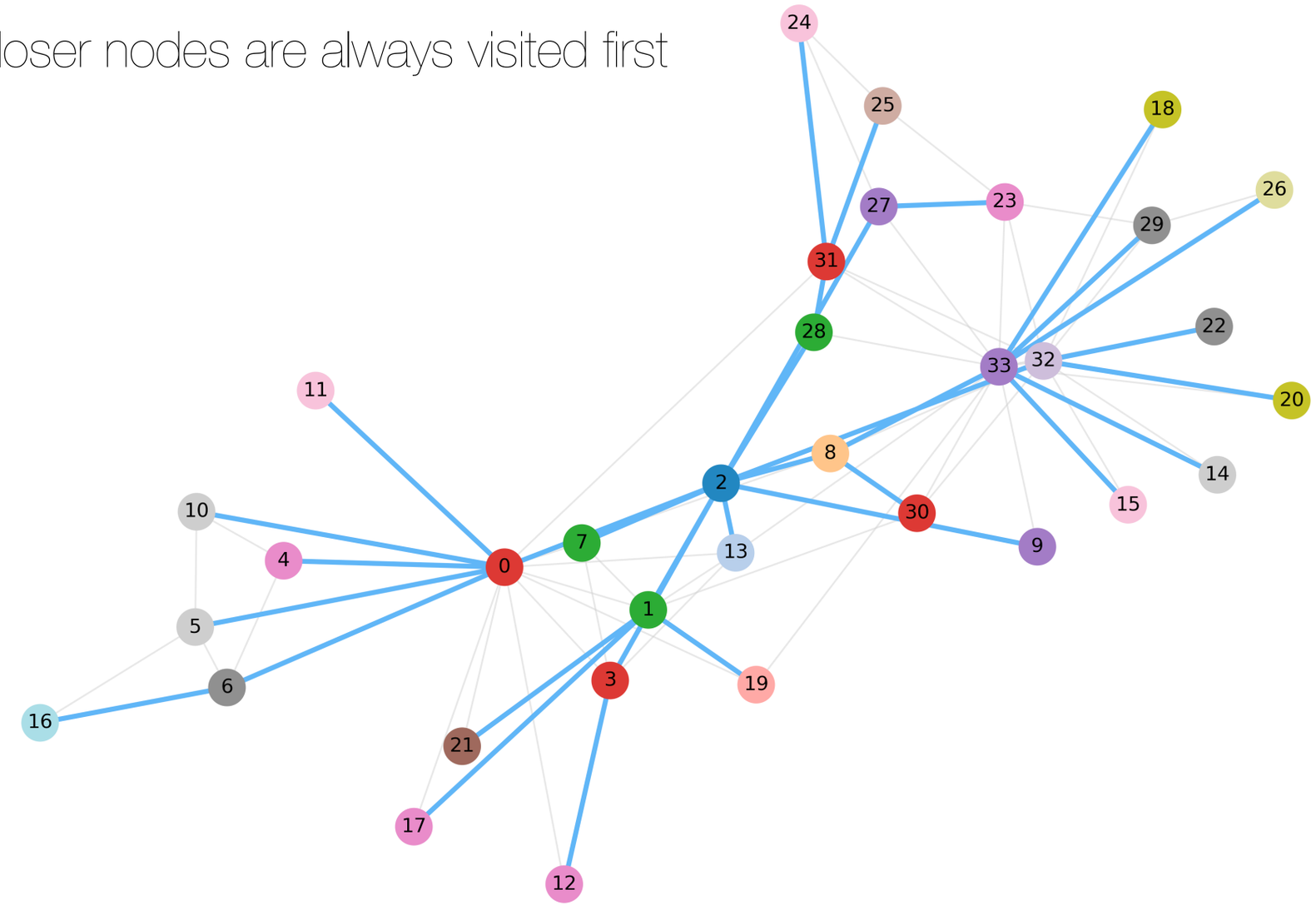
- Assumes weighted graphs
    - Unweighted graphs can be handled by assigning a weight of 1 to each edge
- All weights must be positive. Negative weights will cause infinite loops as you can always reduce your current "distance" estimate by traversing a negative weight edge.
- Can be particularly slow on some topologies.
- In unweighted graphs, it becomes similar to Breath First Search



en.wikipedia.org/wiki/Dijkstra's_algorithm

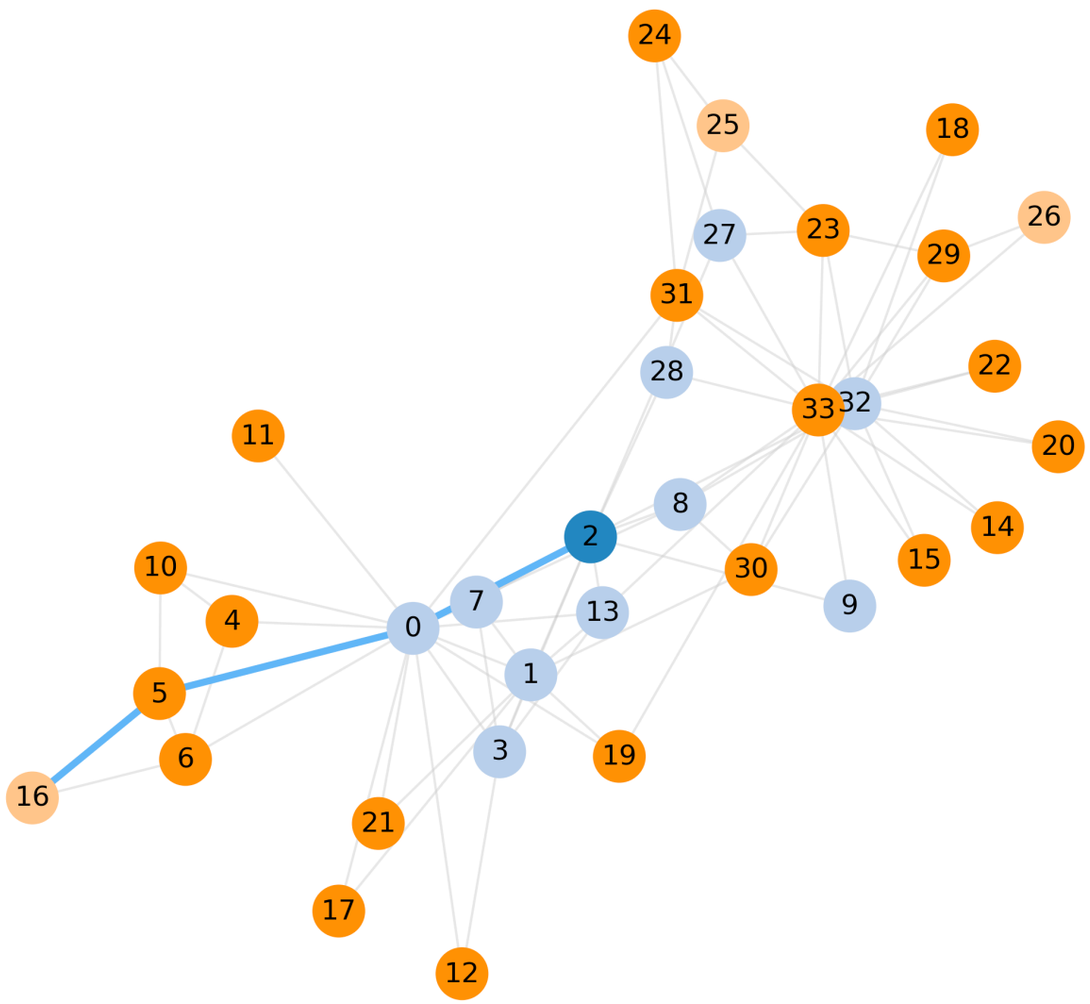
Dijkstra's Algorithm

- Here we use the Euclidean distance between the nodes as the weight
- Note how closer nodes are always visited first

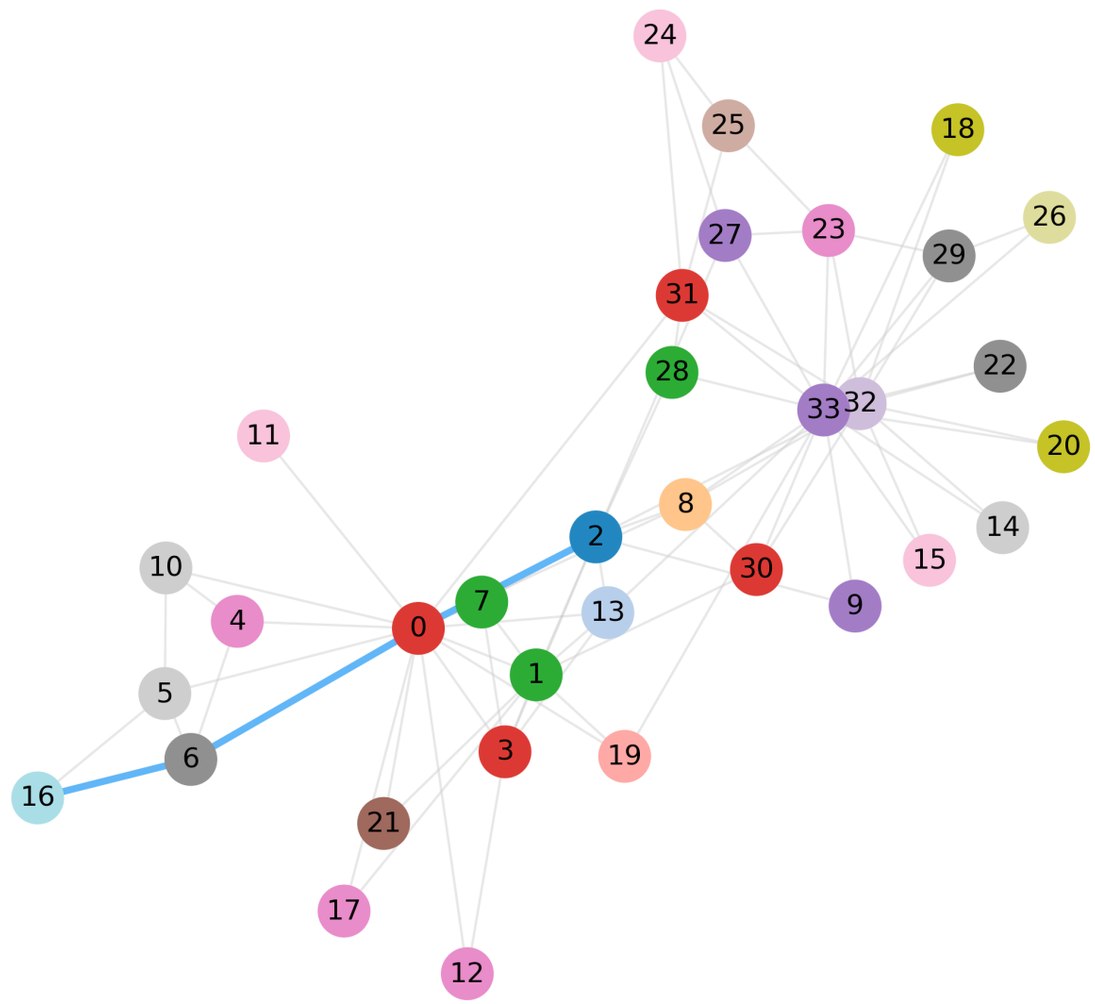


BFS vs Dijkstra's Algorithm

BFS



Dijkstra



Driving directions

- Applying Dijkstras algorithm to the OpenStreetMap graph, we're able to obtain driving directions

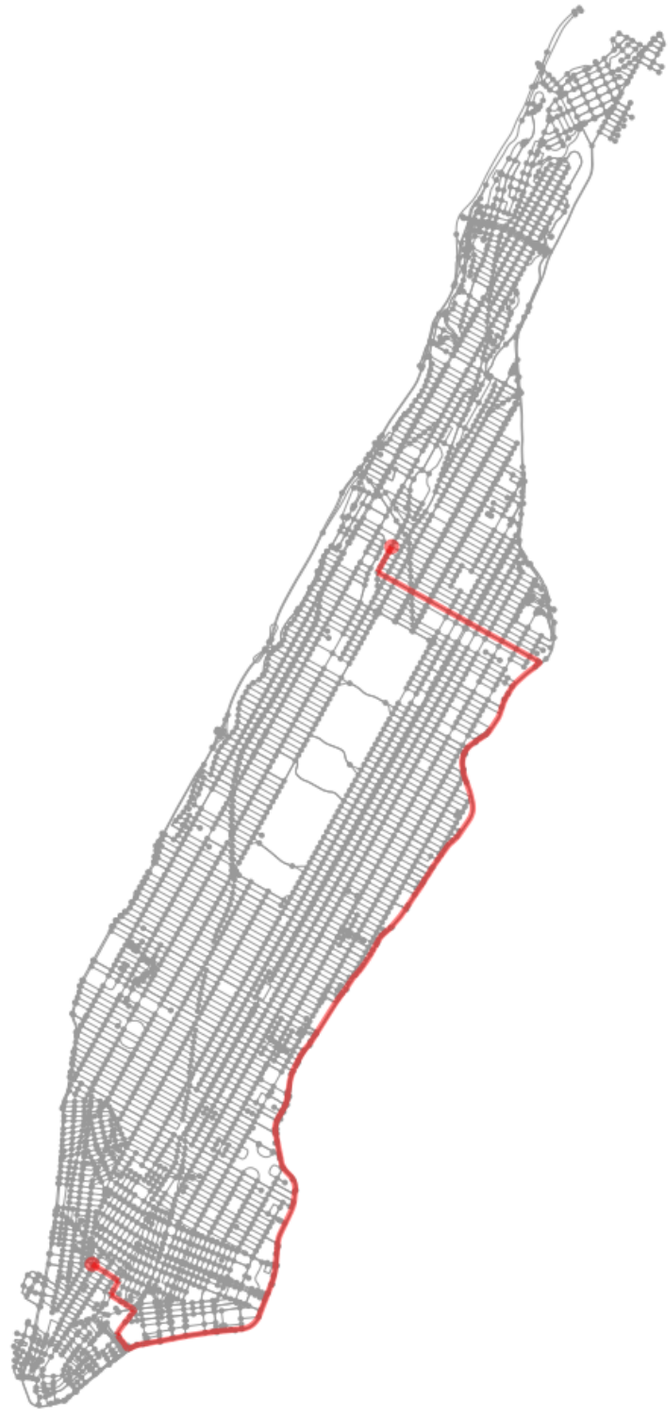
- The simplest case is to use the physical distance between two intersections as the weight, but there are also other options:

    - Type of traffic allowed (cars, bikes, pedestrians, etc)

    - Travel time (longer stretches of road with higher speed limits might be preferable)

    - Congestion levels at certain times of the day

    - etc



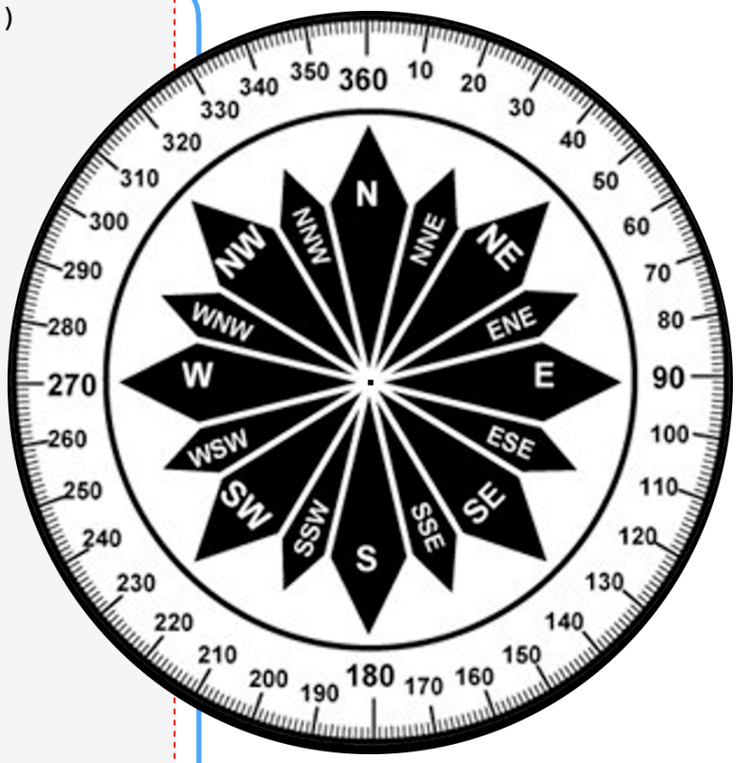
Driving directions

- osmnx also provides us with the "bearing" of each edge, the angle the edge is traveling with respect to North

- By checking the change in bearing from one edge to another, we can detect when we turn left or right

- The combination of changes in bearing with changes in street names allow us to provide turn by turn directions

```
print("Drive %1.1fm down %s" % (directions[0][1], directions[0][0]))

for i in range(1, len(directions)):
    change = directions[i][2]-directions[i-1][2]

    # Find the shortest difference between the two bearings
    if change < -180:
        change += 360
    elif change > 180:
        change -= 360

    if np.abs(change) < 5:
        msg = 'Continue onto'
    elif change > 0:
        if change > 25:
            msg = 'Turn right onto'
        else:
            msg = 'Turn slight right onto'
    elif change < 0:
        if change < -25:
            msg = 'Turn left onto'
        else:
            msg = 'Turn slight left onto'

    print('%s %s and drive for %1.1fm' % (msg, directions[i][0],
                                          directions[i][1]))

print("Arrive at your destination")
```



Code - Navigating Graphs

https://github.com/DataForScience/G4DS

3. Patterns and Structure in Graphs

BitCoin

- Introduced by Satoshi Nakamoto in 2009
- Provided the first anonymous peer-to-peer blockchain based crypto-currency, initiating the crypto-currency revolution
- Currently still the dominant crypto-currency with the largest market capitalization

Cryptos: 11,746 Exchanges: 405 Market Cap: $2,079,319,605,905 24h Vol: $227,858,546,811 Dominance: BTC: 42.5% ETH: 19.5% ETH Gas: 208 Gwei English USD

CoinMarketCap Cryptocurrencies Exchanges NFT Portfolio Watchlist Calendars Products Learn Log In Sign up Search /

🎉 Want to take a sneak peek of our enhanced homepage? Yes, switch to new home Maybe later

Today's Cryptocurrency Prices by Market Cap

The global crypto market cap is $2.08T, a 11.83% decrease over the last day. Read More

News of the Day
Sept. 7: Bitcoin Crashes!

Free Airdrops
Join $21,000 SHREW Airdrop!

Watchlist Portfolio | Cryptocurrencies Categories DeFi NFT Play To Earn Polkadot BSC Solana Avalanche Show rows 100 Filters Customize

#	Name	Price	24h %	7d %	Market Cap	Volume(24h)	Circulating Supply	Last 7 Days
1	Bitcoin BTC Buy	$46,927.64	9.30%	0.90%	$881,962,716,418	$65,141,664,514 1,389,263 BTC	18,809,437 BTC	
2	Ethereum ETH Buy	$3,438.36	12.78%	0.73%	$403,801,281,682	$37,572,720,761 10,926,484 ETH	117,429,028 ETH	
3	Cardano ADA	$2.40	15.33%	13.53%	$76,700,799,428	$10,901,856,290 4,552,056,680 ADA	32,026,324,425 ADA	

Kaggle Bitcoin Transaction Dataset

https://www.kaggle.com/xblock/bitcoin-partial-transaction-dataset

Description

The Bitcoin Partial Transaction Datasets contain three snapshots of Bitcoin transaction data for easier analysis, namly dataset1*2014*11*1500000,* *dataset2*2015*6*1500000 and dataset3*2016*1_1500000. We sample the snapshots from November 2014 to January 2016 with six months as the sampling interval. Each snapshot contains the first 1,500,000 transaction records in its corresponding month, namly Nov. 2014, Jun. 2015 and Jan. 2016.

We also provide a file including the labeled addresses belonging to mixing services, and these addresses were active during the observing time of our snapshots.

Due to the pseudonymous requirements of Bitcoin, it is unlikely to enforce Know-Your-Customer (KYC) processes, which are guidelines in anti-money laundering. However, mixing services in Bitcoin, originally designed to enhance transaction anonymity, have been widely employed for money laundry to complicate trailing illicit fund.

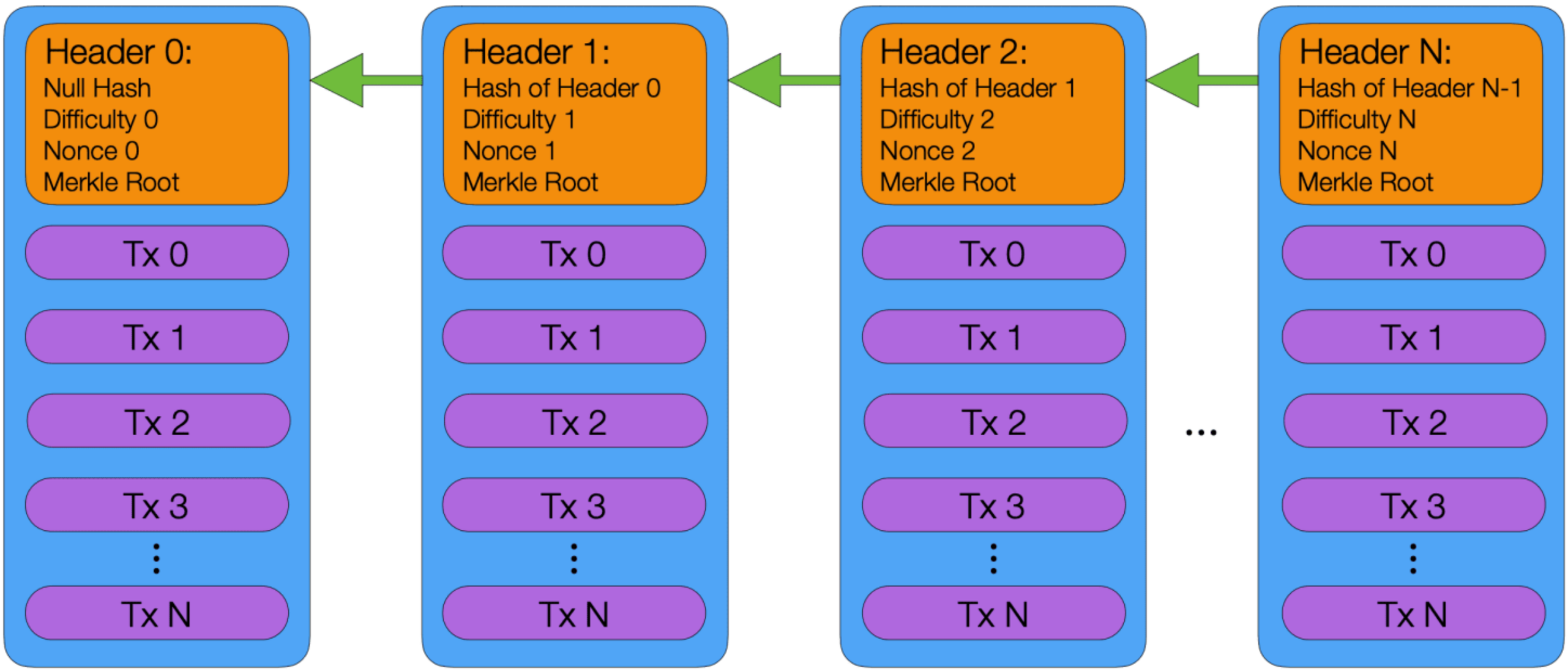
In our work, we study mixing service detection with this dataset. For further study, we can chase up users involved in criminal activities by analyzing users who take part in Bitcoin mixing.

The details of dataset1*2014*11*1500000 are described below. The file structure of dataset2*2015*6*1500000 and dataset3*2016*1_1500000 are similar to EthereumG1. You can know more information from the README file.

For more details about blockchain dataset, please click here.

Blockchain

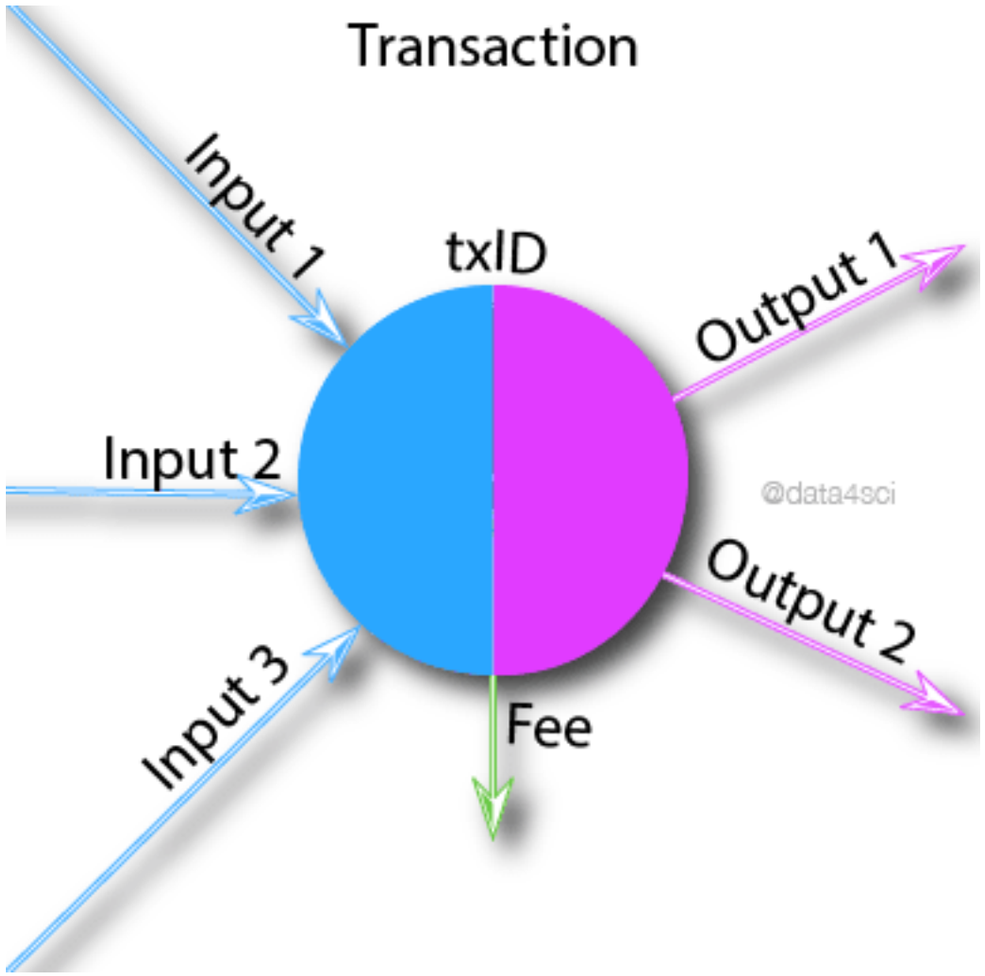
- Each Block contains a



Transactions

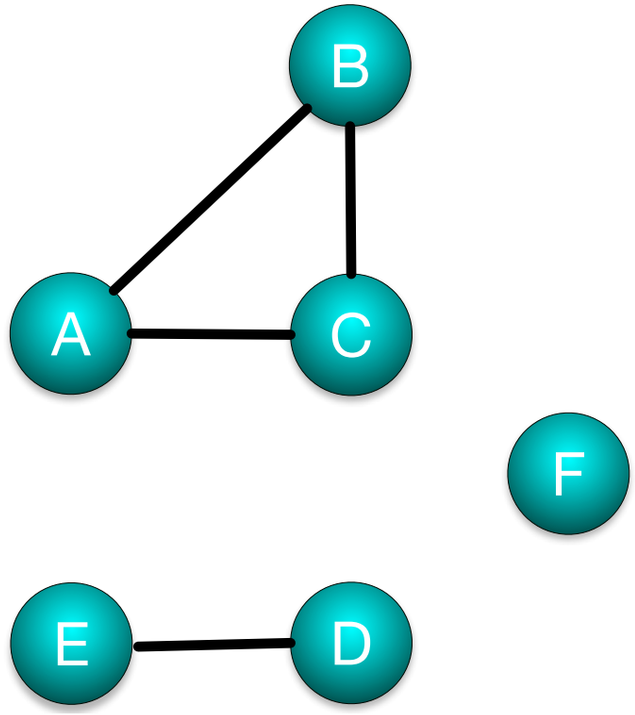
- A transaction (node) combines amounts (weights) associated with inputs (incoming edges) to produce outputs (outgoing edges)

- Outputs of one transaction can be spent as inputs to another



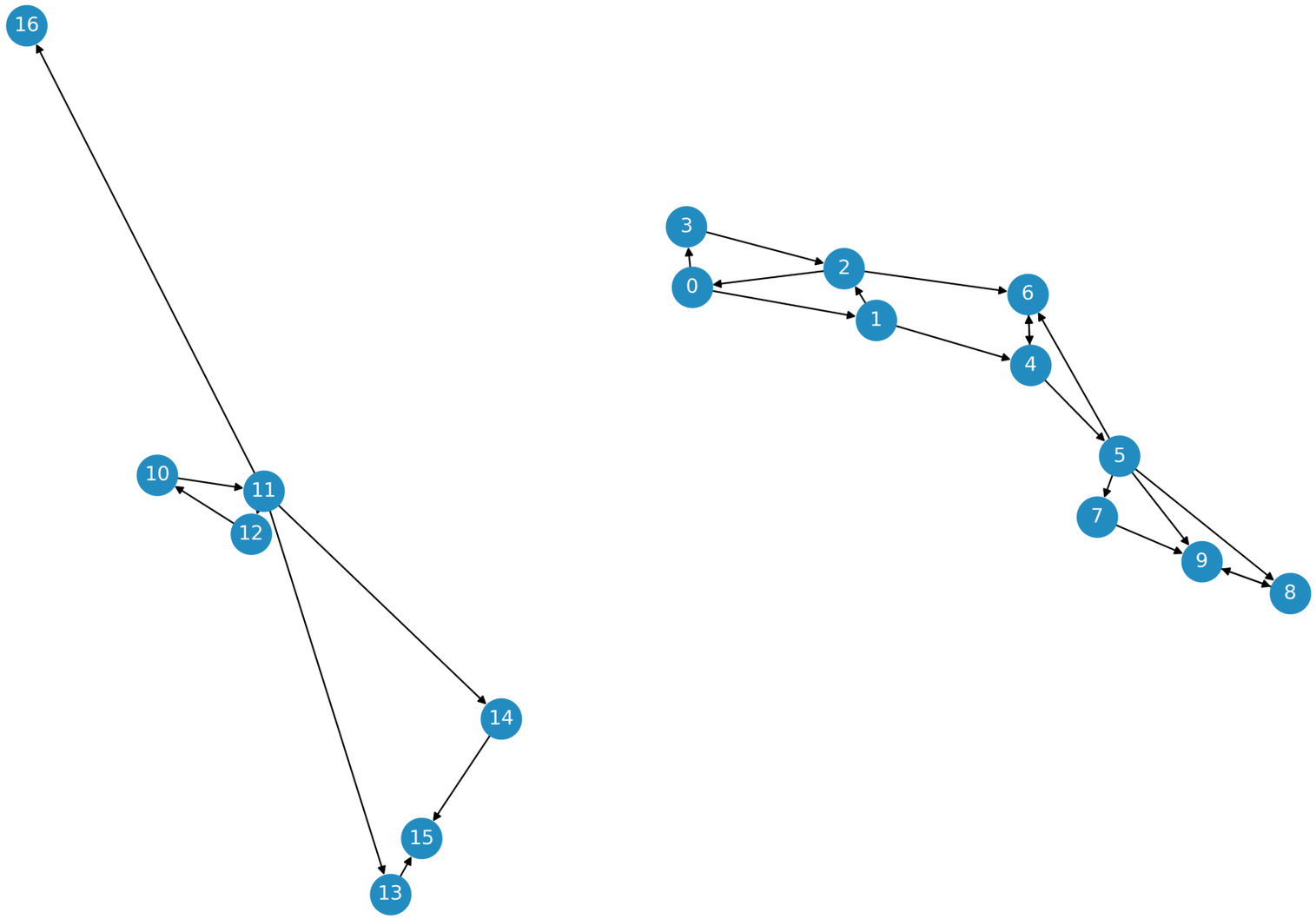
Components

- As we saw, a graph is a set of nodes connected by edges
- However, it is possible to have disconnected nodes (F) and even larger disconnected parts of the network (E and D)
- Each piece of the graph is called a Component.
- When the largest component accounts for a substantial part of the nodes of the graph, it's called the Giant Connected Component.
- Graphs with a single component are called Connected
- In the case of directed graphs, we consider two cases:
  - Strongly-connected - Every node is accessible from every other node following the direction of the edges
  - Weakly-connected - Every node is accessible by disregarding edge directions



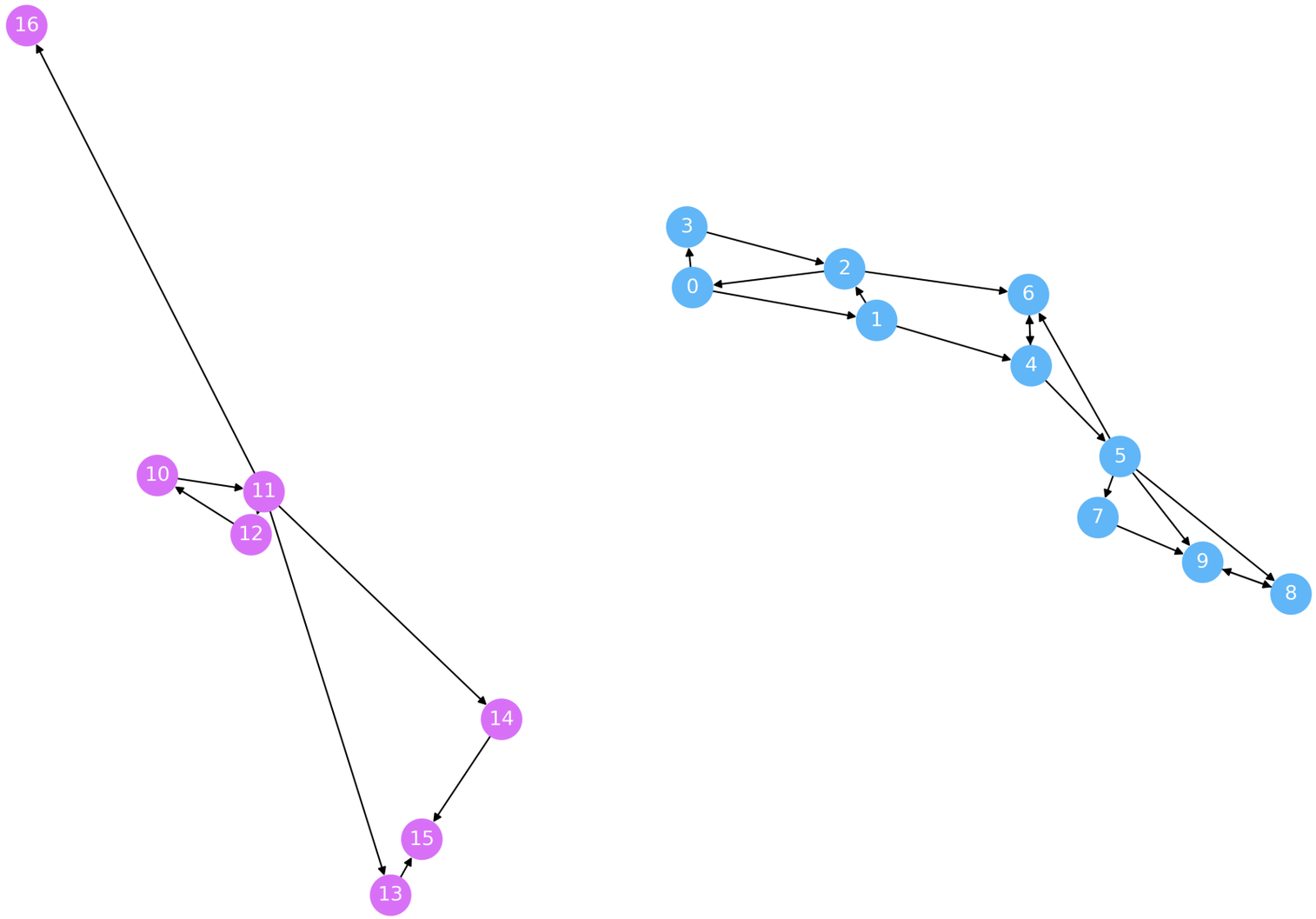
Components

- Let us consider a simple network



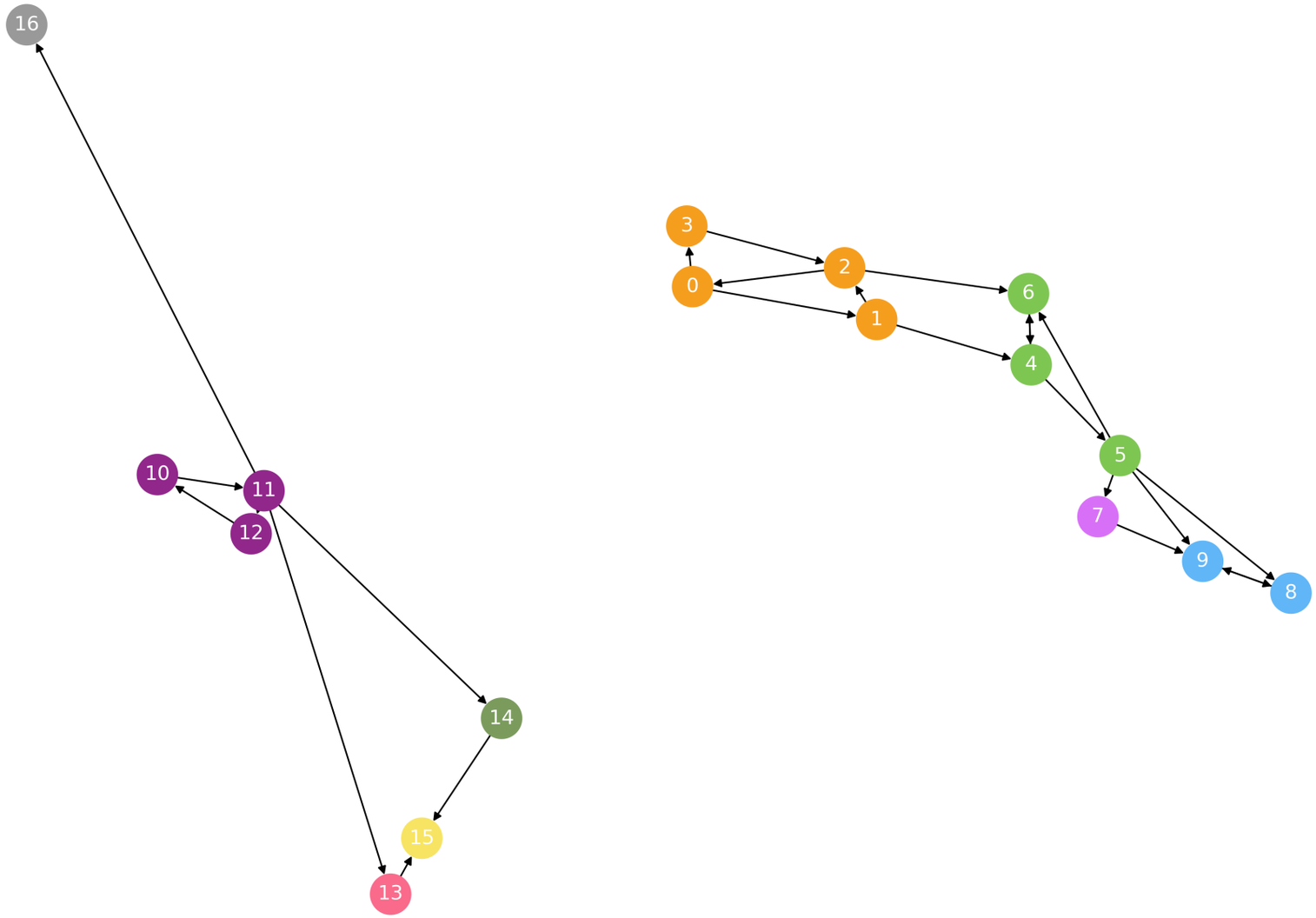
Weakly Connected Components

- The weakly Connected Components are just the components that are isolated from each other



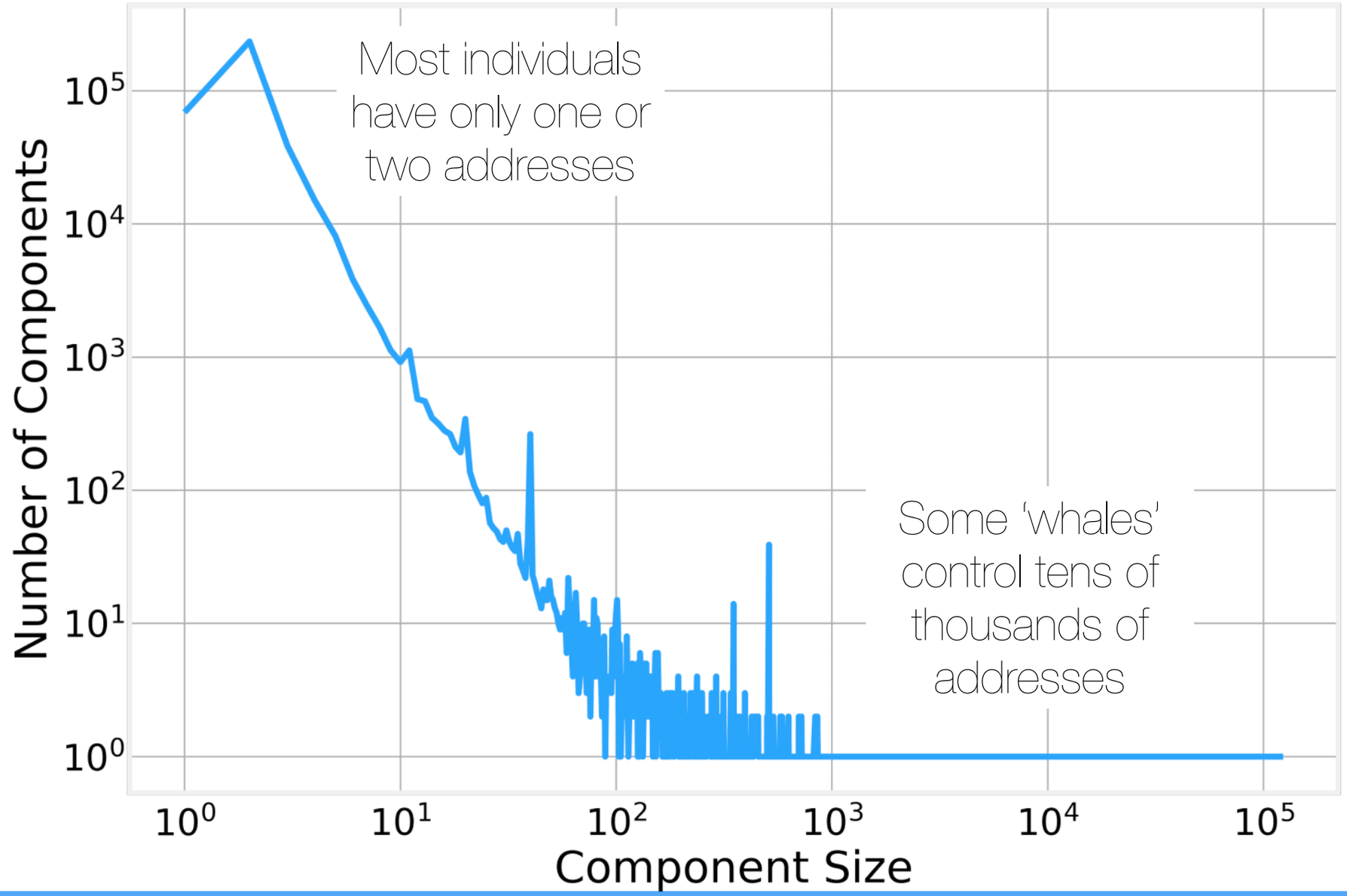
Strongly Connected Components

- Each Weekly Connected Component contains one or more Strongly Connected Components where each node is reachable from every other node along the direction of the edges



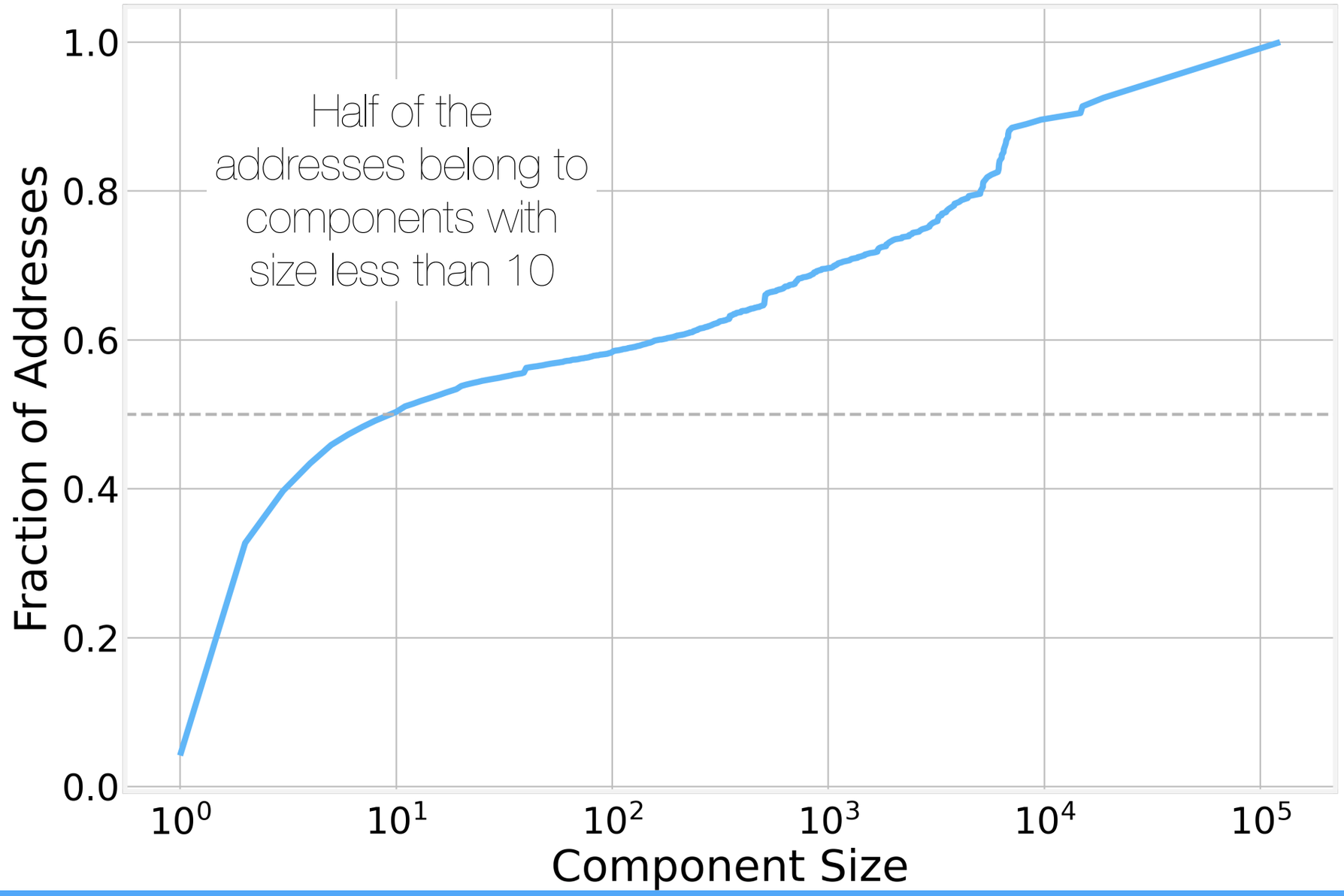
Wallets

- Wallets are addresses controlled by the same individual
- We can identified them as the Weekly Connected Components in our transaction network



Wallets

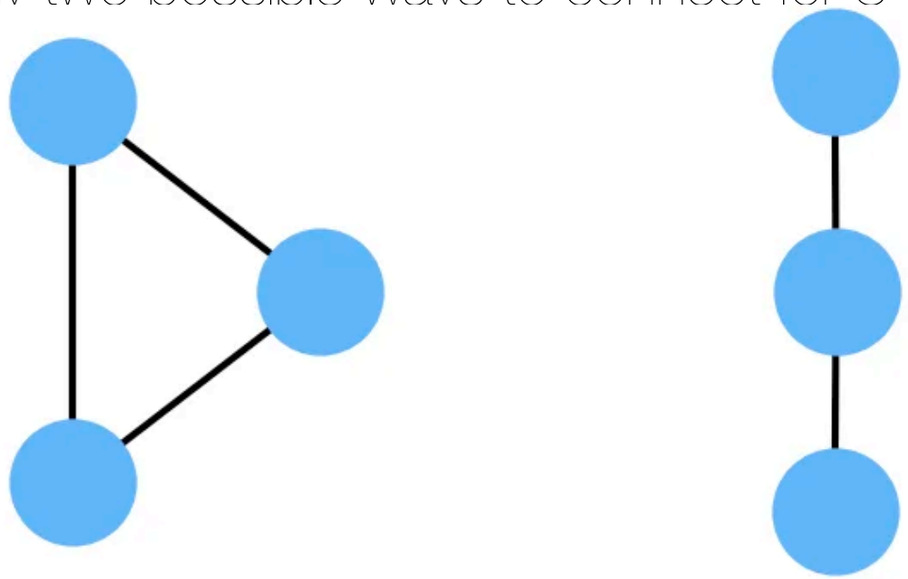
- Wallets are addresses controlled by the same individual
- We can identified them as the Weekly Connected Components in our transaction network



Graph Motifs

- Networks are produced as the result of some natural or artificial process resulting in specific signatures within the structure of empirical graphs.

- For example, there are only two possible ways to connect for 3-nodes without any self loops



- In some networks, one of these might be much more common than the other, indicating a structural constraint on your network.

- These kinds of common connectivity patterns are known as "network motifs" and have been the focus of much attention specially in the study of biological networks.

Graph Motifs

Trans. Comp. Bio and Bioinfo. 3, 347 (2006)

- Introduced by Wernicke in 2006
- The basic idea is to:
  - start with each edge in turn
  - grow subgraphs by adding new edges one at a time

Algorithm: ENUMERATESUBGRAPHS(G, k) (ESU)
Input: A graph $G = (V, E)$ and an integer $1 \leq k \leq |V|$.
Output: All size-k subgraphs in G.

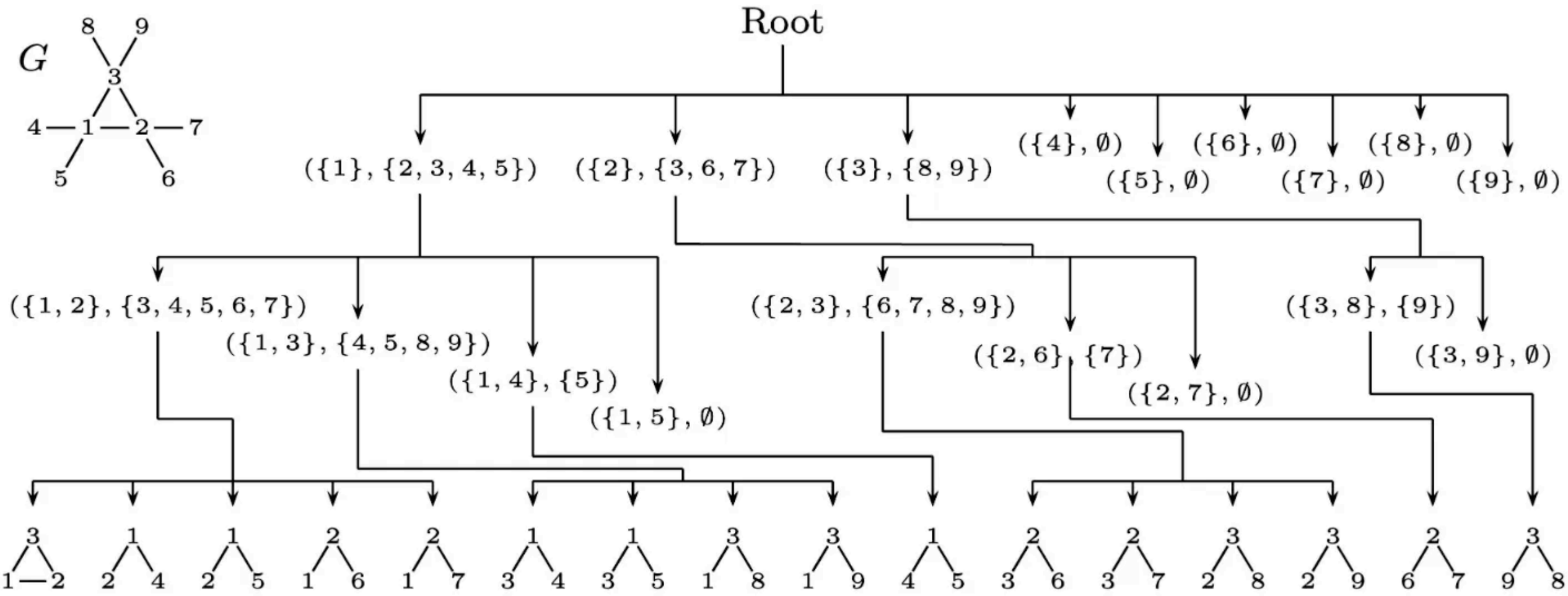
01 **for** each vertex $v \in V$ **do**
02 $\quad V_{Extension} \leftarrow \{u \in N(\{v\}) : u > v\}$
03 $\quad$ **call** EXTENDSUBGRAPH$(\{v\}, V_{Extension}, v)$
04 **return**

EXTENDSUBGRAPH$(V_{Subgraph}, V_{Extension}, v)$
E1 **if** $|V_{Subgraph}| = k$ **then output** $G[V_{Subgraph}]$ and **return**
E2 **while** $V_{Extension} \neq \emptyset$ **do**
E3 $\quad$ Remove an arbitrarily chosen vertex w from $V_{Extension}$
E4 $\quad V'_{Extension} \leftarrow V_{Extension} \cup \{u \in N_{excl}(w, V_{Subgraph}) : u > v\}$
E5 $\quad$ **call** EXTENDSUBGRAPH$(V_{Subgraph} \cup \{w\}, V'_{Extension}, v)$
E6 **return**

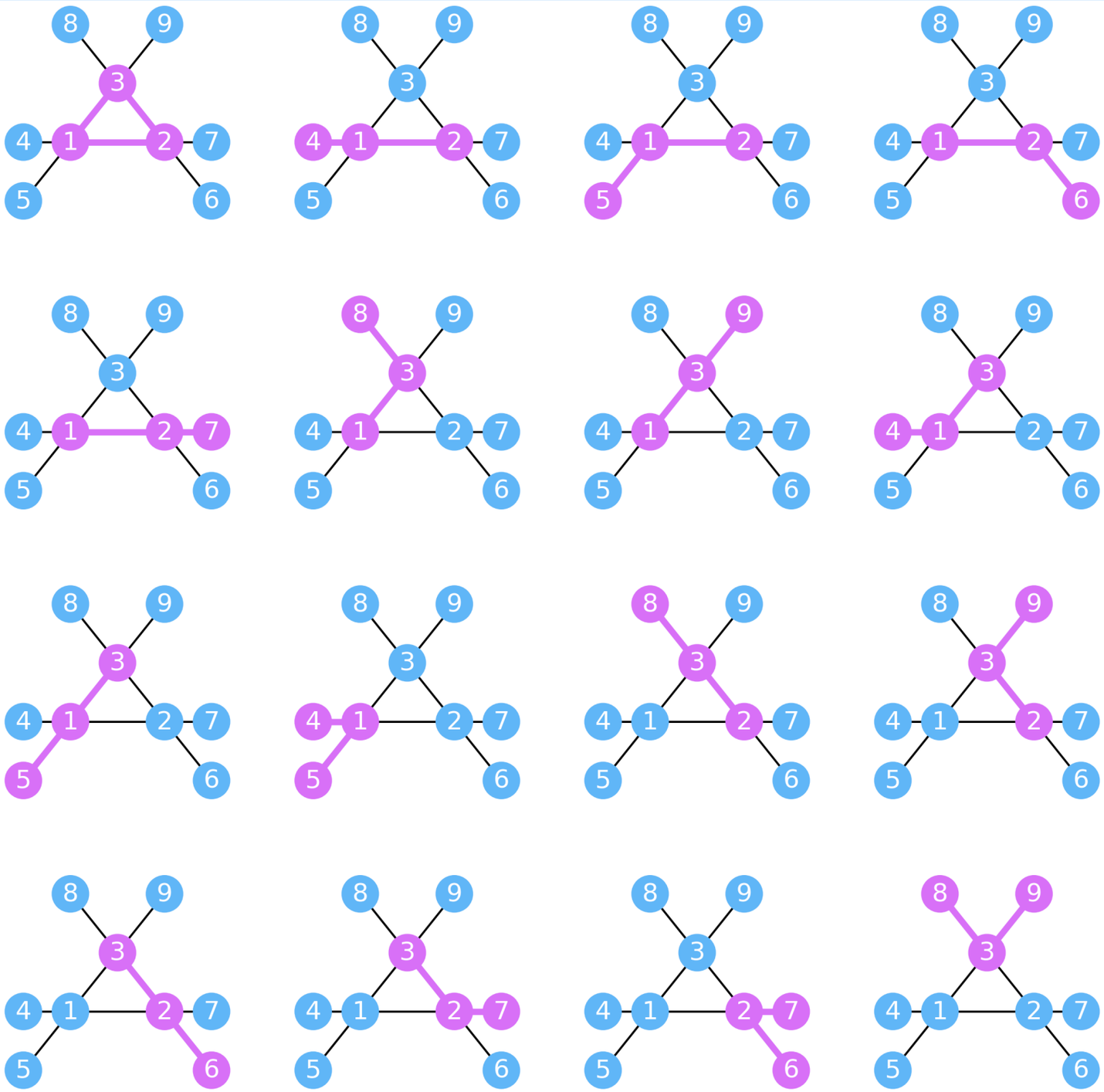
Graph Motifs

Trans. Comp. Bio and Bioinfo. 3, 347 (2006)

- Introduced by Wernicke in 2006
- The basic idea is to:
  - start with each edge in turn
  - grow subgraphs by adding new edges one at a time

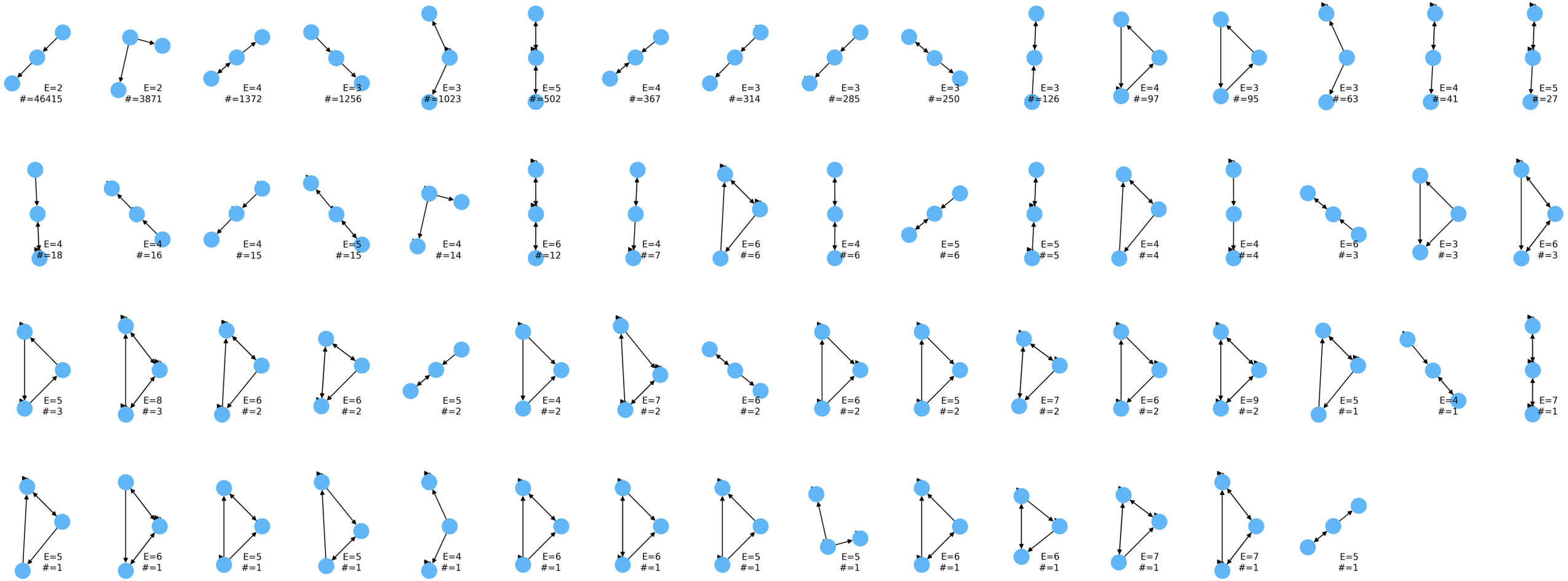


Graph Motifs



Bitcoin Network Motifs

- BitCoin has 62 individual motifs with just 3 nodes, highlighting the complexity of the behaviors we are able to observe in the crypto space



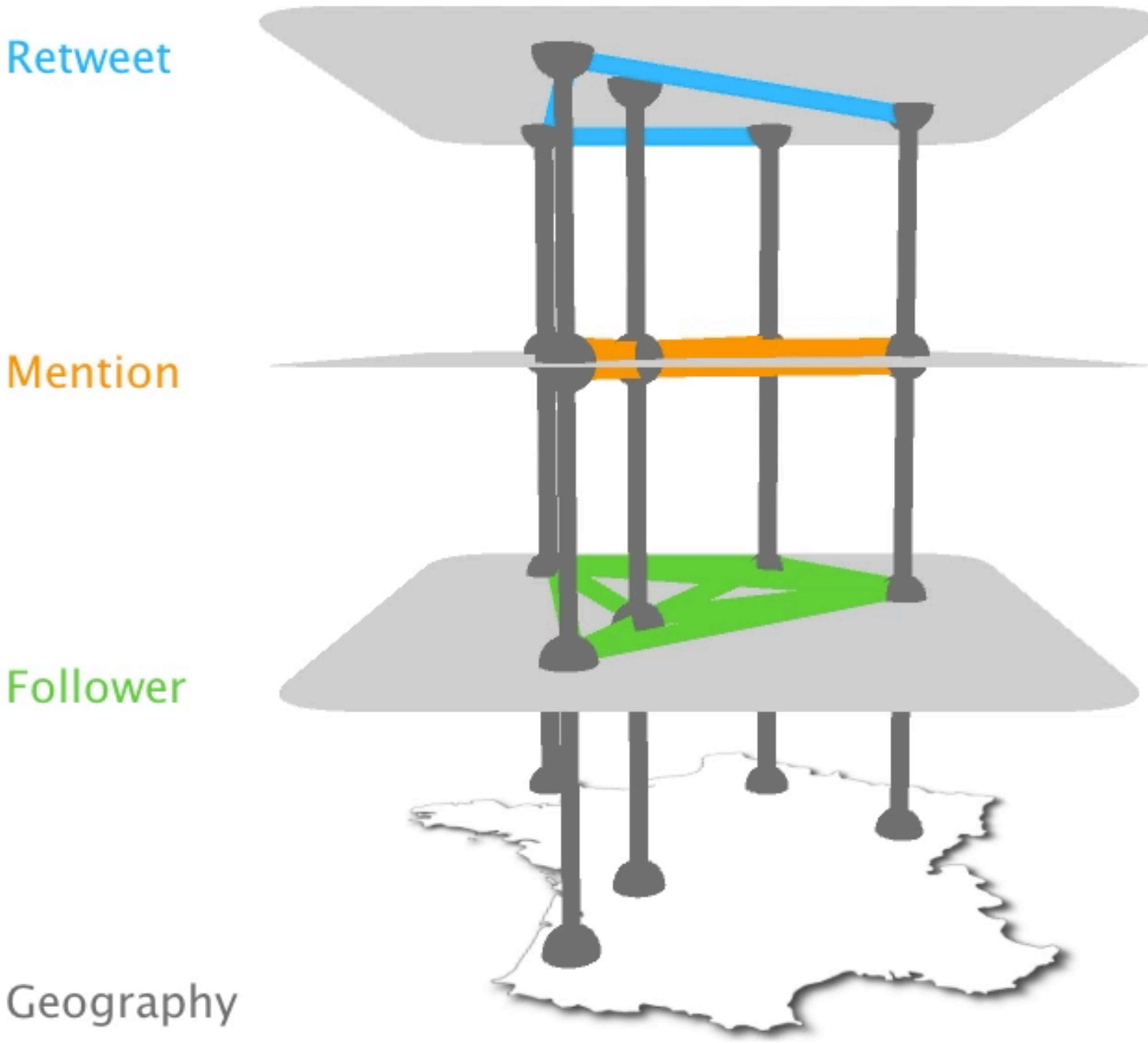
Code - Patterns and Structure

https://github.com/DataForScience/G4DS

4. Social Structure

Tweets can convey a lot of information

Twitter



Higgs Twitter Dataset

https://snap.stanford.edu/data/higgs-twitter.html

By *Jure Leskovec*

STANFORD UNIVERSITY

- SNAP for C++
- SNAP for Python
- SNAP Datasets
- BIOSNAP Datasets
- What's new
- People
- Papers
- Projects
- Citing SNAP
- Links
- About
- Contact us

Higgs Twitter Dataset

Dataset information

The Higgs dataset has been built after monitoring the spreading processes on Twitter before, during and after the announcement of the discovery of a new particle with the features of the elusive Higgs boson on 4th July 2012. The messages posted in Twitter about this discovery between 1st and 7th July 2012 are considered.

The four directional networks made available here have been extracted from user activities in Twitter as:

1. re-tweeting (retweet network)
2. replying (reply network) to existing tweets
3. mentioning (mention network) other users
4. friends/followers social relationships among user involved in the above activities
5. information about activity on Twitter during the discovery of Higgs boson

It is worth remarking that the user IDs have been anonimized, and the same user ID is used for all networks. This choice allows to use the Higgs dataset in studies about large-scale interdependent/interconnected multiplex/multilayer networks, where one layer accounts for the social structure and three layers encode different types of user dynamics .

Note that this dataset has been updated on **Mar 31 2015**. If you downloaded a previous version, please update it, results could differ.

For more information about data collection, please refer to our paper.

Clustering

- How likely are the friends of my friends to know each other?

- The possible number of pairwise combinations of k_i nodes is:

$$\binom{k_i}{2} \equiv \frac{k_i\left(k_i-1\right)}{2}$$

- And the clustering coefficient of node i is then:

$$C_i = \frac{2N_{\Delta i}}{k_i\left(k_i-1\right)}$$

where $N_{\Delta i}$ is the number of triangles of which node i is a part of (the number of my friends that are also friends of each other)

- Tendentially high in Social Networks:

	WWW	Citations	Co-author	Ham Radio	Prison	High School Romance
Number of Nodes	325729	396	81217	44	67	572
Randomness: r	0.57	0.63	4.7	5.0	∞	∞
Avg. In-Degree: m	4.6	5.0	.84	3.5	2.7	.83
Avg. Clustering	.11	.07	.16	.47	.31	-

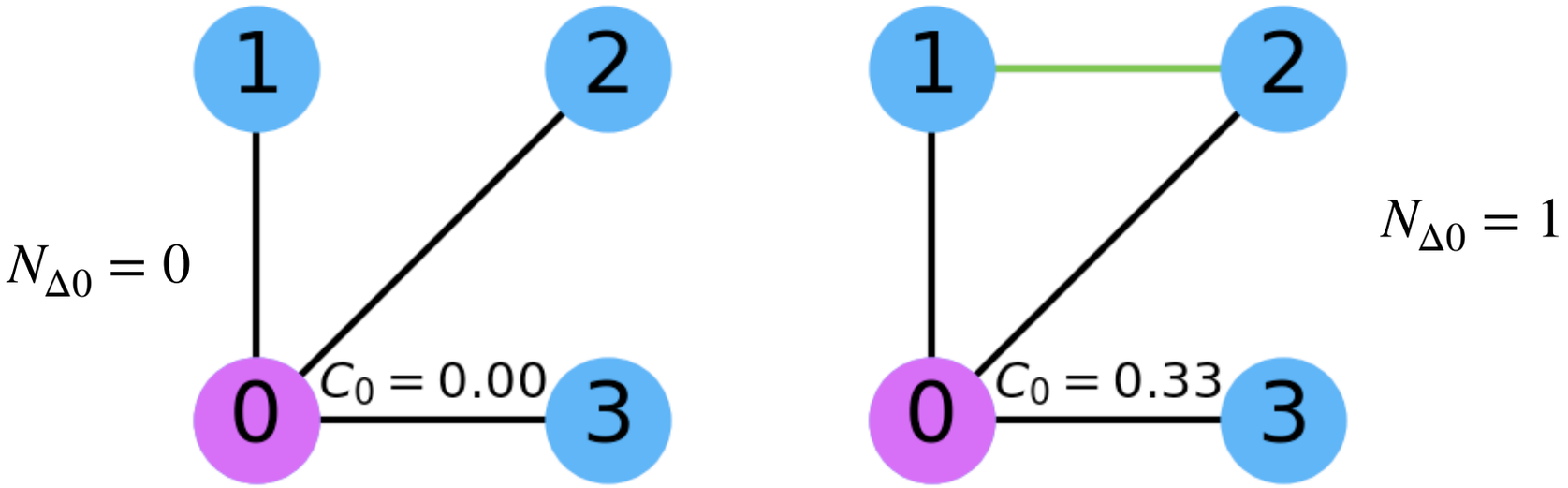
typically one or more orders of magnitude higher than expected in a random graph

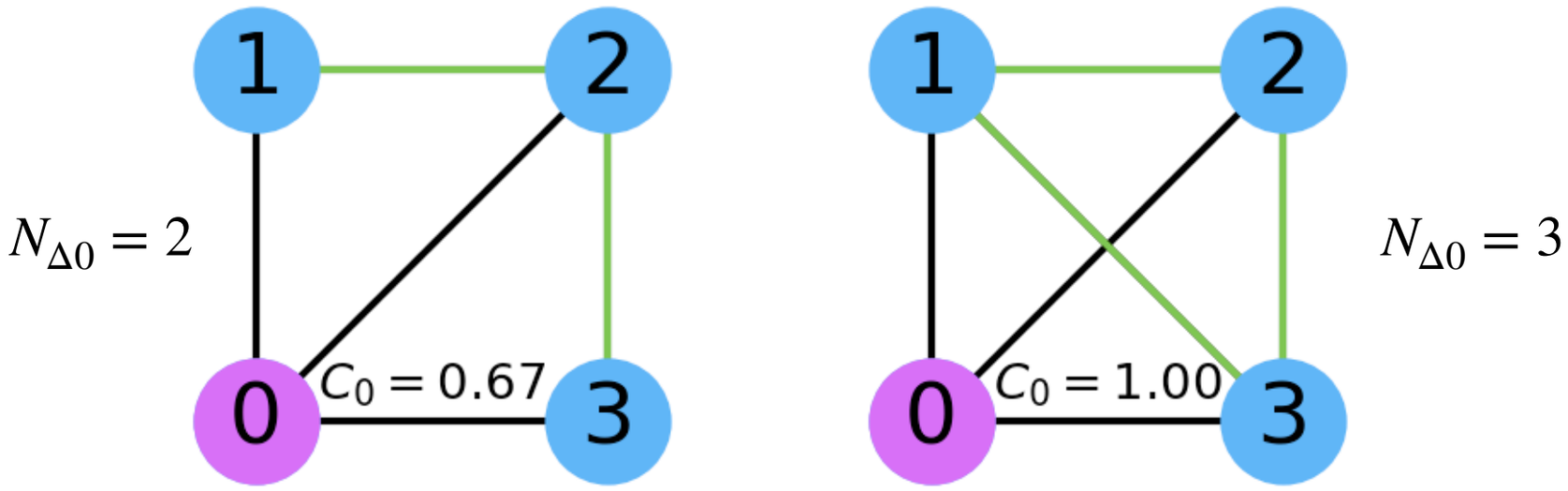
- An important factor in Community Structure

Clustering

$$C_i = \frac{2N_{\Delta i}}{k_i\left(k_i - 1\right)}$$

$k_0 = 3$

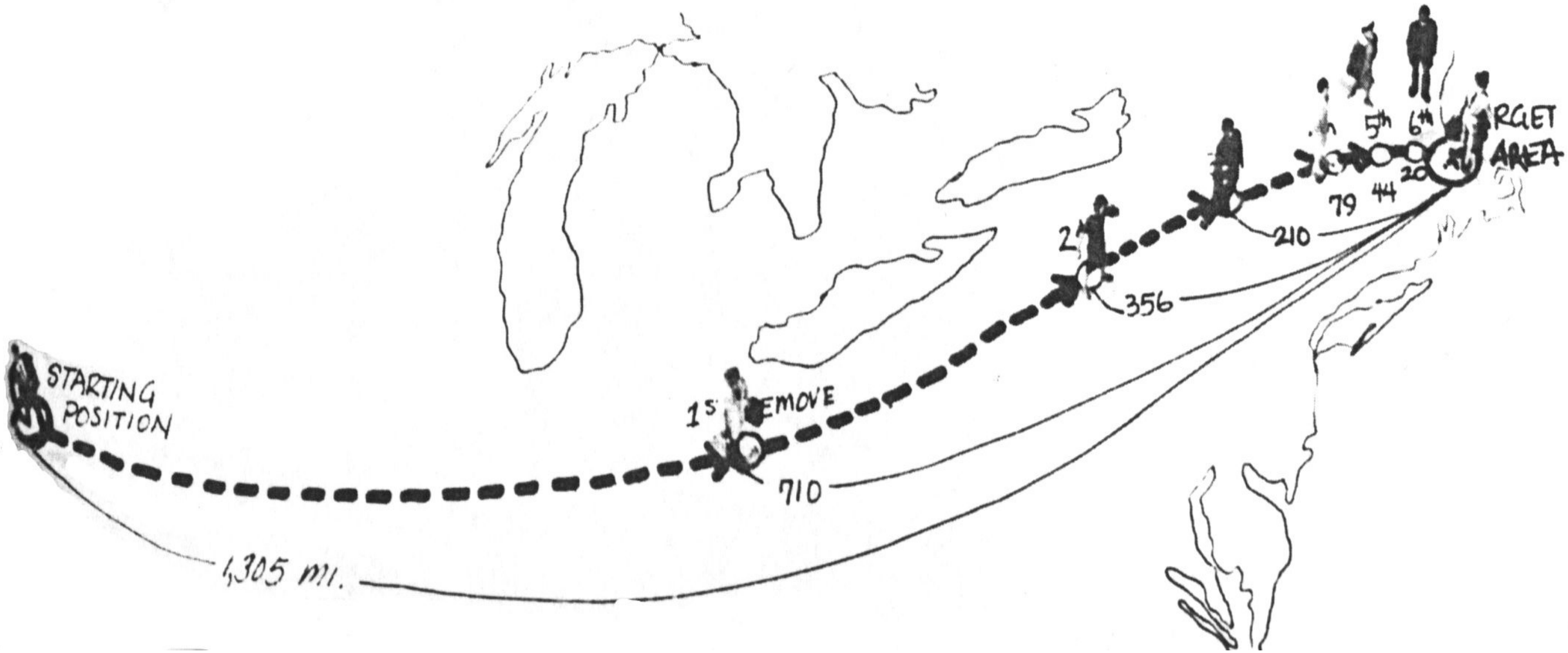




Network Diameter

- In 1967, the Social Psychologist Stanley Milgram decided to conduct an experiment to answer a simple question:

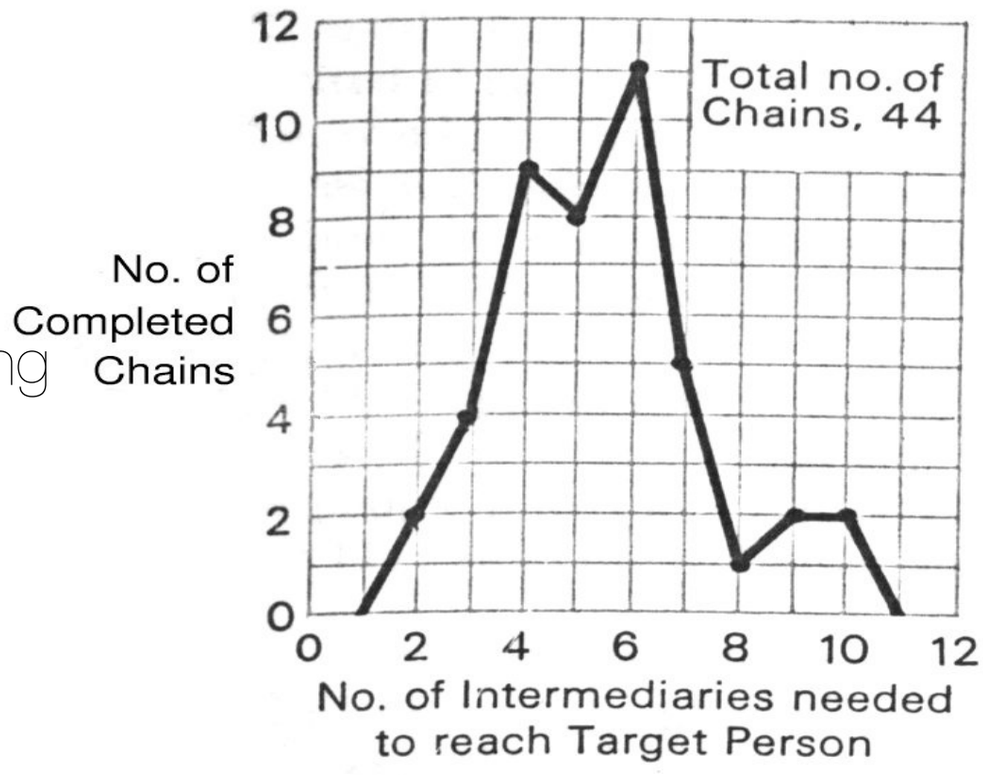
- How many friendship "steps" separate, on average, any two people in the US?



Network Diameter

- Choose participants in Kansas and Nebraska and ask them to try to send a letter to a stock broker in Boston, Massachusetts
- Pass letter in hand to someone they knew well and they think is likely to know the target
- Only 25% of the letters reached their destination
- Attrition causing longer chains to be down sampled
- Self-selection bias (seeds were chosen by answering a journal ad)
- Disconnected components?

6 degrees of separation

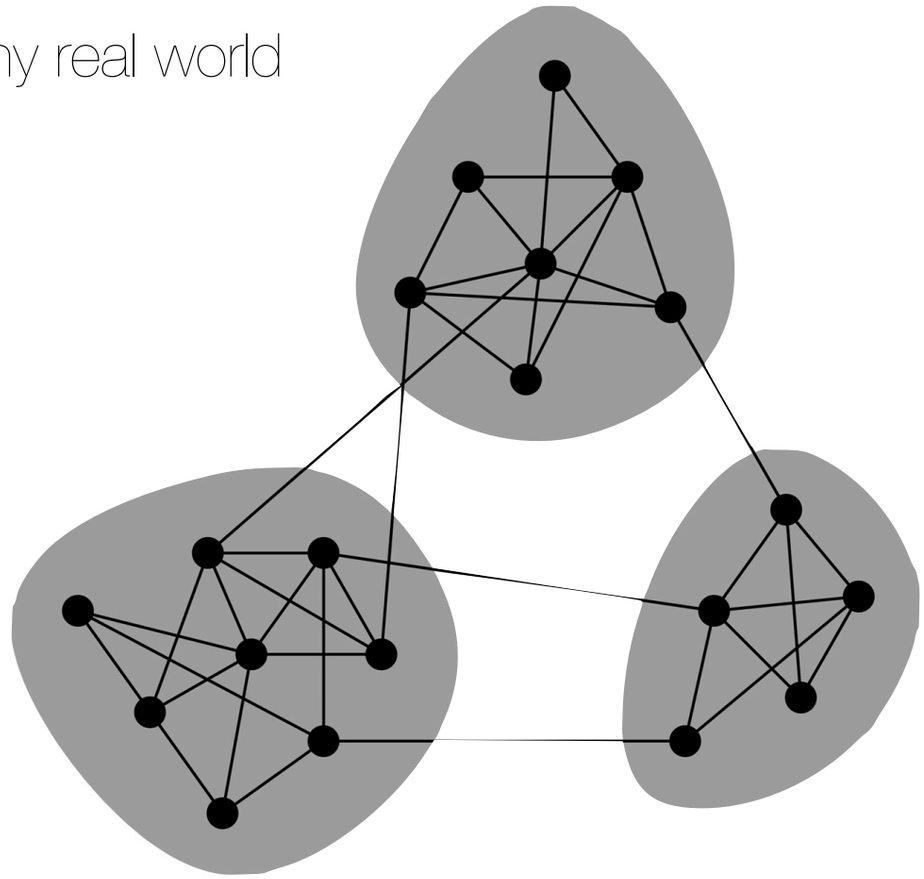


In the Nebraska Study the chains varied from two to 10 intermediate acquaintances with the median at five.

Community Structure

PNAS 103, 8577 (2006)

- Some networks are naturally divided into communities
- Finding communities in graphs is a hard problem with many real world applications:
  - Parallelization of Algorithms
  - Optimization of computer chip design
  - Distribution of MORPG players among game servers
  - Identifying social structure within a large social network
  - etc…



- Many community detection algorithms focus on different strategies to optimize the modularity for a given number of communities.
- See Phys. Rep.486, 75 (2010) for an extensive review of community detection algorithms

Network Modularity

- Empirically, it is clear that some network regions are denser than others
- Intuition:

  "Communities" should have more links within the community than without

- We define the modularity of a network as being:

$$Q = \sum_{i,j} \left[\frac{A_{ij}}{2m} - \frac{k_i k_j}{(2m)^2} \right]$$

- Where:
  - A_{ij} are the elements of the adjacency matrix
  - k_i, k_j are the degrees of nodes i and j, respectively
  - m is the total number of edges
  - The sum is taken only over nodes within the same community
- The modularity will be a large if all

Network Modularity

- Empirically, it is clear that some network regions are denser than others
- Intuition:

  "Communities" should have more links within the community than without

- We define the modularity of a network as being:

$$Q = \sum_{i,j} \left[\frac{A_{ij}}{2m} - \frac{k_i k_j}{(2m)^2} \right]$$

Probability that a node of degree k_i is connected to a node of degree k_j

- Where:
  - A_{ij} are the elements of the adjacency matrix
  - k_i, k_j are the degrees of nodes i and j, respectively
  - m is the total number of edges
  - The sum is taken only over nodes within the same community
- The modularity will be a large if all connected nodes belong to the same community

Code - Social Structure

https://github.com/DataForScience/G4DS

5. Recommendations and Link Prediction

MovieLens

https://movielens.org/

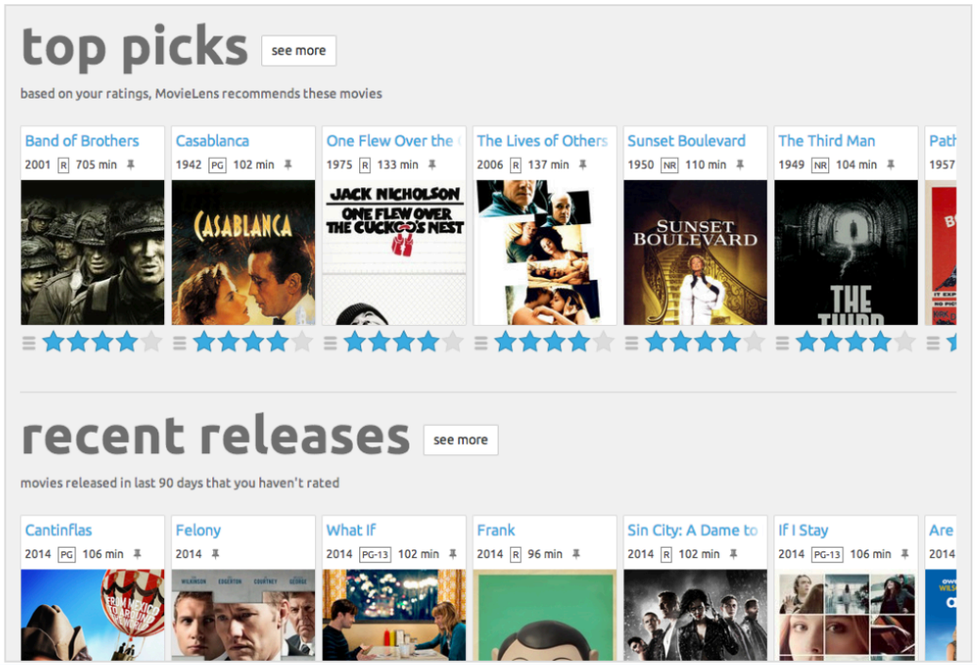
movielens

Non-commercial, personalized movie recommendations.

sign up now or sign in

recommendations

MovieLens helps you find movies you will like. Rate movies to build a custom taste profile, then MovieLens recommends other movies for you to watch.



MovieLens DataSet

https://grouplens.org/datasets/movielens/latest/

- MovieLens rating data made available by GroupLens Research
- Datasets include 5-star ratings and free-text tags generated by the users

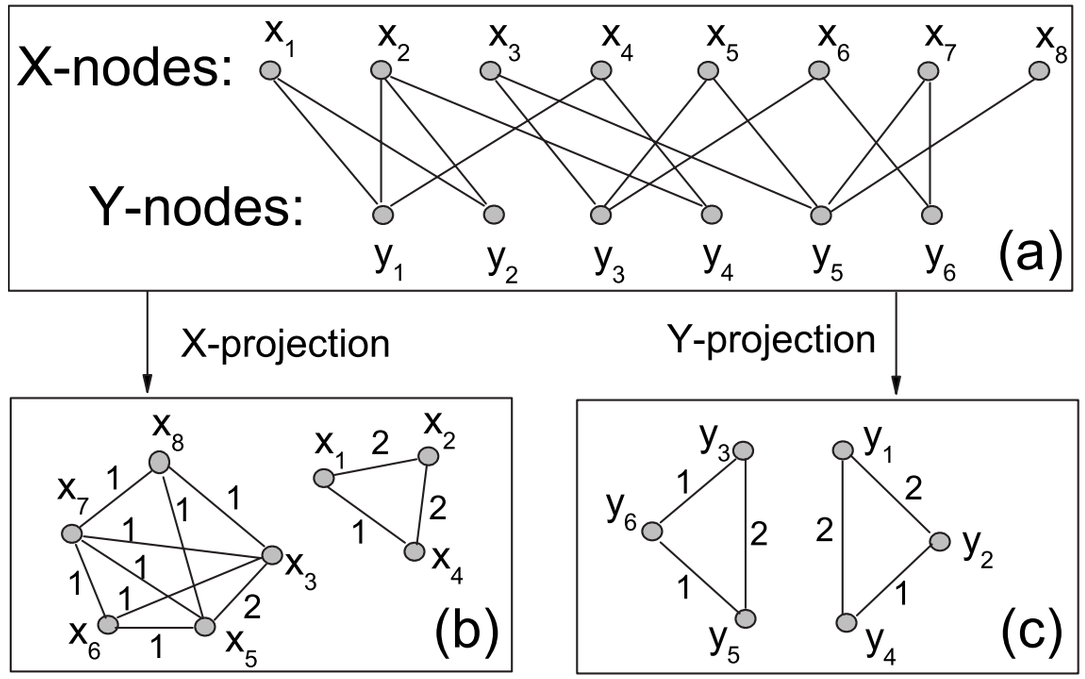
MovieLens

GroupLens Research has collected and made available rating data sets from the MovieLens web site (https://movielens.org). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

Seeking permission? If you are interested in obtaining permission to use MovieLens datasets, please first read the terms of use that are included in the README file. Then, please fill out this form to request use. We typically do not permit public redistribution (see Kaggle for an alternative download location if you are concerned about availability).

Bipartite Networks

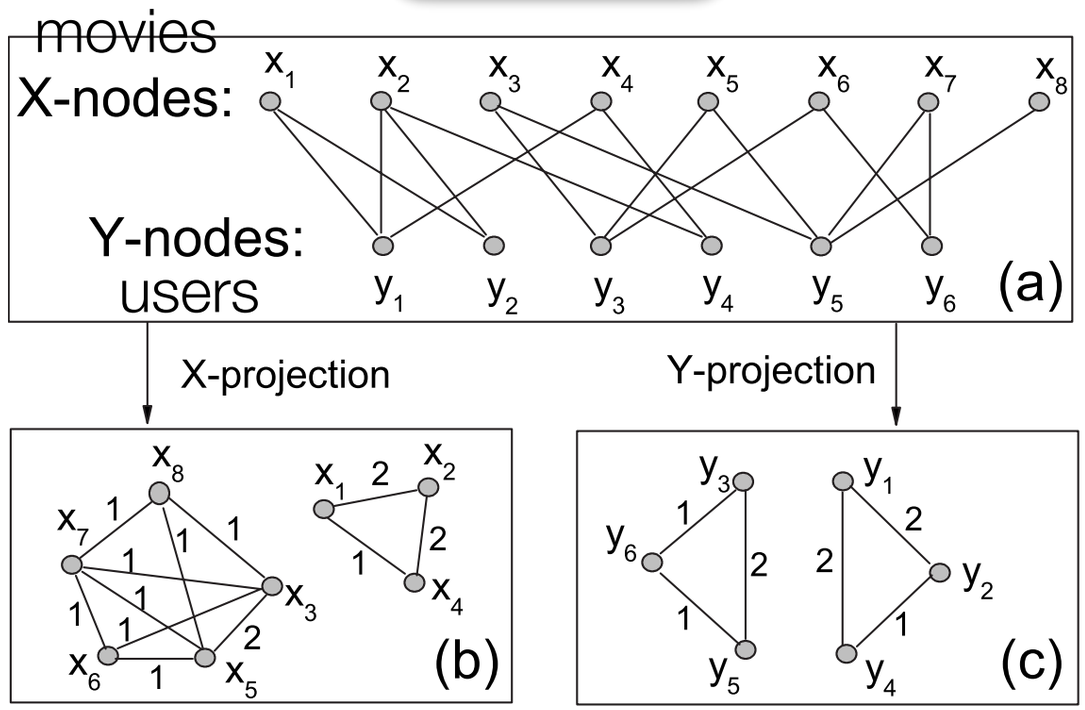
- Two kinds of nodes
- Edges are only allowed between nodes of different kinds
- Typically represent affiliation or collaboration:
  - actors in movies
  - authors in papers
  - users reviewing movies
  - etc



Bipartite Networks

movie-user network

- Two kinds of nodes [users - movies]
- Edges are only allowed between nodes of different kinds
- Typically represent affiliation or collaboration:
  - actors in movies
  - authors in papers
  - users reviewing movies
  - etc



user-user network

movie-movie network

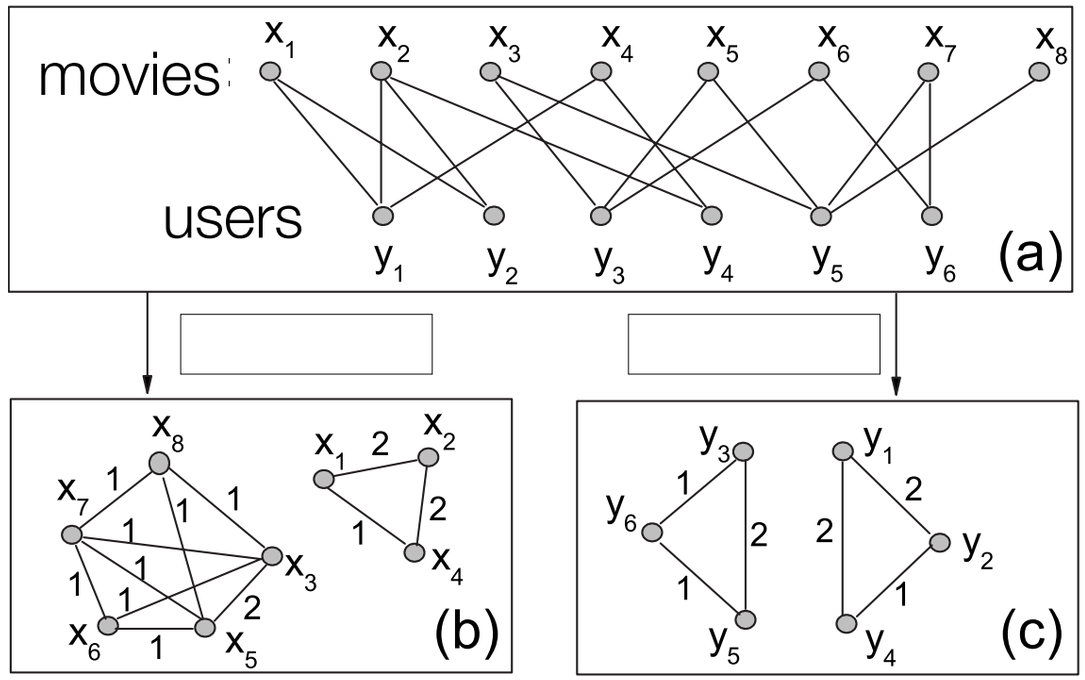
Bipartite Network Projection

- A common step is to project bipartite networks to a single node type
- Projected networks contain similarity information for each kind of node

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{matrix} \text{movies} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix}$$

users

movie-user network



user-user network

$$X = AA^T$$

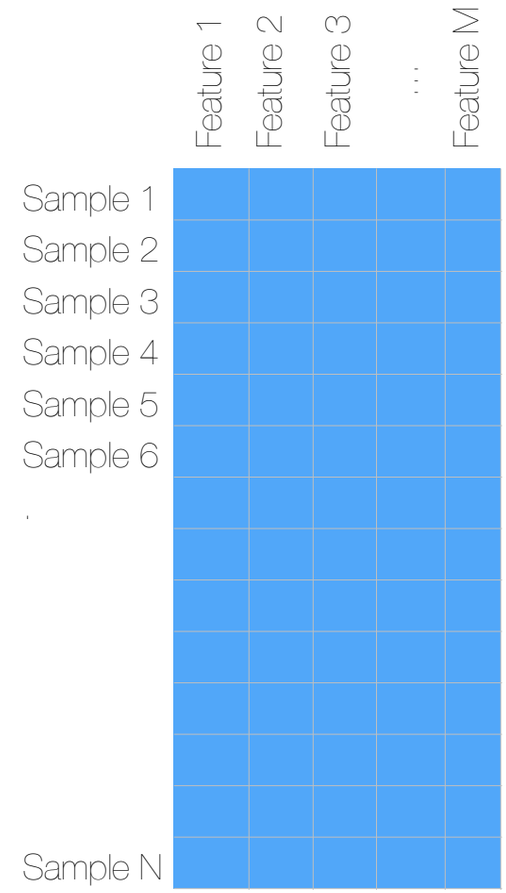
movie-movie network

$$Y = A^T A$$

Bipartite Networks - Similarities

We can treat the (weighted) adjacency matrix as a feature matrix!

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$



And generate a new matrix based on the similarities between items

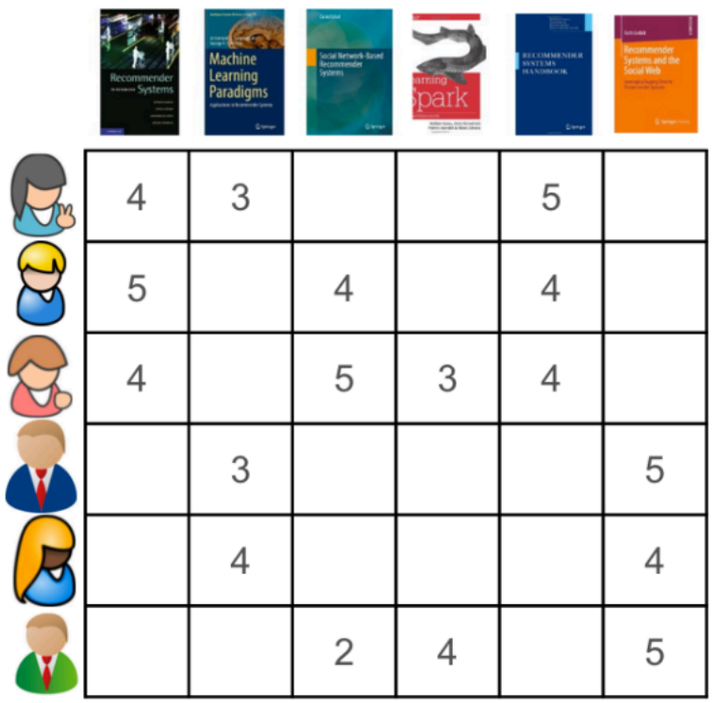
Node Similarity

- A common choice of similarity metric to compare two vectors is the cosine-similarity:

$$\text{sim}\left(\vec{u},\vec{v}\right) = \frac{\vec{u}\cdot\vec{v}}{||\vec{u}||\ ||\vec{v}||}$$

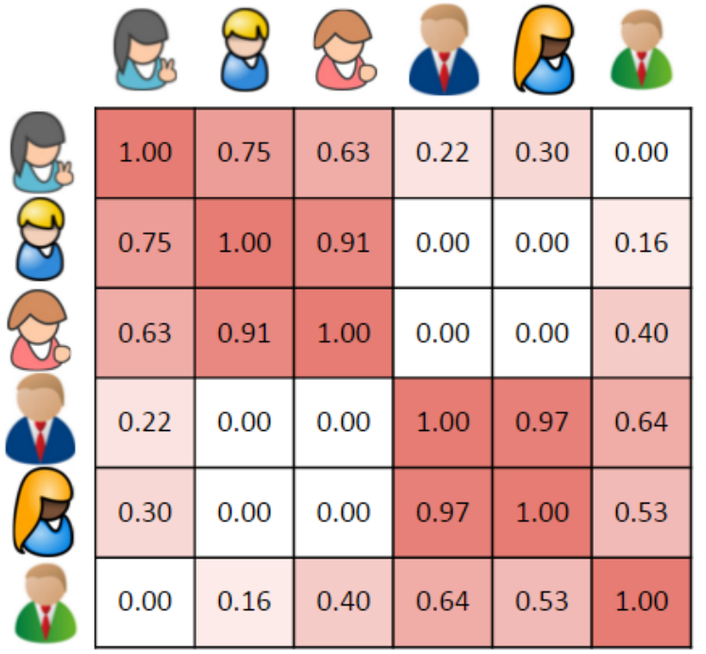
- we can easily measure the similarity between every pair of documents in our corpus to generate a square similarity matrix.

Bipartite Networks - Similarities



Sparse

Dense



RECOMMENDER SYSTEM ALGORITHMS

INPUT

4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

Introduction to Recommender Systems
Machine learning Paradigms
Social Network-based Recommender Systems
Learning Spark
Recommender Systems Handbook
Recommender Systems and the Social Web

FOUR RECOMMENDER ALGORITHMS ARE FED THE SAME INPUT AND PRODUCE DIFFERENT OUTPUTS.
THEIR DEFINITIONS OF USER AND/OR ITEM SIMILARITY ACCOUNT FOR MOST OF THE DIFFERENCES.

USER-BASED COLLABORATIVE FILTERING

1.00	0.75	0.63	0.22	0.30	0.00
0.75	1.00	0.91	0.00	0.00	0.16
0.63	0.91	1.00	0.00	0.00	0.40
0.22	0.00	0.00	1.00	0.97	0.64
0.30	0.00	0.00	0.97	1.00	0.53
0.00	0.16	0.40	0.64	0.53	1.00

(0.7 x) + (0.6 x) =

already rated by user

(0.7 x 4 + 0.6 x 5) / (0.7 + 0.6) = 4.5

(0.6 x 3) / 0.6 = 3.0

already rated by user

ITEM-BASED COLLABORATIVE FILTERING

1.00	0.27	0.79	0.32	0.98	0.00
0.27	1.00	0.00	0.00	0.34	0.65
0.79	0.00	1.00	0.69	0.71	0.18
0.32	0.00	0.69	1.00	0.32	0.49
0.98	0.34	0.71	0.32	1.00	0.00
0.00	0.65	0.18	0.49	0.00	1.00

(4 x) + (3 x) + (5 x) =

already rated by user

(0.8 x 4 + 0.7 x 5) / (0.8 + 0.7) = 4.5

already rated by user

(0.7 x 3) / 0.7 = 3.0

CONTENT-BASED FILTERING

1.00	0.00	0.58	0.00	0.67	0.58
0.00	1.00	0.00	0.41	0.00	0.00
0.58	0.00	1.00	0.00	0.58	0.75
0.00	0.41	0.00	1.00	0.00	0.00
0.67	0.00	0.58	0.00	1.00	0.58
0.58	0.00	0.75	0.00	0.58	1.00

(4 x) + (3 x) + (5 x) =

already rated by user

(0.4 x 3) / 0.4 = 3.0

already rated by user

(0.6 x 4 + 0.6 x 5) / (0.6 + 0.6) = 4.5

HYBRID

0.4 x (UB CF) + (0.3 x (IB CF) + (0.3 x (CB) =

(0.4 x 4.5 + 0.3 x 4.5) / (0.4 + 0.3) = 4.5

(0.3 x 3.0 + 0.3 x 4.5) / (0.3 + 0.3) = 3.8

(0.4 x 3.0 + 0.3 x 3.0) / (0.4 + 0.3) = 3.0

OUTPUTS

Code - Recommendations

https://github.com/DataForScience/G4DS

Question

- How was the technical level?
    - 1 — Too Low (too many details)
    - 2 — Low
    - 3 — Just Right
    - 4 — High
    - 5 — Too High (not enough details)

Question

- How was the level of Python code/explanations?
    - 1 — Too Low (too many details)
    - 2 — Low
    - 3 — Just Right
    - 4 — High
    - 5 — Too High (not enough details)

Events

graphs4sci.substack.com

Deep Learning for Everyone

Jun 20, 2023 - 10am-2pm (PST)

VIDEO

Times Series Analysis for Everyone

★★★★★ 1 review

By Bruno Goncalves

https://bit.ly/Timeseries_LL

TIME TO COMPLETE:
6h

TOPICS:
Time Series

PUBLISHED BY:
Pearson

PUBLICATION DATE:
November 2021

Start

6 Hours of Video Instruction

The perfect introduction to time-based analytics

Overview

Times Series Analysis for Everyone LiveLessons covers the fundamental tools and techniques for the analysis of time series data. These lessons introduce you to the basic concepts, ideas, and algorithms necessary to develop your own time series applications in a step-by-step, intuitive fashion. The lessons follow a gradual progression, from the more specific to the more abstract, taking you from the very basics to some of the most recent and sophisticated algorithms by leveraging the statsmodels, arch, and Keras state-of-the-art models.

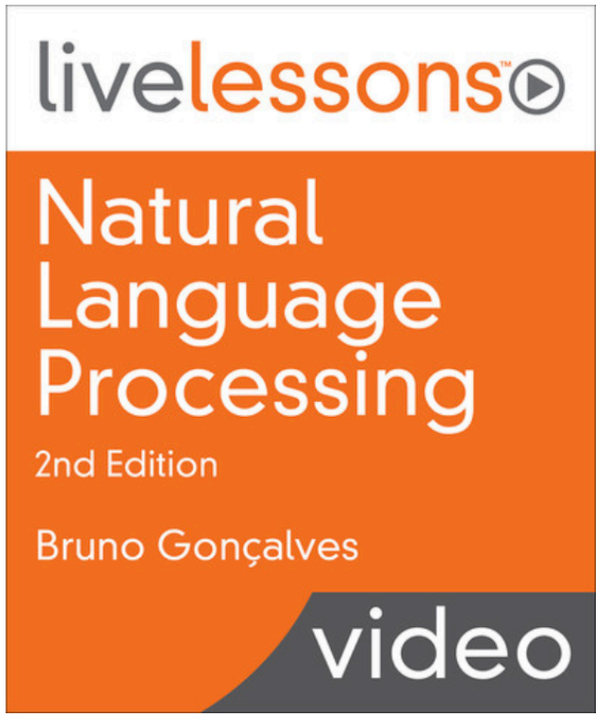
VIDEO

Natural Language Processing, 2nd Edition

Write the first review

By Bruno Goncalves

https://bit.ly/NLP_LL



TIME TO COMPLETE:
5h 23m

TOPICS:
Natural Language Processing

PUBLISHED BY:
Addison-Wesley Professional

PUBLICATION DATE:
October 2021

Start

5 Hours of Video Instruction

Overview

Natural Language Processing LiveLessons covers thefundamentals of Natural Language Processing in a simple and intuitive way,empowering you to add NLP to your toolkit. Using the powerful NLTK package, itgradually moves from the basics of text representation, cleaning, topicdetection, regular expressions, and sentiment analysis before moving on to theKeras deep learning framework to explore more advanced topics such as textclassification and sequence-to-sequence models. After successfully completingthese lessons you'll be equipped with a fundamental and practical understandingof state-of-the-art Natural Language Processing tools and algorithms.

END