CIS 600 - Principles of Social Media and Data Mining

Spring 2023

Week 2

Priyantha Kumarawadu

Associate Teaching Professor

Department of Electrical Engineering and Computer Science

Syracuse University

Python for Data Analysis

Outline

- Introduction to Python Libraries
- Reading Data; Selecting and Filtering the Data; Data manipulation, sorting, grouping, rearranging
- Plotting the data
- Descriptive statistics

Python Libraries for Data Science

Many popular Python toolboxes/libraries:

- NumPy
- SciPy
- Pandas
- SciKit-Learn

Visualization libraries

- matplotlib
- Seaborn

All these libraries are installed on the SCC

and many more …

Python Libraries for Data Science

Pandas:

- adds data structures and tools designed to work with table-like data (similar to Series and Data Frames in R)
- provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc.
- allows handling missing data

```
# Series
x = pd.Series([1,3,5,6])
print(x)
```

```
0    1
1    3
2    5
3    6
dtype: int64
```

```
#DataFrame
#Generate random numbers in 3x4 matrix
import numpy as np
df = pd.DataFrame(np.random.randn(3,4))
print(df)
```

```
          0         1         2         3
0  0.670234  0.266334  0.950184 -0.850909
1 -1.233661  0.729577 -0.630627  0.205718
2 -0.178291  1.238178 -0.549750 -0.148647
```

Link: http://pandas.pydata.org/

Python Libraries for Data Science

NumPy:

- introduces objects for multidimensional arrays and matrices, as well as functions that allow to easily perform advanced mathematical and statistical operations on those objects
- provides vectorization of mathematical operations on arrays and matrices which significantly improves the performance
- many other python libraries are built on NumPy

Link: http://www.numpy.org/

Python Libraries for Data Science

SciPy:

- collection of algorithms for linear algebra, differential equations, numerical integration, optimization, statistics and more
- part of SciPy Stack
- built on NumPy

Link: https://www.scipy.org/scipylib/

Python Libraries for Data Science

matplotlib:

- python 2D plotting library which produces publication quality figures in a variety of hardcopy formats
- a set of functionalities similar to those of MATLAB
- line plots, scatter plots, barcharts, histograms, pie charts etc.
- relatively low-level; some effort needed to create advanced visualization

Link: https://matplotlib.org/

Python Libraries for Data Science

SciKit-Learn:

- provides machine learning algorithms: classification, regression, clustering, model validation etc.

- built on NumPy, SciPy and matplotlib

Link: http://scikit-learn.org/

Python Libraries for Data Science

Seaborn:

- based on matplotlib
- provides high level interface for drawing attractive statistical graphics
- Similar (in style) to the popular ggplot2 library in R

Link: https://seaborn.pydata.org/

Loading Python Libraries

In [ ]:
```
#Import Python Libraries
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
import seaborn as sns
```

Press `Shift+Enter` to execute the *jupyter* cell

Data formats

The three most common formats (judging by my completely subjective experience):

1. CSV (comma separate value) files

2. JSON (Javascript object notation) files and strings

3. HTML/XML (hypertext markup language / extensible markup language) files and strings

CSV Files

Refers to any delimited text file (not always separated by commas)

```
"Semester","Course","Section","Lecture","Mini","Last Name","Preferred/First
Name","MI","Andrew ID","Email","College","Department","Class","Units","Grade
Option","QPA Scale","Mid-Semester Grade","Final Grade","Default Grade","Added
By","Added On","Confirmed","Waitlist Position","Waitlist Rank","Waitlisted
By","Waitlisted On","Dropped By","Dropped On","Roster As Of Date"
"F16","15688","B","Y","N","Kolter","Zico","","zkolter","SCS","CS","50","12.0","L",
"4+"," "," ","","reg","1 Jun2016","Y","","","","","","","30 Aug 2016 4:34"
```

If values themselves contain commas, you can enclose them in quotes (our registrar apparently always does this, just to be safe)

```
import pandas as pd
dataframe = pd.read_csv("CourseRoster_F16_15688_B_08.30.2016.csv",
                        delimiter=',', quotechar='"')
```

We'll talk about the pandas library a lot more in later lectures

JSON files / string

JSON originated as a way of encapsulating Javascript objects

A number of different data types can be represented

- Number: `1.0` (always assumed to be floating point)
- String: `"string"`
- Boolean: `true` or `false`
- List (Array): `[item1, item2, item3,…]`
- Dictionary (Object in Javascript): `{"key":value}`

Lists and Dictionaries can be embedded within each other:

```
[{"key":[value1, [value2, value3]]}]
```

Example JSON data

JSON from Github API


```
{
 'login': 'pvirtue',
 'id': 5945661,
 'node_id': 'MDQ6VXNlcjU5NDU2NjE=',
 'avatar_url': 'https://avatars.githubusercontent.com/u/5945661?v=4',
 'gravatar_id': '',
 'url': 'https://api.github.com/users/pvirtue',
 'html_url': 'https://github.com/pvirtue',
 'followers_url': 'https://api.github.com/users/pvirtue/followers',
 'following_url': 'https://api.github.com/users/pvirtue/following{/other_user}',
 'gists_url': 'https://api.github.com/users/pvirtue/gists{/gist_id}',
 'starred_url': 'https://api.github.com/users/pvirtue/starred{/owner}{/repo}',
 'subscriptions_url': 'https://api.github.com/users/pvirtue/subscriptions',
 'organizations_url': 'https://api.github.com/users/pvirtue/orgs',
 'repos_url': 'https://api.github.com/users/pvirtue/repos',
 'events_url': 'https://api.github.com/users/pvirtue/events{/privacy}',
 'received_events_url': 'https://api.github.com/users/pvirtue/received_events',
 'type': 'User',
 'site_admin': False,
 'name': 'Pat Virtue', ...
```

Parsing JSON in Python

Built-in library to read/write Python objects from/to JSON files

```
import json

# load json from a REST API call
headers = {'Authorization': 'token '+token}
response = requests.get("https://api.github.com/user", headers=headers)
data = json.loads(response.content)

json.load(file) # load json from file
json.dumps(obj) # return json string
json.dump(obj, file) # write json to file
```

XML / HTML files

The main format for the web (though XML seems to be loosing a bit of popularity to JSON for use in APIs / file formats)

XML files contain hiearchical content delineated by tags

```
<tag attribute="value">
    <subtag>
        Some content for the subtag
    </subtag>
    <openclosetag attribute="value2”/>
</tag>
```

HTML is syntactically like XML but horrible (e.g., open tags are not always closed), more fundamentally, HTML is mean to describe appearance

Reading data using pandas

```
In [ ]: #Read csv file
        df = pd.read_csv('../input/walmart-analysis-dataset/myCity.csv')
```

Note: The above command has many optional arguments to fine-tune the data import process.

There is a number of pandas commands to read other data formats:

```
pd.read_excel('myfile.xlsx',sheet_name='Sheet1', index_col=None, na_values=['NA'])
pd.read_stata('myfile.dta')
pd.read_sas('myfile.sas7bdat')
pd.read_hdf('myfile.h5','df')
```

Exploring data frames

In [3]:
```
#List first 5 records
df.head()
```

Out[3]:

	rank	discipline	phd	service	sex	salary
0	Prof	B	56	49	Male	186960
1	Prof	A	12	6	Male	93000
2	Prof	A	23	20	Male	110515
3	Prof	A	40	31	Male	131205
4	Prof	B	20	18	Male	104800

Data Frame data types

Pandas Type	Native Python Type	Description
object	string	The most general dtype. Will be assigned to your column if column has mixed types (numbers and strings).
int64	int	Numeric characters. 64 refers to the memory allocated to hold this character.
float64	float	Numeric characters with decimals. If a column contains numbers and NaNs(see below), pandas will default to float64, in case your missing value has a decimal.
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	Values meant to hold time data. Look into these for time series experiments.

Data Frame data types

```
In [4]: #Check a particular column type
        df['salary'].dtype
```

```
Out[4]: dtype('int64')
```

```
In [5]: #Check types for all the columns
        df.dtypes
```

```
Out[4]: rank          object
        discipline    object
        phd           int64
        service       int64
        sex           object
        salary        Int64
        dtype         object
```

Data Frames attributes

Python objects have *attributes* and *methods*.

df.attribute	description
dtypes	list the types of the columns
columns	list the column names
axes	list the row labels and column names
ndim	number of dimensions
size	number of elements
shape	return a tuple representing the dimensionality
values	numpy representation of the data

Data Frames methods

Unlike attributes, python methods have *parenthesis.*
All attributes and methods can be listed with a *dir()* function: `dir(df)`

df.method()	description
head([n]), tail([n])	first/last n rows
describe()	generate descriptive statistics (for numeric columns only)
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation
sample([n])	returns a random sample of the data frame
dropna()	drop all the records with missing values

Selecting a column in a Data Frame

Method 1: Subset the data frame using column name:

```
df['sex']
```

Method 2: Use the column name as an attribute:

```
df.sex
```

Note: there is an attribute *rank* for pandas data frames, so to select a column with a name "rank" we should use method 1.

Data Frames *groupby* method

Using "group by" method we can:

- Split the data into groups based on some criteria
- Calculate statistics (or apply a function) to each group
- Similar to dplyr() function in R

In [ ]:
```
#Group data using rank
df_rank = df.groupby(['rank'])
```

In [ ]:
```
#Calculate mean value for each numeric column per each group
df_rank.mean()
```

rank	phd	service	salary
AssocProf	15.076923	11.307692	91786.230769
AsstProf	5.052632	2.210526	81362.789474
Prof	27.065217	21.413043	123624.804348

Data Frames *groupby* method

Once groupby object is create we can calculate various statistics for each group:

```
In [ ]: #Calculate mean salary for each professor rank:
        df.groupby('rank')[['salary']].mean()
```

	salary
rank	
AssocProf	91786.230769
AsstProf	81362.789474
Prof	123624.804348

Note: If single brackets are used to specify the column (e.g. salary), then the output is Pandas Series object. When double brackets are used the output is a Data Frame

Data Frames *groupby* method

groupby performance notes:

- no grouping/splitting occurs until it's needed. Creating the *groupby* object only verifies that you have passed a valid mapping
- by default the group keys are sorted during the *groupby* operation. You may want to pass sort=False for potential speedup:

In [ ]:
```
#Calculate mean salary for each professor rank:
df.groupby(['rank'], sort=False)[['salary']].mean()
```

Data Frame: filtering

To subset the data we can apply Boolean indexing. This indexing is commonly known as a filter. For example if we want to subset the rows in which the salary value is greater than $120K:

In [ ]:
```
#Calculate mean salary for each professor rank:
df_sub = df[ df['salary'] > 120000 ]
```

Any Boolean operator can be used to subset the data:

> greater; >= greater or equal;

< less; <= less or equal;

== equal; != not equal;

In [ ]:
```
#Select only those rows that contain female professors:
df_f = df[ df['sex'] == 'Female' ]
```

Data Frames: Slicing

There are a number of ways to subset the Data Frame:

- one or more columns
- one or more rows
- a subset of rows and columns

Rows and columns can be selected by their position or label

Data Frames: Slicing

When selecting one column, it is possible to use single set of brackets, but the resulting object will be a Series (not a DataFrame):

In [ ]:
```
#Select column salary:
df['salary']
```

When we need to select more than one column and/or make the output to be a DataFrame, we should use double brackets:

In [ ]:
```
#Select column salary:
df[['rank','salary']]
```

Data Frames: Selecting rows

If we need to select a range of rows, we can specify the range using ":"

In [ ]:
```
#Select rows by their position:
df[10:20]
```

Notice that the first row has a position 0, and the last value in the range is omitted:
So for 0:10 range the first 10 rows are returned with the positions starting with 0 and ending with 9

Data Frames: method loc

If we need to select a range of rows, using their labels we can use method loc:

In [ ]:
```
#Select rows by their labels:
df_sub.loc[10:20,['rank','sex','salary']]
```

Out[ ]:

	rank	sex	salary
10	Prof	Male	128250
11	Prof	Male	134778
13	Prof	Male	162200
14	Prof	Male	153750
15	Prof	Male	150480
19	Prof	Male	150500

Data Frames: method iloc

If we need to select a range of rows and/or columns, using their positions we can use method iloc:

In [ ]:
```
#Select rows by their labels:
df_sub.iloc[10:20,[0, 3, 4, 5]]
```

Out[ ]:

	rank	service	sex	salary
26	Prof	19	Male	148750
27	Prof	43	Male	155865
29	Prof	20	Male	123683
31	Prof	21	Male	155750
35	Prof	23	Male	126933
36	Prof	45	Male	146856
39	Prof	18	Female	129000
40	Prof	36	Female	137000
44	Prof	19	Female	151768
45	Prof	25	Female	140096

Data Frames: method iloc (summary)

```
df.iloc[0]  # First row of a data frame
df.iloc[i]  #(i+1)th row
df.iloc[-1] # Last row
```

```
df.iloc[:, 0]  # First column
df.iloc[:, -1] # Last column
```

```
df.iloc[0:7]        #First 7 rows
df.iloc[:, 0:2]     #First 2 columns
df.iloc[1:3, 0:2]   #Second through third rows and first 2 columns
df.iloc[[0,5], [1,3]]  #1st and 6th rows and 2nd and 4th columns
```

Data Frames: Sorting

We can sort the data by a value in the column. By default the sorting will occur in ascending order and a new data frame is return.

In [ ]:
```
# Create a new data frame from the original sorted by the column Salary
df_sorted = df.sort_values( by ='service')
df_sorted.head()
```

Out[ ]:

	rank	discipline	phd	service	sex	salary
55	AsstProf	A	2	0	Female	72500
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000

Data Frames: Sorting

We can sort the data using 2 or more columns:

In [ ]:
```
df_sorted = df.sort_values( by =['service', 'salary'], ascending = [True, False])
df_sorted.head(10)
```

Out[ ]:

	rank	discipline	phd	service	sex	salary
52	Prof	A	12	0	Female	105000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
55	AsstProf	A	2	0	Female	72500
57	AsstProf	A	3	1	Female	72500
28	AsstProf	B	7	2	Male	91300
42	AsstProf	B	4	2	Female	80225
68	AsstProf	A	4	2	Female	77500

Missing Values

Missing values are marked as NaN

In [ ]:
```
# Read a dataset with missing values
flights = pd.read_csv("~/python/data_analysis/flights.csv")
```

In [ ]:
```
# Select the rows that have at least one missing value
flights[flights.isnull().any(axis=1)].head()
```

Out[ ]:

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute
330	2013	1	1	1807.0	29.0	2251.0	NaN	UA	N31412	1228	EWR	SAN	NaN	2425	18.0	7.0
403	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EHAA	791	LGA	DFW	NaN	1389	NaN	NaN
404	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EVAA	1925	LGA	MIA	NaN	1096	NaN	NaN
855	2013	1	2	2145.0	16.0	NaN	NaN	UA	N12221	1299	EWR	RSW	NaN	1068	21.0	45.0
858	2013	1	2	NaN	NaN	NaN	NaN	AA	NaN	133	JFK	LAX	NaN	2475	NaN	NaN

Missing Values

There are a number of methods to deal with missing values in the data frame:

df.method()	description
dropna()	Drop missing observations
dropna(how='all')	Drop observations where all cells is NA
dropna(axis=1, how='all')	Drop column if all the values are missing
dropna(thresh = 5)	Drop rows that contain less than 5 non-missing values
fillna(0)	Replace missing values with zeros
isnull()	returns True if the value is missing
notnull()	Returns True for non-missing values

Missing Values

- When summing the data, missing values will be treated as zero
- If all values are missing, the sum will be equal to NaN
- cumsum() and cumprod() methods ignore missing values but preserve them in the resulting arrays
- Missing values in GroupBy method are excluded (just like in R)
- Many descriptive statistics methods have *skipna* option to control if missing data should be excluded . This value is set to *True* by default (unlike R)

Aggregation Functions in Pandas

Aggregation - computing a summary statistic about each group, i.e.

- compute group sums or means
- compute group sizes/counts

Common aggregation functions:

min, max
count, sum, prod
mean, median, mode, mad
std, var

Aggregation Functions in Pandas

agg() method are useful when multiple statistics are computed per column:

In [ ]:
```
flights[['dep_delay','arr_delay']].agg(['min','mean','max'])
```

Out[ ]:

	dep_delay	arr_delay
min	-16.000000	-62.000000
mean	9.384302	2.298675
max	351.000000	389.000000

Basic Descriptive Statistics

df.method()	description
describe	Basic statistics (count, mean, std, min, quantiles, max)
min, max	Minimum and maximum values
mean, median, mode	Arithmetic average, median and mode
var, std	Variance and standard deviation
sem	Standard error of mean
skew	Sample skewness
kurt	kurtosis

Graphics to explore the data

Seaborn package is built on matplotlib but provides high level interface for drawing attractive statistical graphics, similar to ggplot2 library in R. It specifically targets statistical data visualization

To show graphs within Python notebook include inline directive:

In [ ]:
```
%matplotlib inline
```

Graphics

	description
distplot	histogram
barplot	estimate of central tendency for a numeric variable
violinplot	similar to boxplot, also shows the probability density of the data
jointplot	Scatterplot
regplot	Regression plot
pairplot	Pairplot
boxplot	boxplot
swarmplot	categorical scatterplot
factorplot	General categorical plot

Basic statistical Analysis

statsmodel and scikit-learn - both have a number of function for statistical analysis

The first one is mostly used for regular analysis using R style formulas, while scikit-learn is more tailored for Machine Learning.

statsmodels:

- linear regressions
- ANOVA tests
- hypothesis testings
- many more ...

scikit-learn:

- kmeans
- support vector machines
- random forests
- many more ...

See examples in the Tutorial Notebook

Next Week

- Next lecture on Social Networks: Graph Representation

Any Question?

CIS 600 - Principles of Social Media and Data Mining

Spring 2023

Week 2

Priyantha Kumarawadu

Associate Teaching Professor

Department of Electrical Engineering and Computer Science

Syracuse University

Graphs for Social Network Analysis

Outline

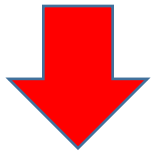
- Graph Basics
- Graph Representation
- Types of Graphs
- Connectivity in Graphs
- Graph Algorithms
- Settings for Large Network

Example: Graphs for Exploratory Analysis

- EUROVISION 2018 Vote Analysis (Credits: Dima Goldenberg)

	Rank	Running order	Country	Total	Albania	Austria	Belarus	Belgium	Croatia	Cyprus	...	Hungary	Moldova	Armenia	Czech Republic	Georgia	Montenegro
0	1	22	Israel	529	6	19	8	16	16	10	...	16	22	18	22	15	1
1	2	25	Cyprus	436	20	1	15	11	8	0	...	7	13	19	8	10	6
2	3	5	Austria	342	2	0	10	15	0	2	...	11	3	7	5	9	0
3	4	11	Germany	340	14	16	0	7	3	3	...	1	8	5	3	7	0
4	5	26	Italy	308	24	10	4	6	10	15	...	6	8	3	2	5	12

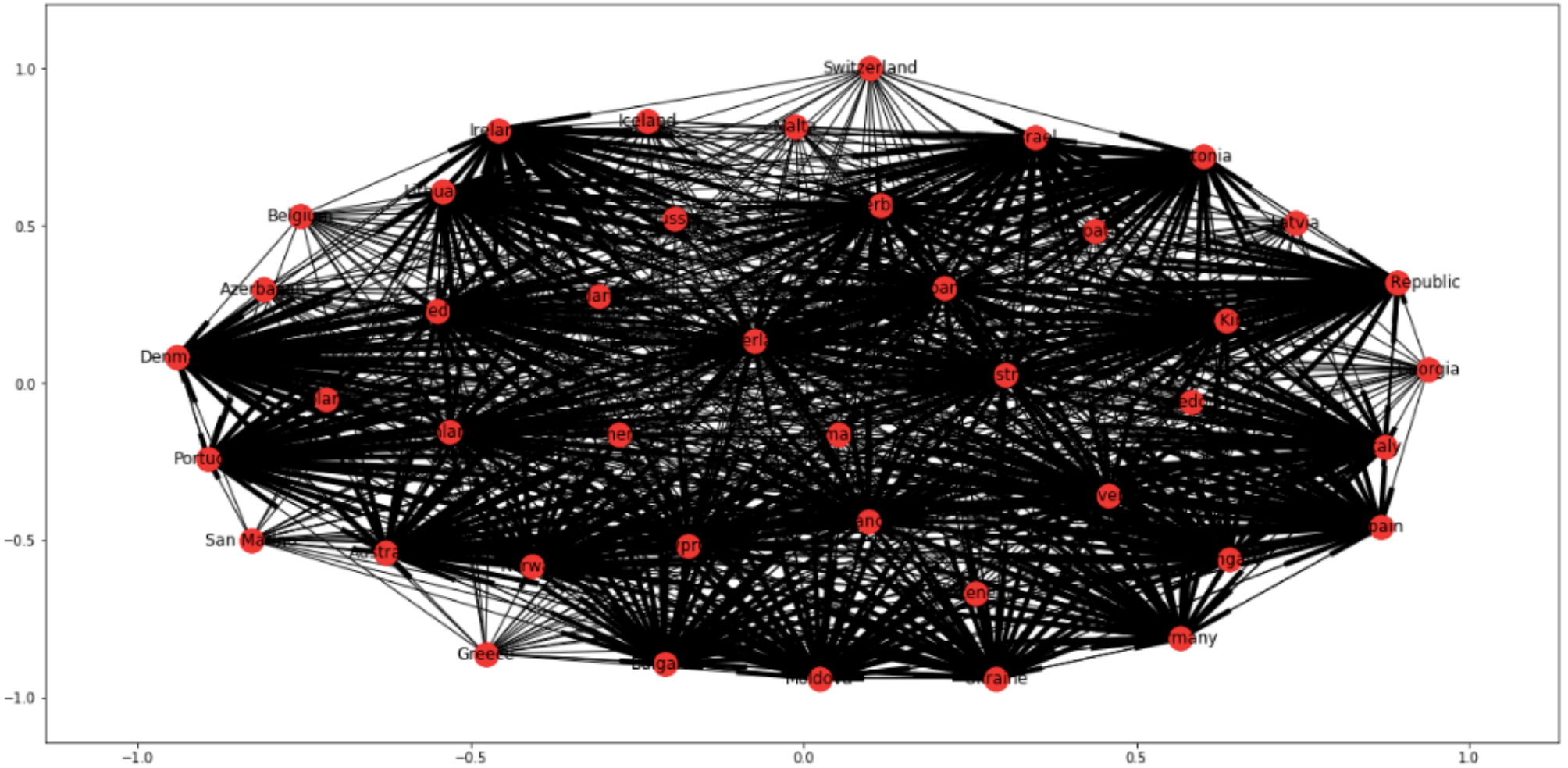
5 rows × 47 columns



	Rank	Running order	Country	Total	Source Country	points
0	1	22	Israel	529	Albania	6
1	2	25	Cyprus	436	Albania	20
2	3	5	Austria	342	Albania	2
3	4	11	Germany	340	Albania	14
4	5	26	Italy	308	Albania	24

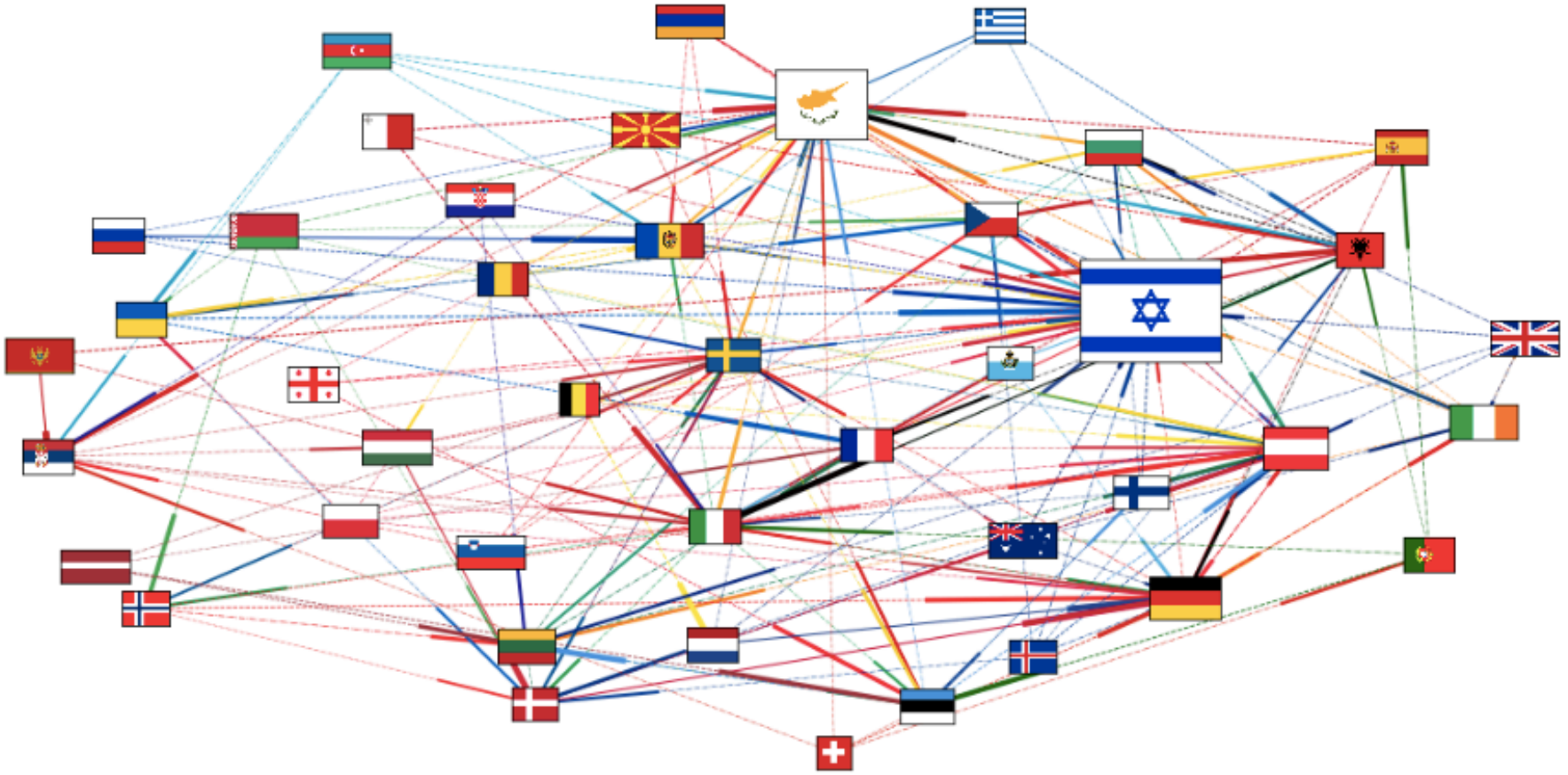
Example: Graphs for Exploratory Analysis

- EUROVISION 2018 Vote Analysis (Credits: Dima Goldenberg)



Example: Graphs for Exploratory Analysis

- EUROVISION 2018 Vote Analysis (Credits: Dima Goldenberg)

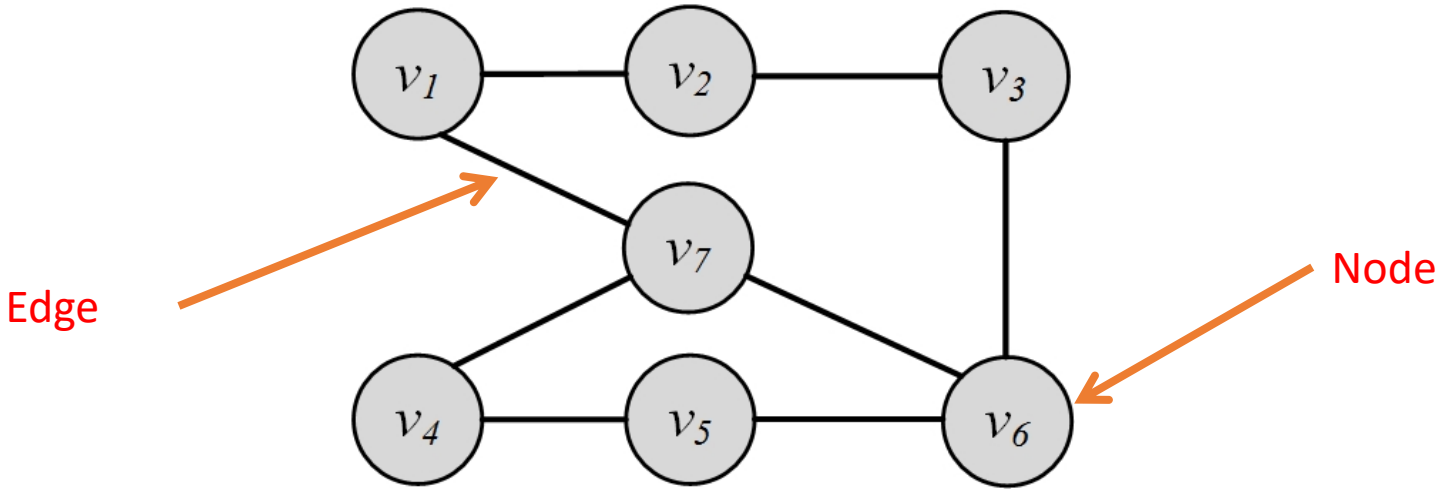


Graph Basics

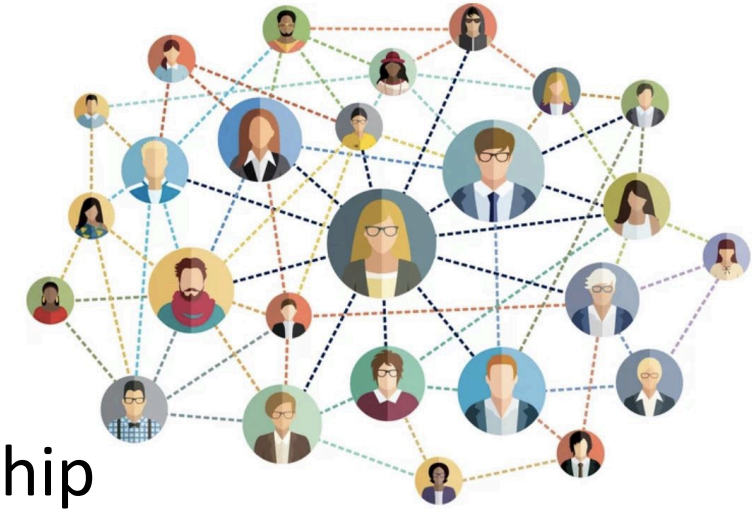
Nodes and Edges

A network is a graph, or a collection of points connected by lines

- Points are referred to as **nodes, actors,** or **vertices** (plural of **vertex**)
- Connections are referred to as **edges** or **ties**



Nodes



In a social friendship graph

People → Nodes Any pair connected → Relationship

Depending on the context, these nodes are called nodes, or actors

- In a web graph, "*nodes*" represent sites and the connection between nodes indicates web-links between them
- In a social setting, these nodes are called actors

$$V = \{v_1, v_2, \ldots, v_n\}$$

- The size of the graph is

$$|V| = \mathbf{n}$$

Edges

Any pair connected → Relationship → Edge

- In a social setting, edges indicate internode relationships

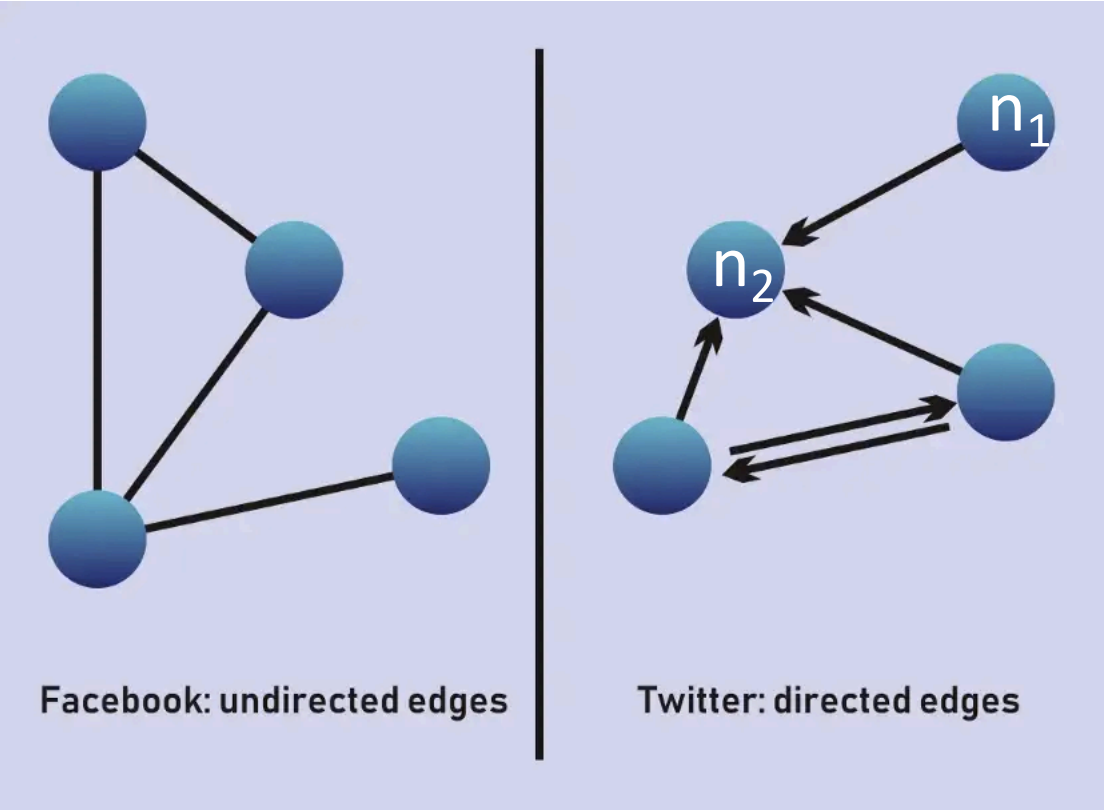
$$E = \{e_1, e_2, \ldots, e_m\}$$

- Number is edges (size of the edge-set) is denoted as

$$|E| = \mathbf{m}$$

Directed Edges

- Edges can have directions
- Edges are represented using their endpoints. $e\ (n_1, n_2)$
- In undirected graphs both representations are the same



Neighborhood and Degree (In-degree, out-degree)

For any node v, in an undirected graph, the set of nodes it is connected to via an edge is called its neighborhood and is represented as $N(v)$

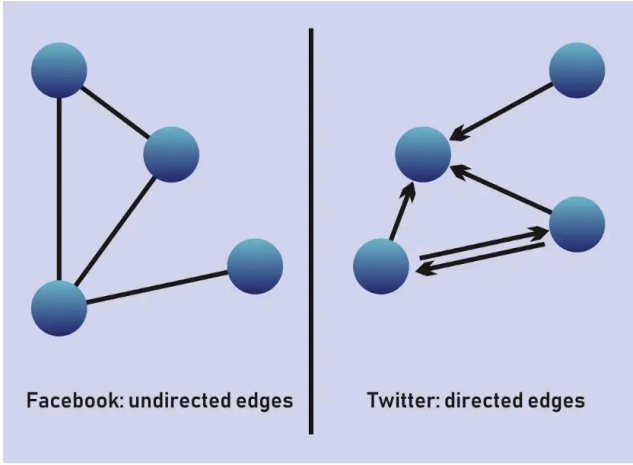
- *In directed graphs we have incoming neighbors $N_{in}(v)$ (nodes that connect to v) and outgoing neighbors $N_{out}(v)$.*

The number of edges connected to one node is the degree of that node (the size of its neighborhood)

- Degree of a node i is usually presented using notation d_i

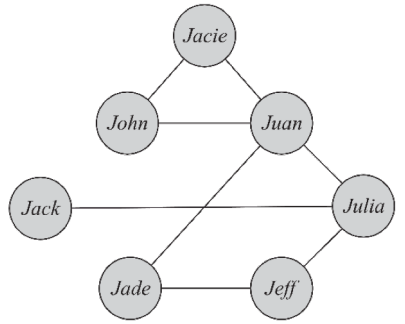
In Directed graphs:

- In-degrees is the number of edges pointing towards a node d_i^{in}
- Out-degree is the number of edges pointing away from a node d_i^{out}



Degree Distribution

When dealing with very large graphs, how nodes' degrees are distributed is an important concept to analyze and is called ***Degree Distribution***



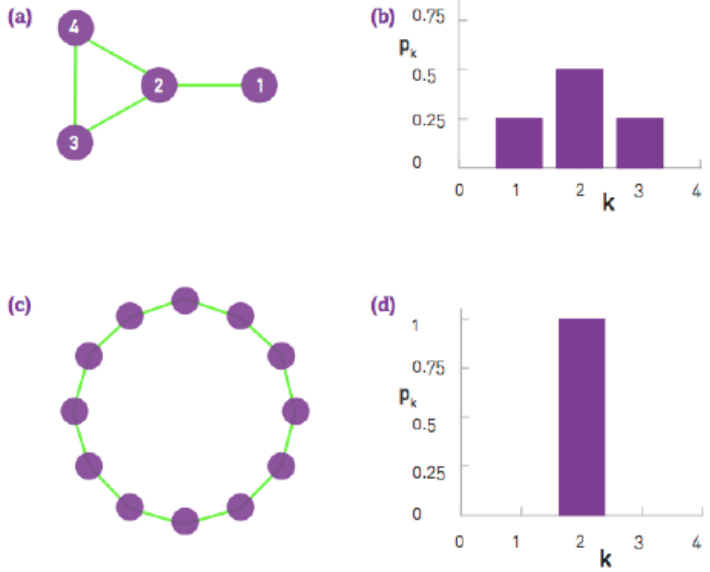
$$p_1 = \frac{1}{7}, p_2 = \frac{4}{7}, p_3 = \frac{1}{7}, p_4 = \frac{1}{7}$$

$$\pi(d) = \{d_1, d_2, \ldots, d_n\}$$

$$p_d = \frac{n_d}{n}$$

n_d is the number of nodes with degree d

$$\sum_{d=0}^{\infty} p_d = 1$$



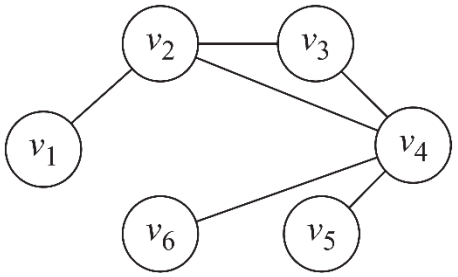
Graph Representation

- Adjacency Matrix
- Adjacency List
- Edge List

Mathematical Representation

- Should not lose information
- Able to manipulate easily by computers
- Can use with mathematical methods which can be applied on them easily
- An edge $e_{ij} = (v_i, v_j)$ is a pair of vertices (ordered pair for directed graph)
- Adjacency Matrix (a.k.a. sociomatrix)

$$A_{ij} = \begin{cases} 1, \text{ if there is an edge between nodes } v_i \text{ and } v_j \\ 0, \text{ otherwise} \end{cases}$$



(a) Graph

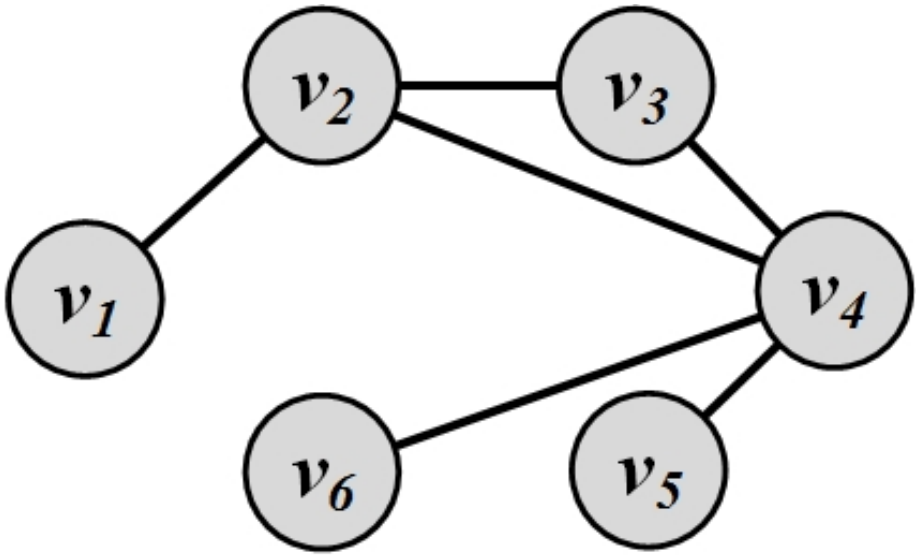
	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	0	0	0
v_2	1	0	1	1	0	0
v_3	0	1	0	1	0	0
v_4	0	1	1	0	1	1
v_5	0	0	0	1	0	0
v_6	0	0	0	1	0	0

(b) Adjacency Matrix

Social media networks have very sparse Adjacency Matrices

Adjacency List

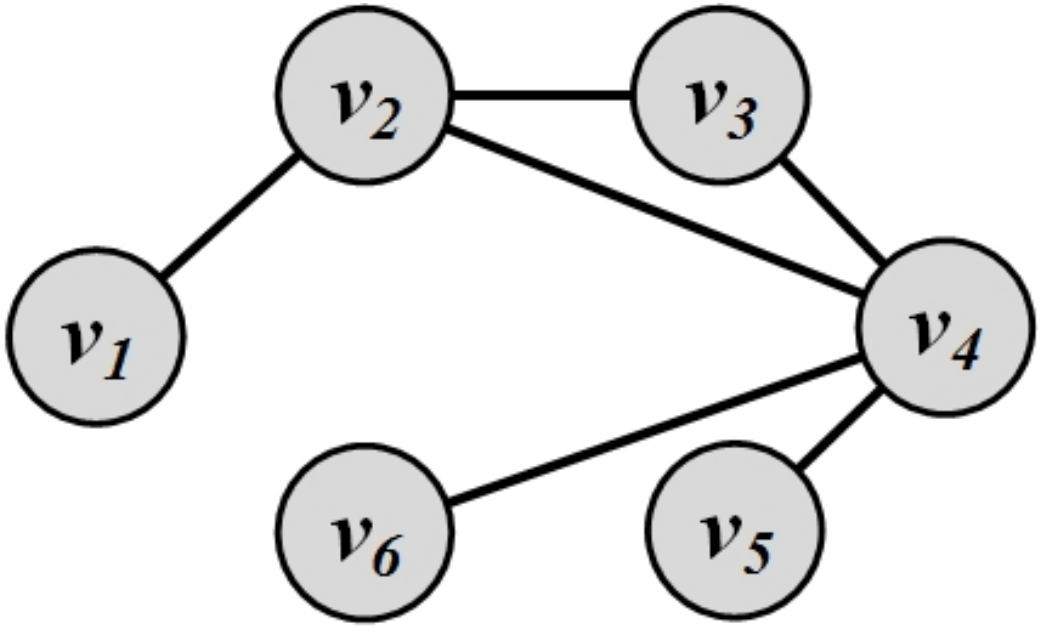
- maintain a list of all the nodes that it is connected to
- The list is usually sorted based on the node order or other preferences



Node	Connected To
v_1	v_2
v_2	v_1, v_3, v_4
v_3	v_2, v_4
v_4	v_2, v_3, v_5, v_6
v_5	v_4
v_6	v_4

Edge List

- An **edge list** is a list or array of pairs which represents the edges of a graph, with each pair consisting of the two unique IDs of the nodes involved



(v_1, v_2)
(v_2, v_3)
(v_2, v_4)
(v_3, v_4)
(v_4, v_5)
(v_4, v_6)

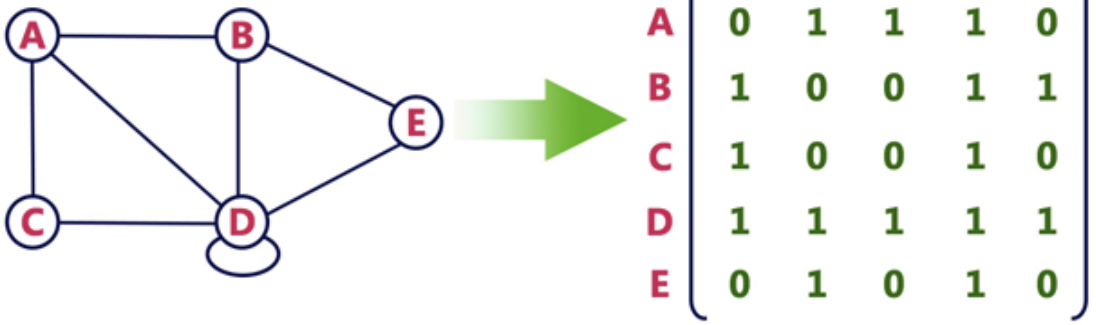
Types of Graphs

- Null and Empty Grapps
- Directed and Undirected Graphs
- Simple/Multigraph and Weighted Graphs
- Signed Graph, Webgraphs

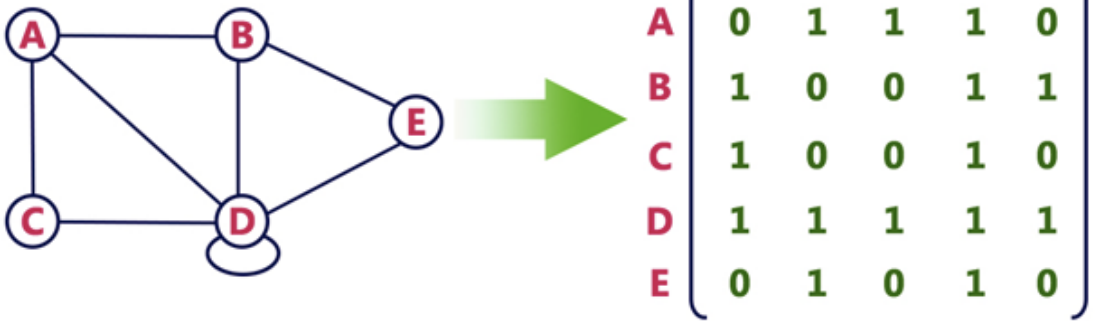
Null Graph or Empty Graph

- A null graph is **a graph in which there are no edges between its vertices**. A null graph is also called empty graph.

Directed and Undirected Graphs



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



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

- The adjacency matrix for directed graphs is often not symmetric ($\boldsymbol{A \neq A^T}$)
  - $\boldsymbol{A_{ij} \neq Aji}$
  - We can have equality though

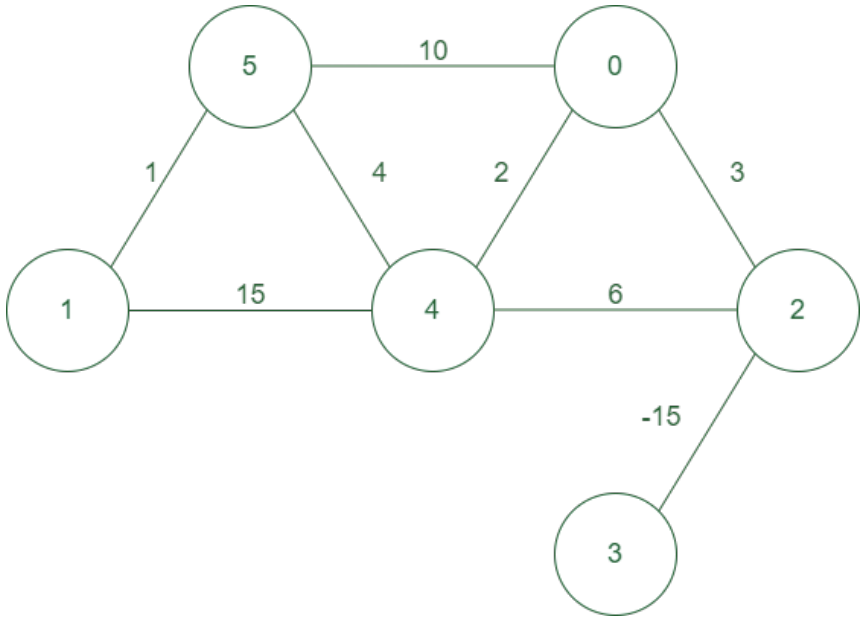
The adjacency matrix for undirected graphs is symmetric ($\boldsymbol{A = A^T}$)

Representing Graphs in a Computer

Weighted Graphs

- The edges are assigned some weights which represent cost, distance and many other relative measuring units.

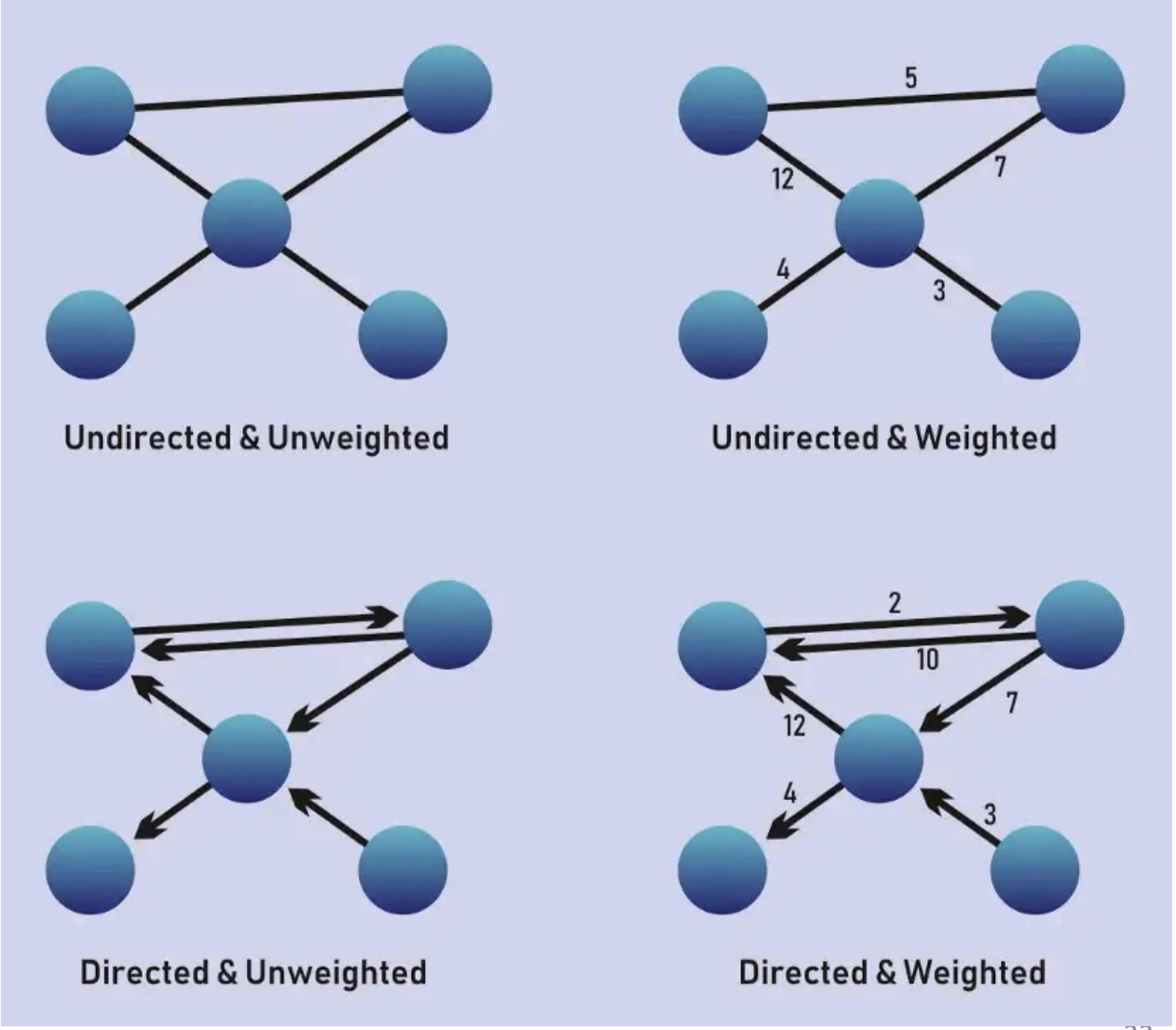
$$A_{ij} = \begin{cases} w_{ij} \text{ or } w(i,j), w \in R \\ 0, \text{There is no edge between } v_i \text{ and } v_j \end{cases}$$



Social network graphs: We can find which all users are connected in a network both directly(direct connection) and indirectly(indirect connection). In social media, for example, to represent features like close friends (Instagram) which is not the same as all friends (implemented using weighted graphs)

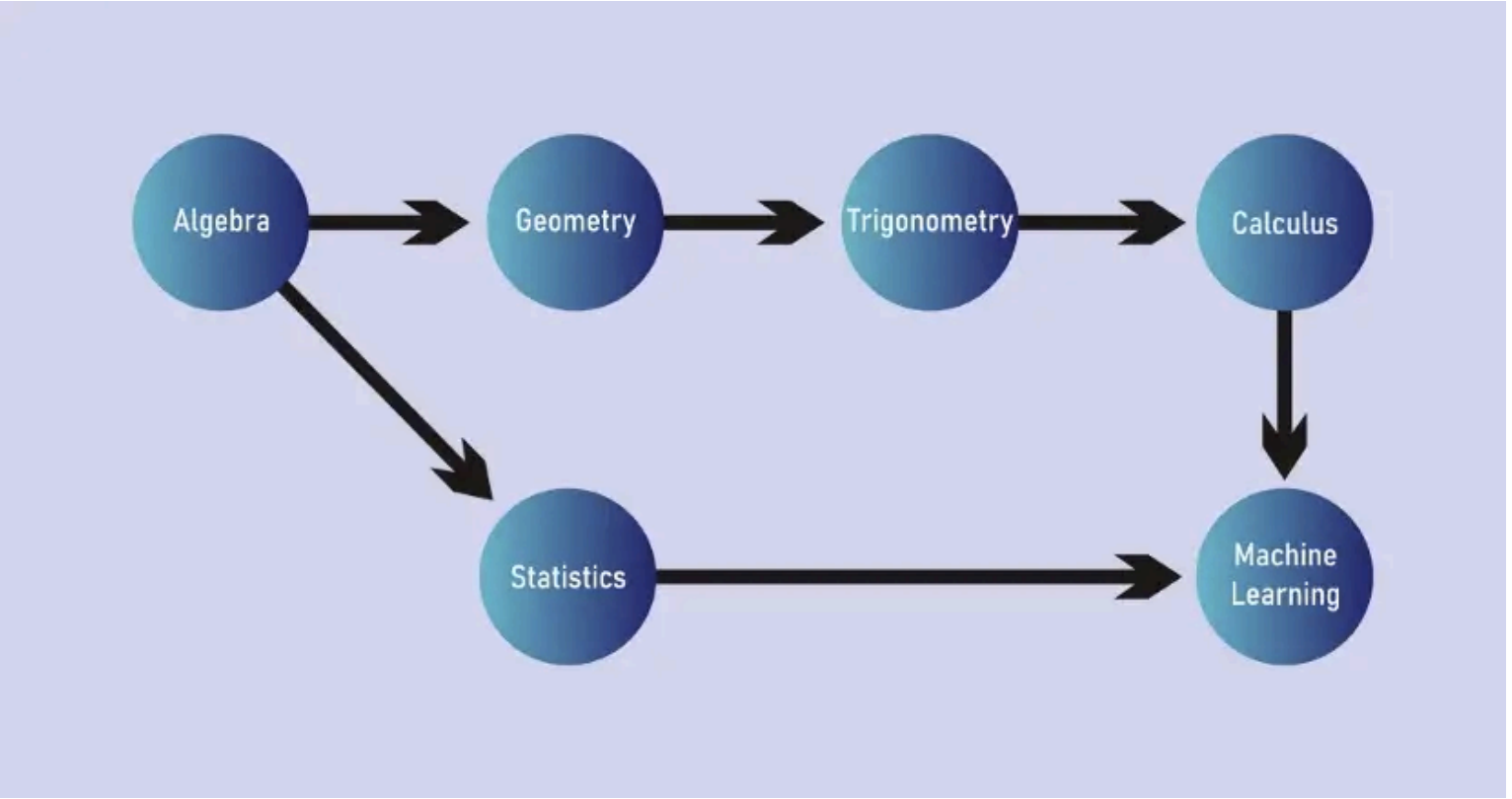
Weighted and Unweighted Graphs

- Undirected & Unweighted: relationships do not have magnitude and are bidirectional.
- Undirected & Weighted: relationships have a magnitude and are bidirectional.
- Directed & Unweighted: relationships do not have magnitude and are one way.
- Directed & Weighted: relationships have a magnitude and are one way.



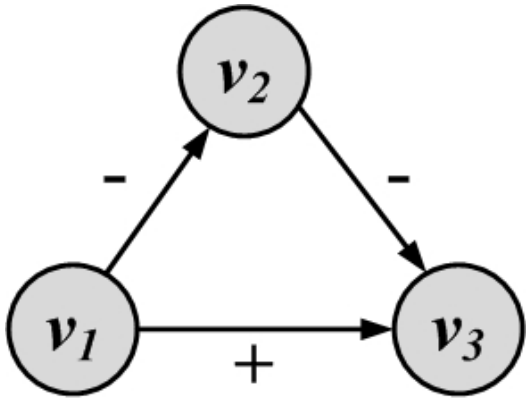
Example: Directional Graphs

- College Courses and Prerequisites



Signed Graphs

- Signed graphs have been used for a long time in social network analysis to model opposite relationships.
- When weights are binary (0/1, -1/1, +/-) we have a signed graph



- Modeling diplomatic relations between countries, a positive edge can represent cooperation and a negative edge can represent some kind of political tension.
- To represent friends or foes

Webgraphs

- **Webgraph** is a graph having static HTML pages as nodes (vertices) and directed hyperlinks among the pages as edges
- Directed graph
- Nodes represent sites and edges represent links between sites.
- Two sites can have multiple links pointing to each other and can have loops (links pointing to themselves)
- Applications:
  - **Search algorithms** that rank results based on the hyperlinks between pages.
  - **SPAM detection methods** which identity networks of web pages that are published in order to trick search engines.
  - **graph analysis algorithms** and can use the hyperlink graphs for testing the scalability and performance of their tools.
  - **Web Science researchers** who want to analyze the linking patterns within specific topical domains in order to identify the social mechanisms that govern these domains.

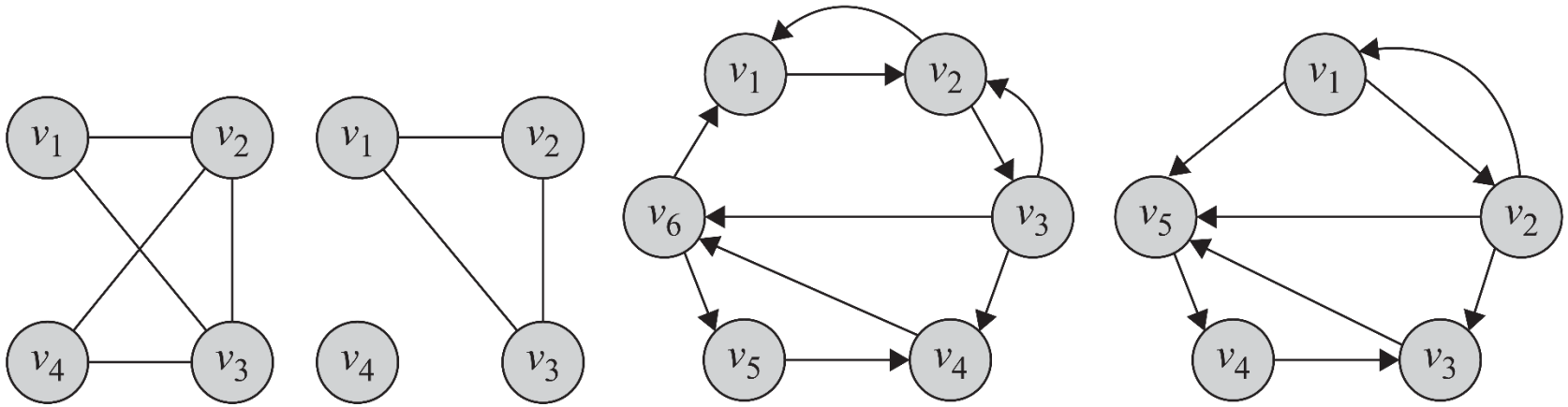
Connectivity in Graphs

- **A node v_i is connected to node v_j** (or reachable from v_j) if it is adjacent to it or there exists a path from v_i to v_j.

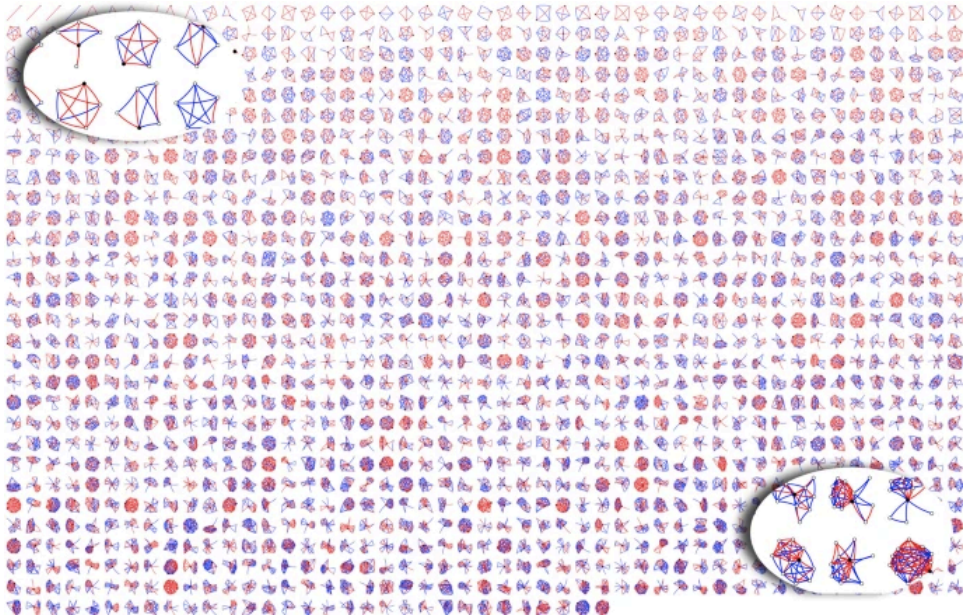
- **A graph is connected**, if there exists a path between any pair of nodes in it
  - In a directed graph, **a graph is strongly connected** if there exists a directed path between any pair of nodes
  - In a directed graph, **a graph is weakly connected** if there exists a path between any pair of nodes, without following the edge directions

- A graph is **disconnected,** if it not connected.

Connectivity Example



(a) Connected (b) Disconnected (c) Strongly connected (d) Weakly connected



- Social network assessment to quantify patients' personal network structure and health characteristics
- Participants' personal social network. Each small network has a black circle that represents the participant who is surrounded by white circles who are the network members. The lines connecting the circles are red if the relationship is strong and blue if the relationship is weak.

https://www.nature.com/articles/s41467-018-06408-6

Adjacent Nodes and Incident Edges

Two nodes are adjacent if they are connected via an edge.

Two edges are incident, if they share on end-point

When the graph is directed, edge directions must match for edges to be incident

An edge in a graph can be traversed when one starts at one of its end-nodes, moves along the edge, and stops at its other end-node.

Walk, Path, Trail, Tour and Cycle

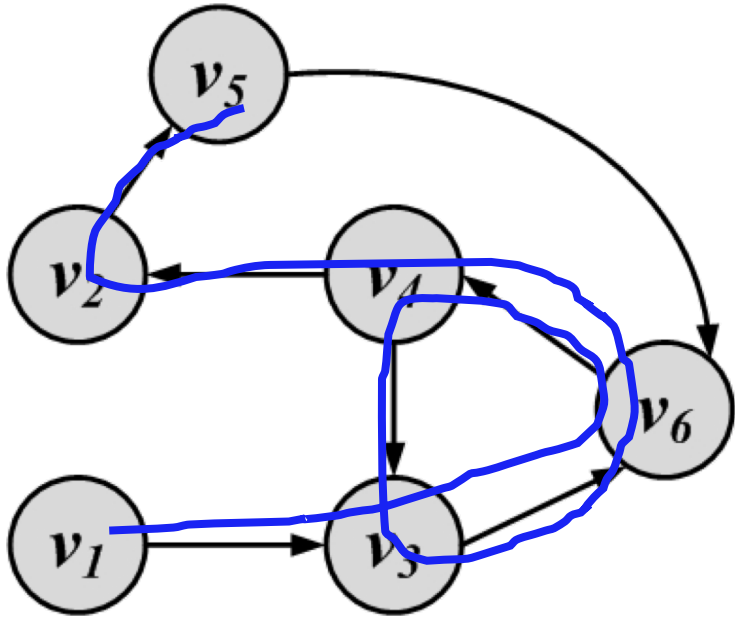
Walk: A walk is a sequence of incident edges visited one after another

- **Open walk**: A walk does not end where it starts
- **Closed walk**: A walk returns to where it starts

- Representing a walk:
  - A sequence of edges: $e_1, e_2, \dots, en$
  - A sequence of nodes: $v_1, v_2, \dots, vn$

- Length of walk:

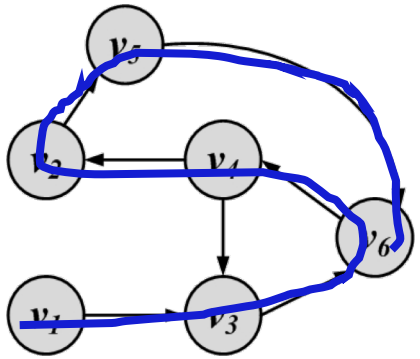
  the number of visited edges



Length of walk= 8

Trail

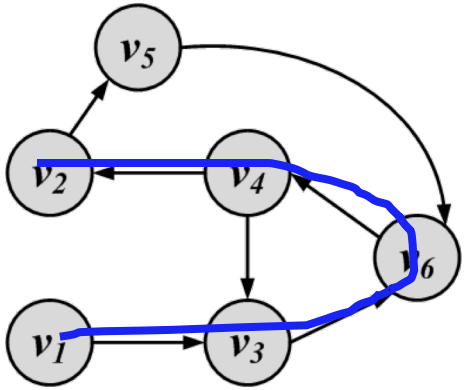
- A trail is a walk where **no edge** **is visited more than once** and all walk edges are distinct



- A closed trail (one that ends where it starts) is called a **tour** or **circuit**

Path

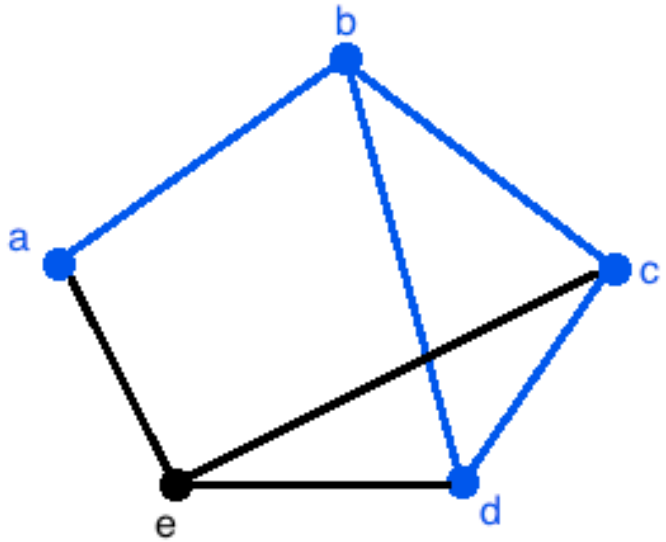
- A walk where **nodes and edges** **are distinct** is called a **path** and a closed path is called a **cycle**
- The length of a path or cycle is the number of edges visited in the path or cycle



Length of path= 4

Walk

A Walk is defined as a sequence of alternating vertices and edges such as v0,e1,v1,e2,...,ek,vk where each edge ei={vi−1,vi}. The **Length** of this walk is k.



walk abcdcbce,
Length = 4

Random Walk

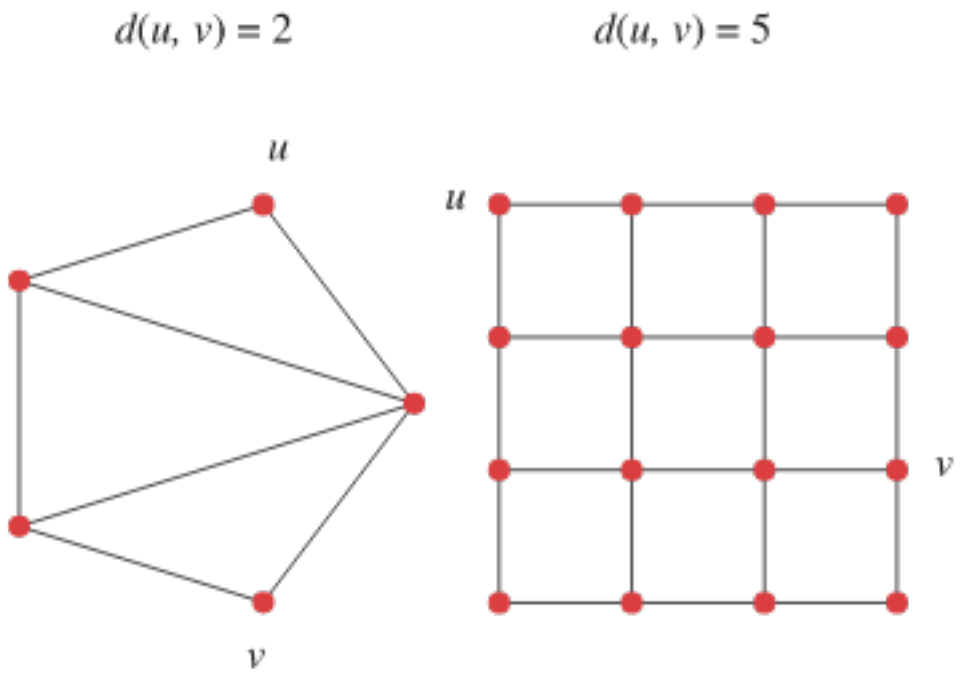
- A random process which describes a path that consists of a succession of random steps on some mathematical space:
  - given a graph and a starting point, select a neighbour at random
  - move to the selected neighbour and repeat the same process till a termination condition is verified
  - the random sequence of points selected in this way is a *random walk* of the graph
  - The weight of an edge can be used to define the probability of visiting it
  - For all edges that start at v_i the following equation holds

- Random walks on directed graphs: the basis of Google's PageRank algorithm

$$\sum_x w_{i,x} = 1, \forall i, j \quad w_{i,j} \geq 0$$

Graph Distance

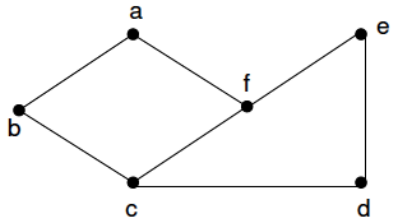
- The distance *d(u,v)* between two vertices *u* and *v* of a finite graph is the minimum length of path connecting them.



Shortest Path

- **Shortest Path** is the path between two nodes that has the shortest length.
  - We denote the length of the shortest path between nodes v_i and v_j as $l_{i,j}$
- The concept of the neighborhood of a node can be generalized using shortest paths. An **n-hop neighborhood** of a node is the set of nodes that are within n hops distance from the node.
- Graph diameter is the largest shortest path:

$$D = max_{i,j}\, d_G(v_i, v_j)$$



Graph diameter = 3

Node pair	Shortest Distance
a, b	1
a, c	2
a, d	3
⋮	⋮
b, e	3
⋮	⋮
e, f	1

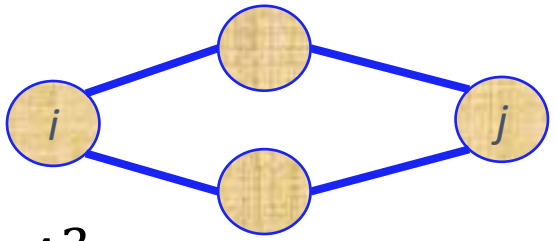
Adjacency Matrix and Connectivity

- Consider the following adjacency matrix

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1n} \\ A_{21} & A_{22} & A_{23} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ A_{d1} & A_{d2} & A_{d3} & \dots & A_{dn} \end{bmatrix}$$

- Number of Common neighbors between node i and node j

$$\sum_k A_{ik} A_{jk} = A_i \cdot A_j$$



- That's element of [ij] of matrix $A \times A^T = A^2$
- Common neighbors are paths of length 2
- Similarly, what is A^3?

Graph Algorithms

- Depth-First Search (DFS)
- Breadth-First Search (BFS)
- Dijkstra’s Algorithm for shortest path

Graph Traversal

- We are interested in surveying a social media site to computing the average age of its users
  - Start from one user;
  - Employ some traversal technique to reach her friends and then friends' friends, …
- The traversal technique guarantees that
  1. All users are visited; and
  2. No user is visited more than once.
- There are two main techniques:
  - **Depth-First Search (DFS)**
  - **Breadth-First Search (BFS)**

Depth-First Search (DFS)

- Depth-First Search (DFS) starts from a node v_i, selects one of its neighbors v_j from $N(v_i)$ and performs Depth-First Search on v_j before visiting other neighbors in $N(v_i)$

- The algorithm can be used both for trees and graphs
  - The algorithm can be implemented using a stack structure
- The waiting list in depth-first, a stack (LIFO)

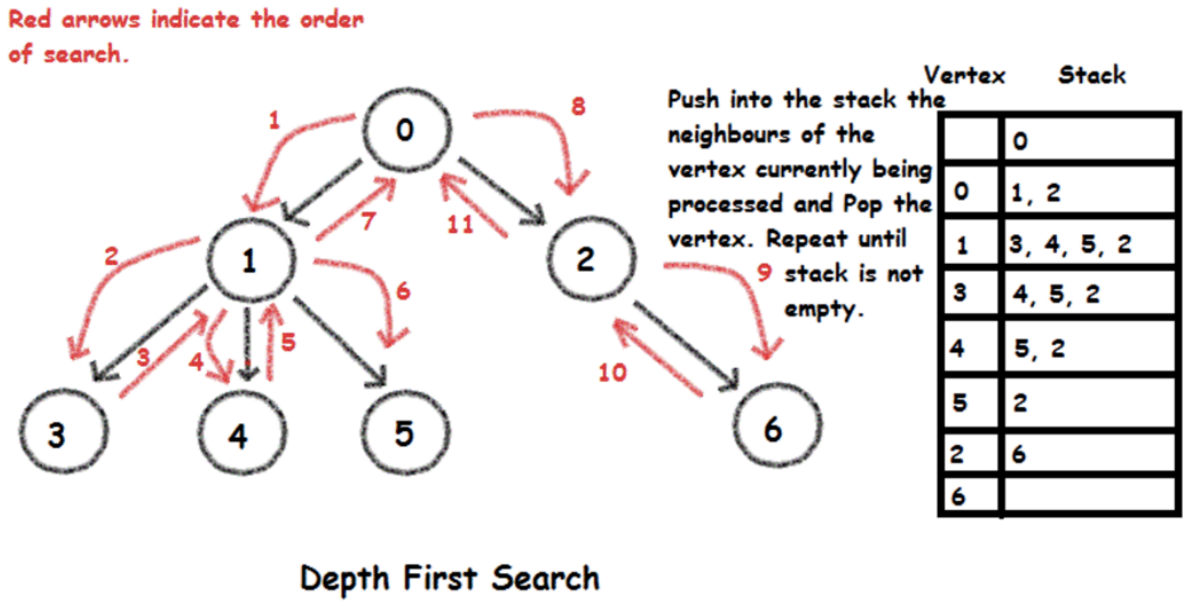
DFS Structure

Algorithm 2.2 Depth-First Search (DFS)

Require: Initial node v, graph/tree $G(V, E)$, stack S

1: **return** An ordering on how nodes in G are visited
2: Push v into S;
3: *visitOrder* = 0;
4: **while** S not empty **do**
5: *node* = pop from S;
6: **if** *node* not visited **then**
7: *visitOrder* = *visitOrder* +1;
8: Mark *node* as visited with order *visitOrder*; `//or print` *node*
9: Push all neighbors/children of *node* into S;
10: **end if**
11: **end while**
12: Return all nodes with their visit order.

DFS


Red arrows indicate the order of search.
Push into the stack the neighbours of the vertex currently being processed and Pop the vertex. Repeat until stack is not empty.
Vertex
Stack
0
1
2
3
4
5
6
1
2
3
4
5
6
7
8
9
10
11
0 | 0
0 | 1, 2
1 | 3, 4, 5, 2
3 | 4, 5, 2
4 | 5, 2
5 | 2
2 | 6
6 |
Depth First Search

Breadth-First Search

- BFS starts from a node and visits all its immediate neighbors first, and then moves to the second level by traversing their neighbors.

- The algorithm can be used both for trees and graphs
  - The algorithm can be implemented using a queue structure
- The waiting list in breadth-first traversal is a queue (FIFO)

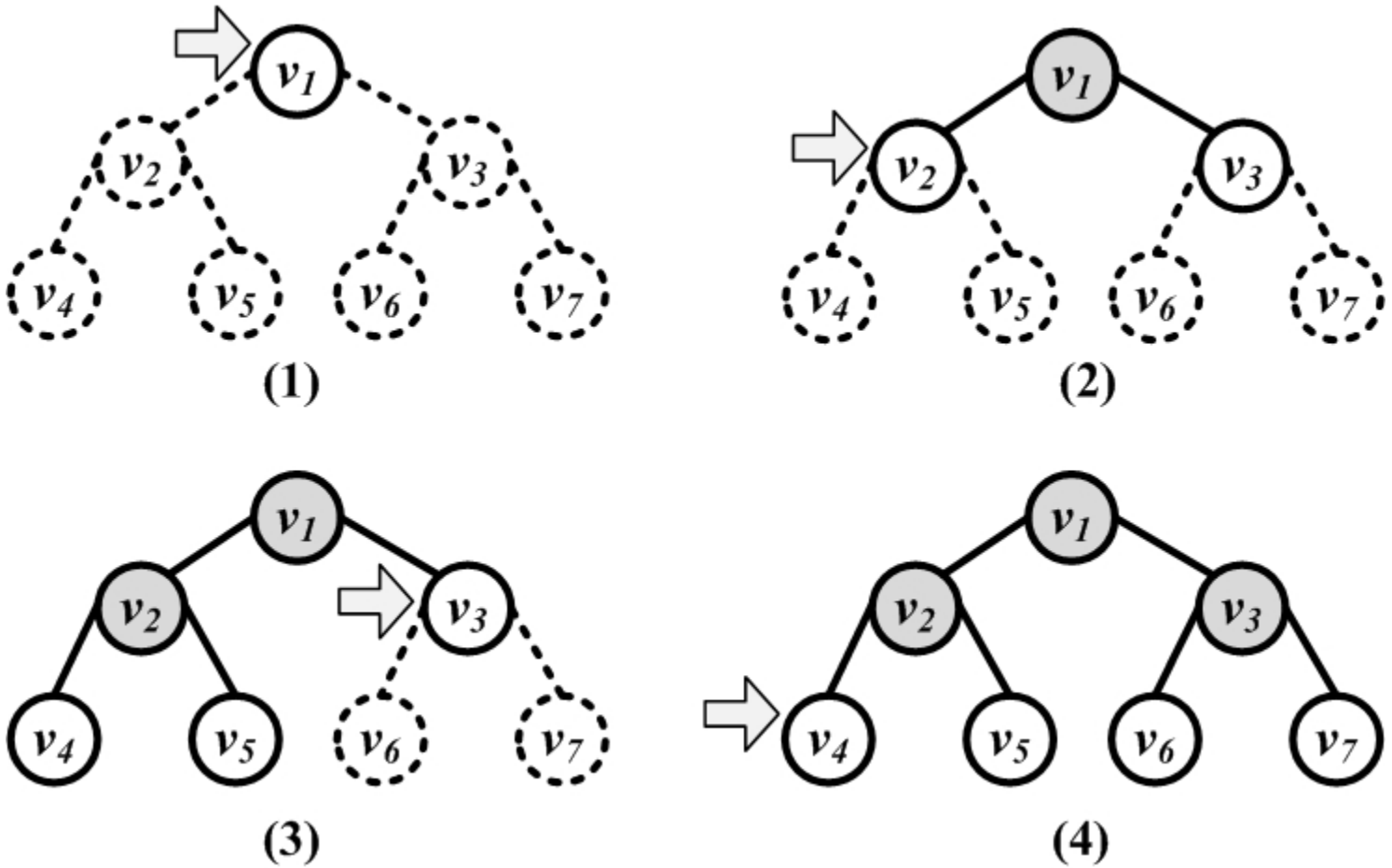
BFS Algorithm

Algorithm 2.3 Breadth-First Search (BFS)

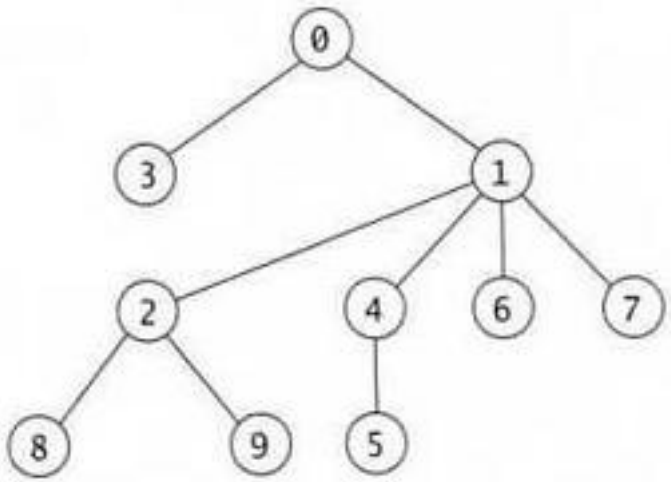
Require: Initial node v, graph/tree $G(V, E)$, queue Q

1: **return** An ordering on how nodes are visited
2: Enqueue v into queue Q;
3: *visitOrder* = 0;
4: **while** Q not empty **do**
5: *node* = dequeue from Q;
6: **if** *node* not visited **then**
7: *visitOrder* = *visitOrder* +1;
8: Mark *node* as visited with order *visitOrder*; `//or print` *node*
9: Enqueue all neighbors/children of *node* into Q;
10: **end if**
11: **end while**

BFS



BFS



Vertex Being Visited	Queue Contents After Visit	Visit Sequence
0	1 3	0
1	3 2 4 6 7	0 1
3	2 4 6 7	0 1 3
2	4 6 7 8 9	0 1 3 2
4	6 7 8 9 5	0 1 3 2 4
6	7 8 9 5	0 1 3 2 4 6
7	8 9 5	0 1 3 2 4 6 7
8	9 5	0 1 3 2 4 6 7 8
9	5	0 1 3 2 4 6 7 8 9
5	empty	0 1 3 2 4 6 7 8 9 5

Dijkstra's Algorithm for the Shortest Path

When a graph is connected, there is a chance that multiple paths exist between any pair of nodes

- In many scenarios, we want the shortest path between two nodes in a graph
  - How fast can I disseminate information on social media?

Dijkstra's Algorithm

- Designed for weighted graphs with non-negative edges
- It finds shortest paths that start from a provided node s to all other nodes
- It finds both shortest paths and their respective lengths

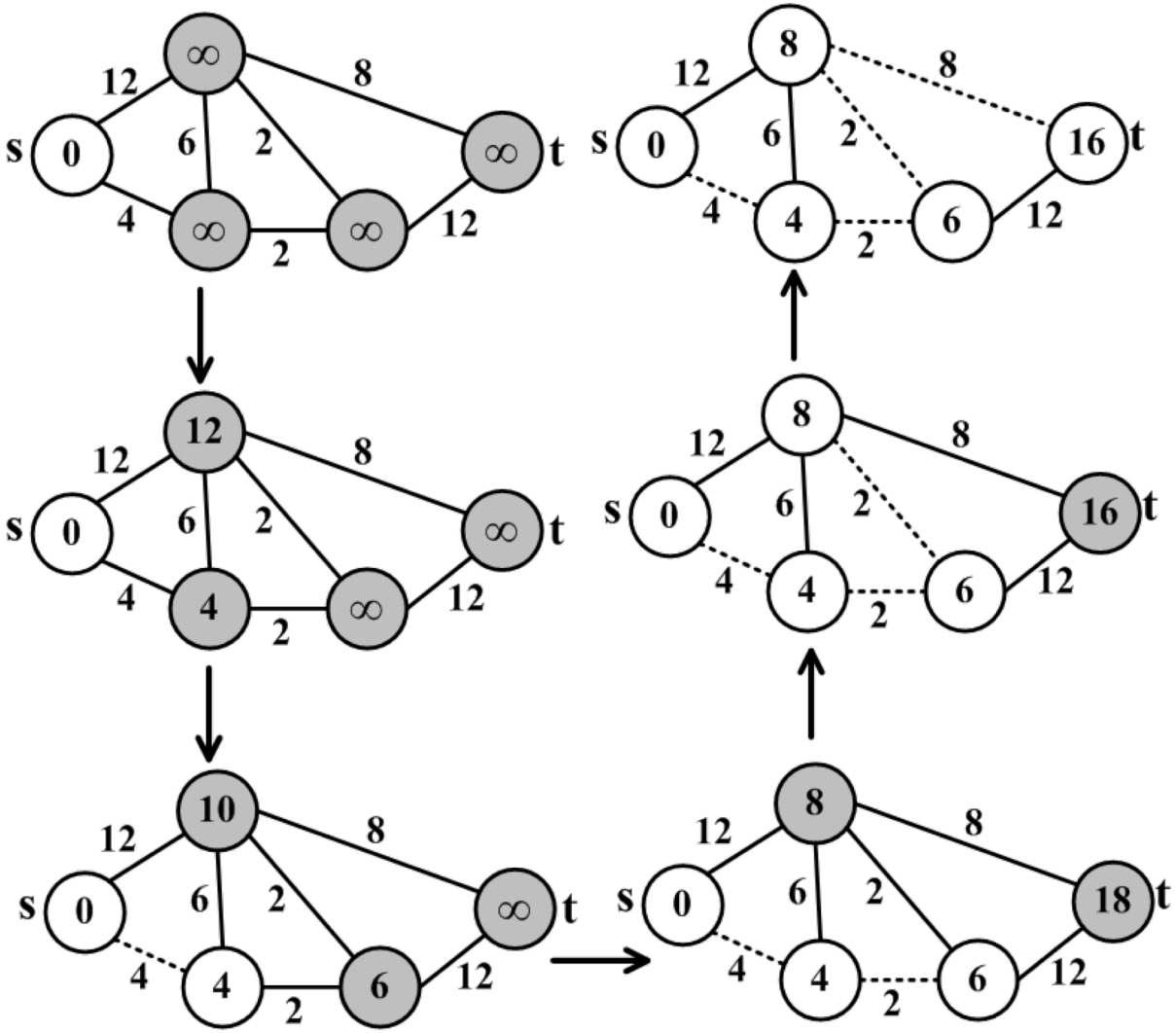
Dijkstra's Algorithm for the Shortest Path

1. Initiation:
   - Assign zero to the source node and infinity to all other nodes
   - Mark all nodes as unvisited
   - Set the source node as current
2. For the current node, consider all of its unvisited neighbors and calculate their *tentative* distances
   - If tentative distance is smaller than neighbor's distance, then Neighbor's distance = tentative distance
3. After considering all of the neighbors of the current node, mark the current node as visited and remove it from the *unvisited set*
4. If the destination node has been marked visited or if the smallest tentative distance among the nodes in the *unvisited set* is infinity, then stop
5. Set the unvisited node marked with the smallest tentative distance as the next "current node" and go to step 2

Tentative distance = current distance + edge weight

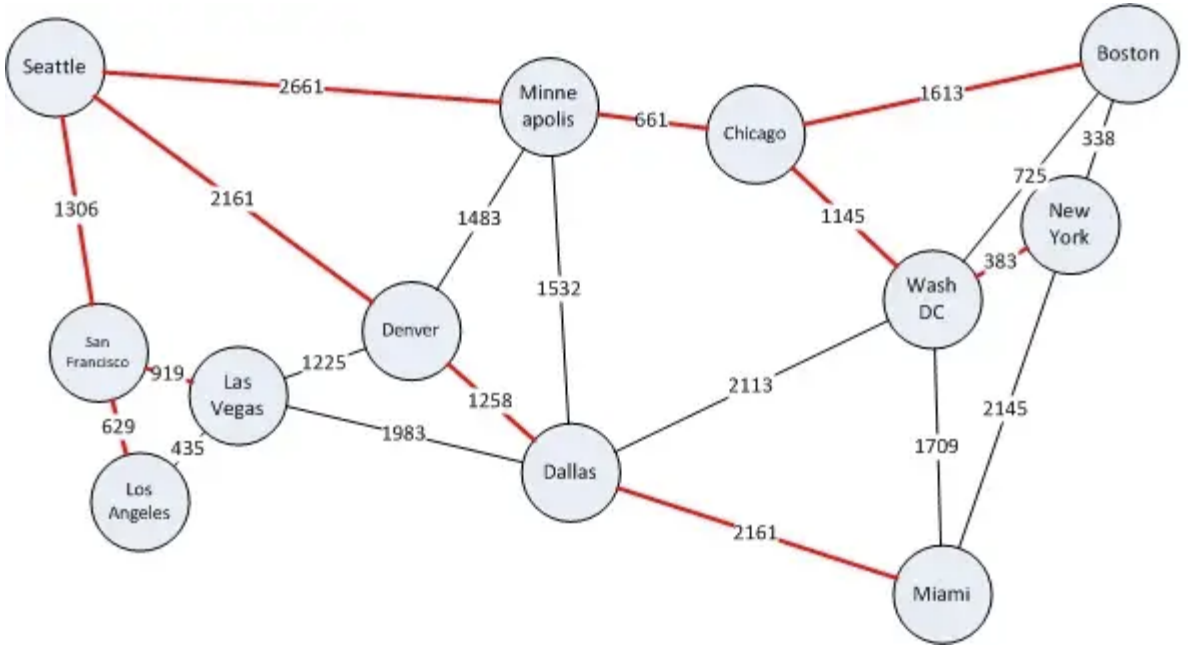
A visited node will never be checked again and its distance recorded now is final and minimal

Dijkstra's Algorithm: Execution Example



Applications

- One common application that uses Dijkstra's algorithm is the airport problem (quickest series of fights)



Dijkstra's Algorithm Notes

- Dijkstra's algorithm is source-dependent
  - Finds the shortest paths between the source node and all other nodes.

- To generate all-pair shortest paths,
  - We can run Dijsktra's algorithm n times, or
  - Use other algorithms such as Floyd-Warshall algorithm.

- If we want to compute the shortest path from source v to destination d,
  - we can stop the algorithm once the shortest path to the destination node has been determined

Settings that Provide Large Network Data

- Manually constructed social networks involving human interactions are small.
- Other settings provide larger data sets representing interactions (which are not necessarily through direct contact).

A. Collaboration Networks: (\Who Works With Whom")

- Co-authorship networks
  - Rich form of interaction over a long period of time (suitable for longitudinal studies).
  - Nodes with high degrees likely to represent in scientists.
- Co-appearance in movies
- Co-membership in Board of Directors of large companies: used to explain business decisions made by companies.

Settings that Provide Large Network Data contd

- B. Networks from Communication Among People: (\Who Talks to Whom")
  - Internet Messenger example [Horovitz & Leskovec, 2008] discussed earlier.
  - Email logs within a company: The most famous example is the Enron data set.
  - Call graphs constructed from phone numbers: Privacy of individuals must be protected.

Settings that Provide Large Network Data contd

C. Information Linkage Graphs:

- Web data:
  - Directed graph with nearly 5 billion nodes.
  - Extremely large for effective processing using commodity hardware.
  - Researchers work with reasonable subsets (e.g. linkage among
  - bloggers, linkage among articles of Wikipedia).
- Citation networks:
  - Useful in tracking the development of disciplines (e.g.
  - identifying “central papers" of a discipline).
  - Also useful for longitudinal studies.

Settings that Provide Large Network Data contd

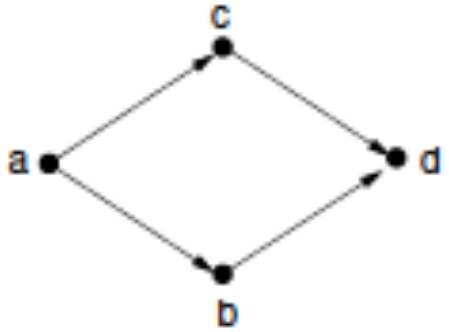
D. Technological Networks:

- Computer networks
- Power grid

E. Networks in the Natural World:

(a) Food Web (\Who Eats Whom" relationship):

- Directed edge x → y indicates that species x eats species y.
- Important in studying cascading extinction of species.



Summary

- Graph Basics
- Graph Representation
- Types of Graphs
- Connectivity in Graphs
- Graph Algorithms
- Settings for Large Network

Next Week

- Next lecture on Social Networks: Network Analysis

Any Question?

CIS 600 - Principles of Social Media and Data Mining

Spring 2023

Week 4

Social Network Analysis Metrices

Priyantha Kumarawadu
Associate Teaching Professor
Department of Electrical Engineering and Computer Science
Syracuse University

Outline

- Types of Network Metrices
  - Node Level
  - Group Level

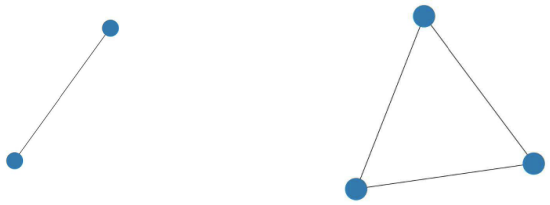
Types of network metrics

- Group level
  - Density
  - Components
    - Isolates
  - Cliques
  - Centralization
    - Degree
    - Closeness
    - Betweenness
- Node level
  - Centrality
    - Degree
      - Indegree
      - Outdegree
    - Closeness
    - Betweenness
- Statistical metrics
  - Multi-dimensional Scaling (MDS)
  - Cluster analysis

Basic Concepts of Social Networks

Social networks represent the **relationship between social entities.**

- **Actor (Nodes):** Each social entity is called an actor. Each entity can be of the same type, as well as of different types.
- **Relation Tie (Edge) :** Ties are the connection between actors.
- **Dyad, Triad, and Subgroup:** A pair of existing or possible two actors are called dyads, while a subset of three existing actors is called triads. Generalizing this logic, a subset of any subset of actors is called a subgroup.
- **Group:** The collection of all actors is called a group
- **Relation:** The collection of ties in the specific group is called relation



Preparing the data

- Most graph-based metrics require symmetric, dichotomous (binary) data
- Symmetrization
  - Minimum (captures reciprocal ties)
  - Maximum (most inclusive)
- Dichotomize
  - Valued data are typically ordinal
  - Mean, median or modal values

Network Density

- **Measure of the prevalence of dyadic linkage or direct tie within a social network.**
- The proportion of ties that exist out of all possible ties
- For valued data **the sum of all tie strengths** divided by all possible ties

For undirected graphs

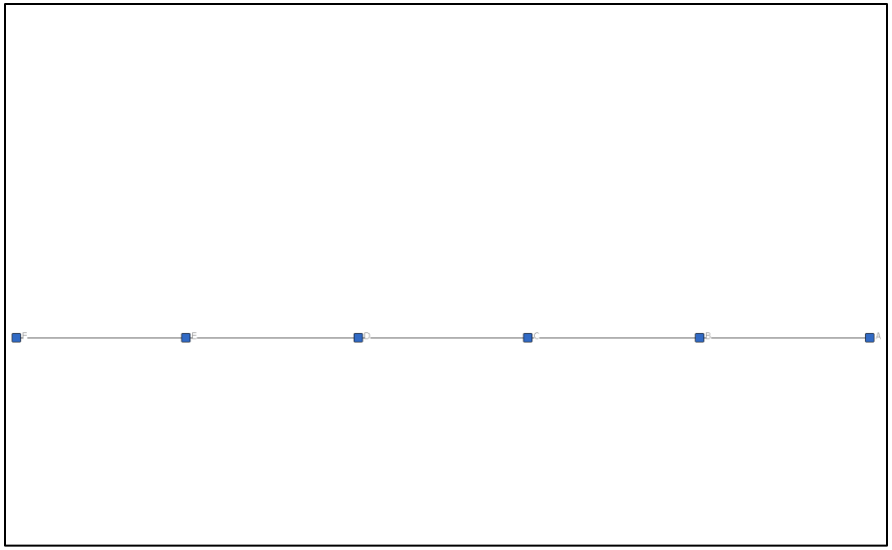
$$\frac{2|E|}{|V|(|V|-1)}$$

For directed graphs

$$\frac{|E|}{|V|(|V|-1)}$$

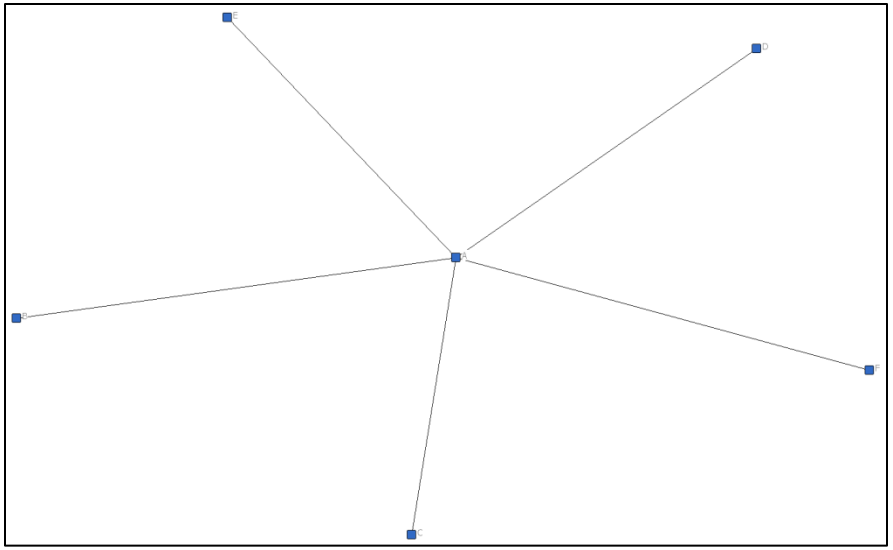
Here |V| is the number of nodes, and |E| is the number of edges in the graph

Density Example



Density = (2*5)/(6*(6-1)) = 0.33

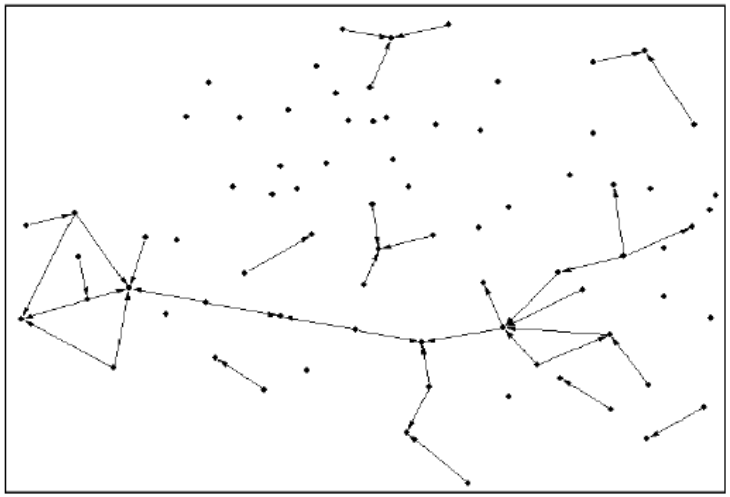
Degree centralization = 10%



Density = (2*5)/(6*(6-1)) = 0.33

Degree centralization = 60%

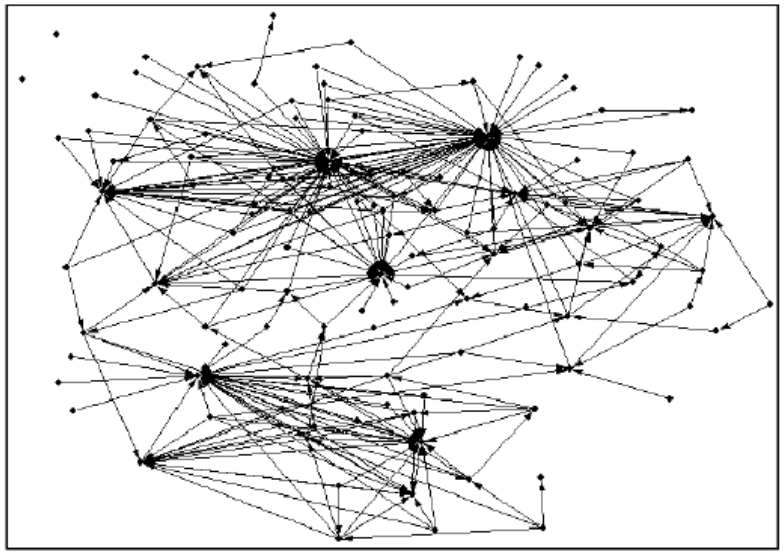
Help With the Rice Harvest



Village 1

Data from

Help With the Rice Harvest



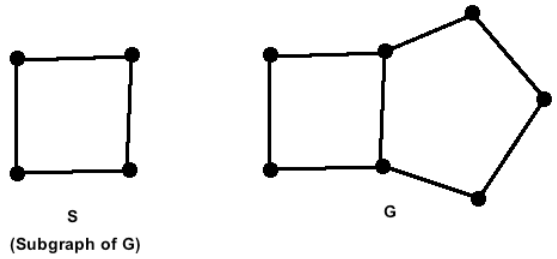
Village 2

Which village is more likely to survive?

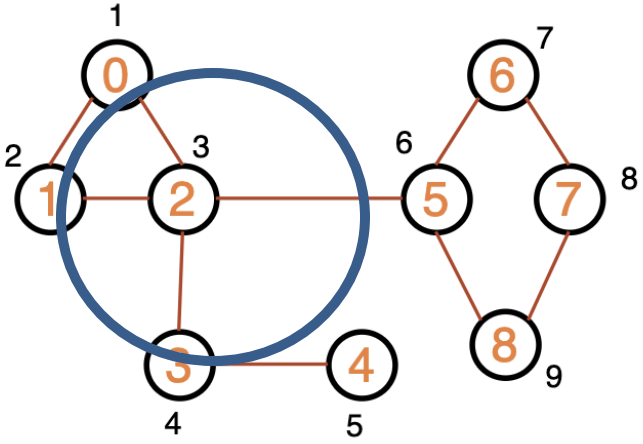
Data from Entwistle et al

Components and Subgraphs

A *subgraph* is a subset of the nodes of a network, and all of the edges linking these nodes. Any group of nodes can form a subgraph

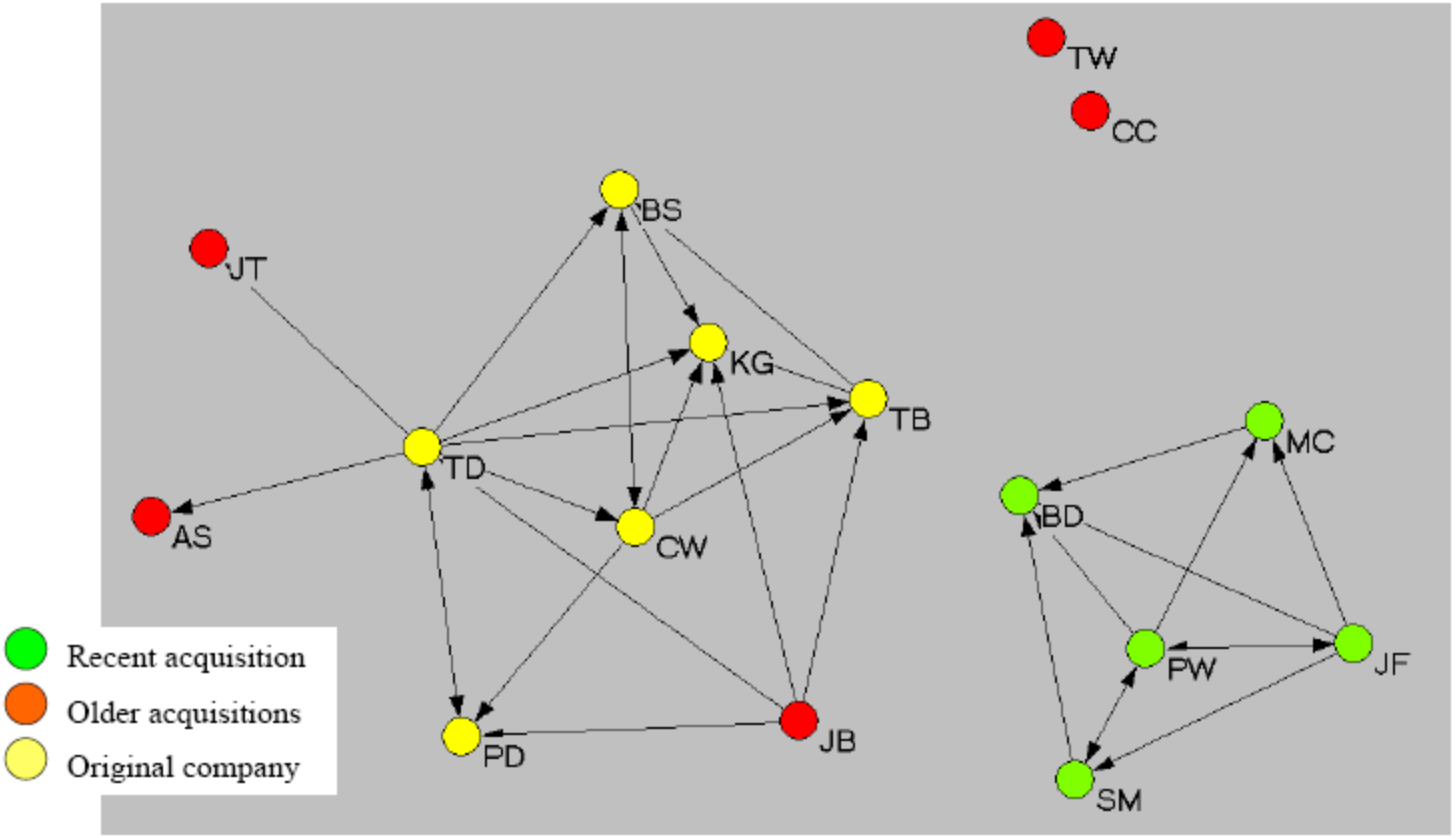


Component subgraphs (or simply *components*) are portions of the network that are disconnected from each other.



A network with 4 components

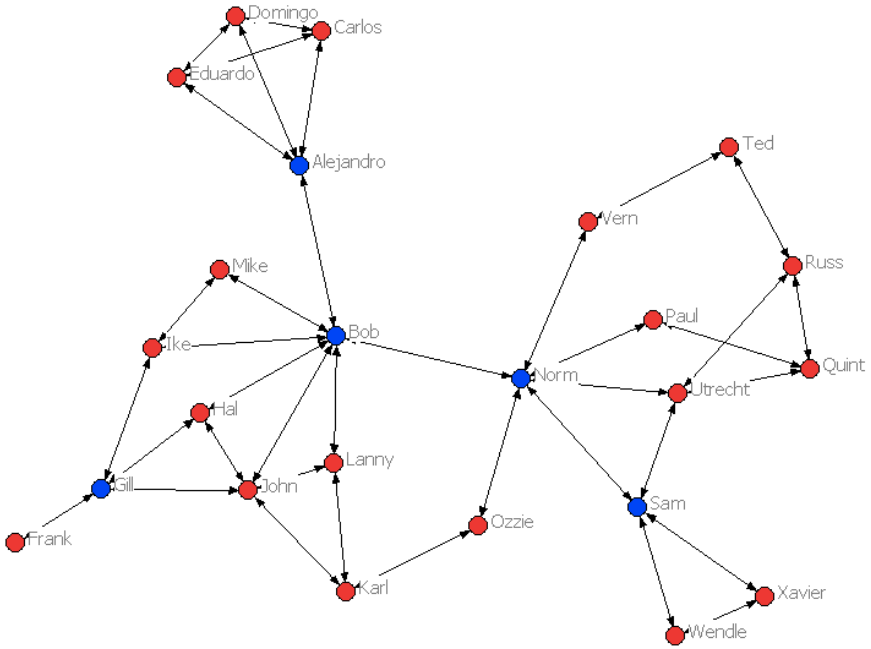
Who you go to so that you can say 'I ran it by ____, and she says ...'



Data drawn from Cross, Borgatti & Parker 2001.

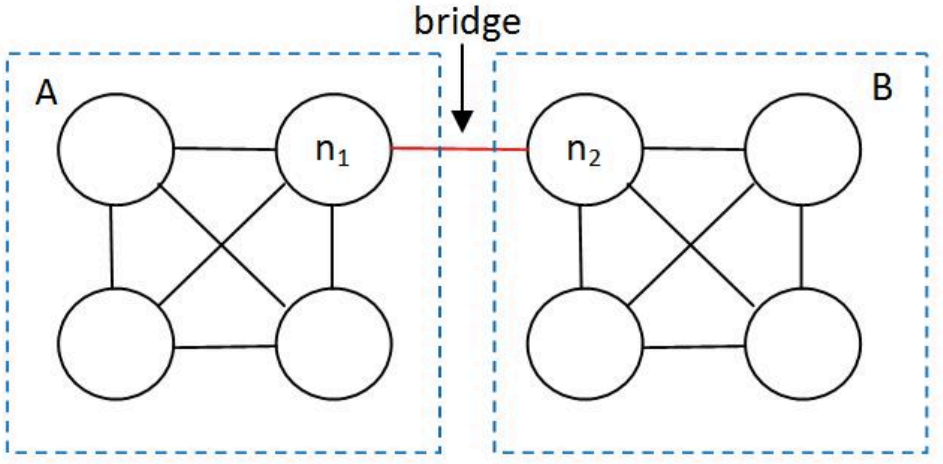
Cutpoint

- is a vertex (Node) whose removal increases the number of components of G*G*.
- Cutpoints are of particular interest when seeking to identify critical positions in flow networks,



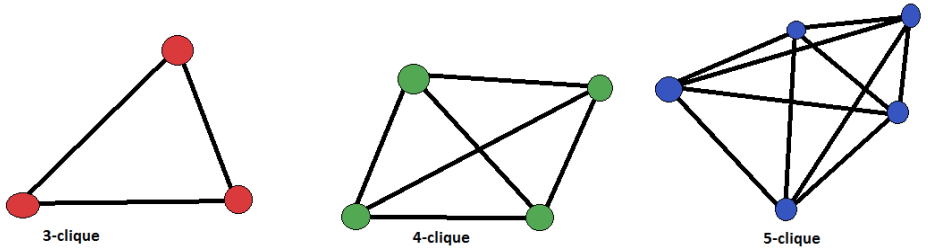
Bridge

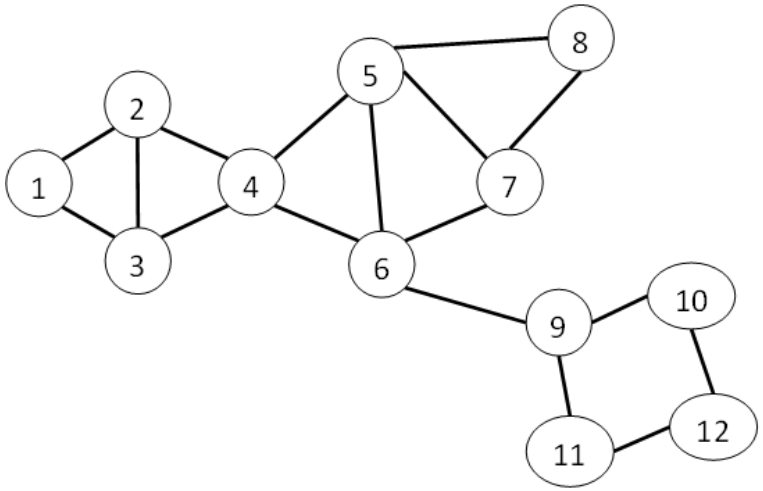
- a bridge is a direct tie between nodes that would otherwise be in disconnected components of the graph



Clique

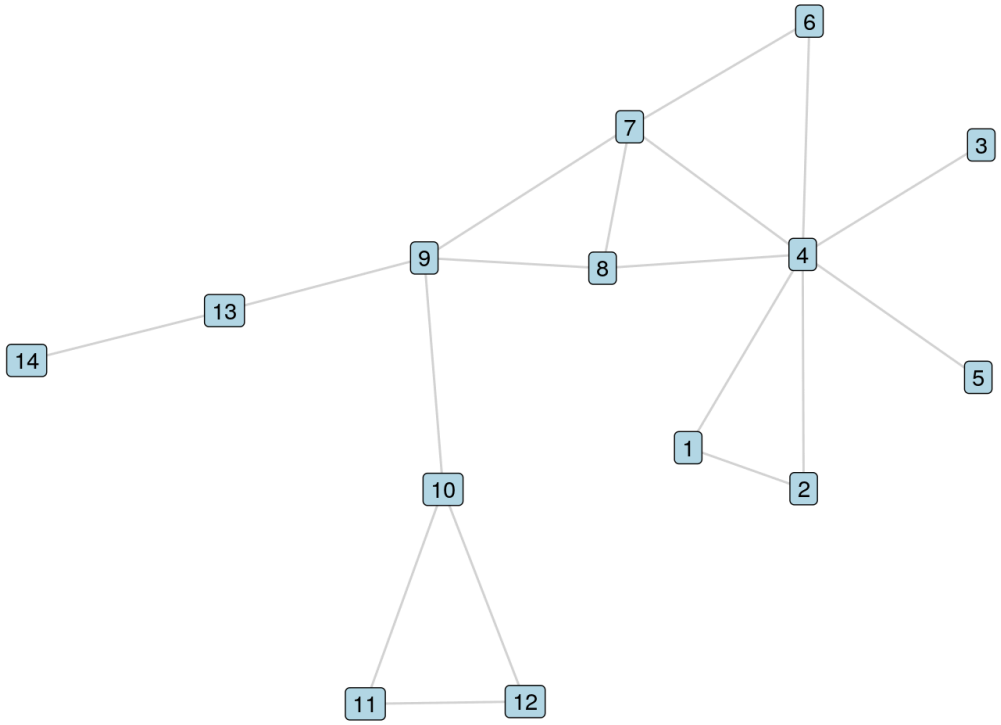
- the **largest subset** of the vertices such that for every pair of vertices in which **every vertex is connected to each other**
- Subset contains at **least 3 vertices**





Path Length and Distance

- **Path Length:** Number of Edges in the Path
- **Distance:** number of edges in a shortest path connecting them

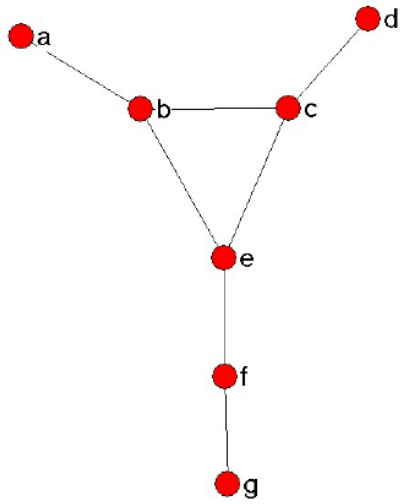


Geodesic Distance

- Geodesic distance between two vertices is the length in terms of the number of edges of the shortest path between the vertices.
- If the graph is weighted, it is a path with the minimum sum of edge weights.

Geodesic Distance Matrix

	a	b	c	d	e	f	g
a	0	1	2	3	2	3	4
b	1	0	1	2	1	2	3
c	2	1	0	1	1	2	3
d	3	2	1	0	2	3	4
e	2	1	1	2	0	1	2
f	3	2	2	3	1	0	1
g	4	3	3	4	2	1	0

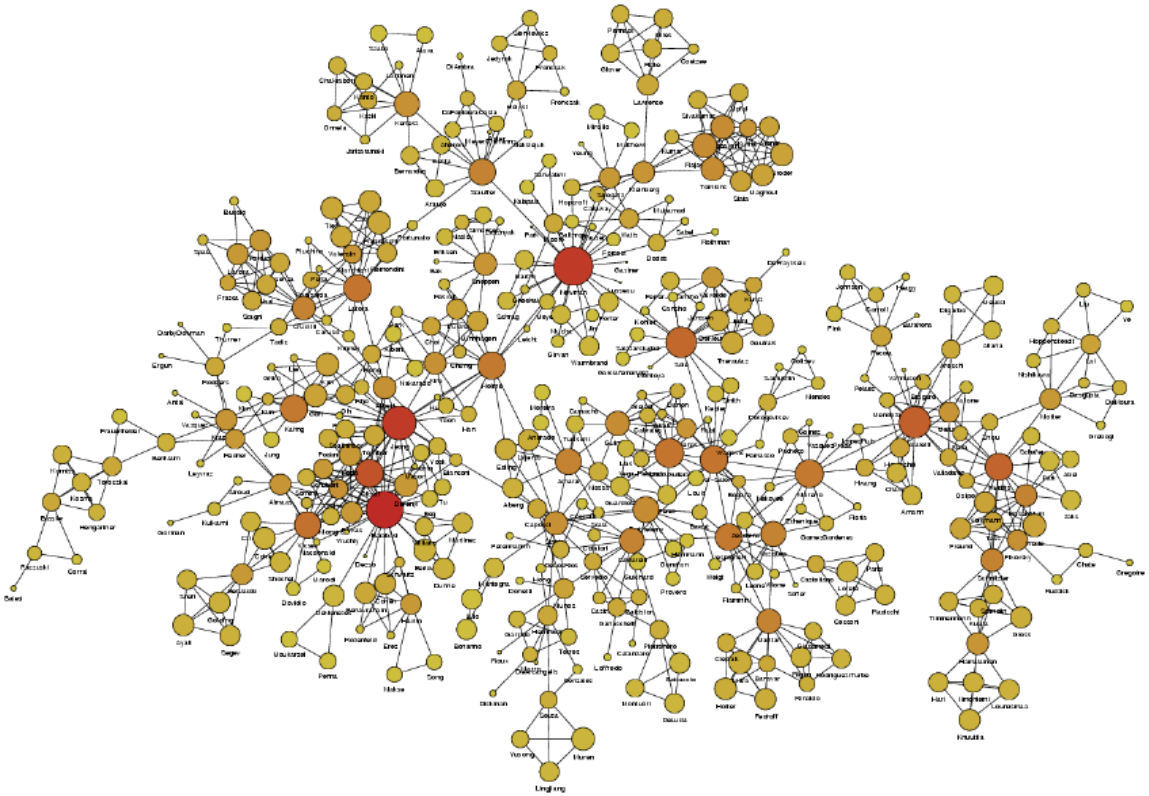


Centrality versus Centralization

- Centrality is a node level metric; each node has a centrality score representing it's position within the network (structurally important actors)
- Centralization is a group level metric indicating the extent to which the network is dominated by one or a few nodes
- How many direct, 'one hop' connections each node has to other nodes in the network.
- Can be used to find very connected individuals, popular individuals, individuals who are likely to hold most information or individuals who can quickly connect with the wider network.

Centrality

- Example: "community centrality" measure with a network of collaborations between scientists working on networks



http://www-personal.umich.edu/~mejn/centrality/

Centrality Measures

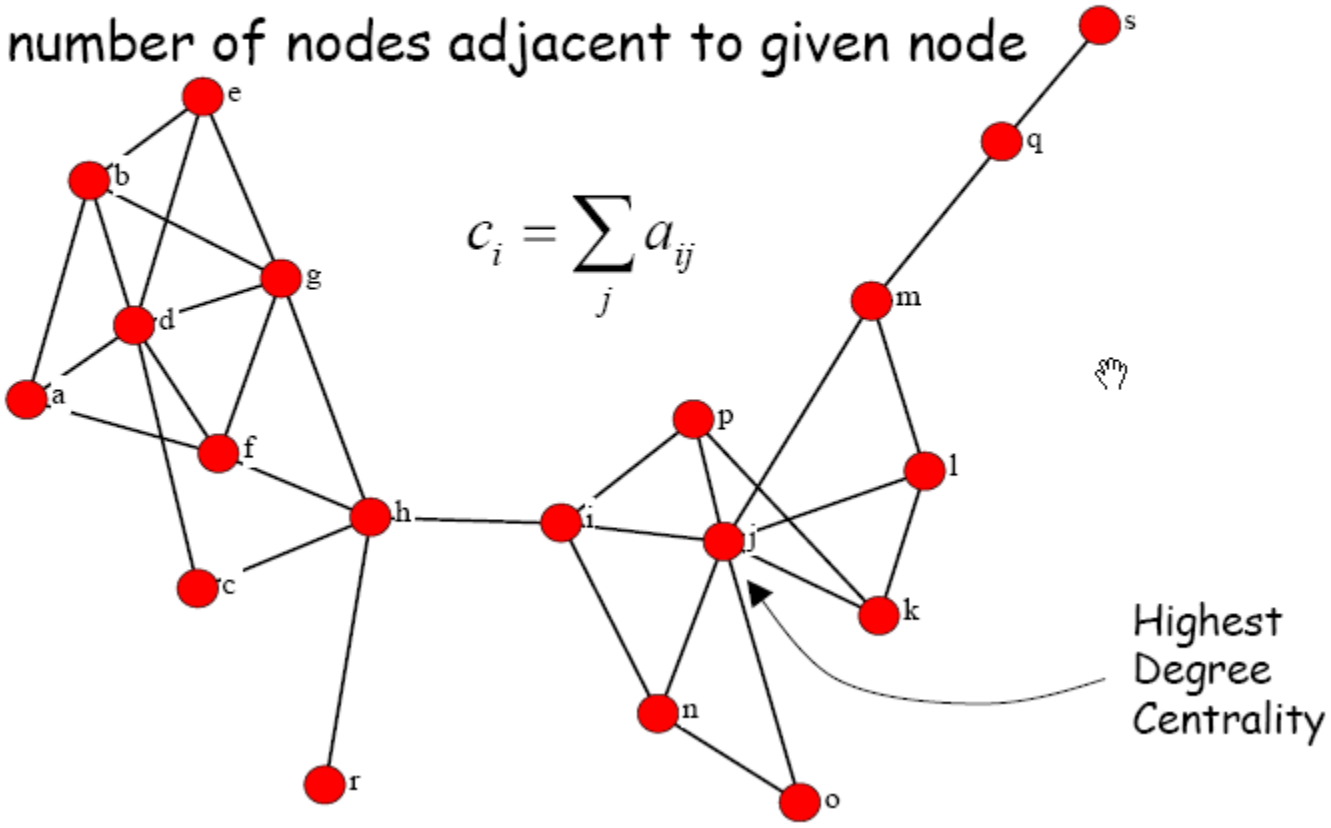
- Degree Centrality.
- Closeness Centrality.
- Betweeness Centrality.
- EigenVector Centrality.

Degree of Centrality

- Number of nodes adjacent to a given node
- How many direct, ‘one hop’ connections each node has to other nodes in the network.
- Can be used to find very connected individuals, popular individuals, individuals who are likely to hold most information or individuals who can quickly connect with the wider network.

Degree Centrality

- The number of nodes adjacent to given node



Closeness Centrality

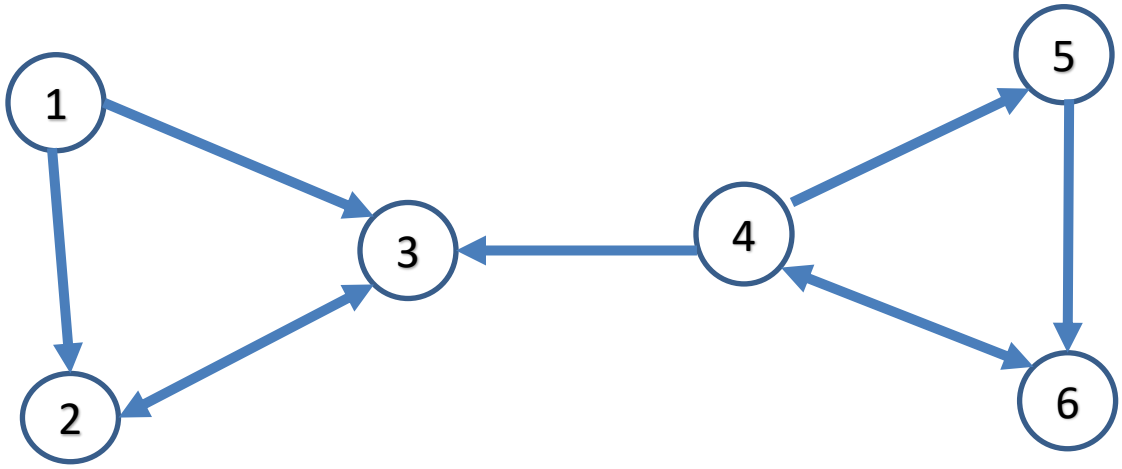
- A measure of the average shortest distance from each vertex to each other vertex
- It is the inverse of the average shortest distance between the vertex and all other vertices in the network.

$$C(x) = \frac{N-1}{\sum_y d(y,x)}.$$

N – Number of Nodes

d(y,x) – the distance (length of the shortest path

Closeness Centrality - Example



$$C(x) = \frac{N-1}{\sum_y d(y,x)}. \quad = \frac{(6-1)}{(1+1+2+3+3)} = 0.5$$

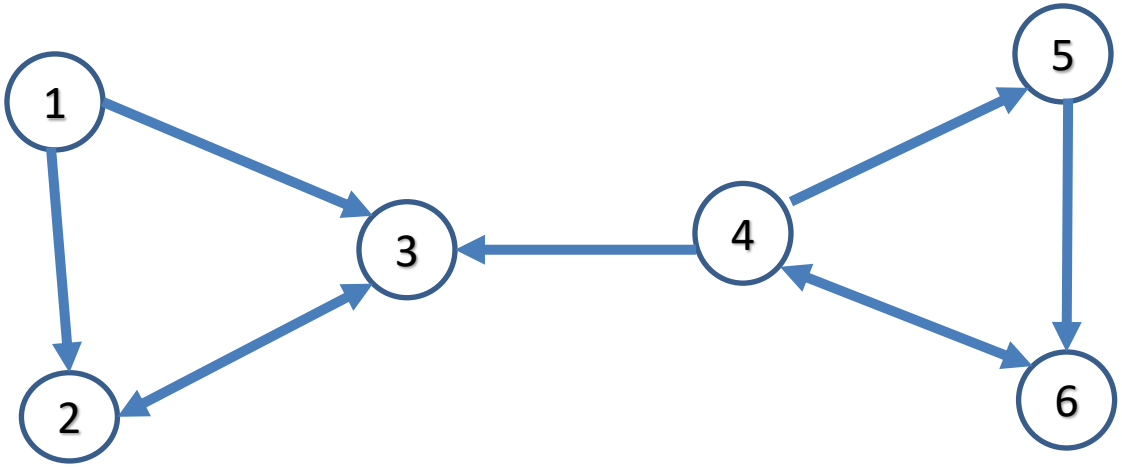
Betweenness Centrality

- measures the extent to which a vertex lies on paths between other vertices.
- Vertices with high betweenness may have considerable influence within a network by virtue of their **control over information passing** between others.
- They are also the ones whose removal from the network will most **disrupt communications** between other vertices because they lie on the largest number of paths taken by messages.

Betweenness Centrality

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

Where g_{jk} = the number of geodesics connecting *jk*, and
$g_{jk}(i)$= the number of geodesics that actor *i* is on.

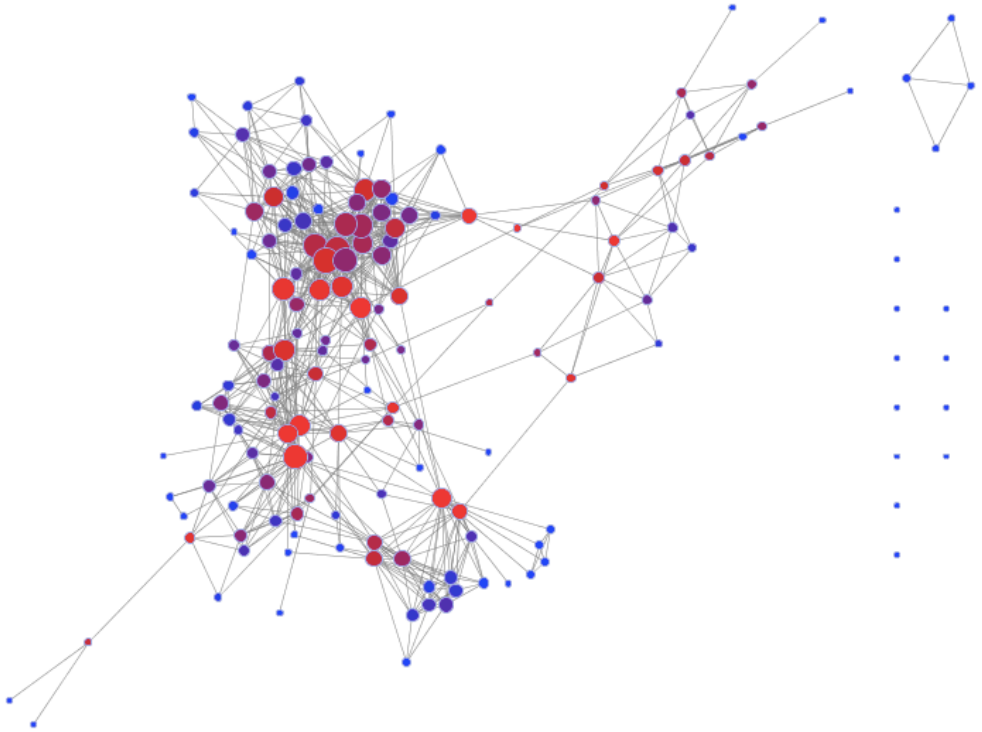


$$C_B(3) = \frac{g_{1,2}(3)}{g_{1,2}} + \frac{g_{1,4}(3)}{g_{1,4}} + \frac{g_{1,5}(3)}{g_{1,5}} + \frac{g_{1,6}(3)}{g_{1,6}} + \frac{g_{2,4}(3)}{g_{2,4}} + \frac{g_{2,5}(3)}{g_{2,5}} + \frac{g_{2,6,}(3)}{g_{2,6}} + \frac{g_{4,5}(3)}{g_{4,5}} + \frac{g_{4,6}(3)}{g_{4,6}} + \frac{g_{5,6}(3)}{g_{5,6}}$$

$$= \frac{0}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{1} = 6$$

Example Facebook network:

- Nodes are sized by degree, and colored by betweenness.
- Can you spot nodes with high betweenness but relatively low degree?
- Explain how this might arise.
- What about high degree but relatively low betweenness?

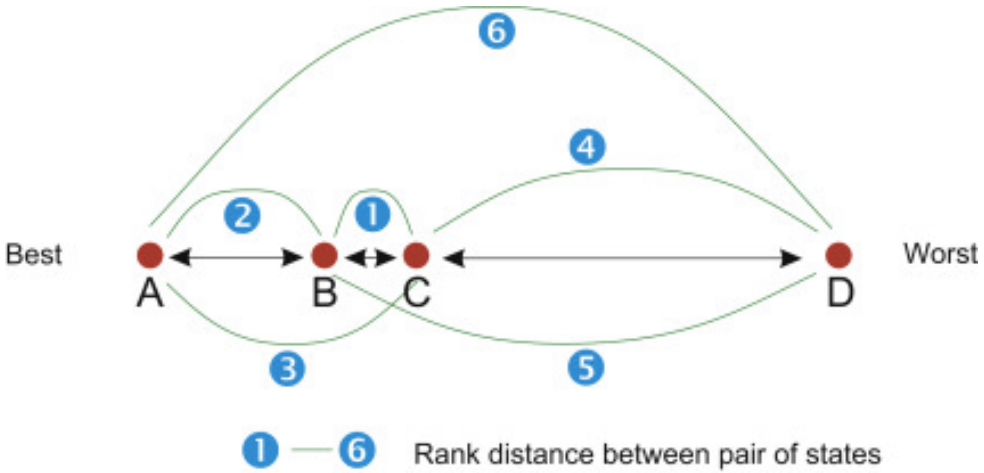


Centrality: to Check your Understanding

- generally different centrality metrics will be positively correlated
- when they are not, there is likely something interesting about the network
- suggest possible topologies and node positions to fit each square

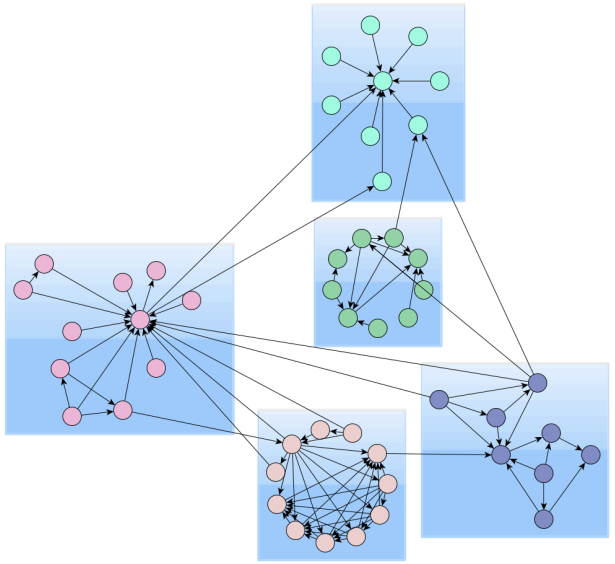
Multi-dimensional Scaling

- A **visual representation of distances or dissimilarities between sets of objects**.
- "Objects" can be colors, faces, map coordinates, political persuasion, or any kind of real or conceptual stimuli
- Objects that are more similar (or have shorter distances) are closer together on the graph than objects that are less similar (or have longer distances).
- Interpreting dissimilarities as distances on a graph, MDS can also serve as a dimension reduction technique for high-dimensional data



Clustering

- the task of assigning a set of objects to communities such that objects in the same community are more similar to each other than to those in other communities.



Summary

- Graph Basics
- Graph Representation
- Types of Graphs
- Connectivity in Graphs
- Graph Algorithms
- Settings for Large Network

Any Question?

CIS 600 - Principles of Social Media and Data Mining

Spring 2023

Week 5

Social Network Models

Priyantha Kumarawadu
Associate Teaching Professor
Department of Electrical Engineering and Computer Science
Syracuse University

Outline

- Properties of Real-world Networks
  - Power-law Distribution
  - High Clustering Coefficient
  - Small Average Path Length
- Network Models
  - Random graph Model
  - Small World Model
  - Preferential Attachment Model

Why should I use network models?

Facebook

May 2011:

- **1.06 Billion** users.

February 2023:

- **2.96 Billion** users
- **Average of 338** friends
- Only **about 28%** are true friends

1. What are the principal underlying processes that help initiate these friendships?

2. How can these seemingly independent friendships form this complex friendship network?

3. In social media there are many networks with millions of nodes and billions of edges.
   - **They are complex and it is difficult to analyze them**

So, what do we do?

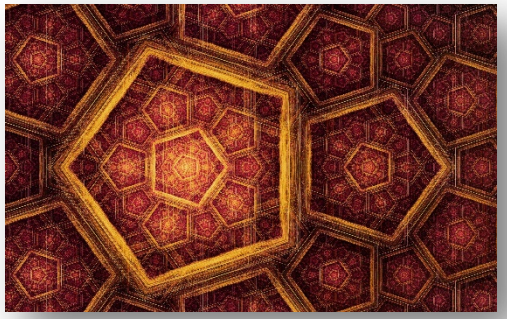
Design models that generate graphs

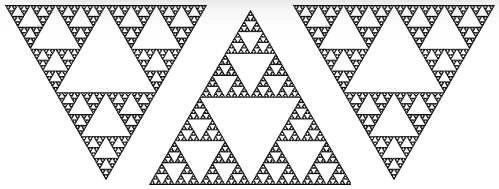
- The generated graphs should be similar to real-world networks.

If we can guarantee that generated graphs are similar to real-world networks:

1. We can analyze simulated graphs instead of real-networks (**cost-efficient**)
2. We can better understand real-world networks by providing concrete mathematical explanations; and
3. We can perform controlled experiments on synthetic networks when real-world networks are unavailable.

What are properties of real-world networks that should be accurately modeled?





Basic Intuition:

Hopefully! Our complex output [social network] is generated by a simple process

Properties of Real-World Networks

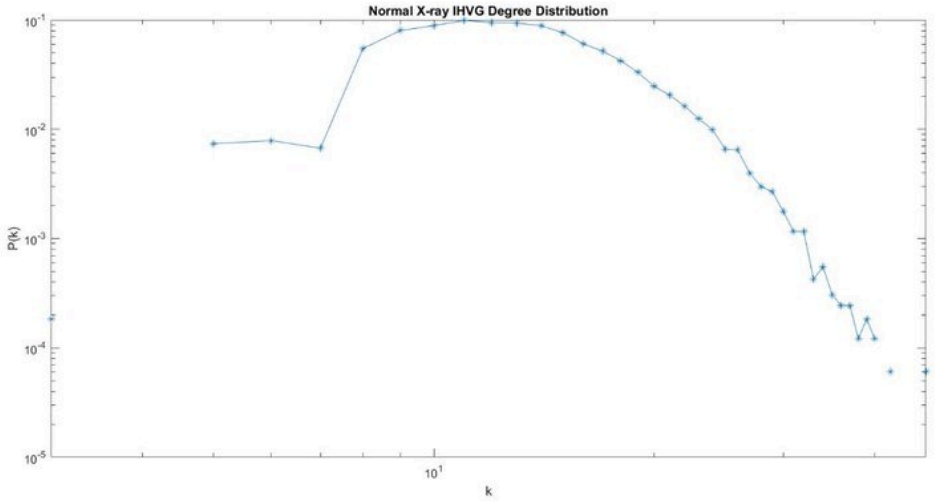
Power-law Distribution

High Clustering Coefficient

Small Average Path Length

Degree Distribution

In a network, the degree of a node in a network is the number of connections it has to other nodes and the degree distribution probability distribution of these degrees over the whole network.

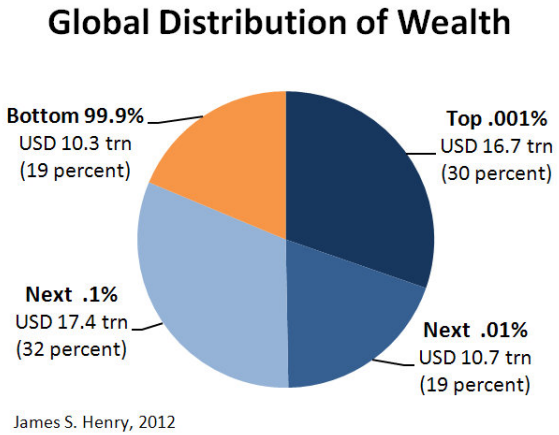


COVID-19 patients classification from chest X-ray using deep learning

Distributions

Wealth Distribution:

- Most individuals have average capitals,
- Few are considered wealthy.
- Exponentially more individuals with average capital than the wealthier ones.



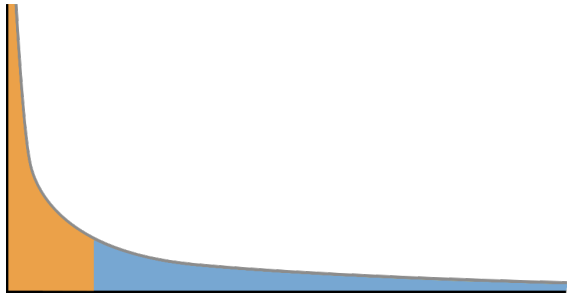
City Population:

- A few metropolitan areas are densely populated
- Most cities have an average population size.

The **Pareto principle** (80–20 rule): 80% of the effects come from 20% of the causes

Social Media:

- We observe the same phenomenon regularly when measuring popularity or interestingness for entities.



Distributions

Site Popularity:

- Many sites are visited less than a 1,000 times a month
- A few are visited more than a million times daily

User Activity:

- Social media users are often active on a few sites
- Some individuals are active on hundreds of sites

Product Price:

- There are exponentially more modestly priced products for sale compared to expensive ones.

Friendships:

- Many individuals with a few friends and a handful of users with thousands of friends

(Degree Distribution)

Power-Law Degree Distribution

- Relative change in one quantity results in proportional change in another quantity
- When the frequency of an event changes as a power of an attribute
  - The frequency follows a **power-law**

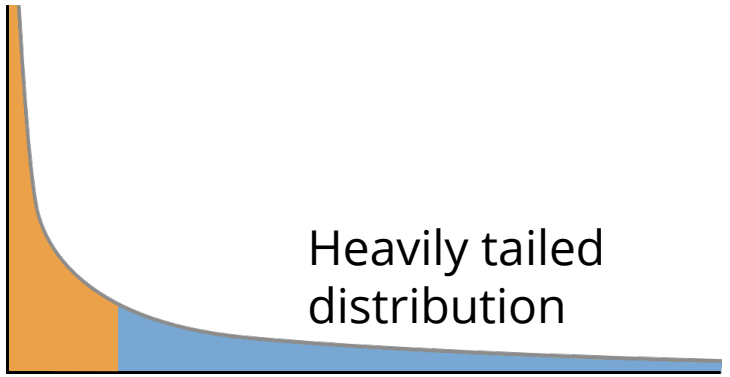
Power-law intercept

The power-law exponent and its value is typically in the range of **[2, 3]**

$$p_d = ad^{-b}$$

Fraction of users with degree d

Node degree



$$\ln p_d = -b \ln d + \ln a$$

Power-Law Distribution: Examples

- **Call networks:**
  - The fraction of telephone numbers that receive k calls per day is roughly proportional to $\mathbf{1/k^2}$
- **Book Purchasing:**
  - The fraction of books that are bought by k people is roughly proportional to $\mathbf{1/k^3}$
- **Scientific Papers:**
  - The fraction of scientific papers that receive k citations in total is roughly proportional to $\mathbf{1/k^3}$
- **Social Networks:**
  - The fraction of users that have in-degrees of k is roughly proportional to $\mathbf{1/k^2}$

Power-Law Distribution

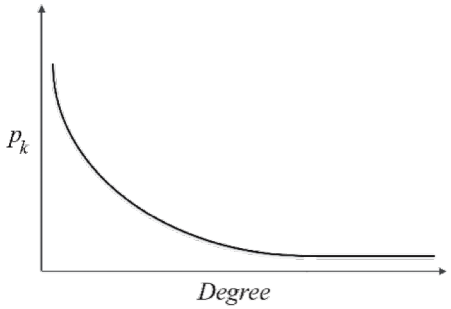
- Many real-world networks exhibit a *power-law* distribution.

  (income distribution,

- Power-laws seem to dominate
  - When the quantity being measured can be viewed as a type of popularity.

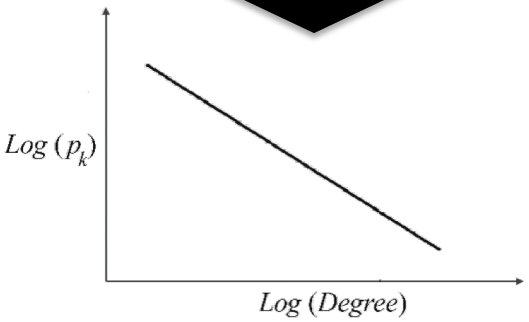
- A power-law distribution
  - **Small occurrences**: common
  - **Large instances**: extremely rare

A typical shape of a power-law distribution



(a) Power-Law Degree Distribution

Log-Log plot



(b) Log-Log Plot of Power-Law Degree Distribution

Power-law Distribution: An Elementary Test

To test whether a network exhibits a power-law distribution

1. Pick a popularity measure and compute it for the whole network
   - Example: number of friends for all nodes
2. Compute p_k, the fraction of individuals having popularity k.
3. Plot a log-log graph, where the x-axis represents $\ln k$ and the y-axis represents $\ln p_k$.
4. If a power-law distribution exists, we should observe a straight line

This is not a systematic approach!

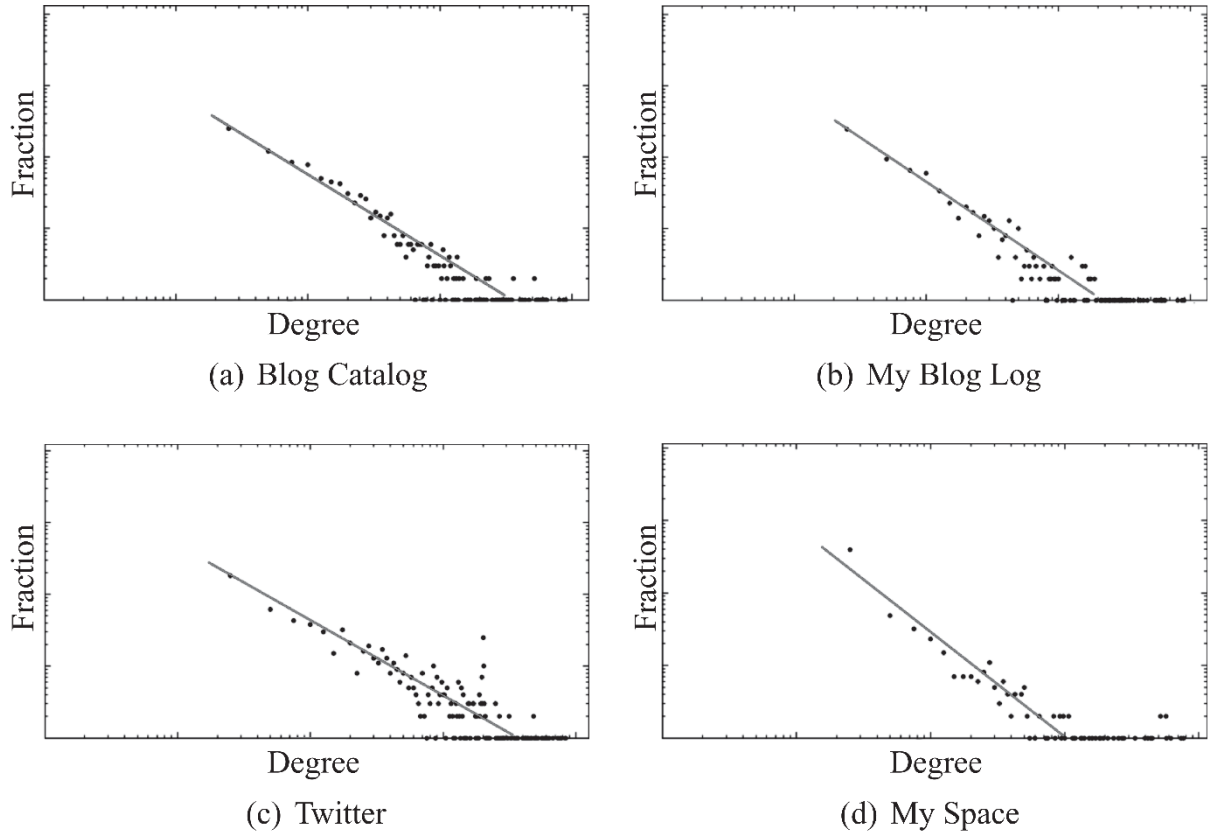
1. Other distributions could also exhibit this pattern
2. The results [estimations for parameters] can be biased and incorrect

For a systematic approach see:

Clauset, Aaron, Cosma Rohilla Shalizi, and Mark EJ Newman. "Power-law distributions in empirical data." *SIAM review 51(4)* (2009): 661-703.

Power-Law Distribution: Real-World Networks

Networks with a power-law degree distribution are called **Scale-Free** networks



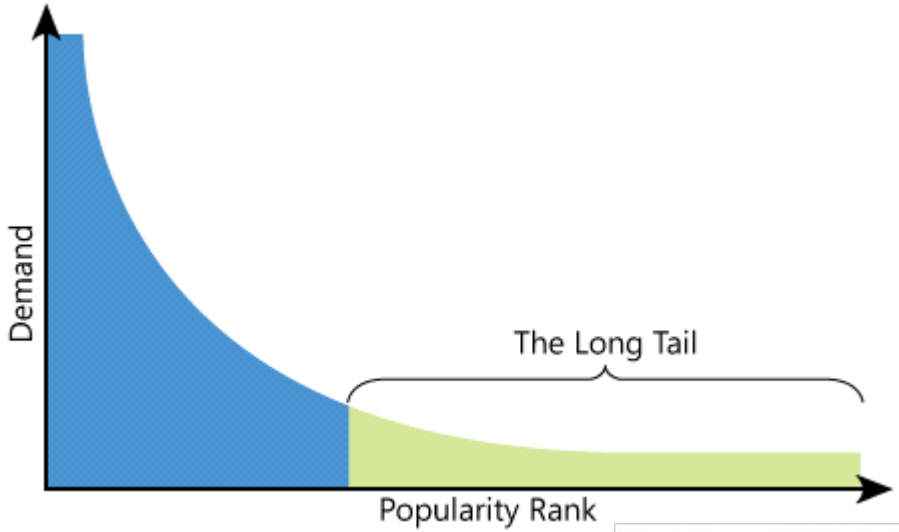
The tail of the power-law distribution is long!

The Looooooong Tail

Are most sales being generated by a small set of items that are enormously popular?

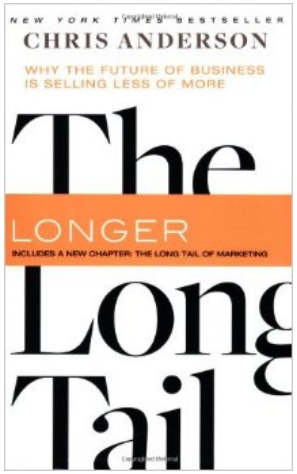
OR

By a much larger population of items that are each individually less popular?



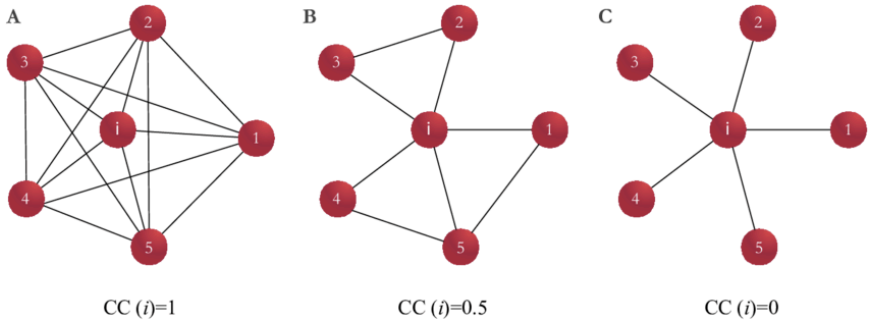
The total sales volume of unpopular items, taken together, is very significant.

- 57% of Amazon's sales is from the long tail



Clustering Coefficient

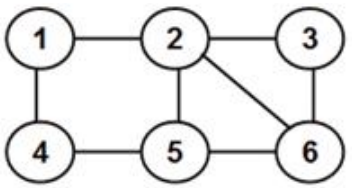
- A measure of the degree to which nodes in a graph tend to cluster together.
- Clustering Coefficient for a specific vertex *C(i)*
  - mean probability that a pair of i's friends are friend of another



Clustering Coefficient

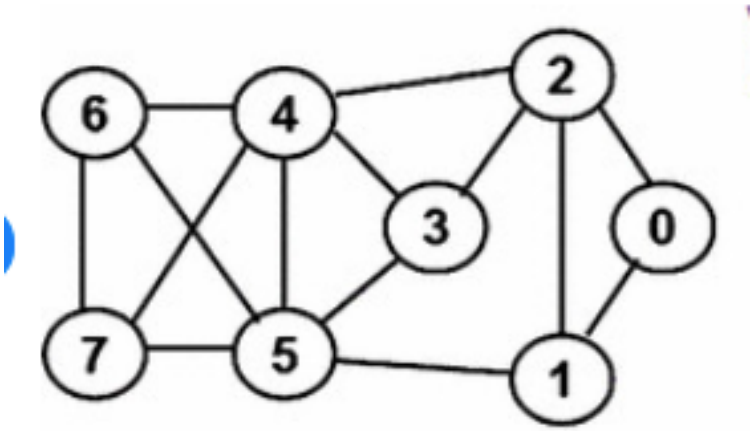
- A measure of the degree to which nodes in a graph tend to cluster together.
- Clustering Coefficient for a specific vertex *C(i)*
  - mean probability that a pair of i's friends are friend of another

$$C(i) = \frac{\#\text{ of pairs of neighbors of } i \text{ that are connected}}{\#\text{ of pairs of neighbors of } i}$$

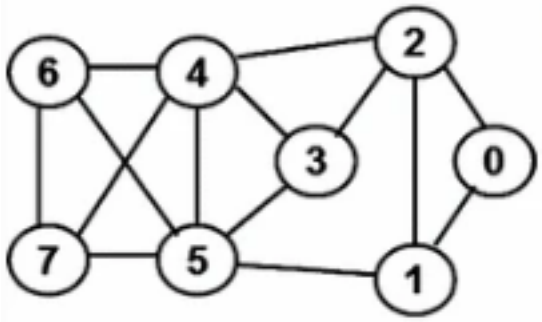


Vertex	Neighbors	# Links connecting the Neighbors	Max. possible # Links connecting the Neighbors	Local Clustering Coefficient
1	2, 4	0	2(1)/2 = 1	0/1 = 0.0
2	1, 3, 5, 6	2	4(3)/2 = 6	2/6 = 0.33
3	2, 6	1	2(1)/2 = 1	1/1 = 1.0
4	1, 5	0	2(1)/2 = 1	0/1 = 0.0
5	2, 4, 6	1	3(2)/2 = 3	1/3 = 0.33
6	2, 3, 5	2	3(2)/2 = 3	2/3 = 0.67

Problem: Calculate clustering coefficient of each node



Answer



Vertex ID, v_i	Degree (k_i) Centrality	Actual # Links among Neighbors	Max. Possible # Links among Neighbors	Local Clustering Coefficient, LCC
0	2	1	2(2-1)/2 = 1	1/1 = 1.0
1	3	1	3(3-1)/2 = 3	1/3 = 0.33
2	4	2	4(4-1)/2 = 6	2/6 = 0.33
3	3	2	3(3-1)/2 = 3	2/3 = 0.67
4	5	5	5(5-1)/2 = 10	5/10 = 0.5
5	5	4	5(5-1)/2 = 10	4/10 = 0.4
6	3	3	3(3-1)/2 = 3	3/3 = 1.0
7	3	3	3(3-1)/2 = 3	3/3 = 1.0

Clustering Coefficient

- In real-world networks, friendships are highly transitive

Facebook

- **May 2011:**
  - Average clustering coefficient of **0.5** for users with **two** friends

  - Friends of a user are often friends with one another
  - These friendships form triads
  - High average [local] clustering coefficient

Web	Facebook	Flickr	LiveJournal	Orkut	YouTube
0.081	0.14 (with 100 friends)	0.31	0.33	0.17	0.13

Clustering Coefficient for Real-World Networks

	Network	Type	n	m	C
Social	Film actors	Undirected	449 913	25 516 482	0.20
	Company directors	Undirected	7 673	55 392	0.59
	Math coauthorship	Undirected	253 339	496 489	0.15
	Physics coauthorship	Undirected	52 909	245 300	0.45
	Biology coauthorship	Undirected	1 520 251	11 803 064	0.088
	Telephone call graph	Undirected	47 000 000	80 000 000	
	Email messages	Directed	59 812	86 300	
	Email address books	Directed	16 881	57 029	0.17
	Student dating	Undirected	573	477	0.005
	Sexual contacts	Undirected	2 810		
Information	WWW nd.edu	Directed	269 504	1 497 135	0.11
	WWW AltaVista	Directed	203 549 046	1 466 000 000	
	Citation network	Directed	783 339	6 716 198	
	Roget's Thesaurus	Directed	1 022	5 103	0.13
	Word co-occurrence	Undirected	460 902	16 100 000	
Technological	Internet	Undirected	10 697	31 992	0.035
	Power grid	Undirected	4 941	6 594	0.10
	Train routes	Undirected	587	19 603	
	Software packages	Directed	1 439	1 723	0.070
	Software classes	Directed	1 376	2 213	0.033
	Electronic circuits	Undirected	24 097	53 248	0.010
	Peer-to-peer network	Undirected	880	1 296	0.012
Biological	Metabolic network	Undirected	765	3 686	0.090
	Protein interactions	Undirected	2 115	2 240	0.072
	Marine food web	Directed	134	598	0.16
	Freshwater food web	Directed	92	997	0.20
	Neural network	Directed	307	2 359	0.18

Source: M. E. J Newman

Average Path Length

How Small is the World?

A rumor is spreading over a social network.

- Assume all users pass it immediately to all of their friends

1. How long does it take to reach almost all of the nodes in the network?
2. What is the maximum time?
3. What is the average time?

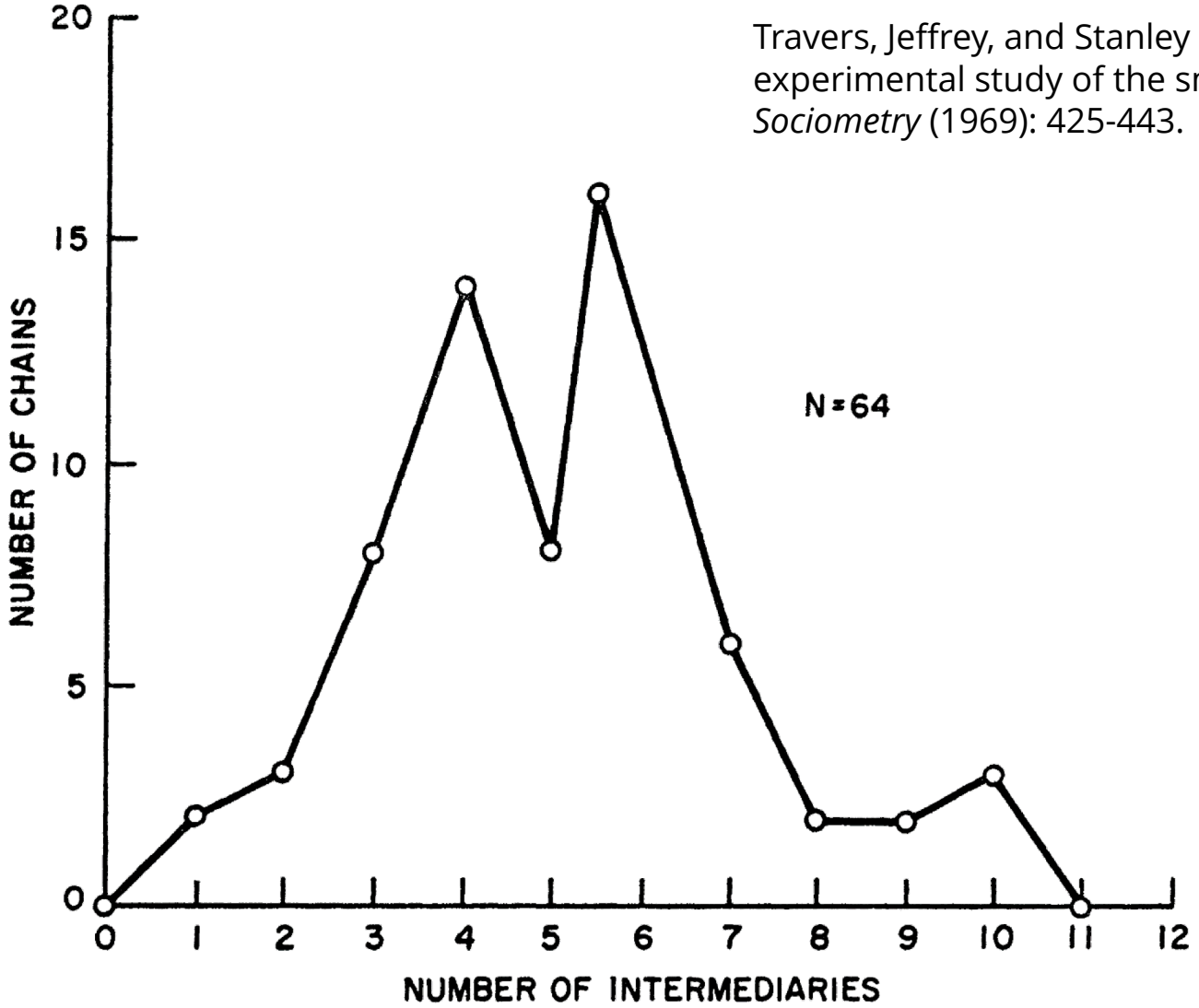
Milgram's Experiment

- 296 random people from Nebraska (196 people) and Boston (100 people) were asked to send a letter (via intermediaries) to a stock broker in Boston
- S/he could only send to people they personally knew, i.e., were on a first-name basis

Stanley Milgram (1933-1984)

Among the letters that found the target (64), the average number of links was around **six**.

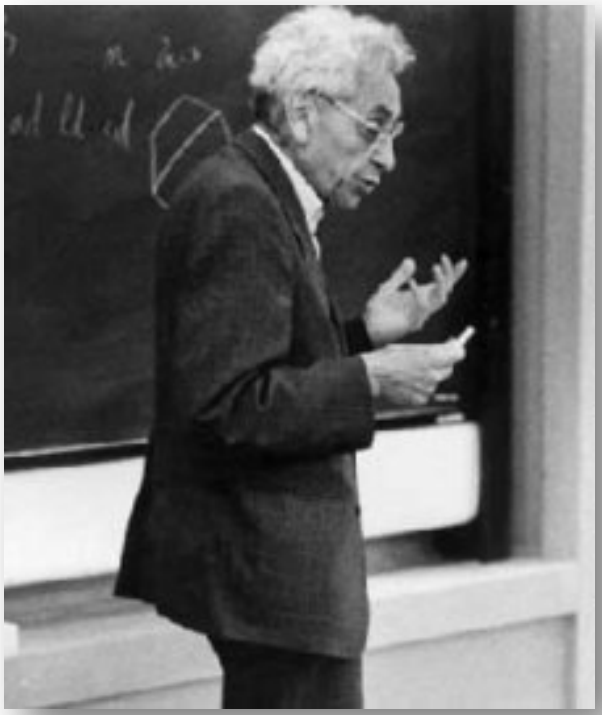
Milgram's Experiment



Travers, Jeffrey, and Stanley Milgram. "An experimental study of the small world problem." *Sociometry* (1969): 425-443.

Average Number of Intermediate people is 5.2

Erdös Number

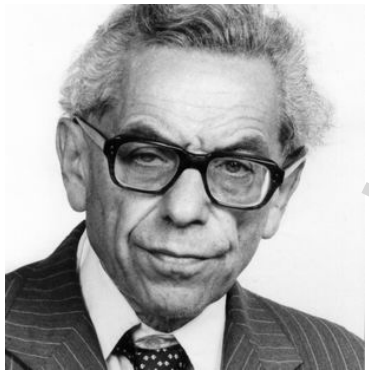


Paul Erdös (1913-1996)

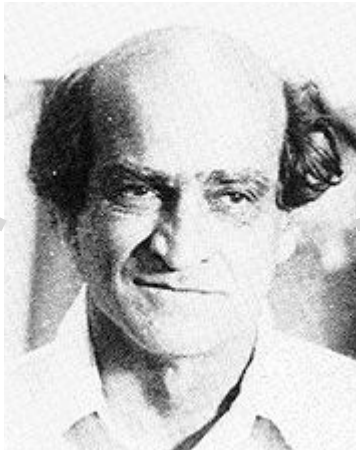
- **Erdös Number:** Number of links required to connect scholars to Erdös, via co-authorship papers
- Erdös wrote 1500+ papers with 507 co-authors.
- The Erdös Number Project allows you to compute your Erdös number:
  - http://www.oakland.edu/enp/
- Connecting path lengths, among mathematicians only:
  - Avg. is **4.65** and Maximum is **13**

Watch Erdös's documentary "*N is a number*" on YouTube

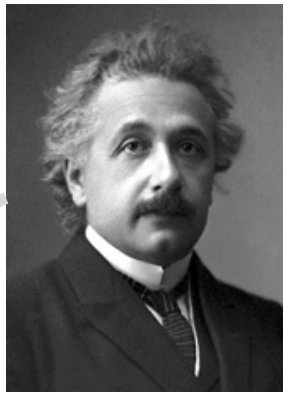
An Example of Erdös number 2 [Einstein]



Paul Erdös (1913-1996)



Ernst Gabor Straus (1922-1983)

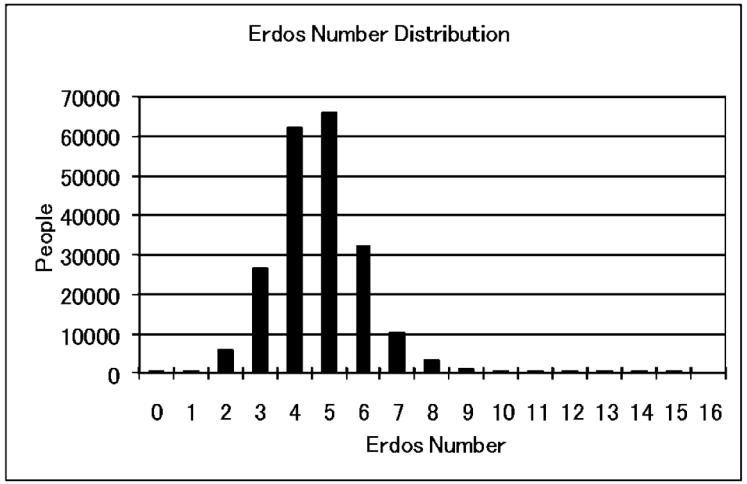


Albert Einstein (1879-1955)

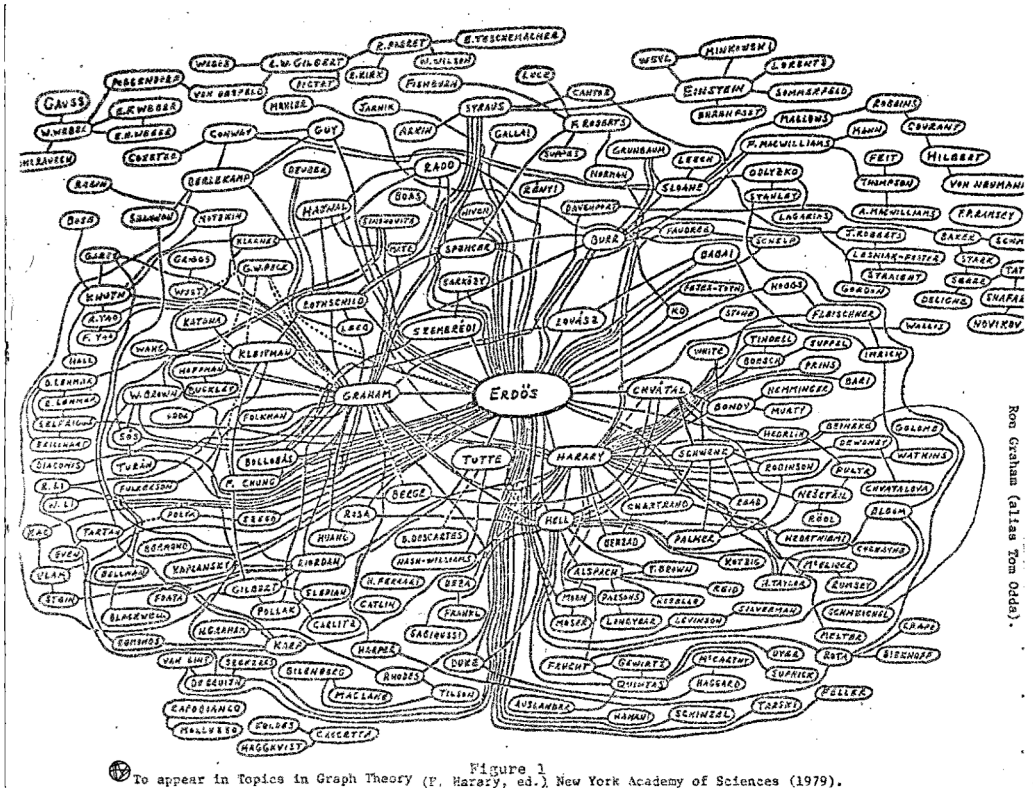
Erdös, Paul, B. Rothschild, and E. G. Straus. "Polychromatic Euclidean Ramsey theorems." *Journal of Geometry* 20.1 (1983): 28-35.

Einstein, Albert, and Ernst Gabor Straus. "A generalization of the relativistic theory of gravitation, II." *Annals of Mathematics* (1946): 731-741.

Erdös number Distribution



- The median Erdös number is **5**
- The mean is **4.65**
- The standard deviation is **1.27**



Erdös Number Project:

http://www.oakland.edu/enp/index.html

The Average Shortest Path

In real-world networks, any two members of the network are usually connected via a short paths.

Facebook

May 2011:

- Average path length was **4.7**
- **4.3** for US users

[Four degrees of separation]

The average path length is small

Web	Facebook	Flickr	LiveJournal	Orkut	YouTube
16.12	4.7	5.67	5.88	4.25	5.10

The Average Shortest Path in Sample Networks

	Network	Type	n	m	ℓ
Social	Film actors	Undirected	449 913	25 516 482	3.48
	Company directors	Undirected	7 673	55 392	4.60
	Math coauthorship	Undirected	253 339	496 489	7.57
	Physics coauthorship	Undirected	52 909	245 300	6.19
	Biology coauthorship	Undirected	1 520 251	11 803 064	4.92
	Telephone call graph	Undirected	47 000 000	80 000 000	
	Email messages	Directed	59 812	86 300	4.95
	Email address books	Directed	16 881	57 029	5.22
	Student dating	Undirected	573	477	16.01
	Sexual contacts	Undirected	2 810		
Information	WWW nd.edu	Directed	269 504	1 497 135	11.27
	WWW AltaVista	Directed	203 549 046	1 466 000 000	16.18
	Citation network	Directed	783 339	6 716 198	
	Roget's Thesaurus	Directed	1 022	5 103	4.87
	Word co-occurrence	Undirected	460 902	16 100 000	
Technological	Internet	Undirected	10 697	31 992	3.31
	Power grid	Undirected	4 941	6 594	18.99
	Train routes	Undirected	587	19 603	2.16
	Software packages	Directed	1 439	1 723	2.42
	Software classes	Directed	1 376	2 213	5.40
	Electronic circuits	Undirected	24 097	53 248	11.05
	Peer-to-peer network	Undirected	880	1 296	4.28
Biological	Metabolic network	Undirected	765	3 686	2.56
	Protein interactions	Undirected	2 115	2 240	6.80
	Marine food web	Directed	134	598	2.05
	Freshwater food web	Directed	92	997	1.90
	Neural network	Directed	307	2 359	3.97

l: average path length

Source: M. E. J Newman

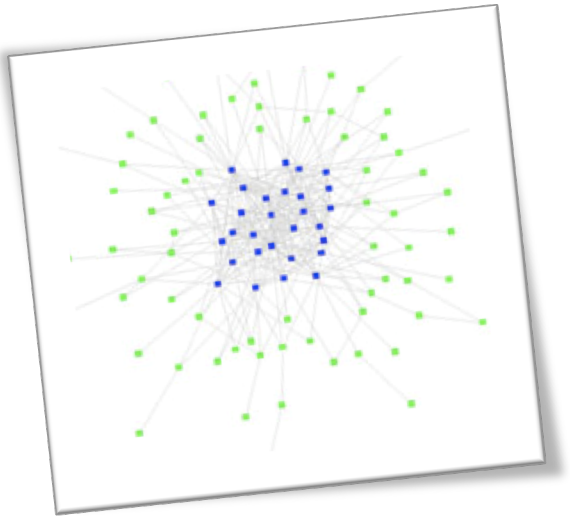
More Properties of Real-World Networks

Friendship Paradox [Feld 1991]

- i.e., you friends, on average, have more friends than you
- **Why?**
  - High degree nodes appear in many averages when averaging over friends
- It holds for 98% of Twitter Users [Hodas et al. 2013]

Core-Periphery Structure

- Dense Core
- Periphery nodes that connects to the core, but not connected among themselves
- Also known as
  - **Jellyfish** or **Octopus** structures



Network Models

- Model-Driven Models!

Random graphs

Small-World Model

Preferential Attachment

Random Graphs

Random Graphs

- We have to assume how friendships are formed
  - The most basic form:

Random Graph assumption:
Edges (i.e., friendships) between nodes (i.e., individuals) are formed randomly.

We discuss two random graph models $G(n,p)$ and $G(n,m)$

1.G(n,m) model, in which n nodes are randomly connected by m edges.
2.G(n,p) model, in which we have a graph of n nodes, and each pair of nodes has probability p of being connected.

Random Graph Model - $G(n, p)$

- Consider a graph with a fixed number of nodes n
- Any of the $\binom{n}{2}$ edges can be formed independently, with probability p
- This model is very simple, every possible edge is created with the same constant probability.
- The graph is called a $G(n, p)$ *random graph*

Proposed independently by Edgar Gilbert and by Solomonoff and Rapoport.

Modeling Random Graphs, Cont.

Similarities:

- In the limit (when n is large), both $G(n,p)$ and $G(n,m)$ models act similarly
  - The expected number of edges in $G(n,p)$ is $\binom{n}{2}p$
  - We can set $\binom{n}{2}p = m$ and in the limit, we should get similar results

Differences:

- The $G(n,m)$ model contains a fixed number of edges
- The $G(n,p)$ model is likely to contain none or all possible edges

Expected Degree

The expected number of edges connected to a node (expected degree) in $G(n, p)$ is $c = (n - 1)p$

Proof.

- A node can be connected to at most $n - 1$ nodes
  - or $n - 1$ edges
- All edges are selected independently with probability p
- Therefore, on average, $(n - 1)p$ edges are selected

- $c = (n - 1)p$ or equivalently,

$$p = \frac{c}{n-1}$$

Expected Number of Edges

The expected number of edges in $G(n, p)$ is $\binom{n}{2}p$

Proof.

– Since edges are selected independently, and we have a maximum $\binom{n}{2}$ edges, the expected number of edges is $\binom{n}{2}p$

The probability of observing m edges

Given the $G(n,p)$ model, the probability of observing m edges is the binomial distribution

$$P(|E| = m) = \binom{\binom{n}{2}}{m} p^m (1-p)^{\binom{n}{2}-m}$$

Proof.

- m edges are selected from the $\binom{n}{2}$ possible edges.
- These m edges are formed with probability p^m and other edges are not formed (to guarantee the existence of only m edges) with probability

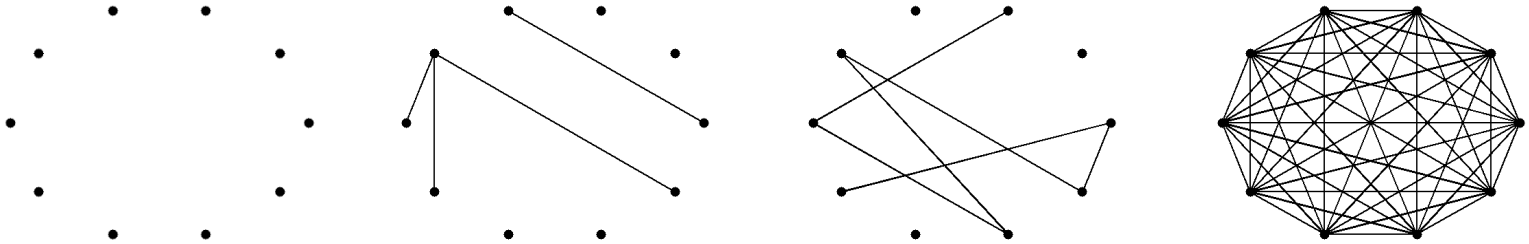
$$(1-p)^{\binom{n}{2}-m}$$

Evolution of Random Graphs

The Giant Component

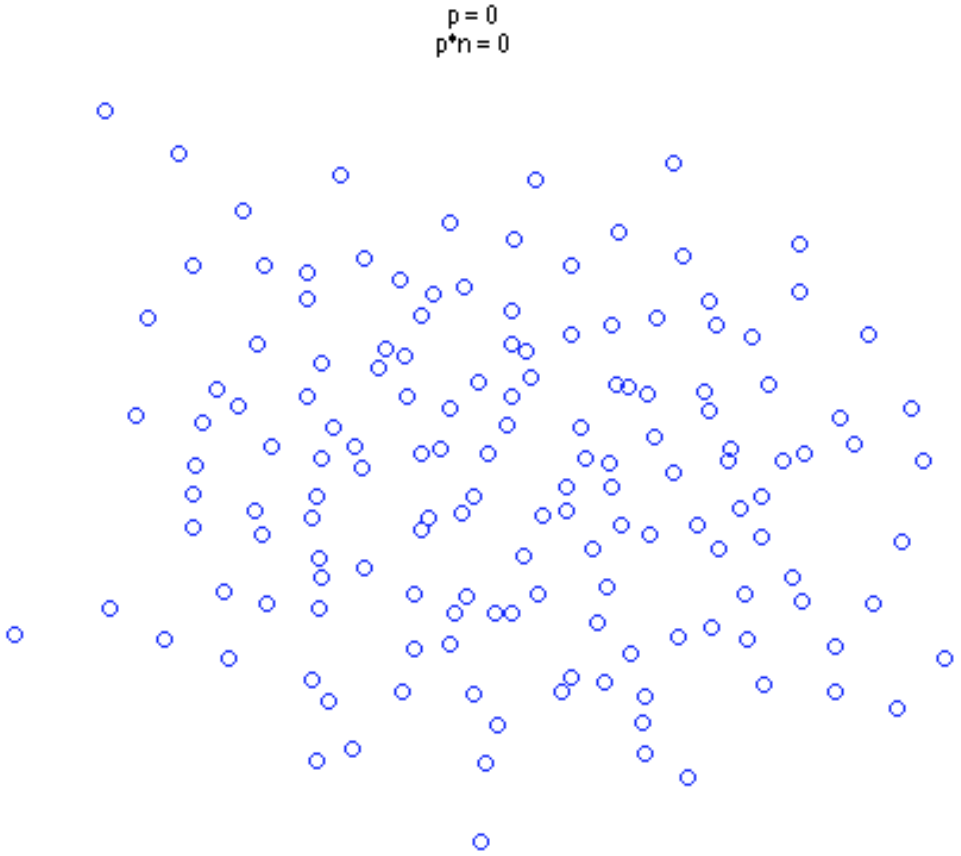
- In random graphs, as we increase p, a large fraction of nodes start getting connected
  - i.e., we have a path between any pair
- This large fraction forms a connected component:
  - **Largest connected component**, also known as the **Giant component**
- In random graphs:
  - $p = 0$
    - the size of the giant component is 0
  - $p = 1$
    - the size of the giant component is n

The Giant Component



Probability (p)	0.0	0.055	0.11	1.0
Average Node Degree (c)	0.0	0.8	≈1	n-1=9
Diameter	0	2	6	1
Giant Component Size	0	4	7	10
Average Path Length	0.0	1.5	2.66	1.0

Demo ($n = 150$)



From *David Gleich*

1st Phase Transition (Rise of the Giant Component)

- **Phase Transition**: the point where diameter value starts to shrink in a random graph
  - We have other phase transitions in random graphs
    - E.g., when the graph becomes connected

- The phase transition we focus on happens when
  - average node degree $c = 1$ (or when $p = 1/(n-1)$)

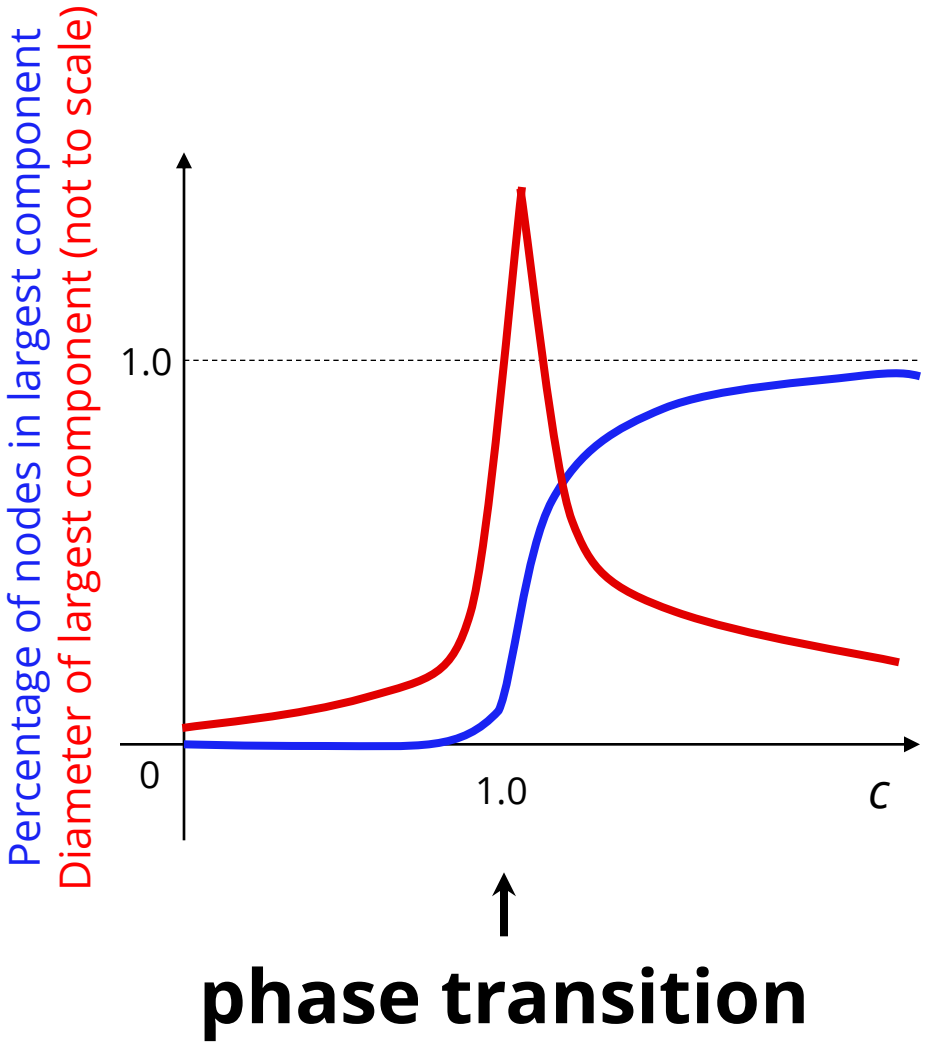
- At this Phase Transition:
  1. The giant component, which just started to appear, starts to grow, and
  2. The diameter, which *just* reached its maximum value, starts decreasing.

Random Graphs

If $c < 1$:
- **small**, isolated clusters
- **small** diameters
- **short** path lengths

At $c = 1$:
- a ***giant component*** appears
- diameter **peaks**
- path lengths are **long**

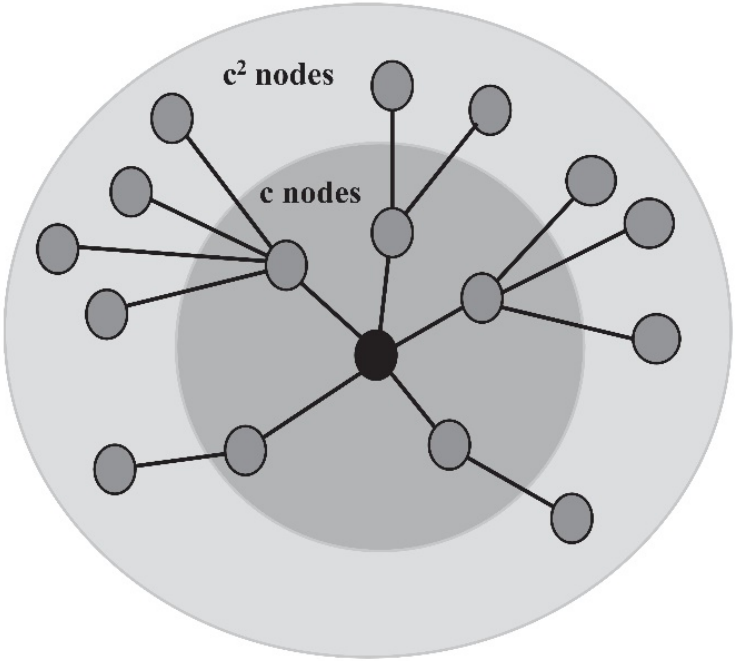
For $c > 1$:
- almost **all** nodes **connected**
- diameter **shrinks**
- path lengths **shorten**



Why $c = 1$? [Rough Idea]

Consider a random graph with expected node degree c

- In this graph,
  - Consider any **connected** set of nodes S;
  - Let $S' = V - S$ denote the complement set; and
  - Assume $|S| << |S'|$.
- For any node v in S,
  - If we move one hop away from v, we visit approximately c nodes.
- If we move one hop away from nodes in S,
  - we visit approximately $|S|c$ nodes.



- If S is small, the nodes in S only visit nodes in S' and when moving one hop away from S, the set of nodes *guaranteed to be connected* gets larger by a factor c.
- In the limit, if we want this connected component to become the largest component, then after traveling n hops, its size must grow and we must have

$$c^n \geq 1 \text{ or equivalently } c \geq 1$$

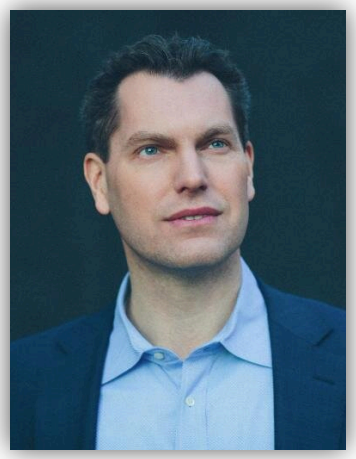
Try out some examples

- https://networkx.org/documentation/networkx-1.10/examples/index.html

Small-World Model

Small-world Model (the Watts-Strogatz (WS) Model)

- A special type of random graph
- a type of graph in which most nodes are not neighbors of one another, but most nodes can be reached from every other by a small number of hops.
- Exhibits small-world properties:
  - Short average path length
  - High clustering coefficient

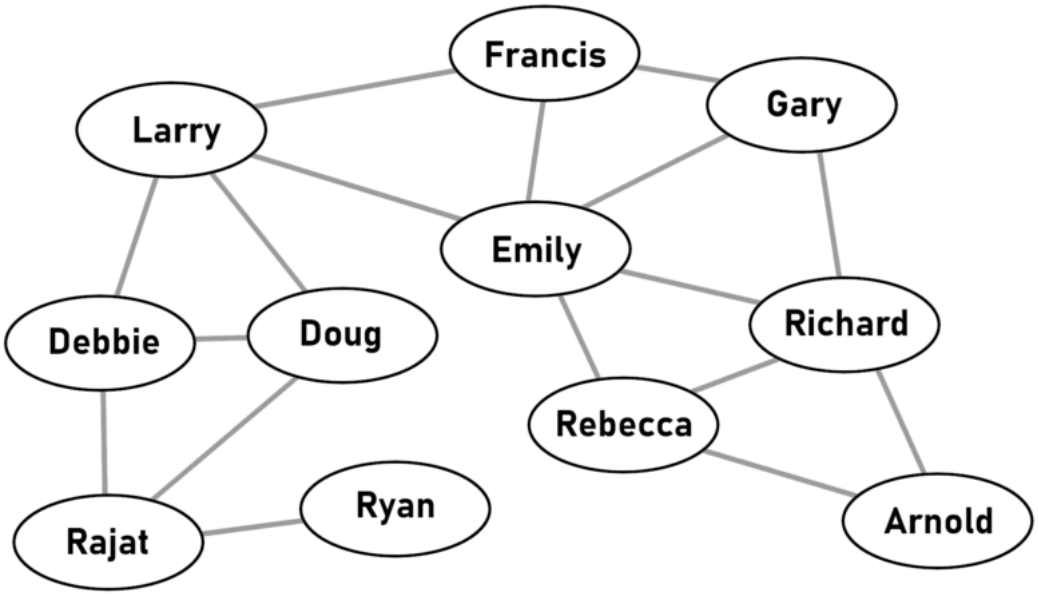


- It was proposed by Duncan J. Watts and Steven Strogatz in their joint 1998 Nature paper

Watts, Duncan J., and Steven H. Strogatz. "Collective dynamics of 'small-world'networks." *nature* 393.6684 (1998): 440-442.

Six Degree of Separation

- The idea that all people are six or fewer social connections away from each other. As a result, a chain of "friend of a friend" statements can be made to connect any two people in a maximum of six steps. It is also known as the **six handshakes rule**



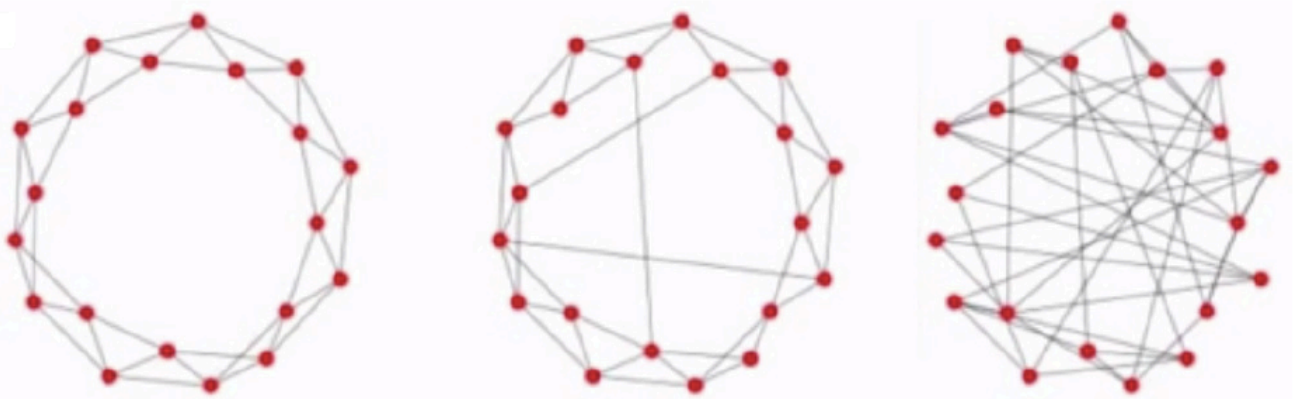
A small social group: is six degrees of separation from Arnold

Example

- Road maps, food chains, electric power grids, metabolite processing networks, networks of brain neurons, voter networks, telephone call graphs, gene regulatory networks

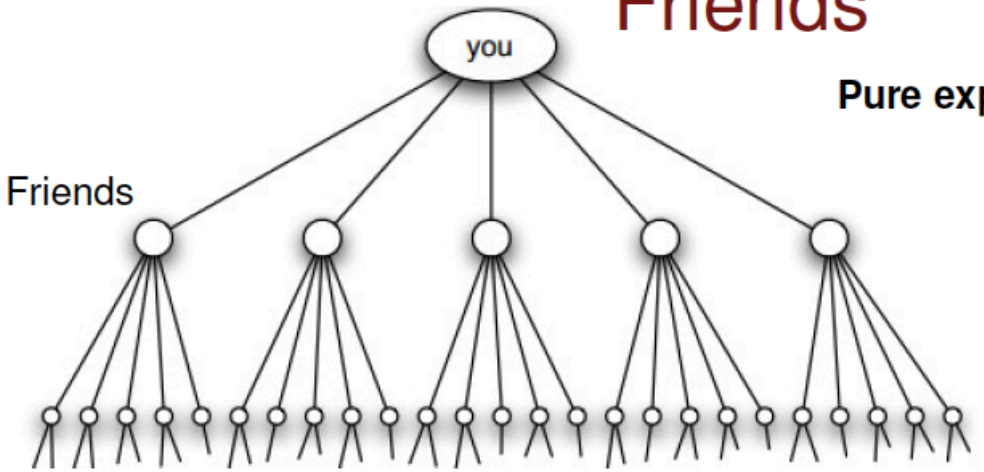
Small Worlds

- Two major properties of small world networks
  - High average clustering coefficient
    - The neighbors of a node are connected to each other
    - Nodes' contacts in a social network tend to know each other.
  - Short average shortest path length
    - Shorter paths between any two nodes in the network

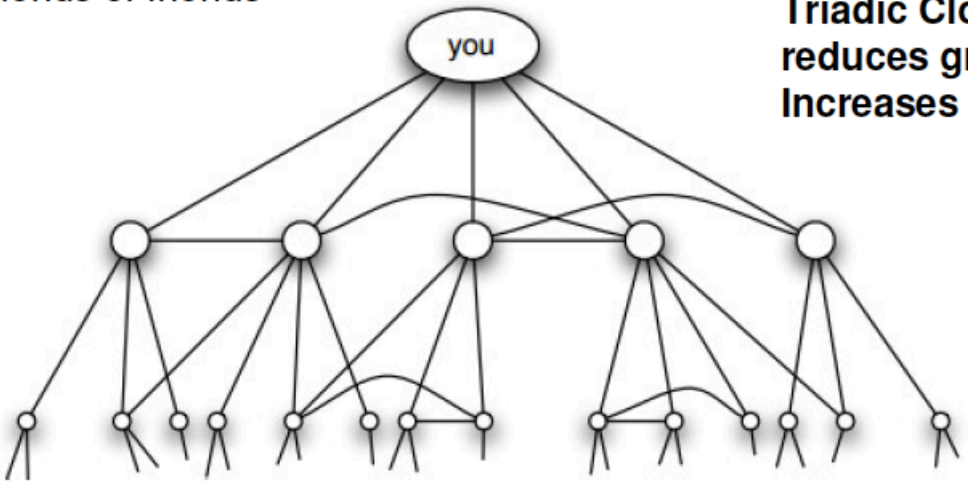


(Regular graph) (Small-world network) (Random graph)

Triadic Closure and Growth Rate of New Friends



Pure exponential growth
No triadic closure
Zero clustering coefficient

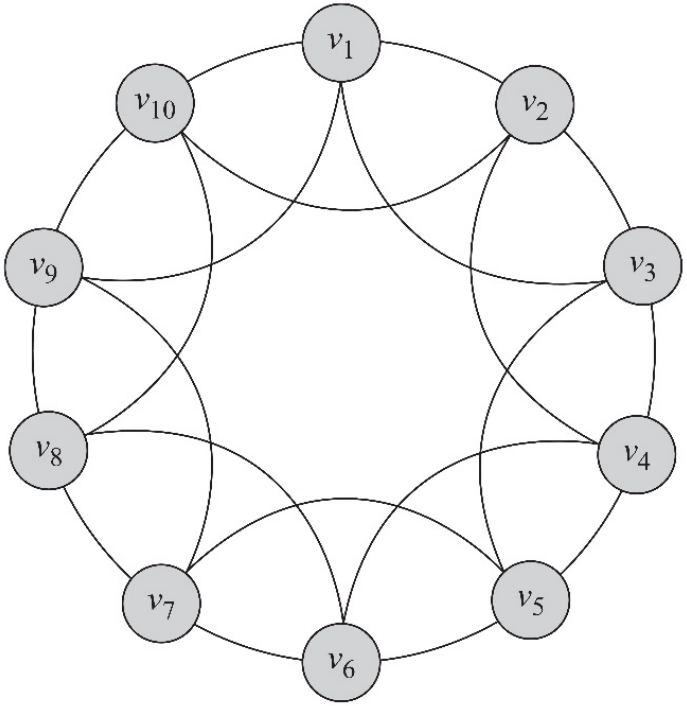


Triadic Closure
reduces growth rate of new friends
Increases clustering coefficient

The principle of triadic closure is that **if two people in a social network have a friend in common, then there is an increased probability that they will become friends at some point in the future**

Small-world Model

- In real-world interactions, many individuals have a limited and often at least, a fixed number of connections
- In graph theory terms, this assumption is equivalent to embedding users in a regular network
- A regular (ring) lattice is a special case of regular networks where there exists a certain pattern on how ordered nodes are connected to one another
- In a regular lattice of degree c, nodes are connected to their previous $c/2$ and following $c/2$ neighbors



Preferential Attachment Model

Preferential Attachment Model

- A measure used to compute the closeness of nodes, based on their shared neighbors.
- **Main assumption:**
  - When a new user joins the network, the probability of connecting to existing nodes is proportional to existing nodes' degrees
  - For the new node v
    - Connect v to a random node v_i with probability

$$P(v_i) = \frac{d_i}{\sum_j d_j}$$

  - Proposed by Albert-László Barabási and Réka Albert
    - A special case of the Yule process

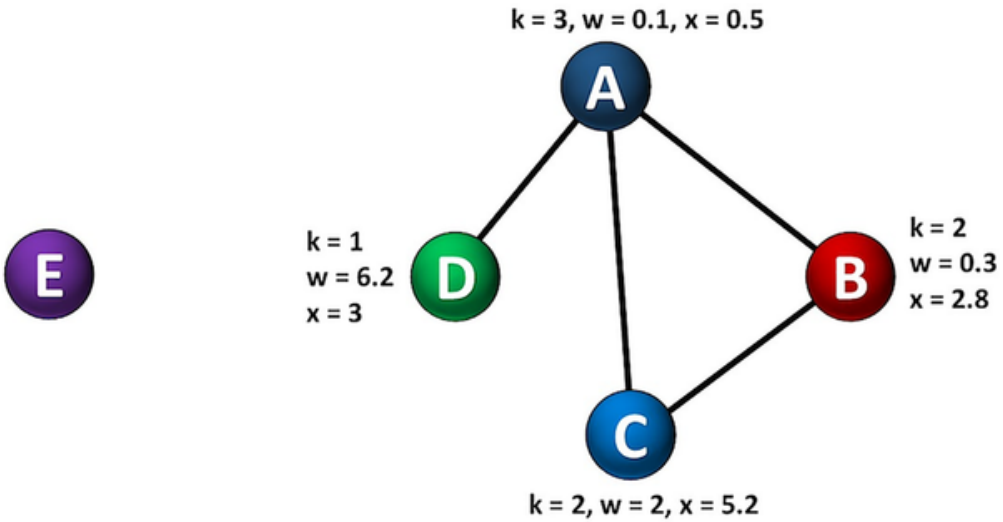
Distribution of wealth in the society:

The rich get richer

Barabási, Albert-László, and Réka Albert. "Emergence of scaling in random networks." *science* 286.5439 (1999): 509-512.

Example

Each existing node in the network (nodes A, B, C, D) has properties such as degree *k* (how many transactions were made), wealth *w* (amount of bitcoins in wallet), and fitness *x* (the potential of node *i* to create new connections, where different types of users in the system have different intrinsic fitness. A new node E might chose to connect to any existing node depending on its preference—based on degree, wealth or fitness.



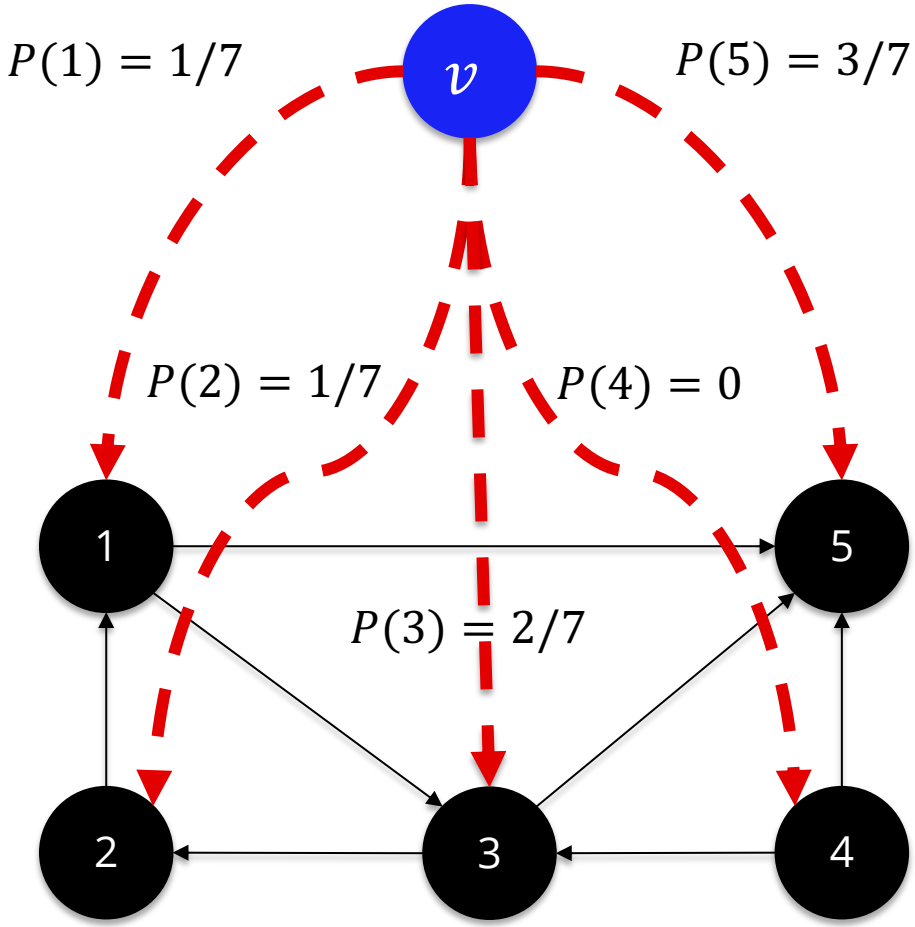
Aspembitova A, Feng L, Melnikov V, Chew LY (2019) Fitness preferential attachment as a driving mechanism in bitcoin transaction network. PLOS ONE 14(8): e0219346. https://doi.org/10.1371/journal.pone.0219346
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0219346

Preferential Attachment: Example

- Node v arrives

$$P(v_i) = \frac{d_i}{\sum_j d_j}$$

- $P(1) = 1/7$
- $P(2) = 1/7$
- $P(3) = 2/7$
- $P(4) = 0$
- $P(5) = 3/7$



Summary

- Properties of Real-world Networks
  - Power-law Distribution
  - High Clustering Coefficient
  - Small Average Path Length
- Network Models
  - Random graph Model
  - Small World Model
  - Preferential Attachment Model

Any Questions?