COMP9312

Data Analytics for Graphs

Dong Wen

2023 Term 2

UNSW SYDNEY

Teaching Team

Dr. Dong Wen (LiC)
cse.unsw.edu.au/~dwen/

Tutors are research students from DKR group in CSE:

- **Yiheng Hu (Admin)**: Graph Neural Networks
- **Simon Sima**: Graph Neural Networks
- **Kaiyu Chen**: Streaming Graph Processing

Course Info

Lectures:

4 PM – 6 PM (Monday) week 1 – 2, 4 – 5, 7 – 10

4 PM – 6 PM (Tuesday) week 1 – 5, 7-10

Tutorials: detailed tutorial notes will be provided on WebCMS3.

Week 2-5, 7-10.

~~**Consultation:** after lectures.~~

No tutorial in Week 1.

Course Info (cont)

LiC Email: dong.wen@unsw.edu.au

Admin: yiheng.hu@unsw.edu.au

For normal questions, we recommend you use the Q&A forums. You are also welcome to contact us via email if something is urgent.

Welcome to book private help sessions.

EdForum: https://edstem.org/au/courses/11987/discussion/

See FAQ on Webcms for the registration link of EdForum.

Webcms3: https://webcms3.cse.unsw.edu.au/COMP9312/23T2/

Moodle: https://moodle.telt.unsw.edu.au/course/view.php?id=76004

Course Info (cont)

Read the Webcms course outline.

Read all previous Webcms notice after you enroll the course.

Teaching/Learning

What we do to support your learning:

- Lectures
- Reference python codes
- Tutorials (starting from Week 2): guides you through the theoretical knowledge and practical skills of the course, solutions will be released after all the tutorials are delivered.
- Private help sessions
- Assignments and projects
- Textbook & papers

Syllabus Overview

Traditional algorithms on Graphs / Graph Database (Week 1 to Week 5)

- Basic concepts of graphs
- Graph traversal – DFS, BFS, connectivity, etc.
- Path and reachability queries
- Subgraph search
- Graph Database

Machine learning methods on Graphs (Weeks 7 to 10)

- Traditional machine learning methods on graphs
- Node Embedding
- Graph neural networks

Your Background

We assume that you …

- Have background in **data structure and algorithms**
- Have experience with programming languages, i.e., **Python** ~~and C/C++~~
- Hopefully, have some knowledge of relational databases

You might have acquired this background in

COMP1511, COMP1531, COMP2521, COMP3311, COMP9024, COMP9311,

Textbook

Lecture notes are sufficient, but the following materials are good:

Reference Books:

- Introduction to algorithms by Thomas H, Cormen, et al
- Cohesive subgraph computation over large sparse graphs: algorithms, data structures, and programming techniques by Lijun Chang and Lu Qin.
- Graph Representation Learning by William L. Hamilton
- Networks, Crowds, and Markets: Reasoning About a Highly Connected World by David Easley and Jon Kleinberg
- Network Science by Albert-László Barabási

Assessment Structure

```
ass1        = mark for assignment 1 (out of 15)
ass2        = mark for assignment 2 (out of 10)
project     = mark for project (out of 25)
exam        = mark for final exam (out of 50)
final_mark  = ass1 + ass2 + project + exam
grade       = HD|DN|CR|PS if final_mark >= 50
            = FL if mark < 50
```

Submitting code generated by Github Copilot, ChatGPT, Google Bard and similar tools will be treated as plagiarism.

https://www.student.unsw.edu.au/notices/2023/02/academic-integrity-reminder-chatgpt

Assignments

Two assignments are **done individually**.

- Released via Webcms
- Submitted via **Moodle**
- Plagiarism-checked (copying solutions -> 0 mark for the assignment)

Assignments (25%):

- Ass 1: Graph Traversal; Reachability/Path Queries; Cohesive Subgraphs (15%) (week 2-4)
- Ass 2: Graph Neural Networks (10%) (week 9-10)

Project

- Released via Webcms
- **Done individually**
- Submitted via Moodle

Projects (25%)

- Designing an algorithm to process graph problems (week 5-8)

Exam and Marks

Exam: 50%

- Graph Algorithms + Graph Neural Network
- Exam is around 3 hours.
- All answers are submitted via Moodle
- If you are ill on the day of the exam, **do not attend** the exam.
- Sample questions will be available in Week 10

Final mark = Ass1 + Ass2 + Project + Exam

Pass the course: Final mark >= 50

Supplementary Exam Policy

Everyone gets **exactly one chance** to pass the Exam.

If you attempt the Exam

- We assume you are fit/healthy enough to take it.
- No 2nd chance exams, even with a medical certificate.

Special consideration:

- If you are ill on the day of the exam, **do not attend** the exam.
- All special consideration requests must be submitted to UNSW.
- All special consideration requests must document how you were effect.

Beyond the course

Research Degrees: (https://research.unsw.edu.au/higher-degree-research-programs)

- PhD (3 -- 3.5 years)
- Master of Philosophy (1.5 -- 2 years)

Feel free to contact me if you are interested in obtaining a research degree.

Requirement:

GPA>80 with education from world top 400 universities. GPA > 75 if from world top 100 universities. Local students are particularly welcome to apply.

Data and Knowledge Research Group in CSE:

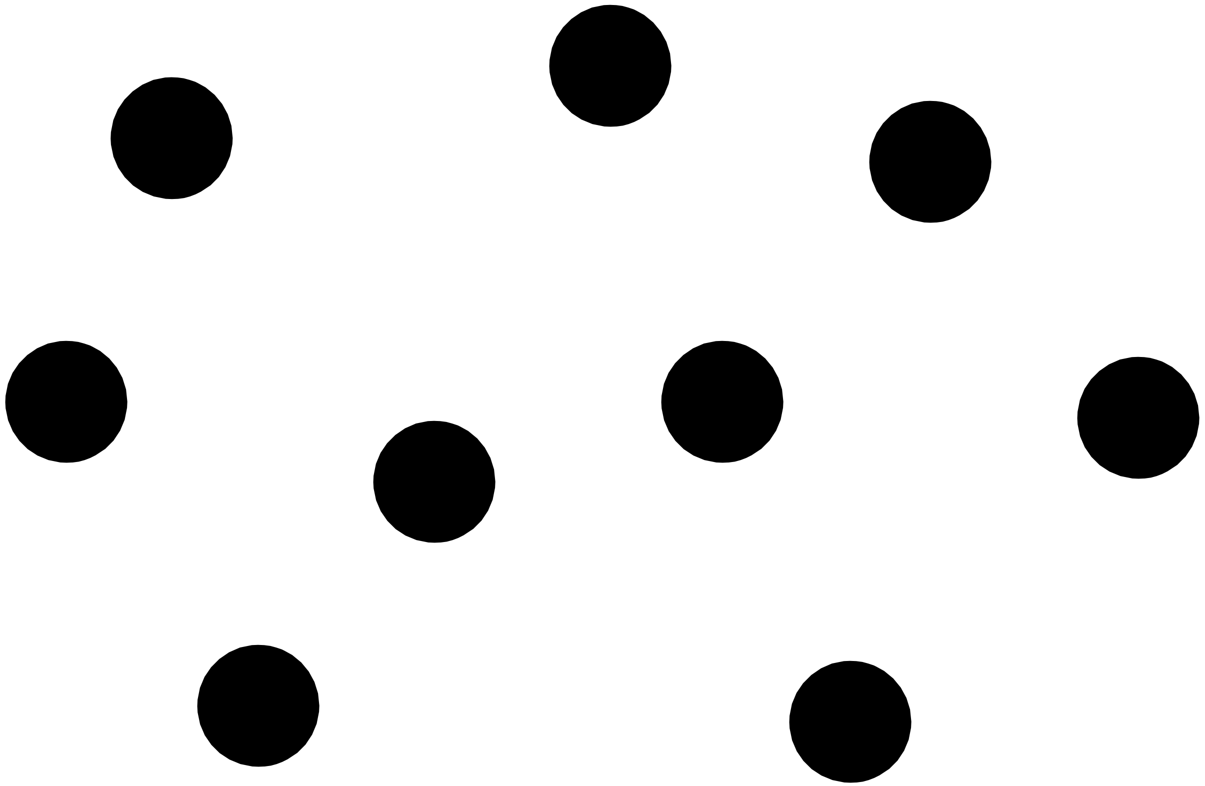
https://unswdb.github.io/

Outline

- **Basic Concepts**
  - What are networks/graphs?
  - What will we learn?
- **Characterization of graphs**
  - How to characterize graphs?
- **Data structure to represent a graph**
- **Analysing of graphs**
  - How to analyse graphs data?
- **Applications of graphs**
  - Applications of graphs analysis

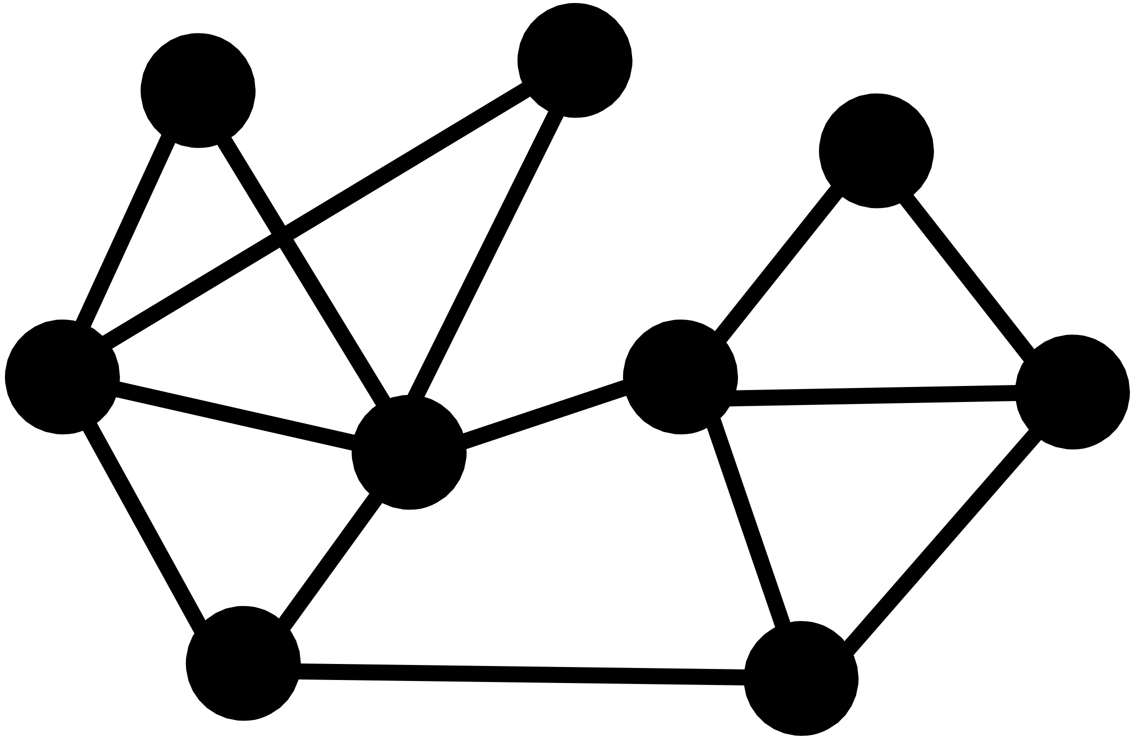
Why Graphs?

Graph is a general language for describing and analyzing entities with relations/interactions.



users, machines, webpages, road intersections, and any entities…

Network / Graph



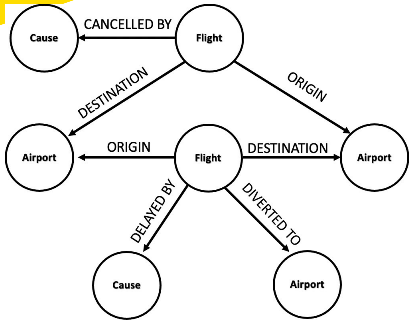
Sometimes the distinction between networks & graphs is blurred

Types of Networks and Graphs

Graphs (also known as Networks):

- **Social networks:**
  - **Society** is a collection of 7+ billion individuals
- **Communication and transactions:**
  - Electronic devices, phone calls, financial transactions
- **Biomedicine:**
  - Interactions between **genes/proteins** regulate life
- **Brain connections:**
  - Our **thoughts** are hidden in the connections between billions of neurons
- . . .

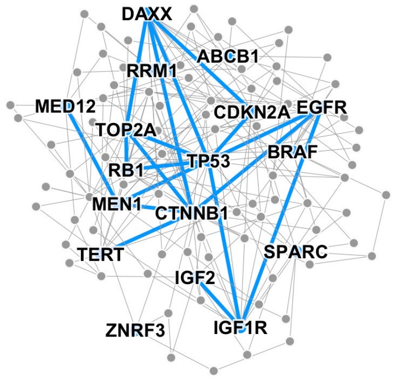
Many Types of Data are Graphs (1)



Event Graphs

Image credit: SalientNetworks

Computer Networks



Disease Pathways

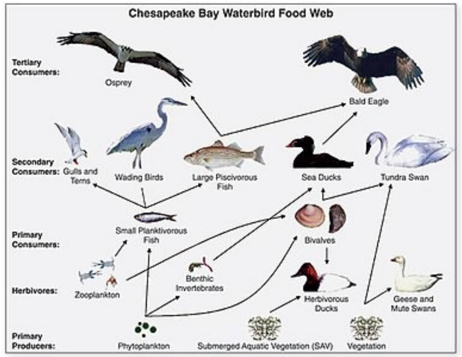


Image credit: Wikipedia

Food Webs

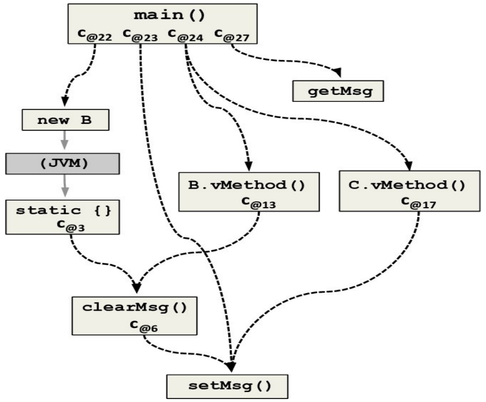


Image credit: ResearchGate

Code Graphs

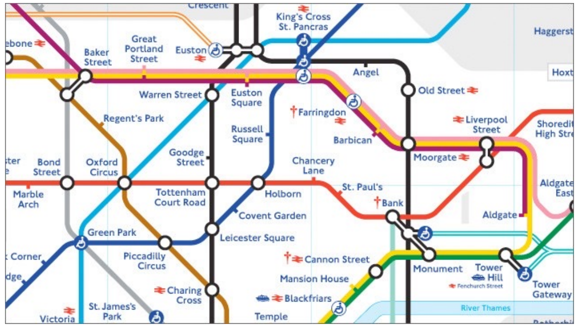


Image credit: visitlondon.com

Underground Networks

Many Types of Data are Graphs (2)

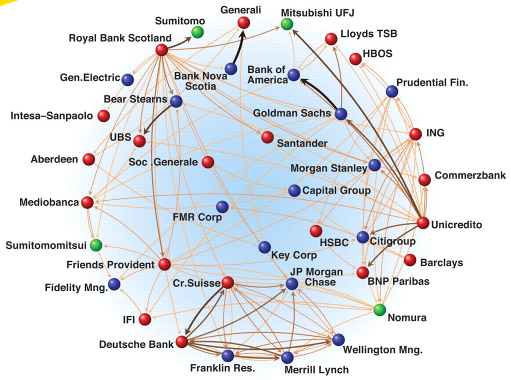


Image credit: Science

Economic Networks

Image credit: Lumen Learning

Communication Networks

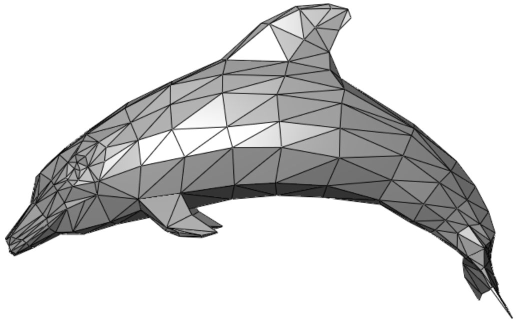
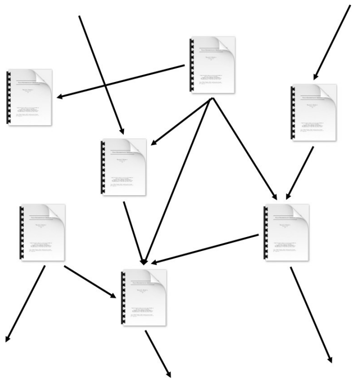


Image credit: Wikipedia

3D Shapes



Citation Networks

Image credit: Missoula Current News

Internet

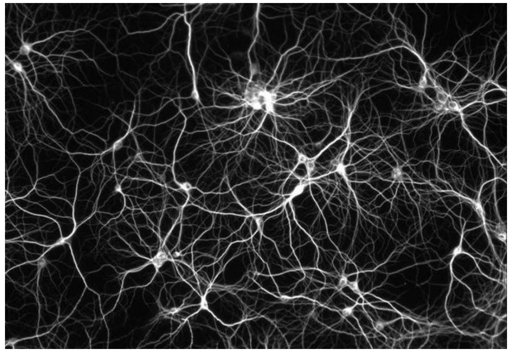


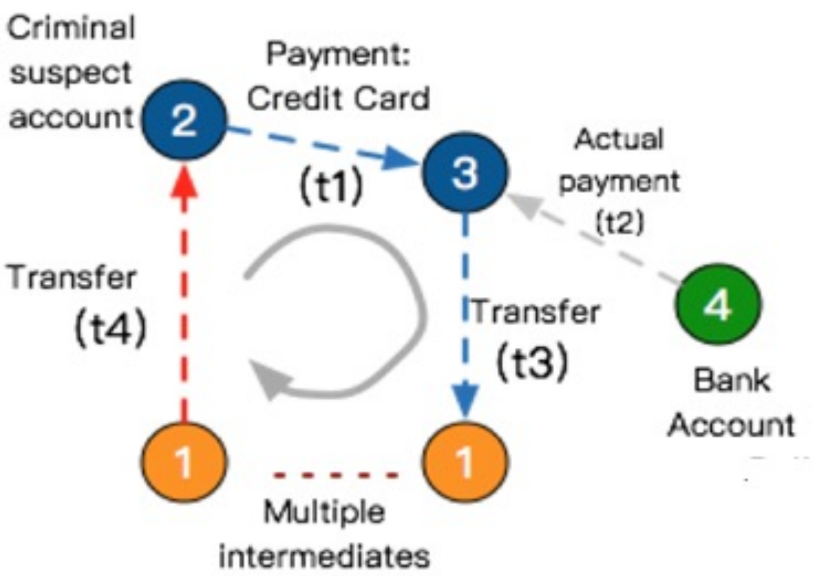
Image credit: The Conversation

Networks of Neurons

Why we study networks/graphs ?

Networks / graphs are everywhere, and we live in a **highly-connected world**.

In many applications, we need analyze **in the context of networks**, not just individuals.



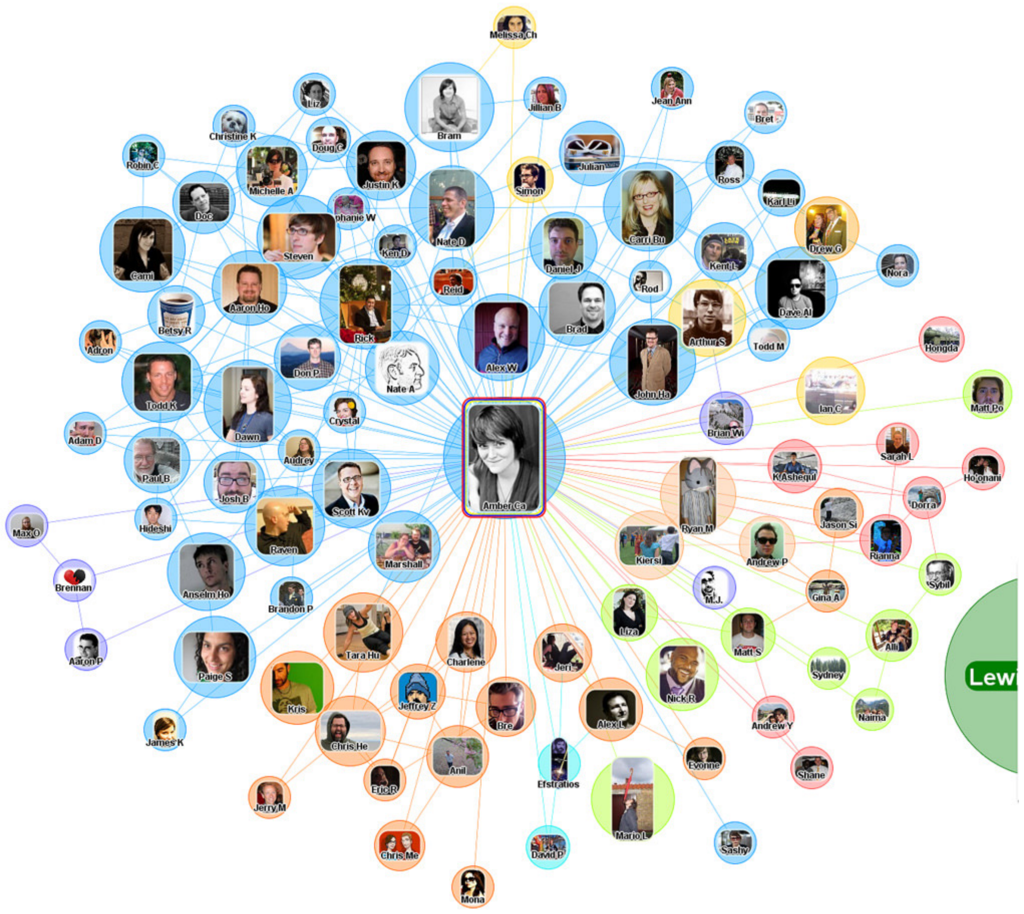
Money laundering detection

Predict the spread of virus

Social Networks – Facebook ego-network

- Find your communities
- Know how to approach a person via Facebook.

e.g., how to increase the chance of getting your friend invitation accepted?

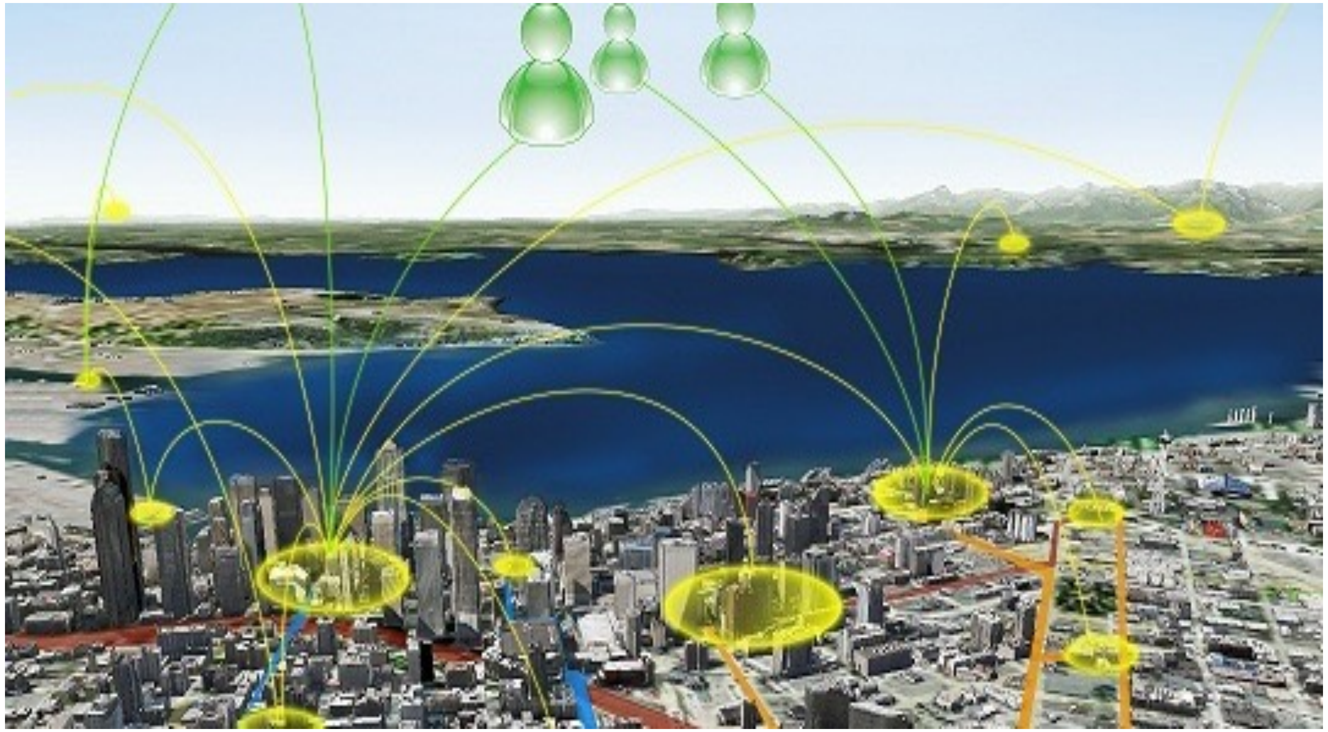


Social Networks – Facebook (location-based visualization in 2011)

Monthly active users: around **1 billion** in **2012** and **2.32 billion** (2017) **-** now around **2.5 billion (2020)**

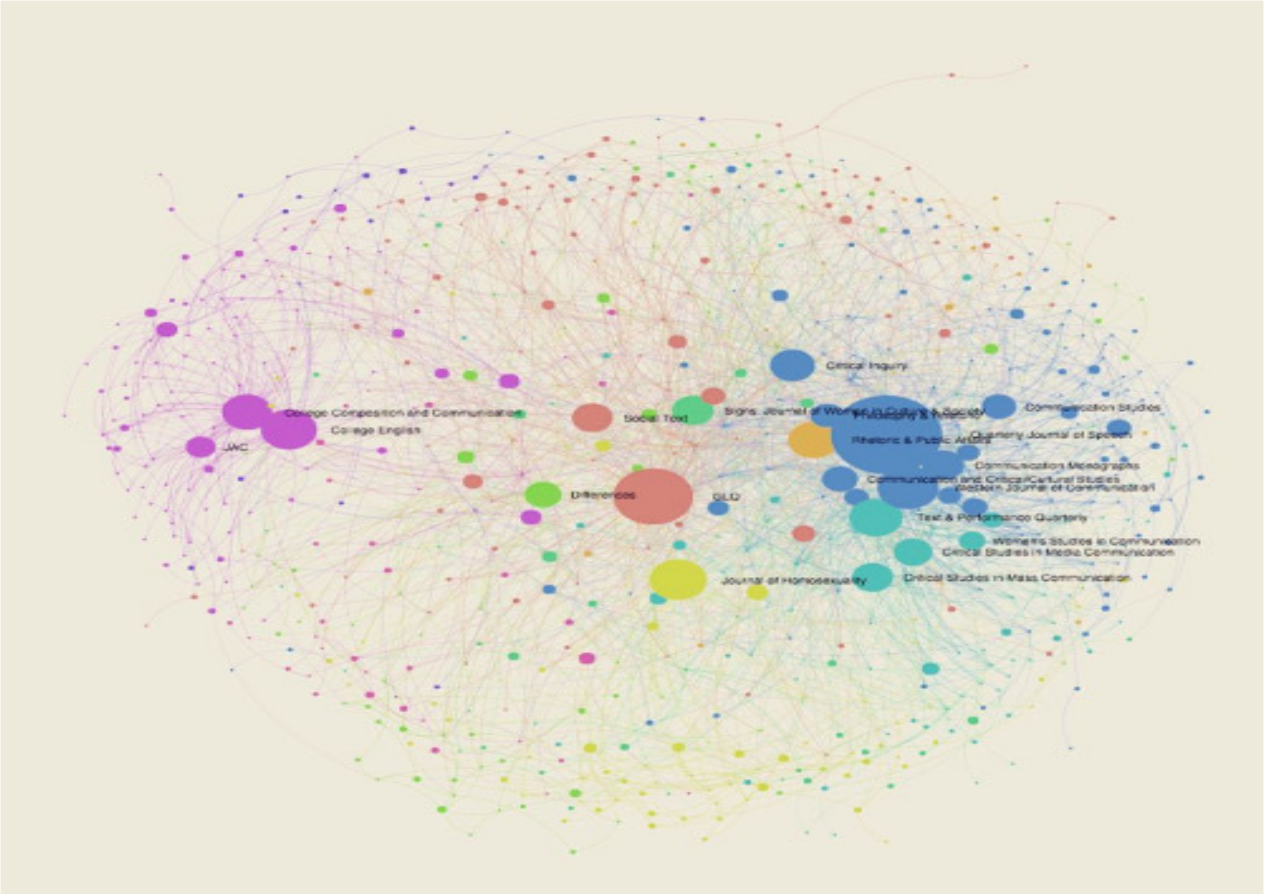
Facebook social graph [Backstrom-Boldi-Rosa-Ugander-Vigna, ***2011****]*

Social Networks – Location Based Social Network (LBSN)



https://www.microsoft.com/en-us/research/project/location-based-social-networks

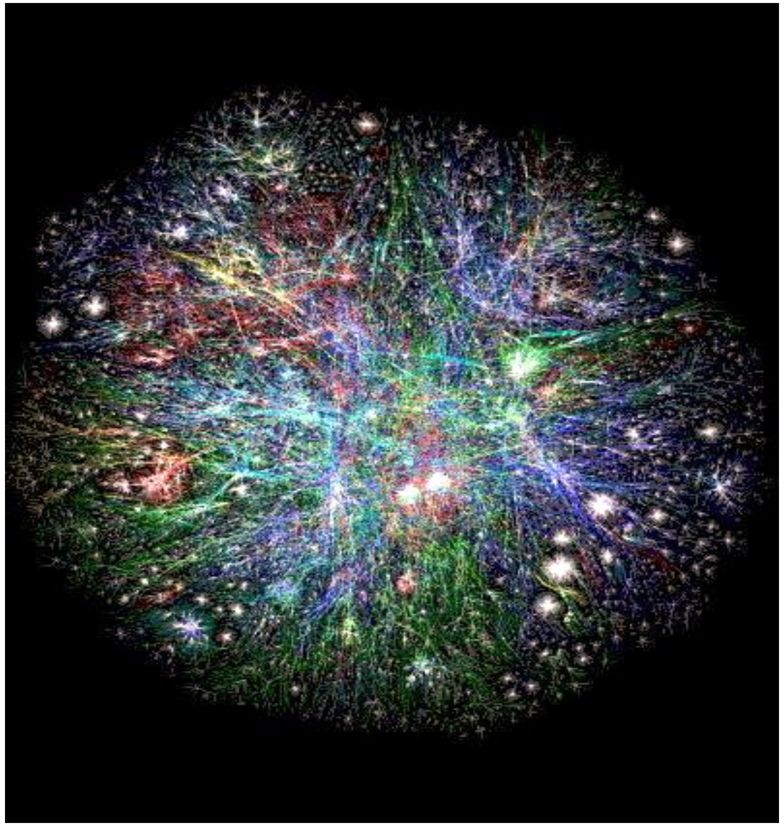
Social Networks – Citation Network



http://michaeljfaris.com/blog/2015/08/2353652015-visualizing-citation-networks/

Citation Network is a social network which contains paper sources and linked by co-citation relationships

Information Networks – WWW



http://www.vlib.us/web/worldwideweb3d.html

3D Map of the World Wide Web

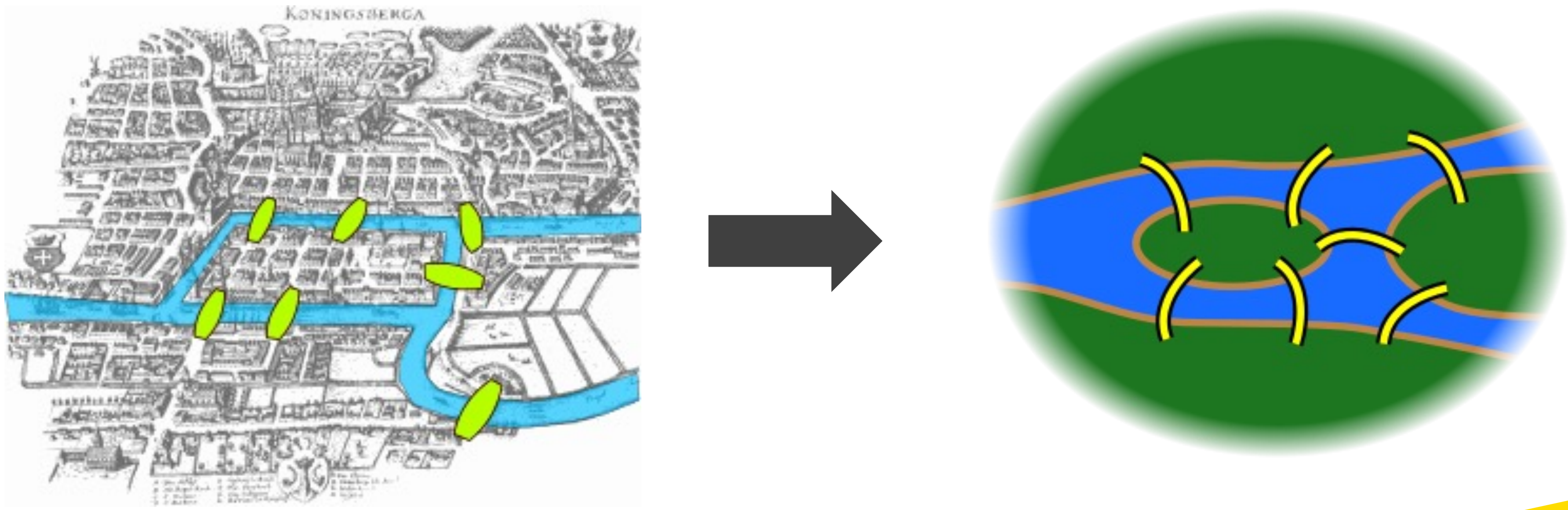
This illustrates in 3-D the actual domains and connections of the world wide web. **Colors** have been added to represent .edu, .gov, .com, etc. domains.

What we learn

Graph Theory

Leonhard Euler in 1735:

- find a roundtrip through the city that would cross each bridge once and only once
- earliest (published) work on networks/graphs, laid the foundations of graph theory



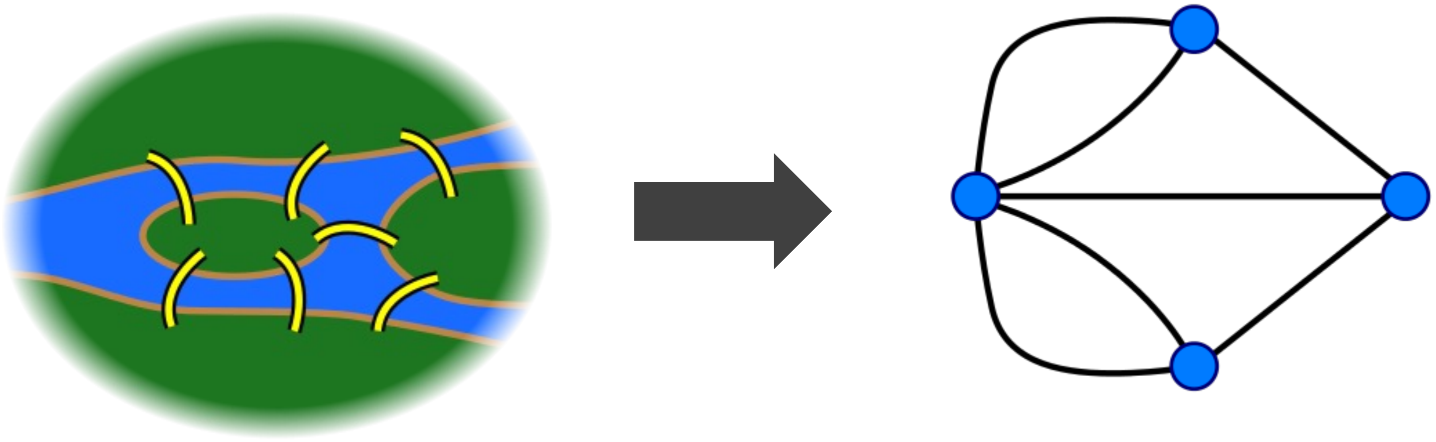
Graph Theory (cont)

Abstraction (Problem):

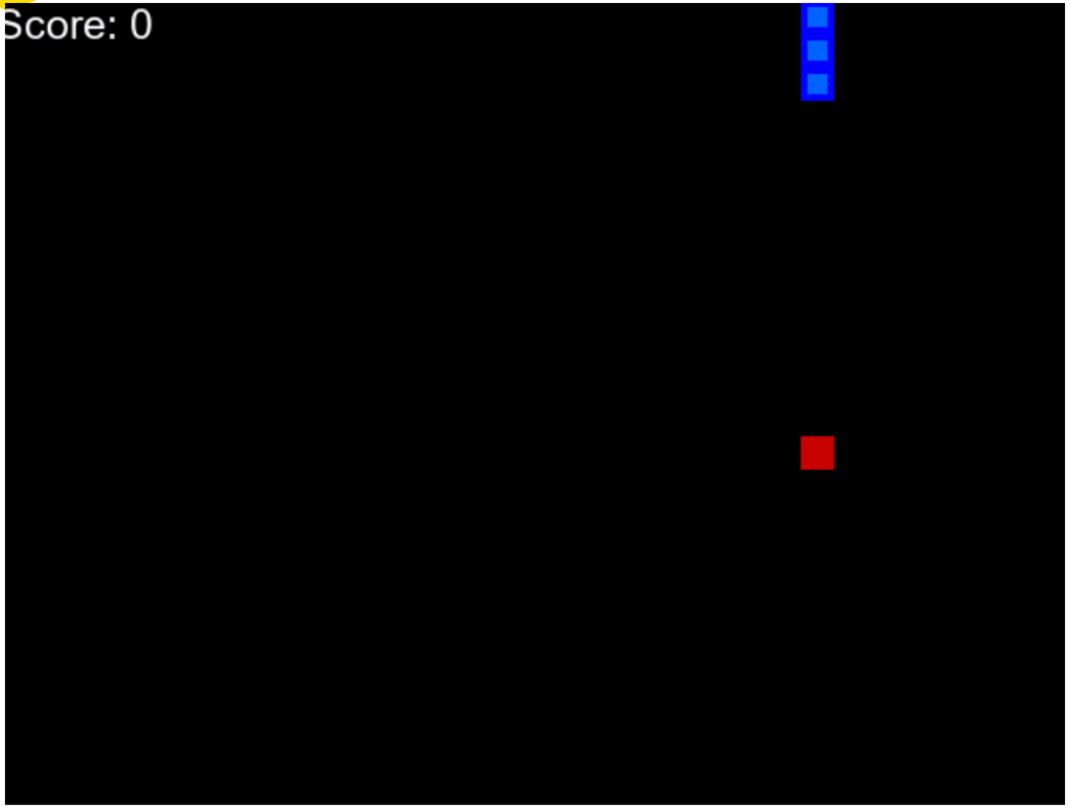
- replaced each land mass with an abstract "vertex" or node, and each bridge with an abstract connection, an "edge”
- Proved that this problem has no solution.

Euler Tour (Solution):

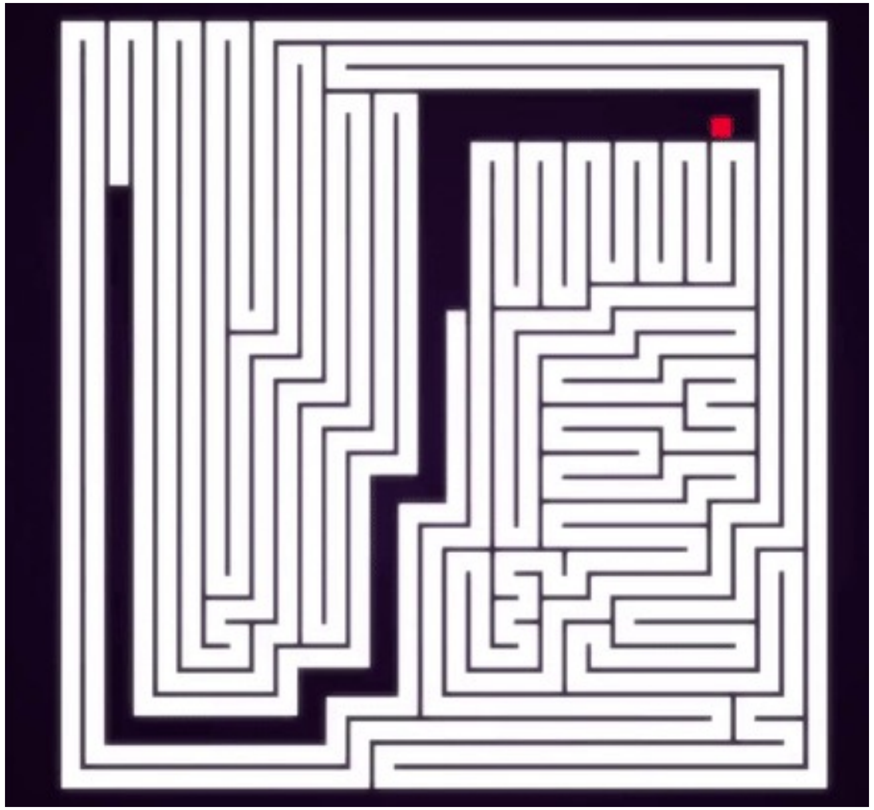
- a necessary and sufficient condition for the walk of the desired form: connected and all vertices with even degrees.



Snake



Does an optimal plan exist?



https://en.wikipedia.org/wiki/Hamiltonian_path

We study …

Graph Theory

Mathematical proofs

Big Graph Algorithms

Efficiency and scalability

Graph Applications

Interdisciplinary tasks

RESEARCH ON GRAPH:

REAL DATA

- Money Transactions
- Protein Interactions
- Internet
- System
- Medical Records
- Phone Calls
- Road Map
- Social Network
- ...

GRAPH MODEL

- Traditional Graph
- Streaming Graph
- Heterogeneous Graph
- Knowledge Graph
- Attributed Graph
- Temporal Graph
- Evolving Graph
- Uncertain Graph
- ...

GRAPH PROBLEM

Link Analysis (relevance): Pagerank, SimRank, Link Prediction, Clustering

Degree/Neighbors (structural): Cluster Coefficient, Triangle, Vertex Cover

Distance/Path (reachability): Reachability, Shortest Path, Spanning Tree, Steiner Tree, K-Hop, Centrality, KNN, Keyword Search

Subgraph Detection: Graph Similarity, Cohesive Subgraph, Subgraph Enumeration

Propagation: Influence Maximization, Information Cascade, Label Propagation

Node mapping (Embedding): Node2Vec, AI4DB

Graph Analytics System

Graph Database

APPLICATION

- Recommendation
- Anomaly Detection
- Visualization
- Cybersecurity
- Fraud Detection
- Marketing
- Legal Reasoning
- Promotion
- Traffic Monitoring
- Virus Control
- ...

We study …

Database
Part 1

- Big data processing
- Time complexity
- External memory
- Distributed systems
- …

Data Mining

- Vertex similarity
- Community search

…

Machine Learning
Part 2

- Graph neural networks
- Node classification
- AI
- …

Programming Languages

C/C++
Efficiency

Java
Distributed Applications
Android, Web …

Javascript
Phone App, Web …

…

Python
Graph Neural Networks
Plugins …

We mainly use Python in this course …

Let’s see some simple codes :)

Python

```
a_set = {“aaa”, “bbbb”, “c”}
a_tuple = (“aaa”,1)
```

List vs Array

```
a_list = [1,2,3,4,5]
b_list = [1,2,“abc”,4,5]
```

~~import numpy as np~~
~~arr = np.array([1, 2, 3, 4, 5])~~

Pre-requisites

- Binary Search
- Balanced Binary Search Tree (AVL-tree & red-black tree)
- Hash Set/Map
- Heap/Priority Queue
- Sorting Algorithm
- …

Warm Up

Given a graph with n vertices and m edges:

- How to compute the shortest distance between two query vertices?
- What is the time complexity of your solution?
- Can you further improve your solution?

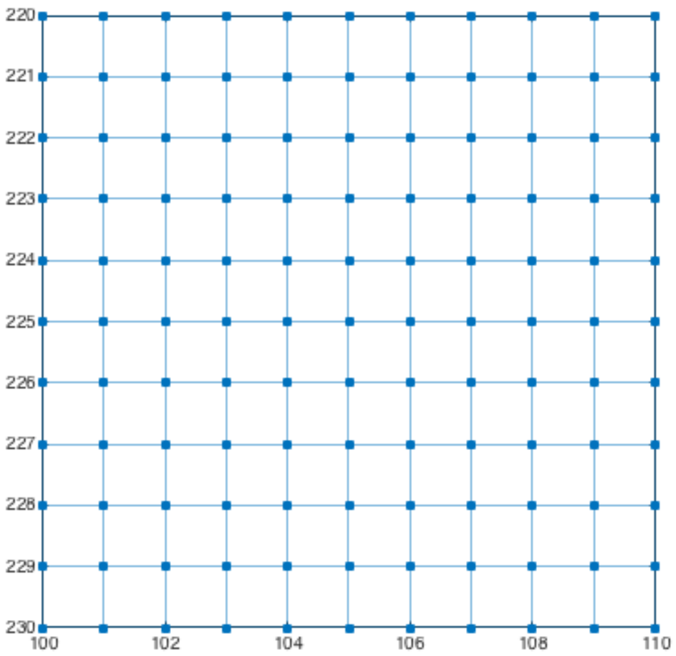
Recap

String, Image, Tree

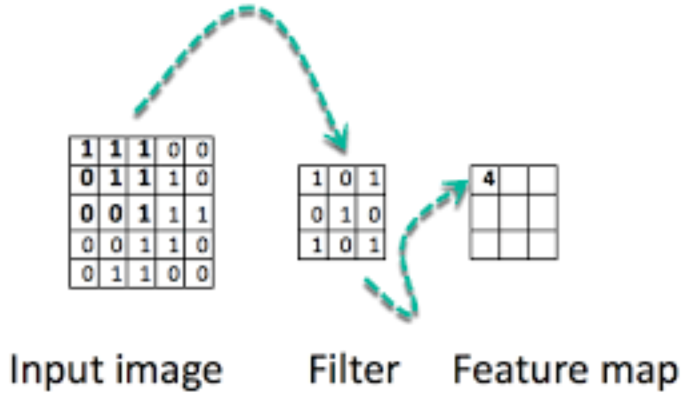
python list [ ] vs array

linked list?

pointers array

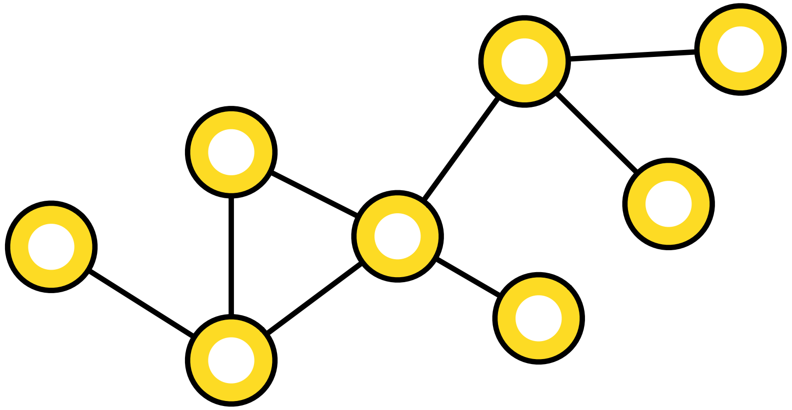


Convolution



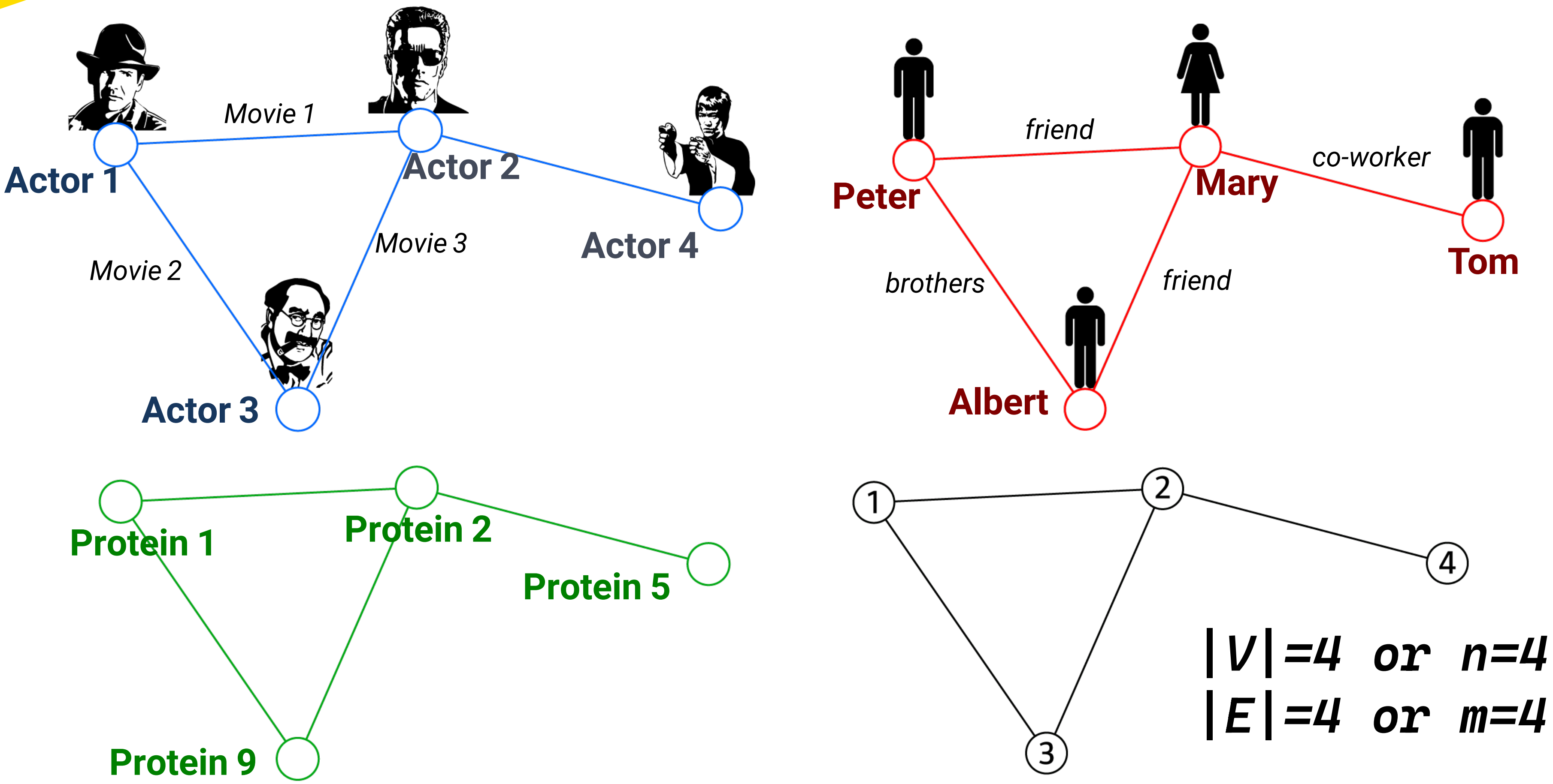
Characterization of graphs

Subgraph / component



- **Objects:** nodes, vertices $V, \; n=|V|$
- **Interactions:** links, edges $E, \; m=|E|$
- **Systems:** networks, graphs $G(V,E)$

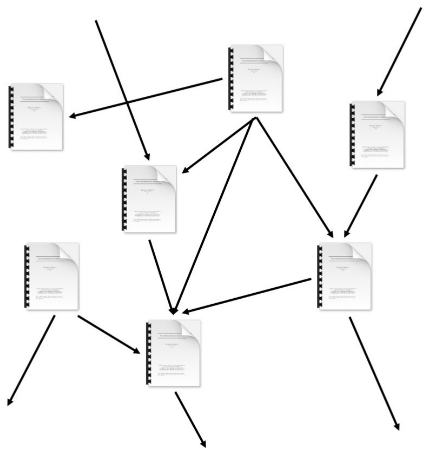
Graphs: A Common Language



Choosing a Proper Representation

- If you connect individuals that work with each other, you will explore a **professional network.**
- If you connect those that have a sexual relationship, you will be exploring **sexual networks.**
- If you connect scientific papers that cite each other, you will be studying the **citation network.**
- **If you connect all papers with the same word in the title, what will you be exploring?**

  It is a graph/network, nevertheless.



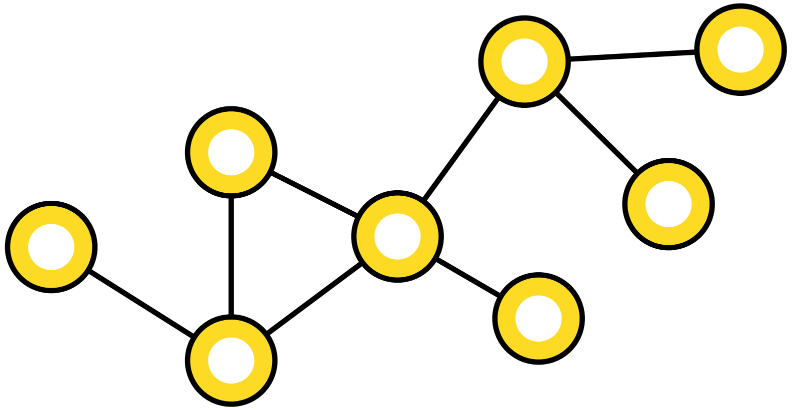
How do you define a graph?

- How to build a graph:
  - What are vertices?
  - What are edges?

- Choice of the proper network representation is important:
  - The way you assign links will determine the nature of the question you can study

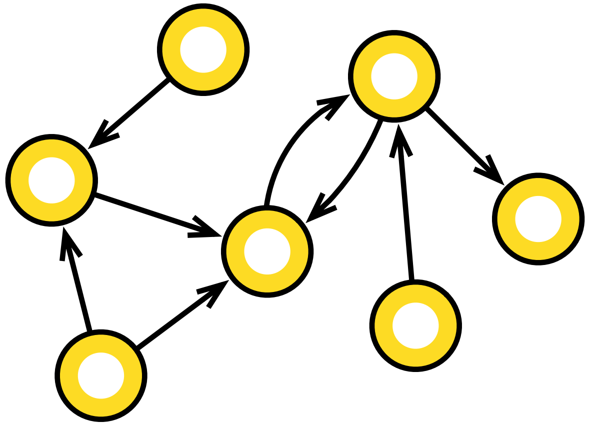
- Consider an email scenario:
  - User (email address, …)
  - Email (from, to, cc, email content)
  - …

Directed vs Undirected



An undirected graph

- Collaborations
- Friendship on Facebook

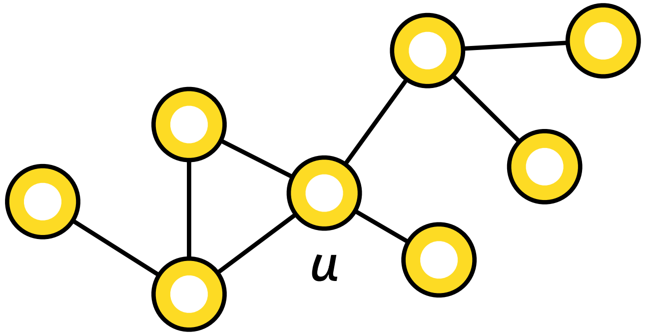


A directed graph

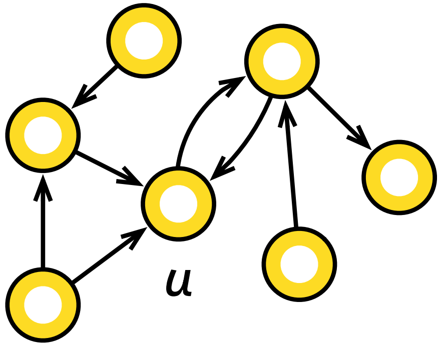
- Phone calls
- Following on Twitter

Node degree

Undirected



Directed



For undirected graphs:

Vertex degree: the number of edges adjacent to the vertex

$deg(u) = 4$

Average degree: $deg_{avg} = 2m/n$

For directed graphs:

In-degree (#in-neighbors) and out-degree (#out-neighbors)

$deg_{in}(u) = 3, \ deg_{out}(u) = 1$

Average out-degree equals to average in-degree

Characterization

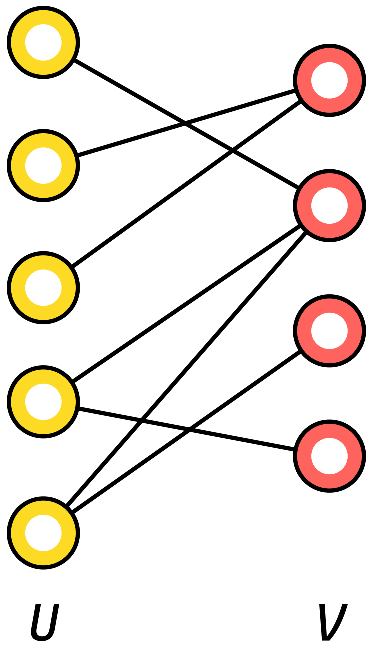
- **Degree**: how many friends do I have?
- **Path**: how far am I from another vertex?
- **Connectivity**: can I reach other vertices?
- **Density**: how dense are they? (Number of edges / Number of possible edges)
- …

Bipartite Graph

Bipartite graph is a graph whose vertices can be divided into two disjoint sets *U* and *V* such that every link connects a vertex in *U* to one in *V*; that is, *U* and *V* are independent sets

Examples:

- Authors-to-Papers (they authored)
- Actors-to-Movies (they appeared in)
- Users-to-Movies (they rated)
- Recipes-to-Ingredients (they contain)

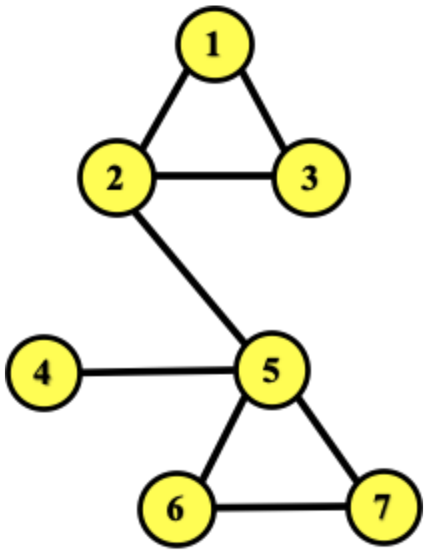


“Folded” networks:

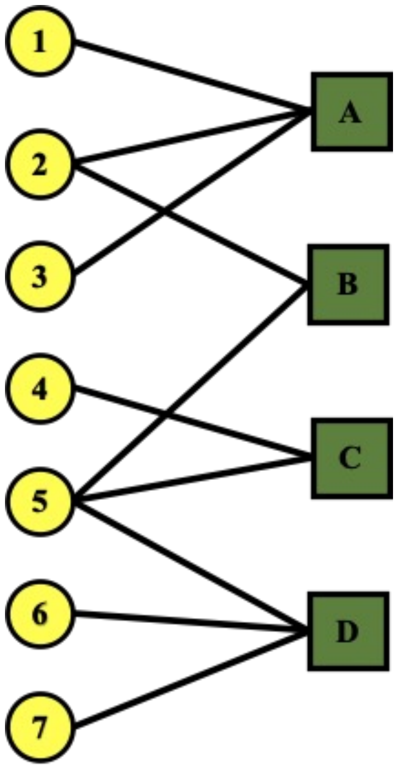
- Author collaboration networks
- Movie co-rating networks

Folded/Projected Bipartite Graphs

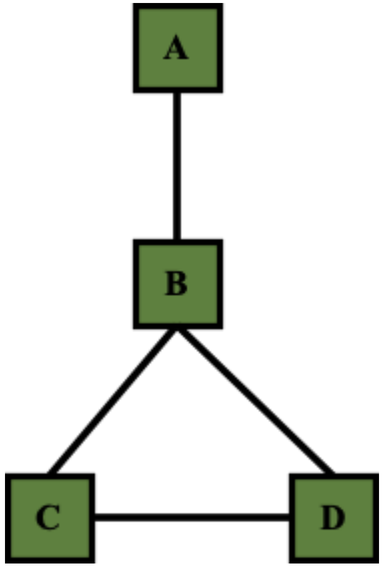
Projection U



U **V**



Projection V



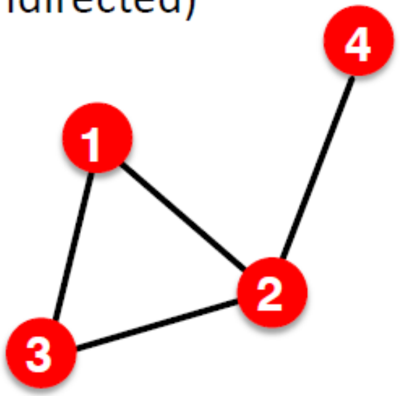
Node and edge attributes

Possible options:

- Weight (*e.g.*, frequency of communication)
- Ranking (best friend, second best friend…)
- Type (friend, relative, co-worker)
- Properties depending on the structure of the rest of the graph: Number of common friends

More Types of Graphs

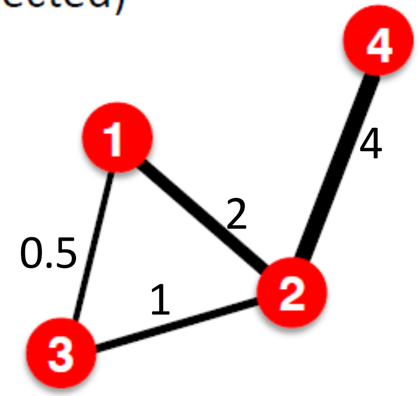
- **Unweighted**
  (undirected)



$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Examples: Friendship, Hyperlink

- **Weighted**
  (undirected)

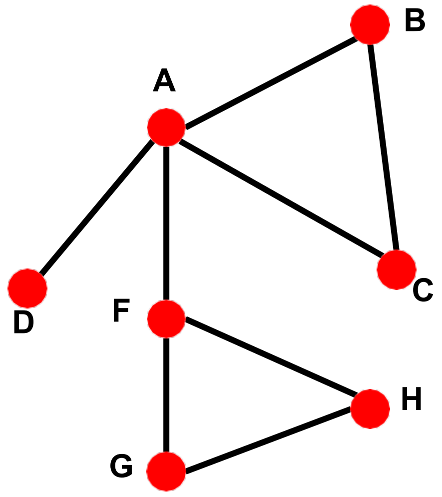


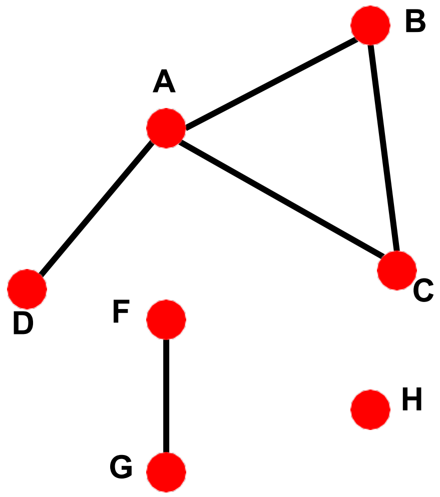
$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

Examples: Collaboration, Internet, Roads

Connectivity of Undirected Graphs

- **Connected (undirected) graph:**
  - Any two vertices can be joined by a path
- A disconnected graph is made up by two or more connected components



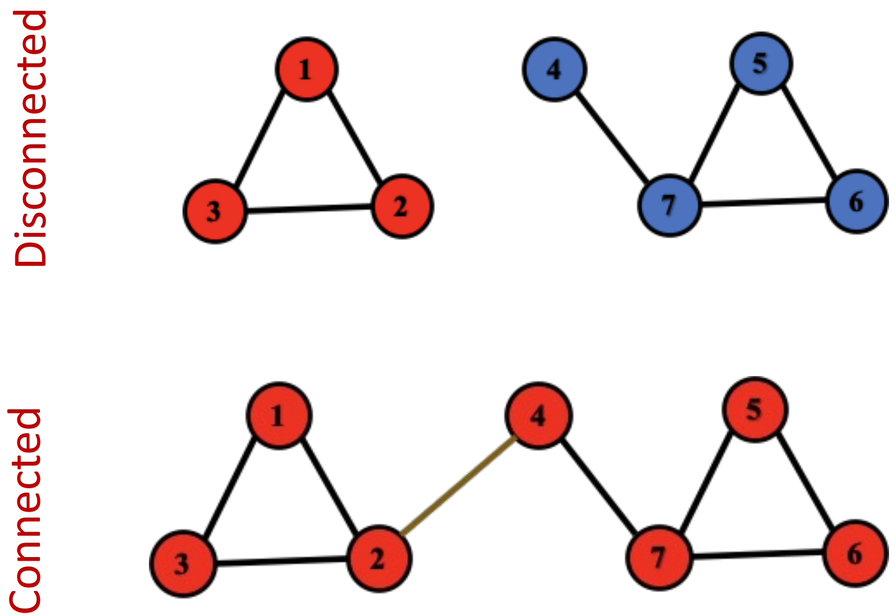


Largest Component:
Giant Component

Isolated node (node H)

Connectivity: Example

The adjacency matrix of a network with several components can be written in a block-diagonal form, so that nonzero elements are confined to squares, with all other elements being zero:

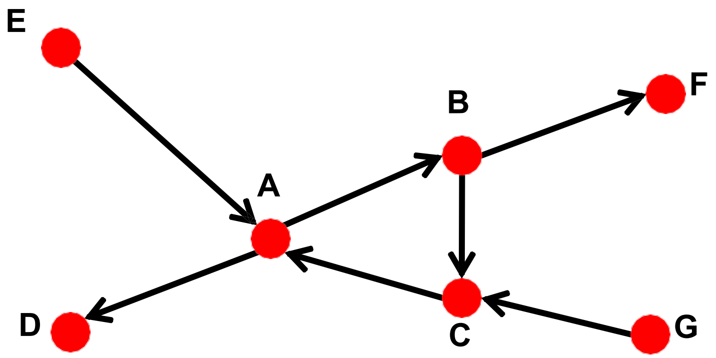


$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

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

Connectivity of Directed Graphs

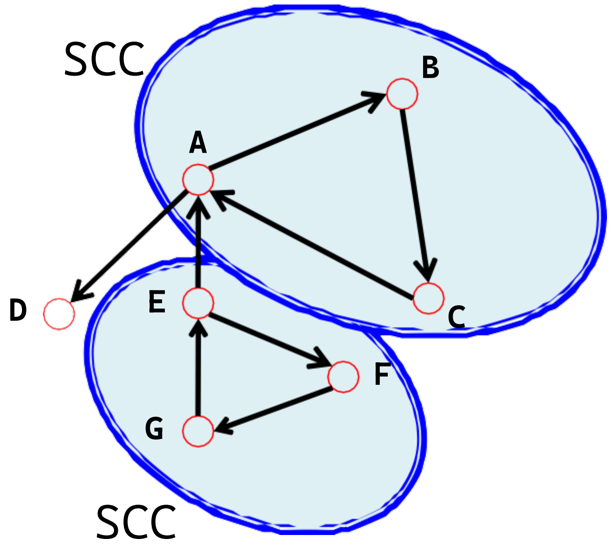
- Strongly connected directed graph
  - has a path from each vertex to every other vertex and vice versa (e.g., A-B path and B-A path)
- Weakly connected directed graph
  - is connected if we disregard the edge directions

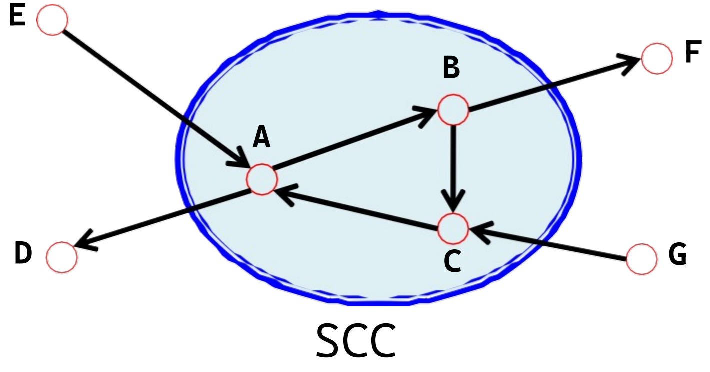


Graph on the left is connected but not strongly connected (e.g., there is no way to get from F to G by following the edge directions).

Connectivity of Directed Graphs

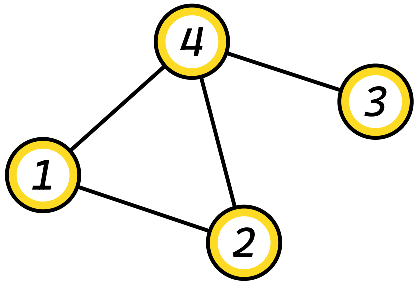
Strongly connected components (SCCs) can be identified, but not every vertex is part of a nontrivial strongly connected component.

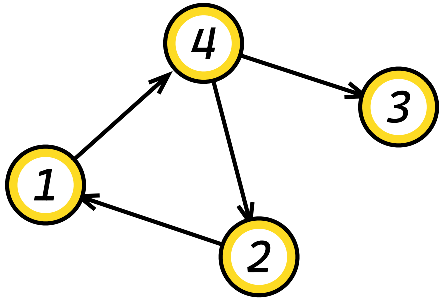




Data structure of a graph

Adjacency Matrix





$A_{ij} = 1$ if there is a link from vertex i to vertex j

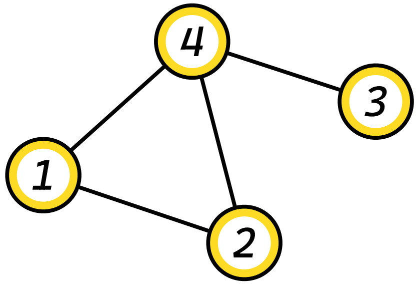
$A_{ij} = 0$ *Otherwise*

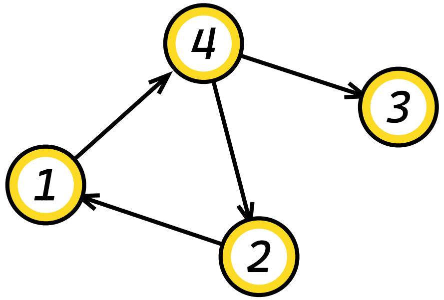
$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

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

Note that for a directed graph (right) the matrix is not symmetric.

Adjacency Matrix (cont)





$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$A_{ij} = A_{ji}$$
$$A_{ii} = 0$$

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

$$A_{ij} \neq A_{ji}$$
$$A_{ii} = 0$$

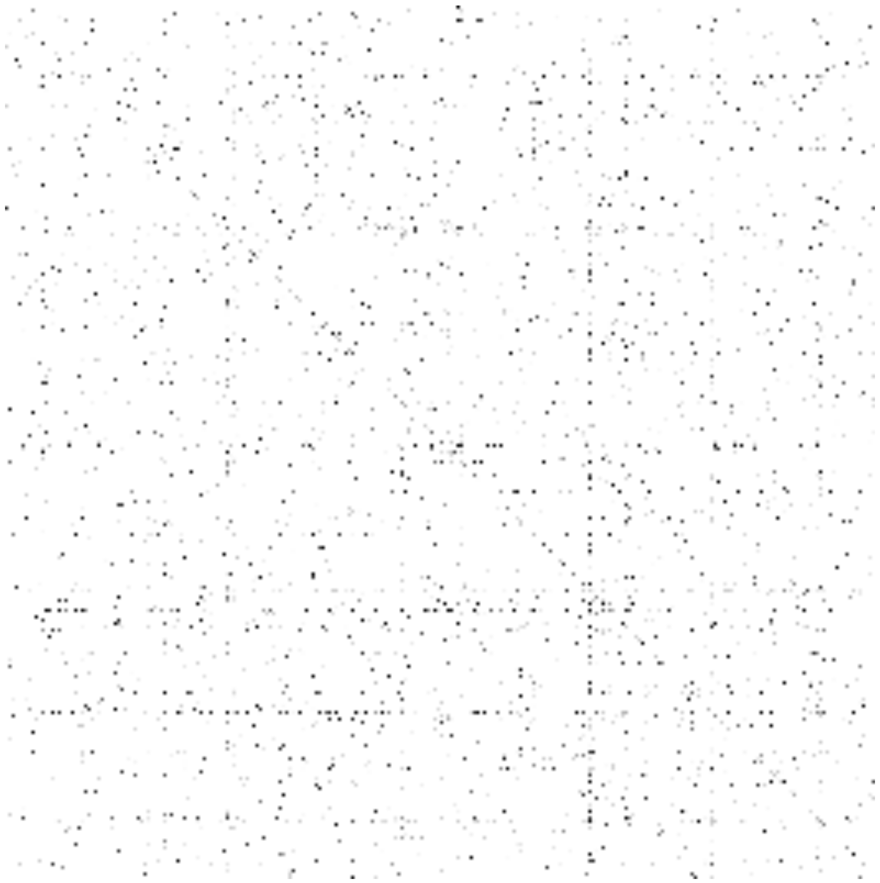
Most real-world networks are sparse

$E \ll E_{max}$ or $deg_{avg} \ll |V|-1$

NETWORK	NODES	LINKS	DIRECTED/ UNDIRECTED	N	L	<k>
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.33
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Phone Calls	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email Addresses	Emails	Directed	57,194	103,731	1.81
Science Collaboration	Scientists	Co-authorship	Undirected	23,133	93,439	8.08
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908	83.71
Citation Network	Paper	Citations	Directed	449,673	4,689,479	10.43
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802	5.58
Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930	2.90

Consequence: Adjacency matrix is filled with zeros!

Real Adjacency Matrices are sparse



Scanning neighbors is inefficient

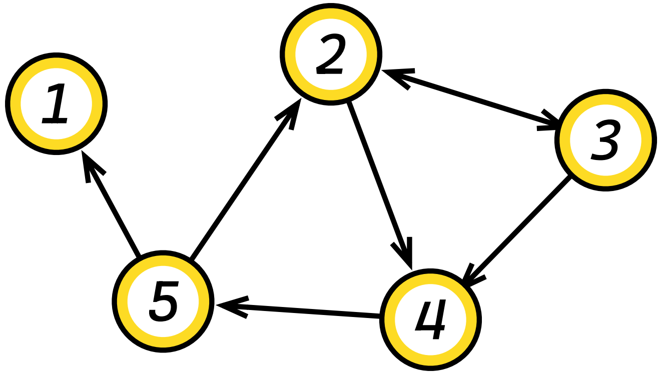
Implement adjacency matrix in Python

Explore real graphs:
http://snap.stanford.edu/data/index.html

Adjacency list

- Easier to work with if network is
  - Large
  - Sparse

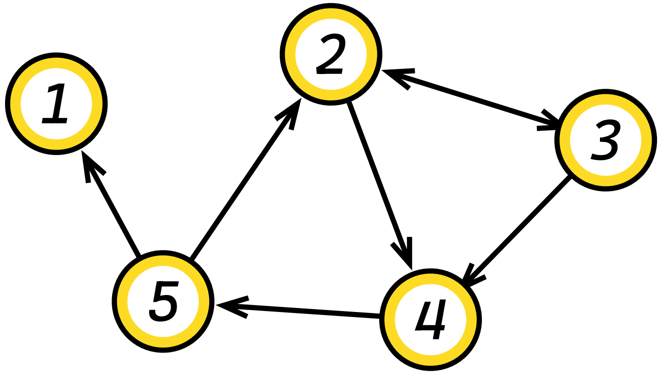
- Allows us to quickly retrieve all neighbors of a given vertex
  - *1:*
  - *2: 3, 4*
  - *3: 2, 4*
  - *4: 5*
  - *5: 1, 2*



Adjacency list (cont)

- Easier to work with if network is
  - Large
  - Sparse

- Allows us to quickly retrieve all neighbors of a given vertex
  - *1:*
  - *2: 3, 4*
  - *3: 2, 4*
  - *4: 5*
  - *5: 1, 2*



Implement adjacency list in Python

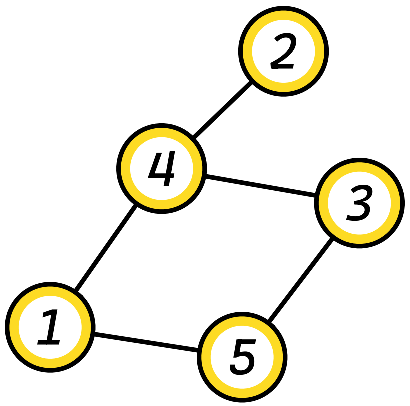
Space-efficient for sparse graphs

Good for graph updates

Efficient enough?

Quick quiz

- Compute the adjacency matrix and the adjacency list of the right graph

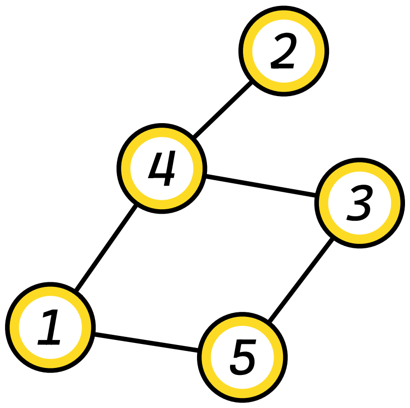


- Degree of each vertex

- Density of the graph

Quick quiz

- Compute the adjacency matrix and the adjacency list of the right graph
  - 1: [0, 0, 0, 1, 1]
  - 2: [0, 0, 0, 1, 0]
  - 3: [0, 0, 0, 1, 1]
  - 4: [1, 1, 1, 0, 0]
  - 5: [1, 0, 1, 0, 0]
  - 1: 4, 5
  - 2: 4
  - 3: 4, 5
  - 4: 1, 2, 3
  - 5: 1, 3
- Degree of each vertex
  - 1: 2
  - 2: 1
  - 3: 2
  - 4: 3
  - 5: 2
- Density of the graph
  5/10 = 0.5



Complexity difference between the adjacency matrix and the adjacency list:

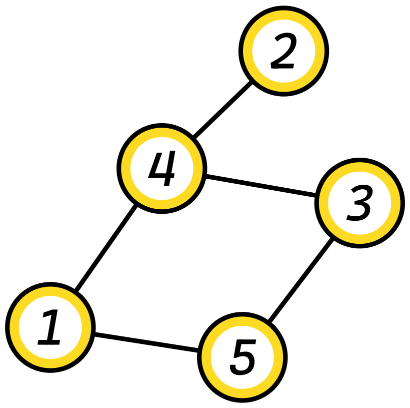
Storage space: ?

Query whether an edge (u, v) exists : ?

Get all the neighbors of a vertex v: ?

Quick quiz

- Compute the adjacency matrix and the adjacency list of the right graph
  - 1: [0, 0, 0, 1, 1]
  - 2: [0, 0, 0, 1, 0]
  - 3: [0, 0, 0, 1, 1]
  - 4: [1, 1, 1, 0, 0]
  - 5: [1, 0, 1, 0, 0]
  - 1: 4, 5
  - 2: 4
  - 3: 4, 5
  - 4: 1, 2, 3
  - 5: 1, 3

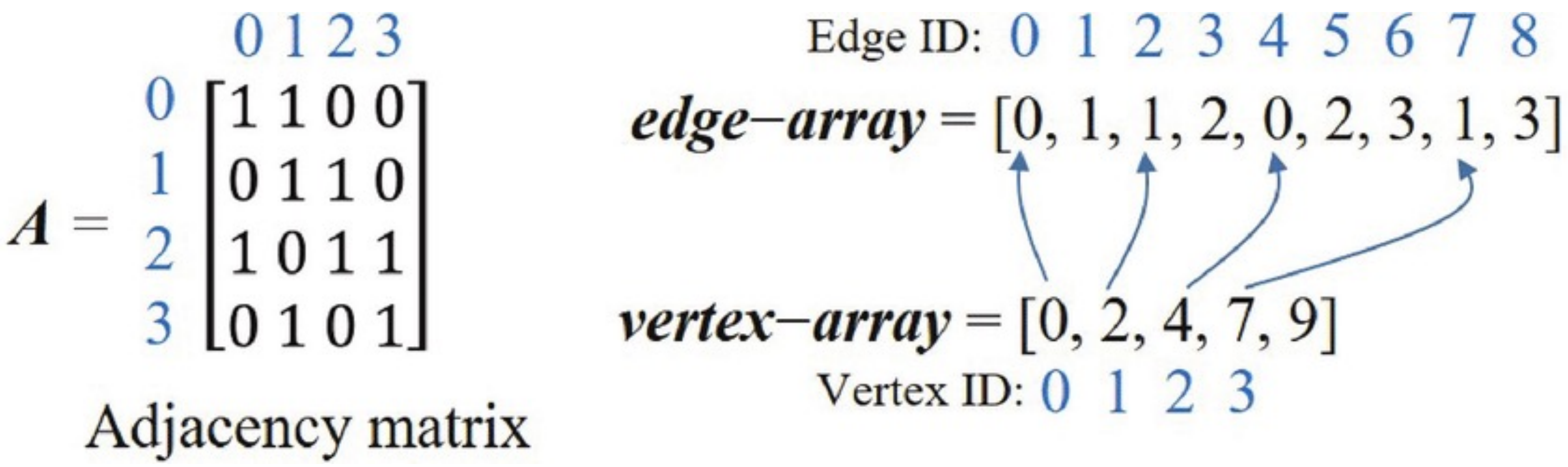


Storage space: O(|V|^2) v.s. O(|V|+|E|)

Query whether an edge (u, v) exists : O(1) v.s. O(deg(u)) → O(min(deg(u), deg(v))) → **Or better?**

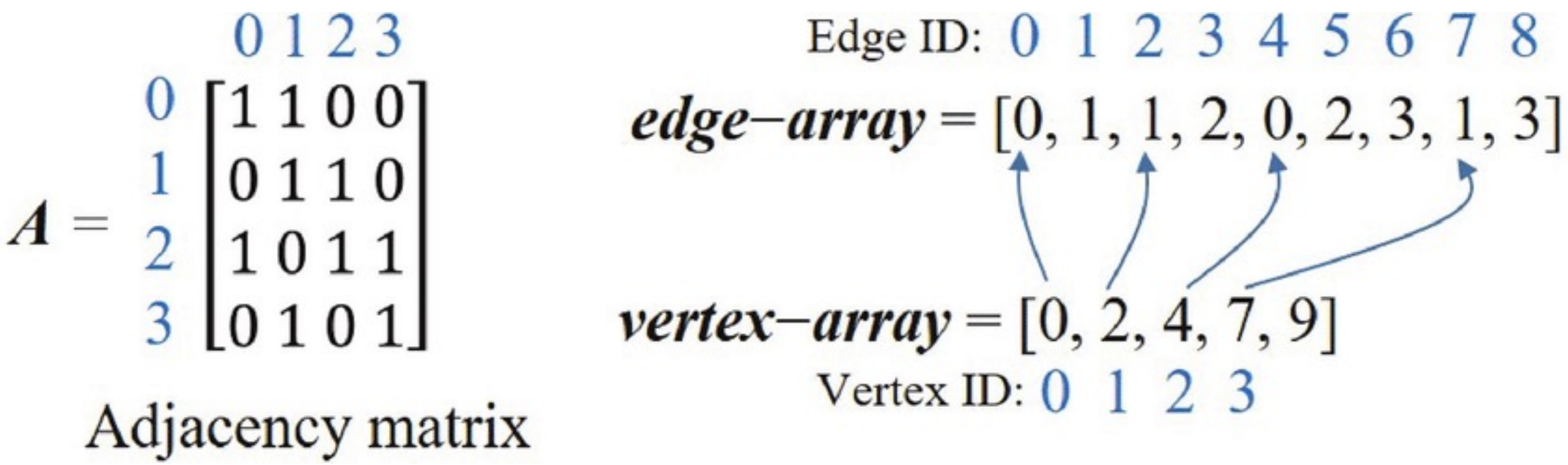
Get all the neighbors of a vertex v: O(|V|) v.s. O(deg(v))

Compressed Sparse Row (CSR)



Compressed Sparse Row (CSR)

Implement CSR in Python



Space-efficient for sparse graphs

Efficient for static graphs

Hard to update

Adjacency list VS CSR

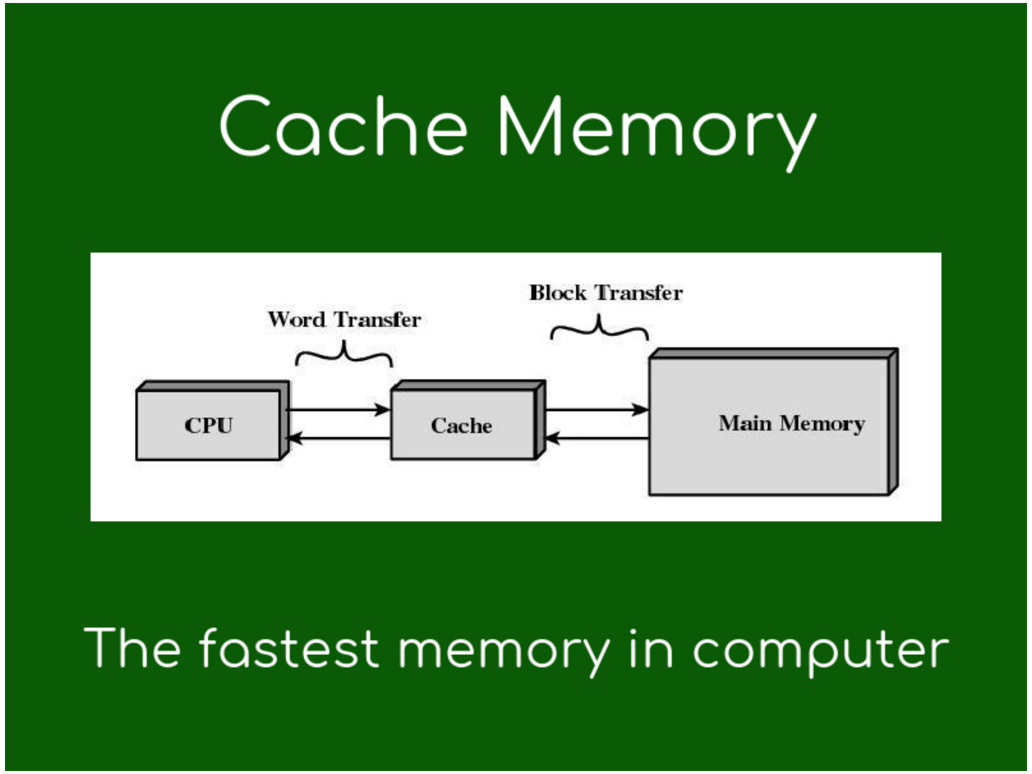
Which one is better?

Adjacency list VS CSR

Which one is better?

How does CPU read data?

Handing graph updates using adjacency list or CSR?



A quick summary for graph storage

Adjacency matrix: **best for updating**, worst for neighbor scanning, **worst for big graphs**

Adjacency list: good for neighbor scanning, ok for updating

CSR: **best for neighbor scanning**, **worst for updating**

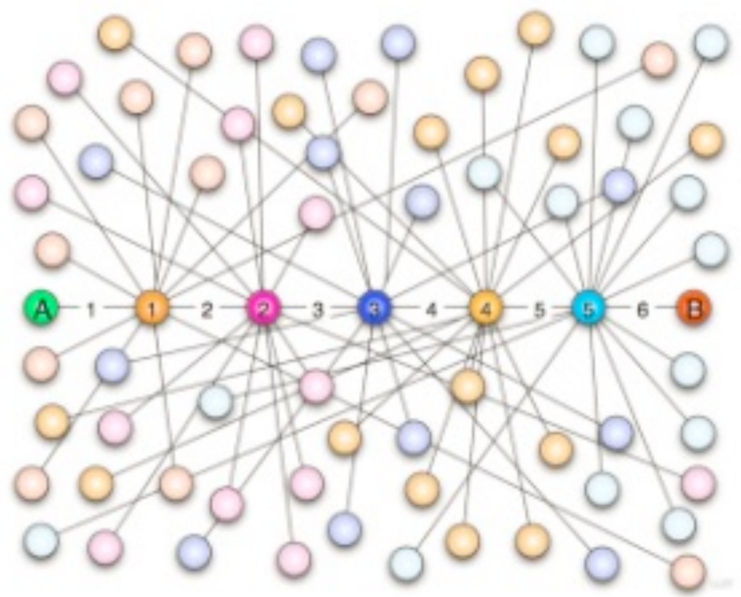
Other structure to store neighbors of each vertex:

- Hash Map: hard to choose bucket number
  - large bucket number: bad for neighbor scanning, good for updating, bad for big graphs
  - small bucket number: good for neighbor scanning, bad for updating, good for big graphs
- Binary search tree: ok for updating

Analyzing Graphs

Searching

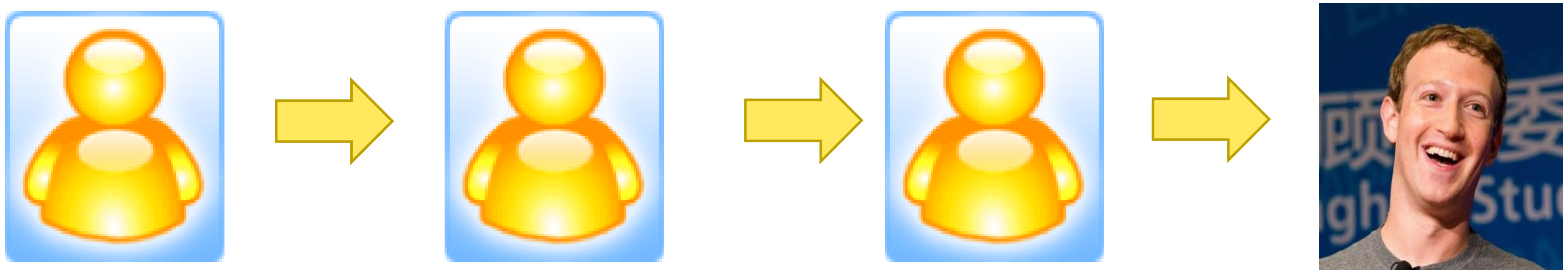
- ❑ Find the (shortest) path between two people on Facebook.



- ❑ Six degree of separation
  : only six hops separate any two people in the world

- ❑ Can you find **a path (visible by public)** between you and Mark Zuckerberg with at least **3** hops in Facebook ?

You



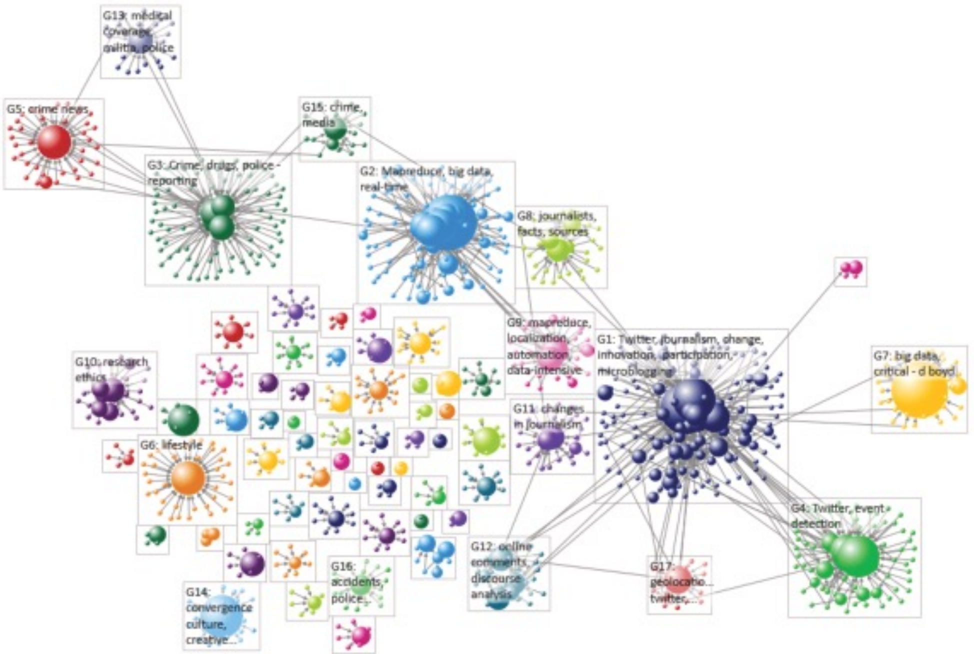
Ranking

Importance of the vertices.

E.g., which is the most influential paper in a co-citation network

https://en.wikipedia.org/wiki/PageRank

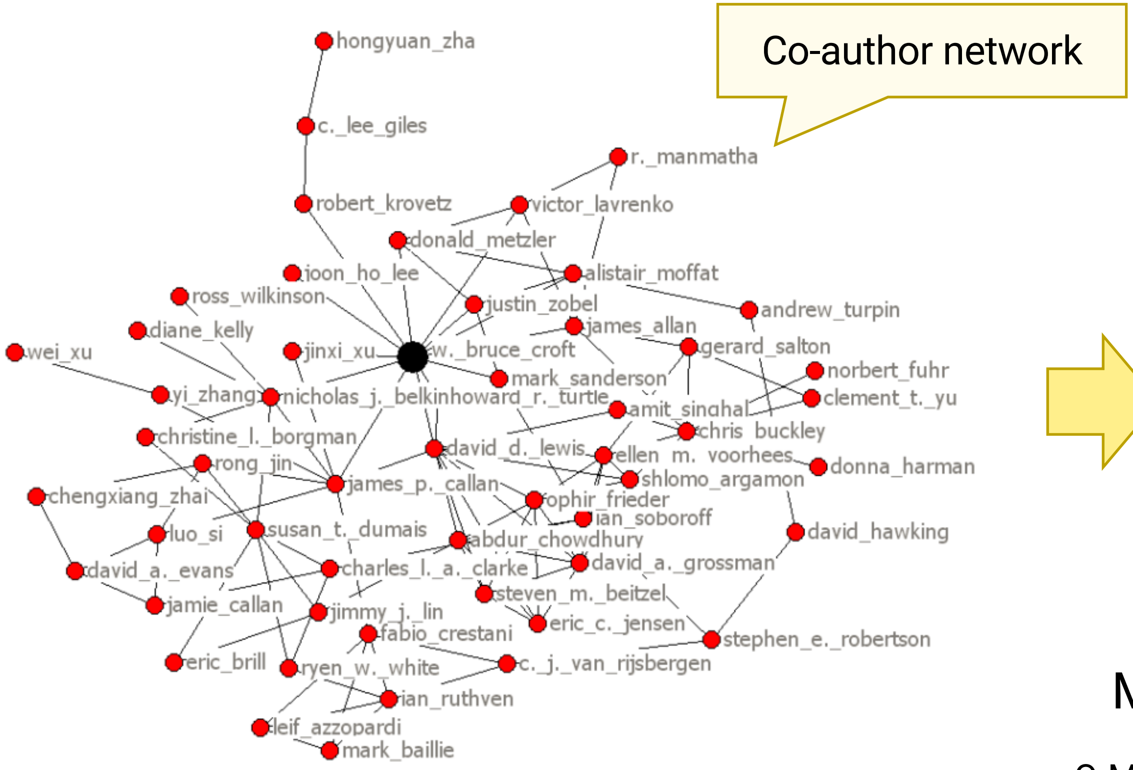
Back Mirror : Nosedive (2016)

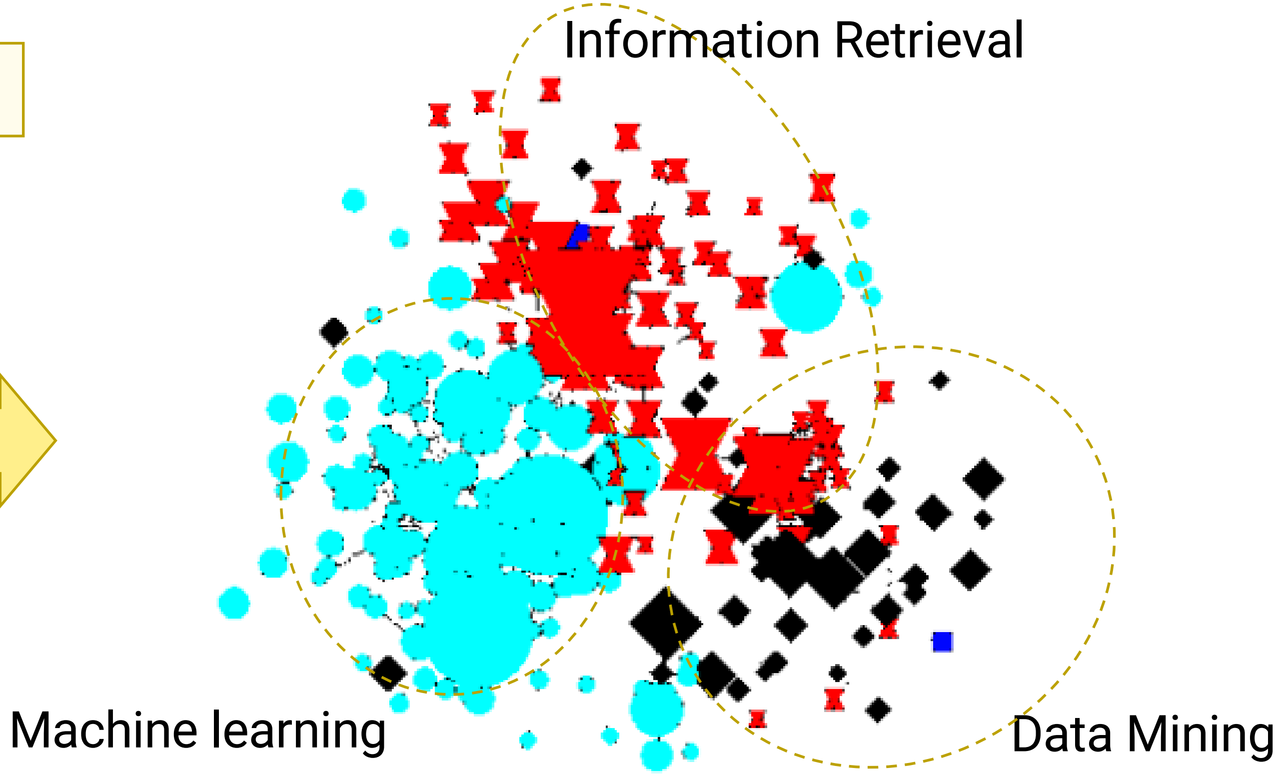


https://www.researchgate.net/

Finding Communities

Who tend to work together

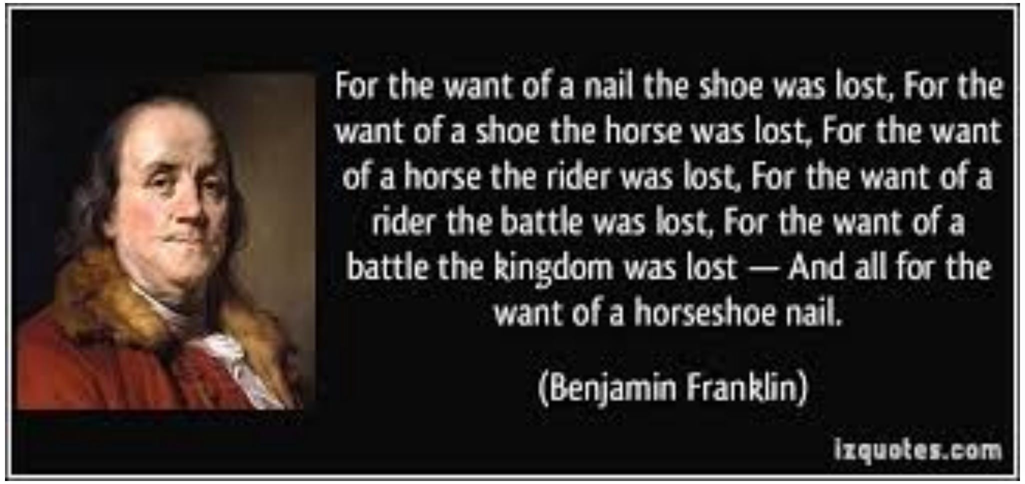


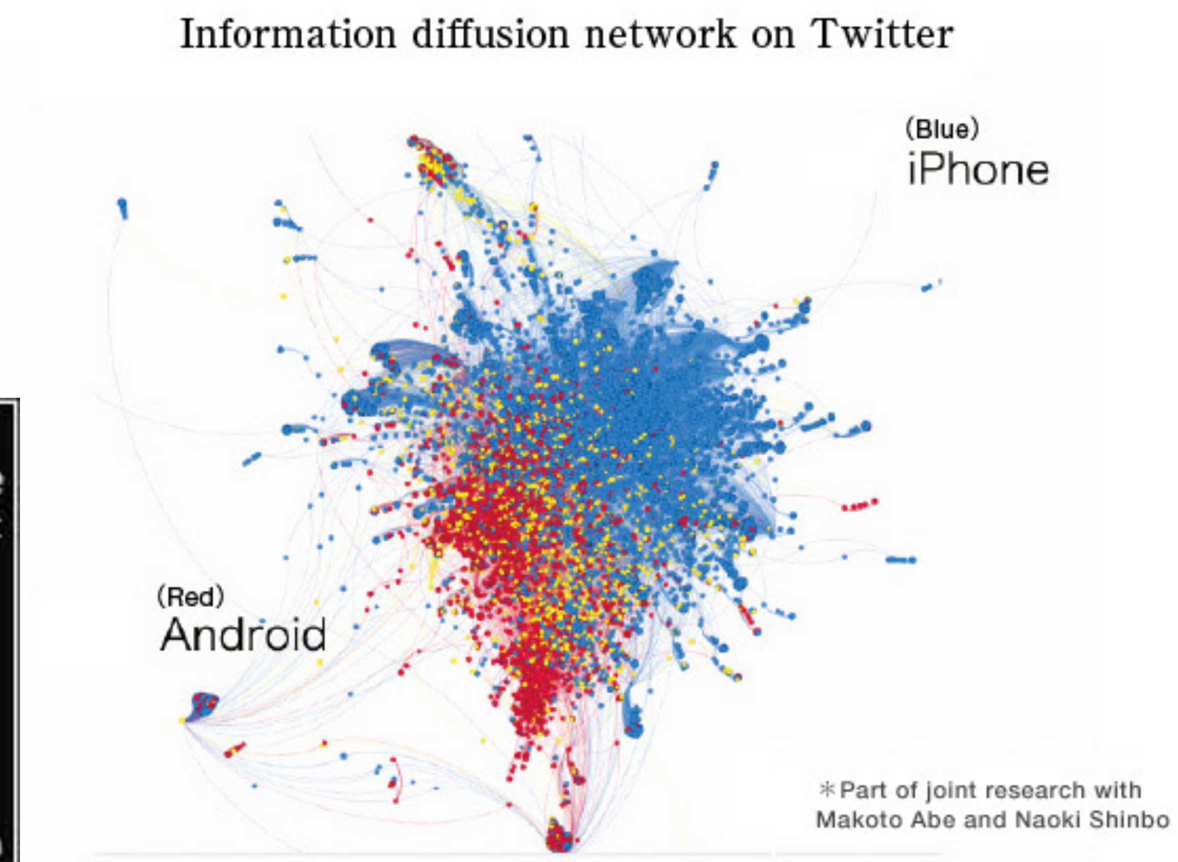


- Q.Mei, D.Cai, D.Zhang, and C.Zhai, Topic Modeling with Hitting Time, WWW 2008

Network Dynamic Analysis

Cascade behaviour from node to node like an epidemic,
– Marketing, online advertising
– News, opinions, rumours, virus
– Adoption of innovation
– Joining a community, buying a book

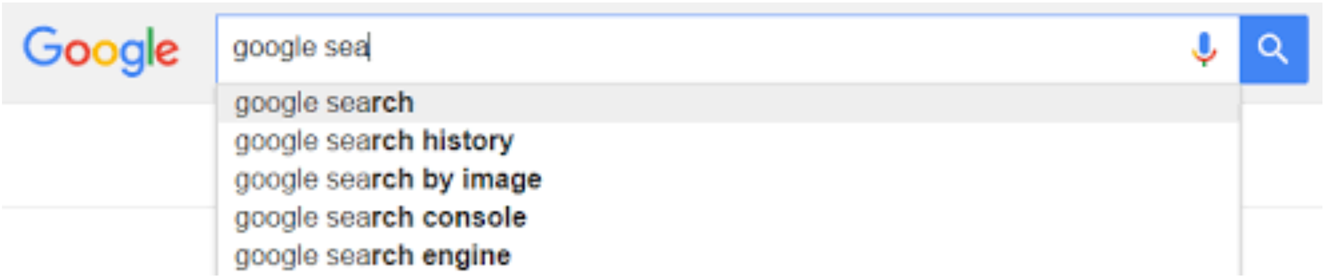




Applications of Graphs

Web search

Yahoo directory



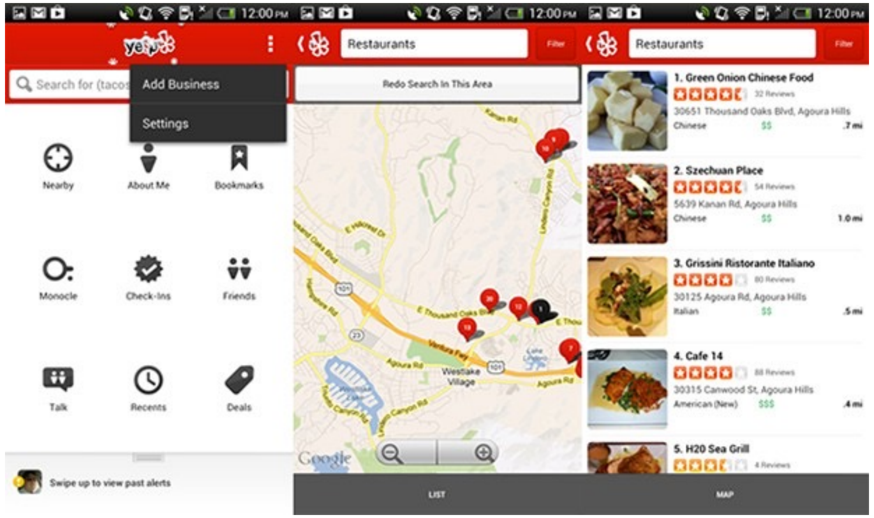
Social search

Social Search is **an enhanced version of web search,** also takes into account **social relationships** between the results and the searcher, such as work for the same companies, belong to the same social groups etc.

Facebook graph search

Recommendation

http://www.kis.kansai-u.ac.jp/res_music_e.html



Learning outcomes

Understand the basic graph structure and how to represent a graph

Know about characterization of graphs

Know about different types of graphs