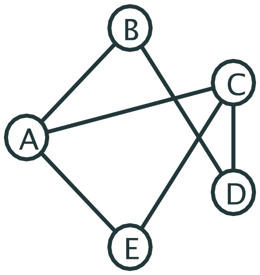
Graph ADT

Lecture 03.01
By Marina Barsky

[What is a graph?]



A graph *G* = (*V, E*) is an Abstract Data Type that consists of 2 sets:

- Set of objects (*vertices*, nodes)

  V = {A, B, C, D, E}

- Relation on set of objects (*edges*)

  E = {(A,B), (A,C), (A,E), (B.D), (C,D), (C,E)}

Running time of Graph algorithms uses **two** numbers:

- *n* = |V|
- *m* = |E|

[Vertices and edges]

- Edge *e* connects vertices *u* and *v*
- Vertices *u* and *v* are end points of edge *e*
- Vertex *u* and edge *e* are incident
- Two edges are also called incident, if they are incident to the same vertex
- *Vertices u* and *v* are adjacent
- *Vertices u* and *v* are neighbors

- This is a dictionary for undirected graph

[The degree of a vertex]

- The degree of a vertex is the number of its incident edges.
  I.e., the degree of a vertex is the number of its neighbors

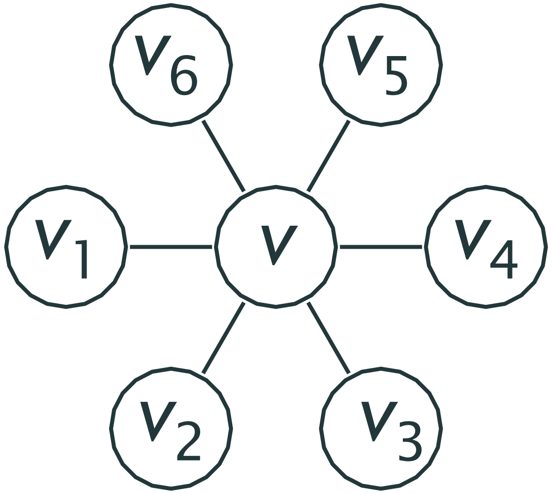
- The degree of a vertex v is denoted by deg(v)

- The degree of a graph is sum of degree of its vertices.
  The degree of undirected graph with m edges is $2m$

Example

The degree of v is 6: $\deg(v) = 6$

The degree of v_6 is 1: $\deg(v_6) = 1$



The degree of *this graph*: $\deg(G) = 2m = 12$

[Directed graphs]

Nodes: {A,B,C,D}

Edges (ordered pairs):
{(A,C),(D,A),(B,D),(C,B)}

Edges (ordered pairs):
{(C,A),(D,A),(B,D),(C,B)}

These two graphs are different

Graphs can model many things

Trivial:

- Mobile networks
- Computer networks
- Social networks

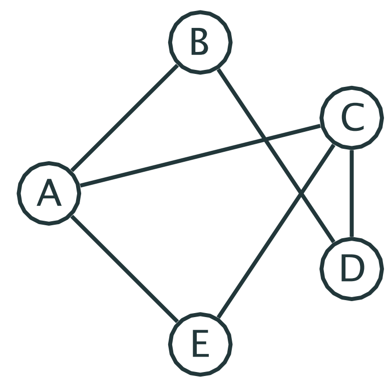
Non-trivial:

- Web pages
- States of the game
- …

Graph: airlines

Graph: airlines

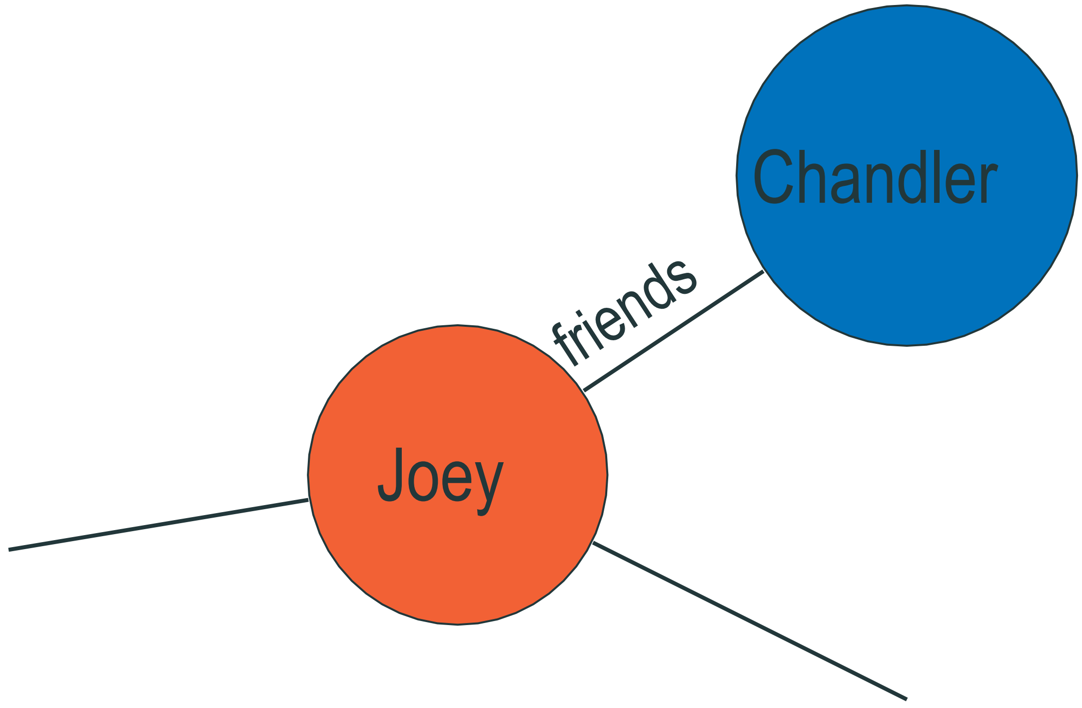
- Is there a direct flight from A to D?
- With one stop?
- With exactly two stops?



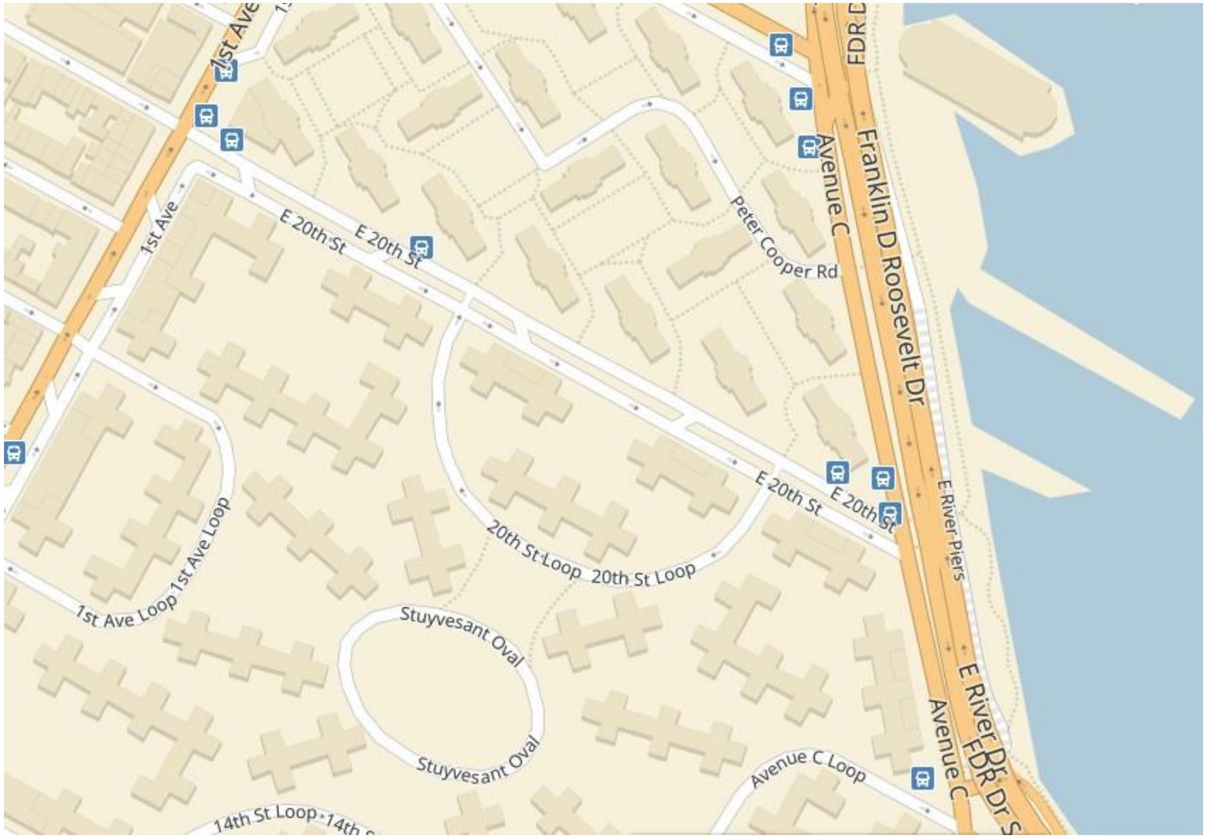
Graph of flights between 5 cities

Facebook graph

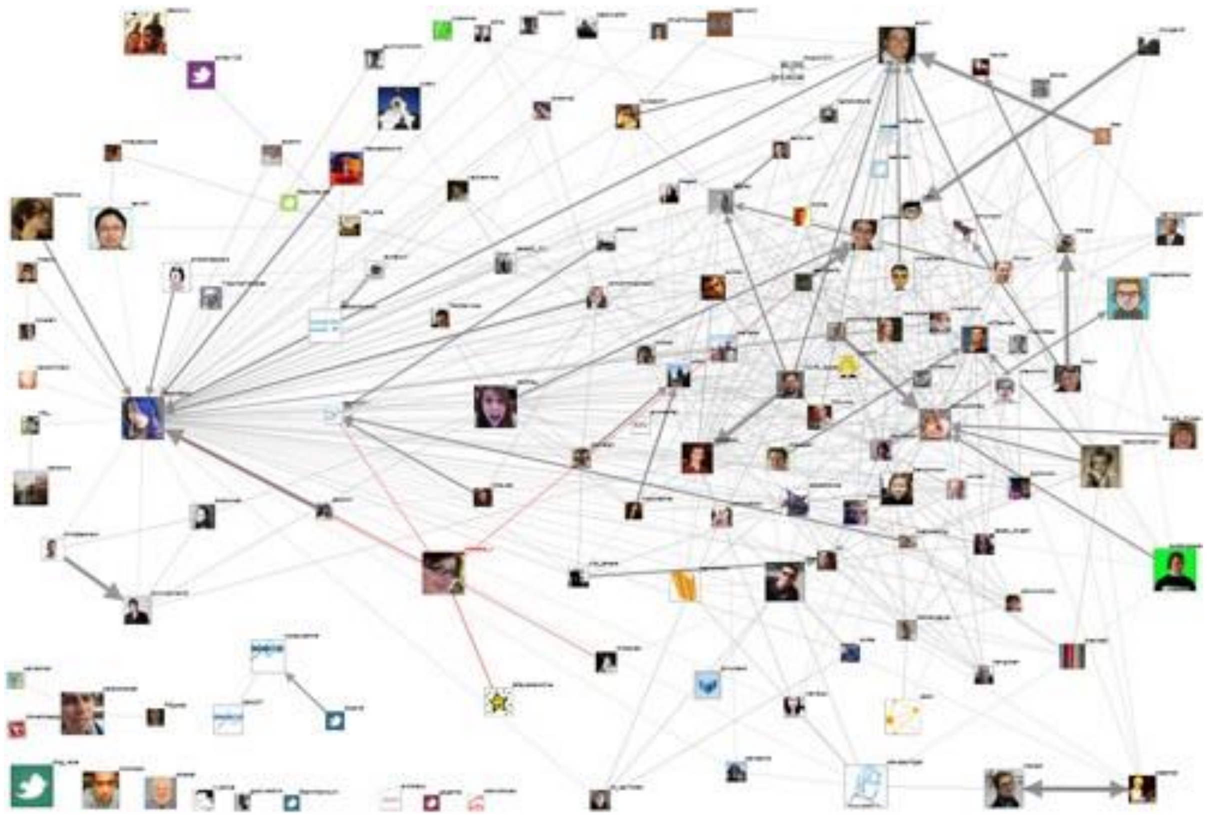
Facebook graph



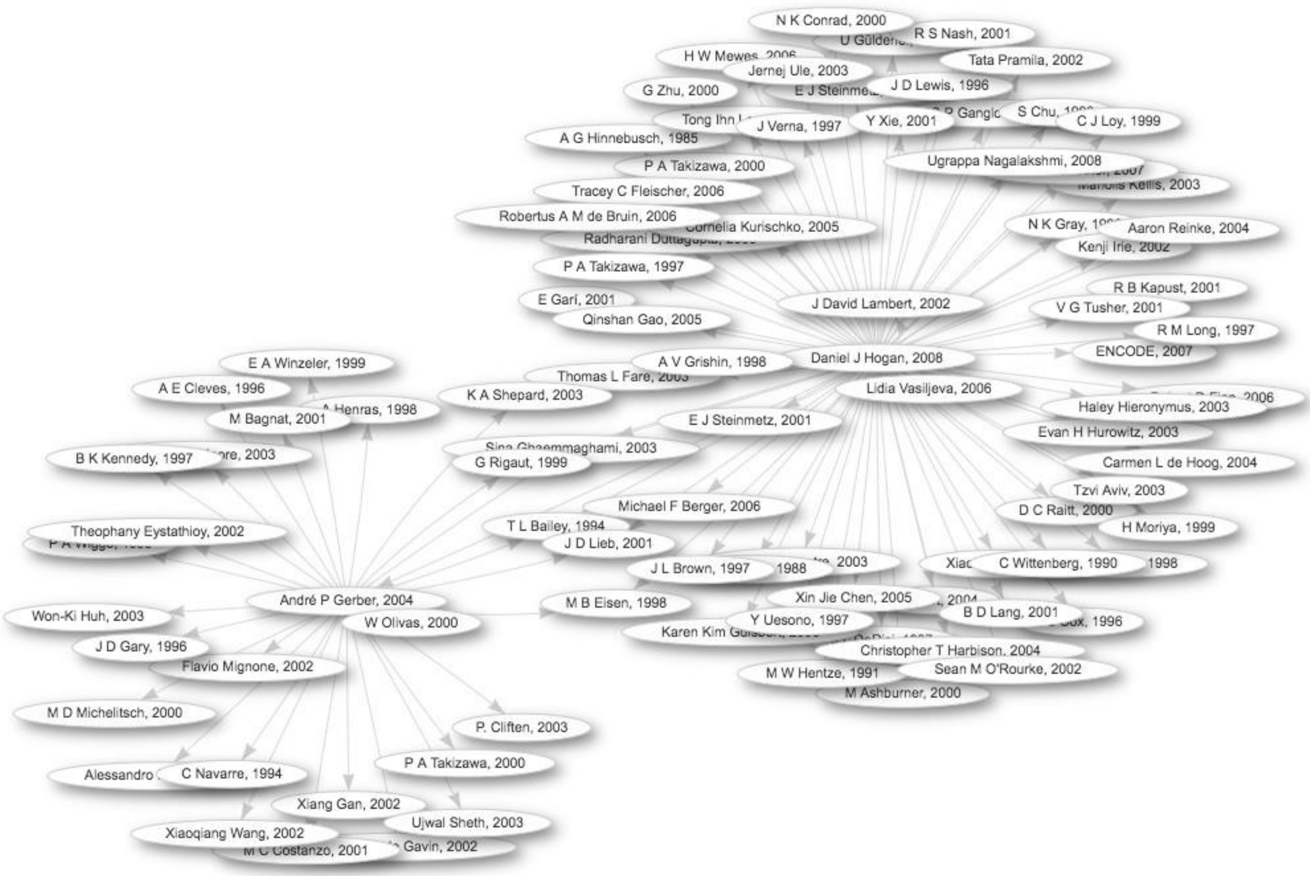
Directed graph: one-way streets



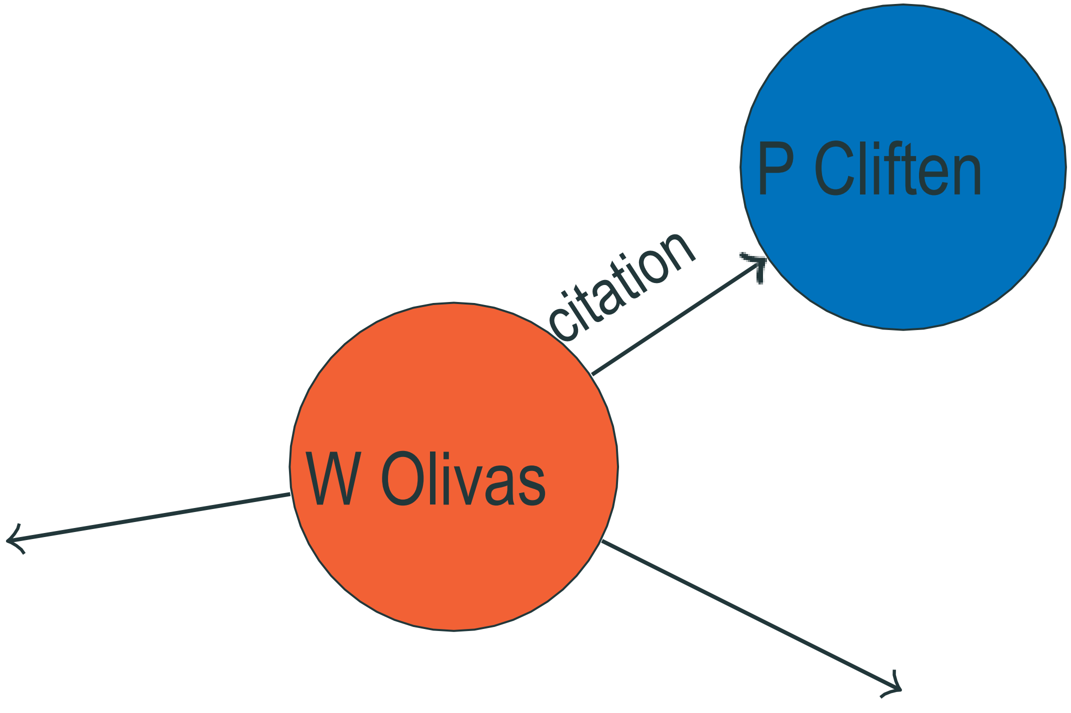
Directed graph: followers



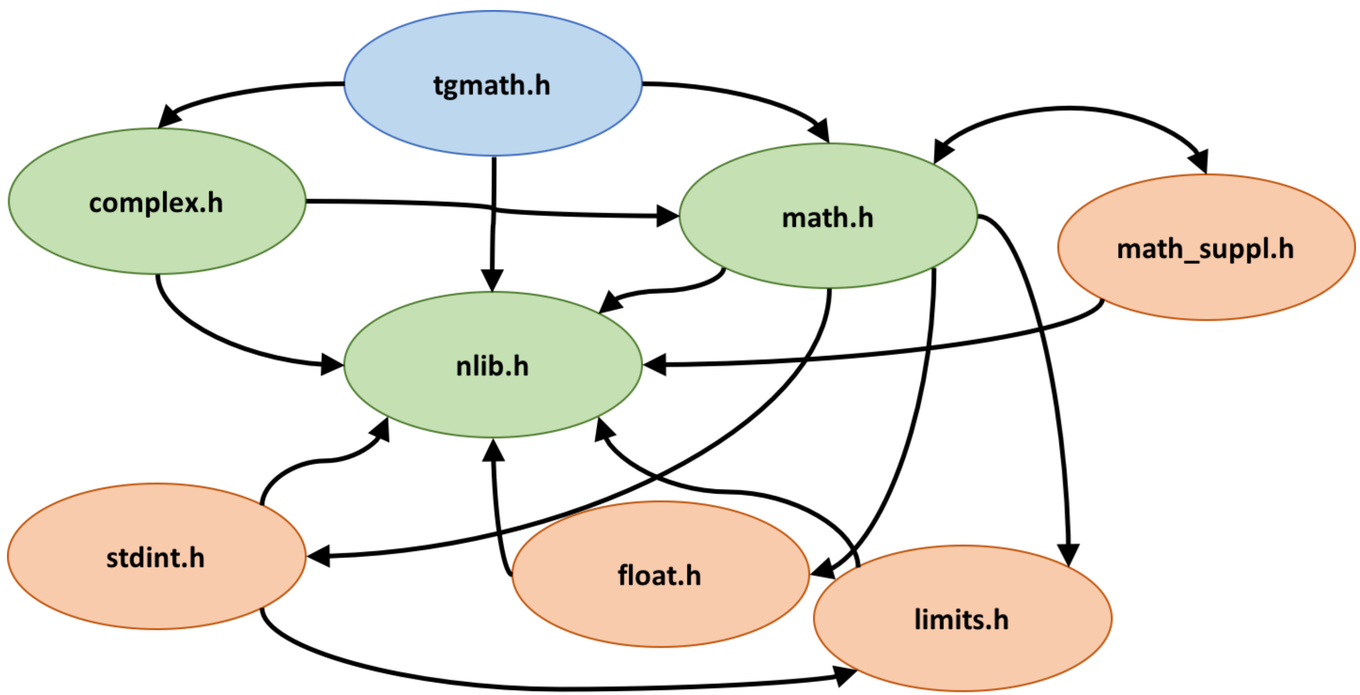
Directed graph: citations



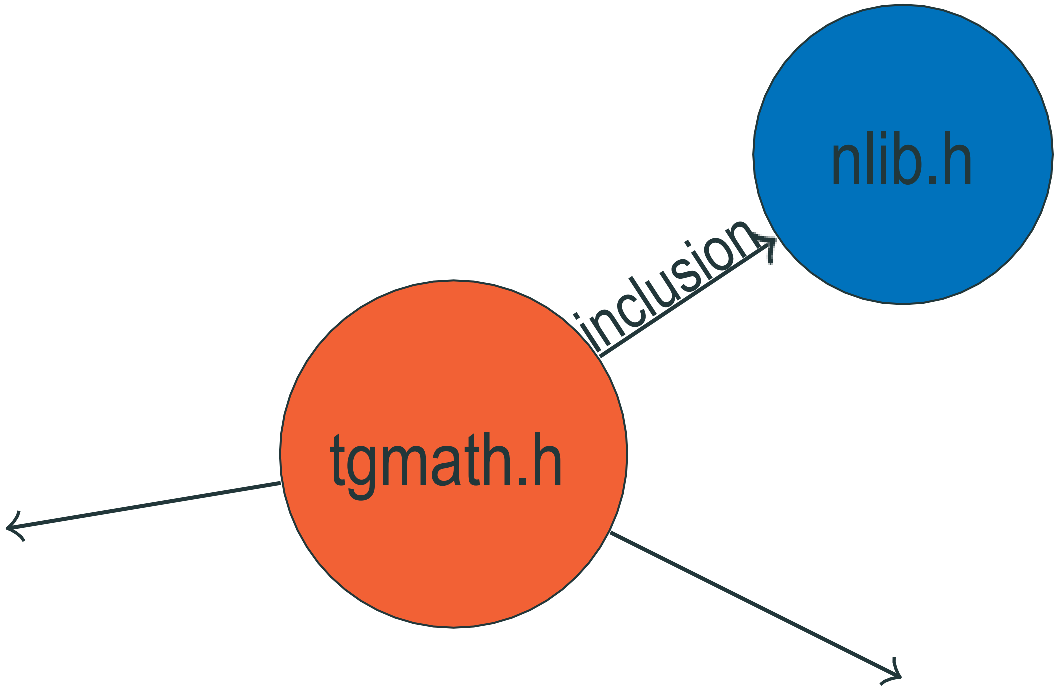
Directed graph: citations



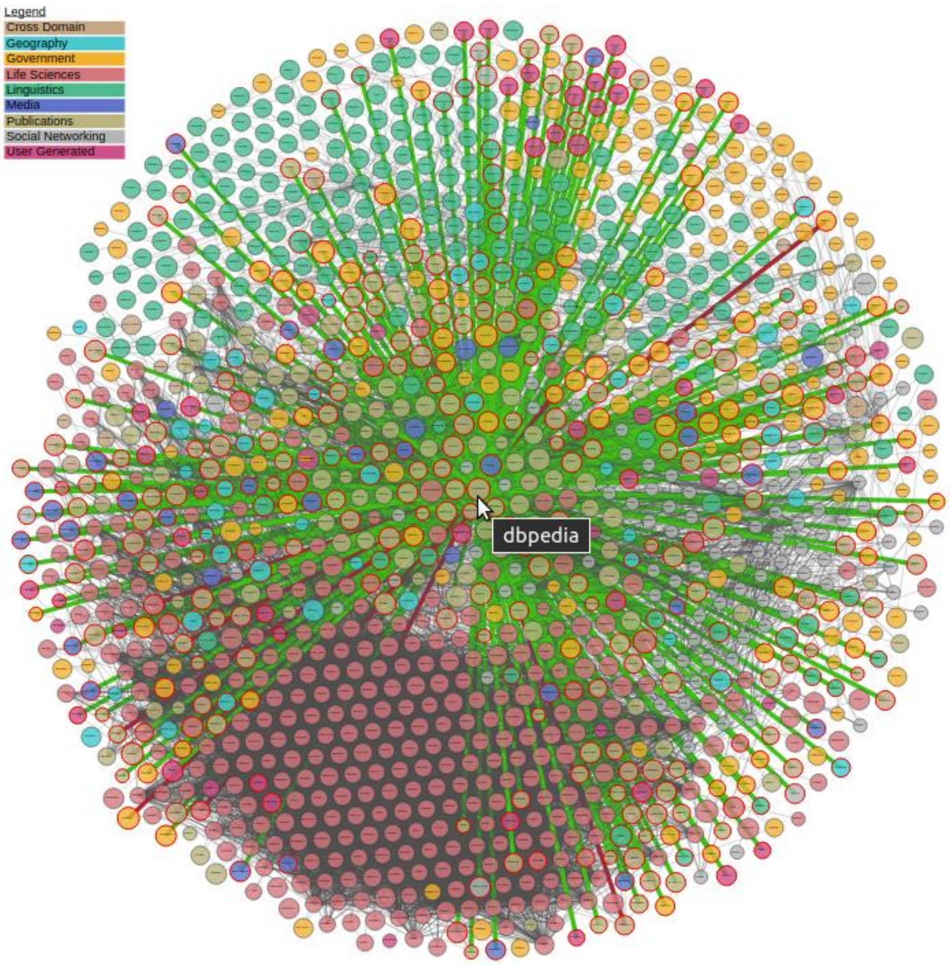
Directed graph: dependencies



Directed graph: dependencies

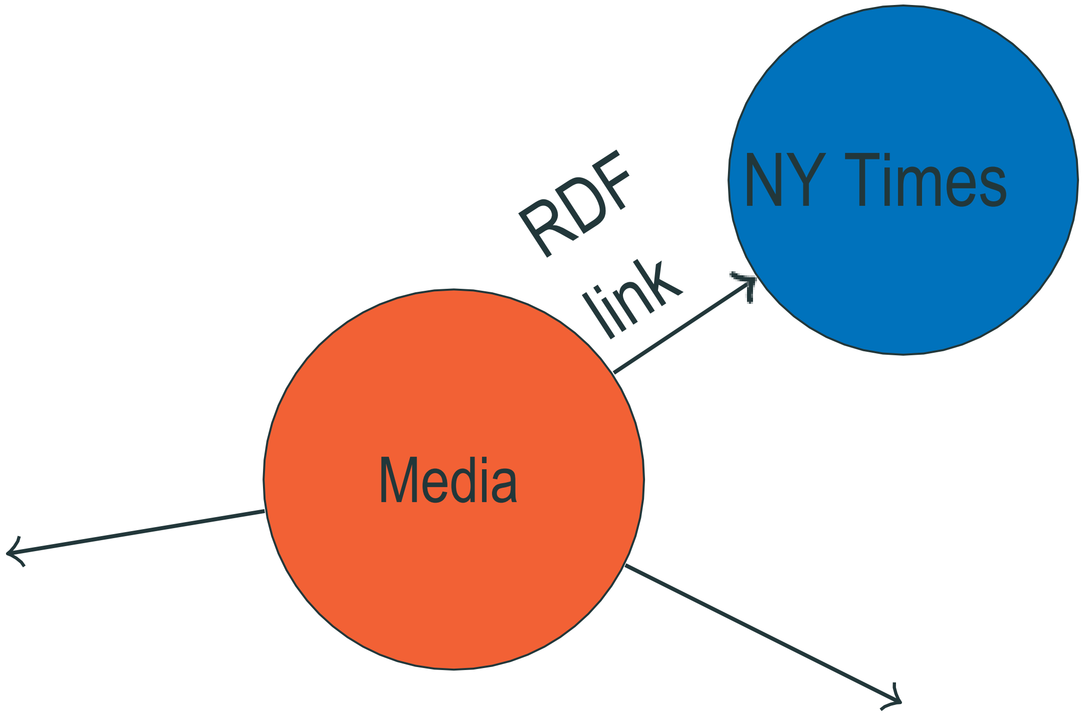


Linked Open Data Diagram

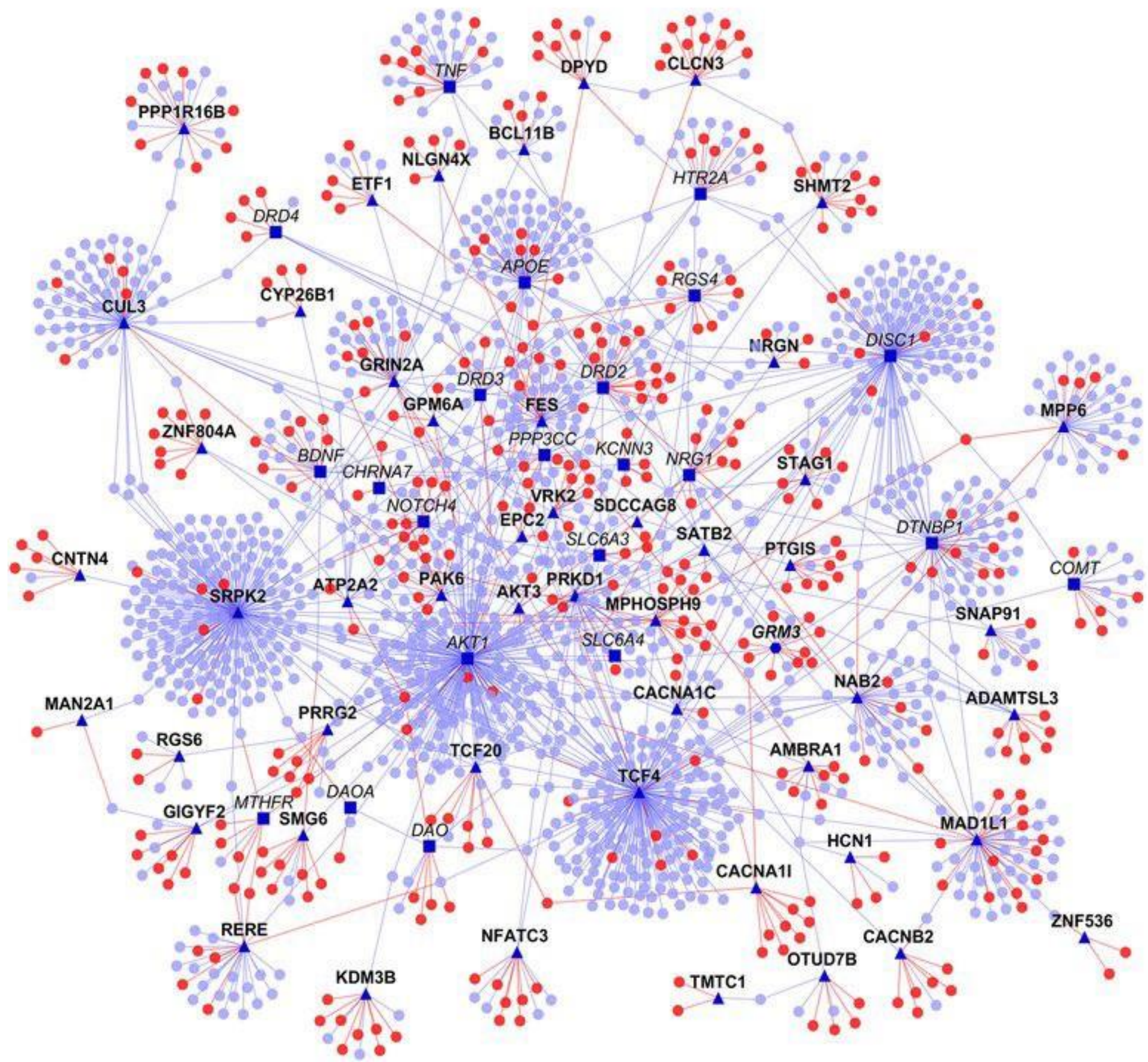


DBpedia: structured cross-domain knowledge

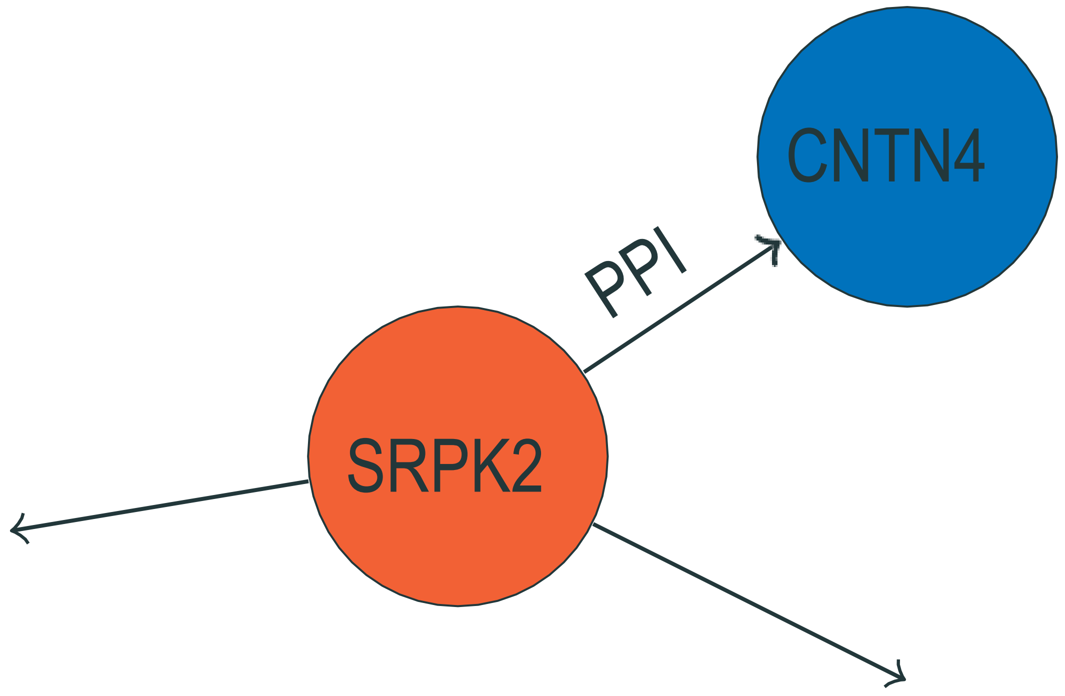
Linked Open Data Diagram



Schizophrenia Protein-Protein Interaction（PPI)

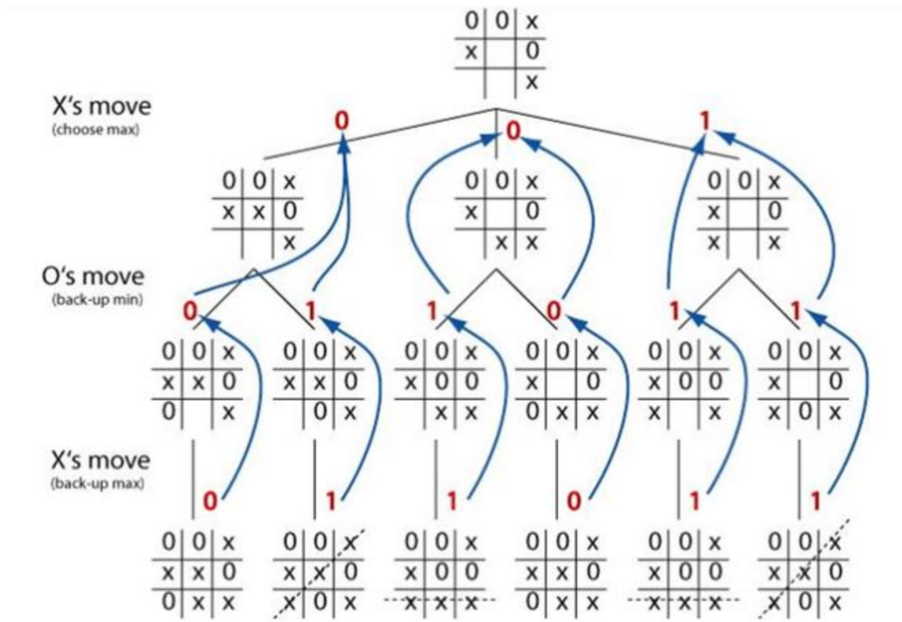


Schizophrenia Protein-Protein Interaction (PPI)

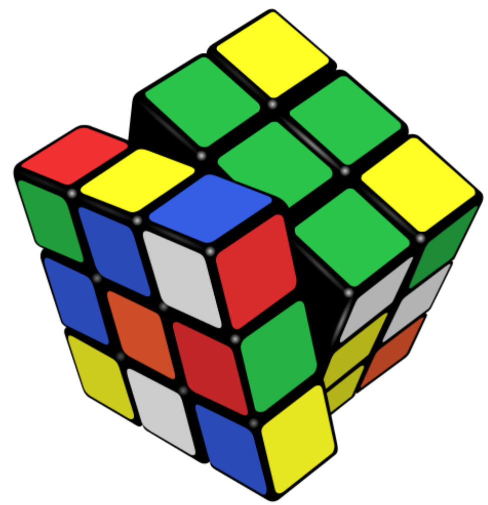


Explicit vs Implicit Graph of states

- A graph is *explicit* if all its vertices and edges are stored.
- Often we work with an *implicit graph* which is conceptual or unexplored.



There are only 3^9 = 19,683 different states in Tic-Tac-Toe. We can store the entire graph and compute the optimal strategy as a path through this graph



The Rubik's Cube has 43 quintillion states.
It can be solved without explicitly listing all vertices (states)

Example: Solving puzzles

With graphs!

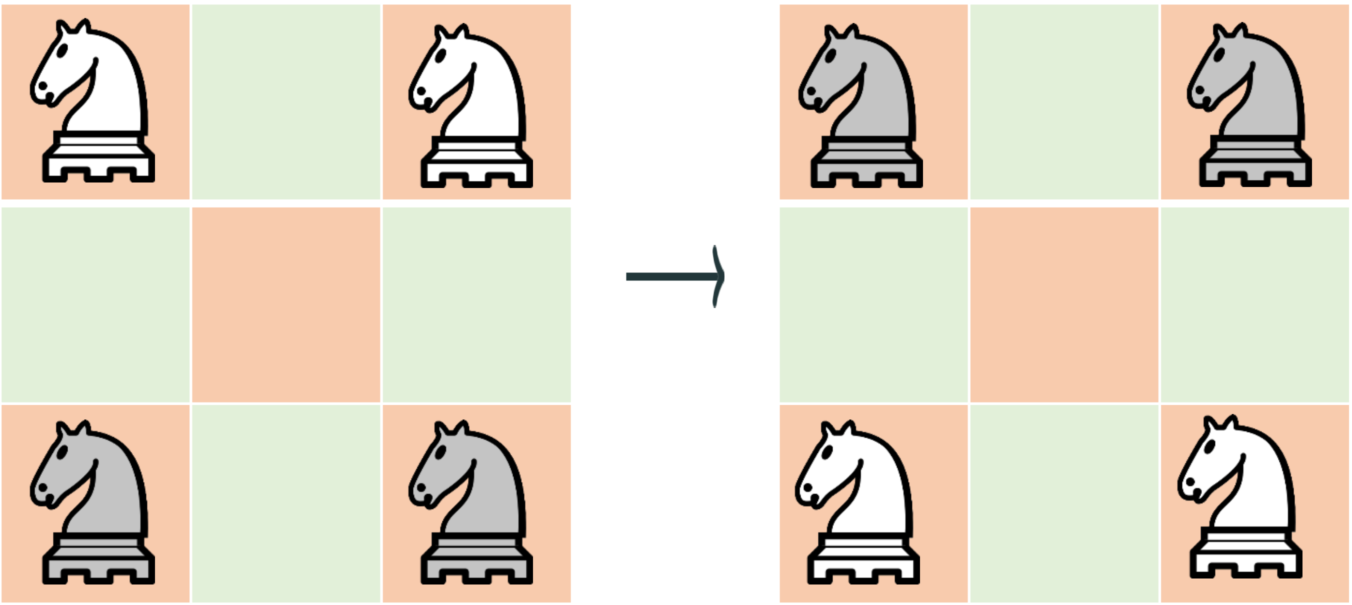
Paolo Guarini di Forli, Italy
15th - 16th Century

Try it out

http://barsky.ca/knights/

Guarini's Puzzle

Paolo Guarini
di Forli, Italy
15th - 16th
Century

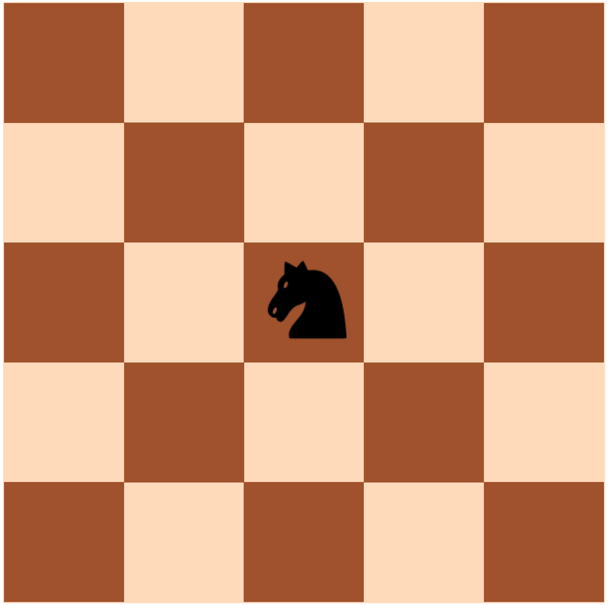


There are four knights on the 3×3 chessboard: the two white knights are at the two upper corners, and the two black knights are at the two bottom corners of the board.

The goal is to switch the knights in the minimum number of moves so that the white knights are at the bottom corners and the black knights are at the upper corners.

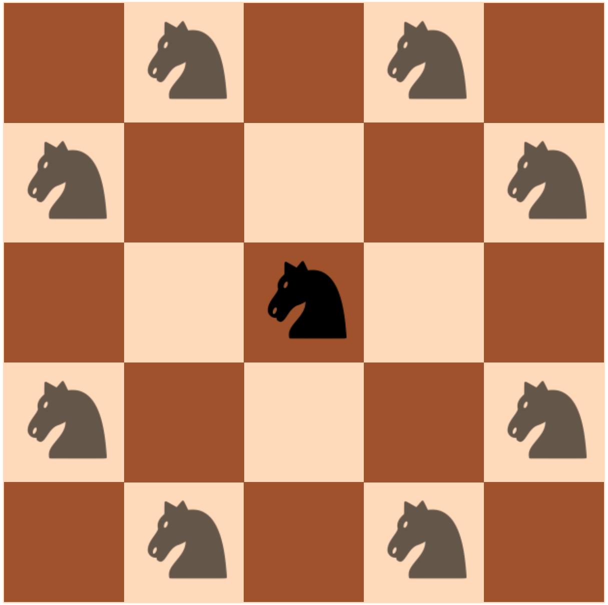
Chess Knight

A chess knight can move in an **L** shape in any direction

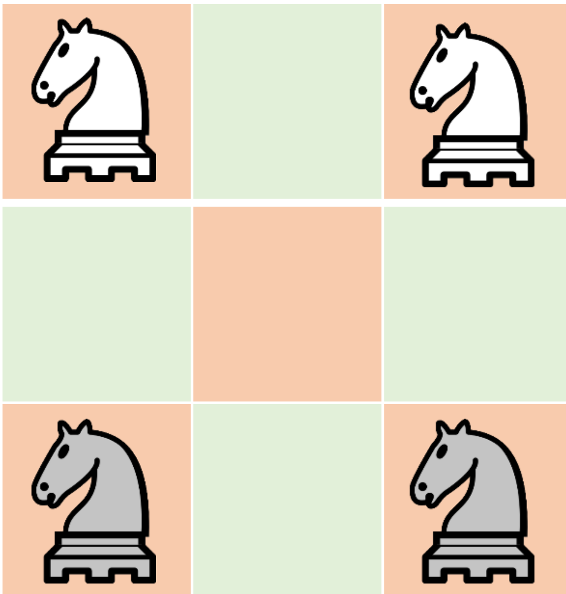


Chess Knight

A chess knight can move in an **L** shape in any direction



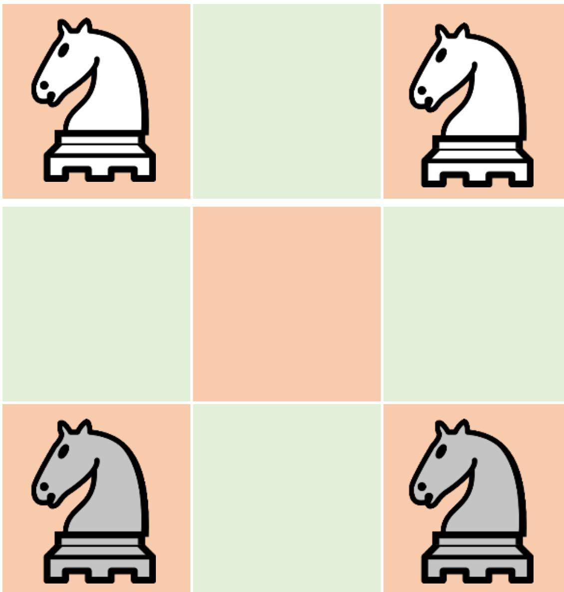
Graph: nodes

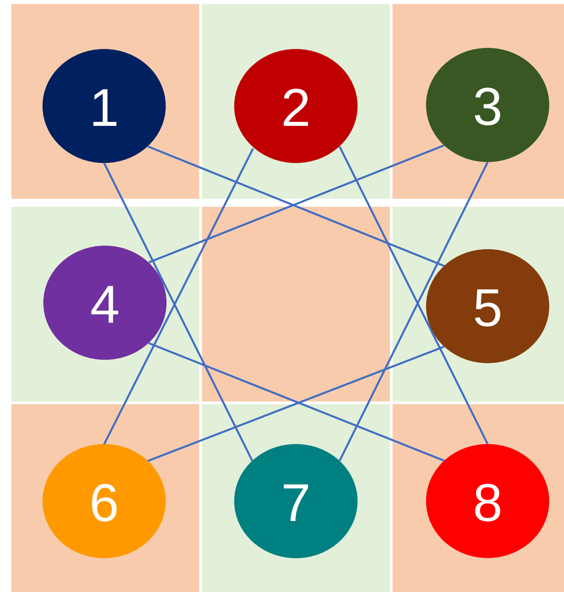


1	2	3
4		5
6	7	8

Each position is a node in a graph

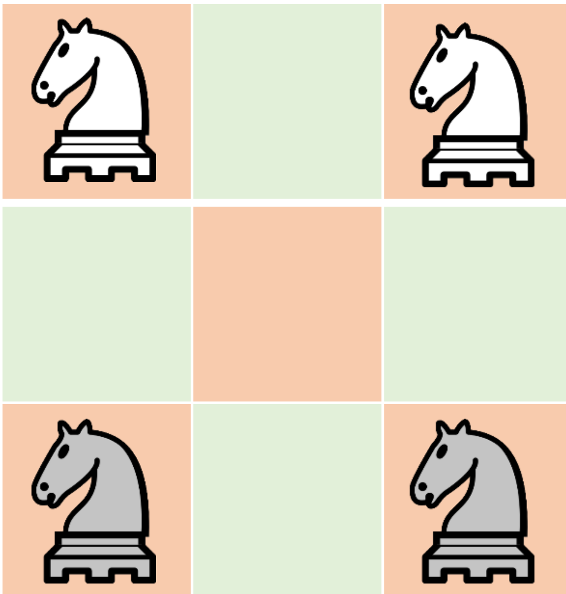
Graph: edges

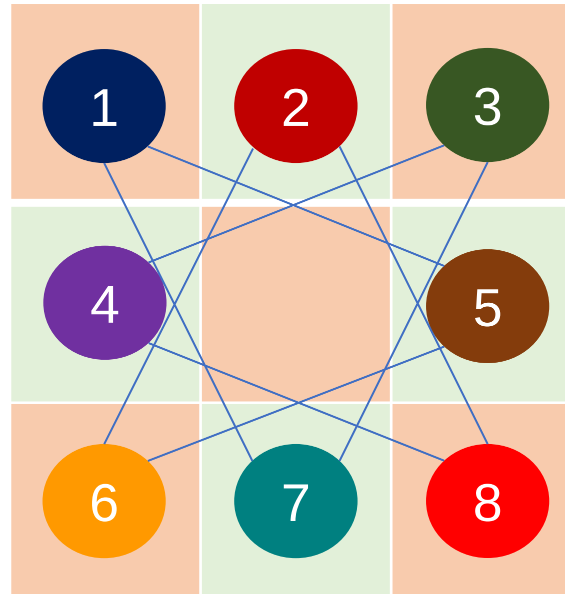




There is an edge between the nodes if you can go from 1 node to another by 1 knight move

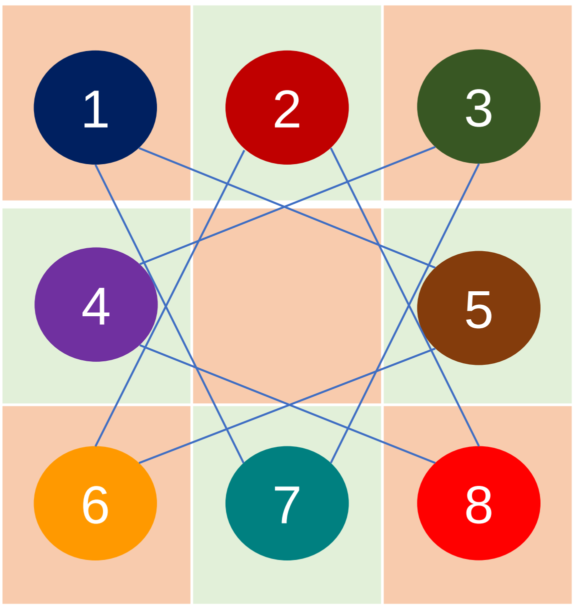
Graph: edges

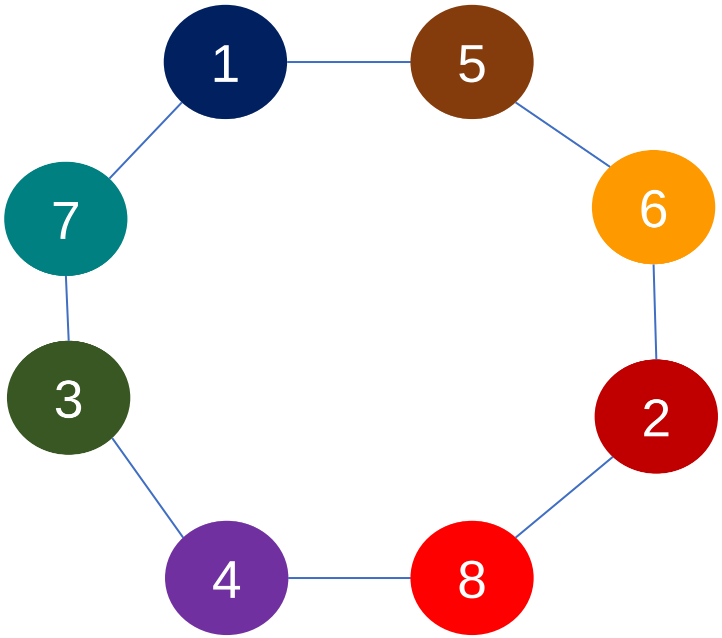




Does it help to solve the puzzle?

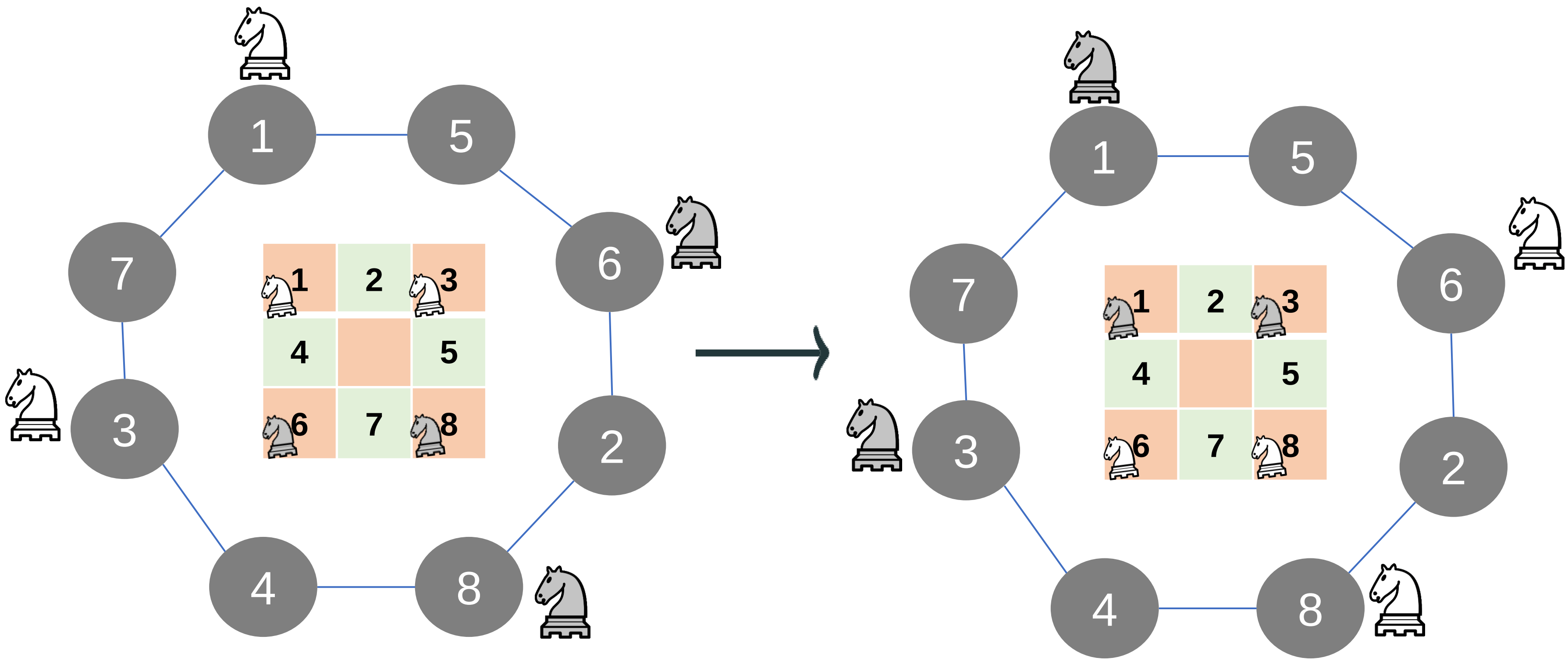
Unfold the graph!





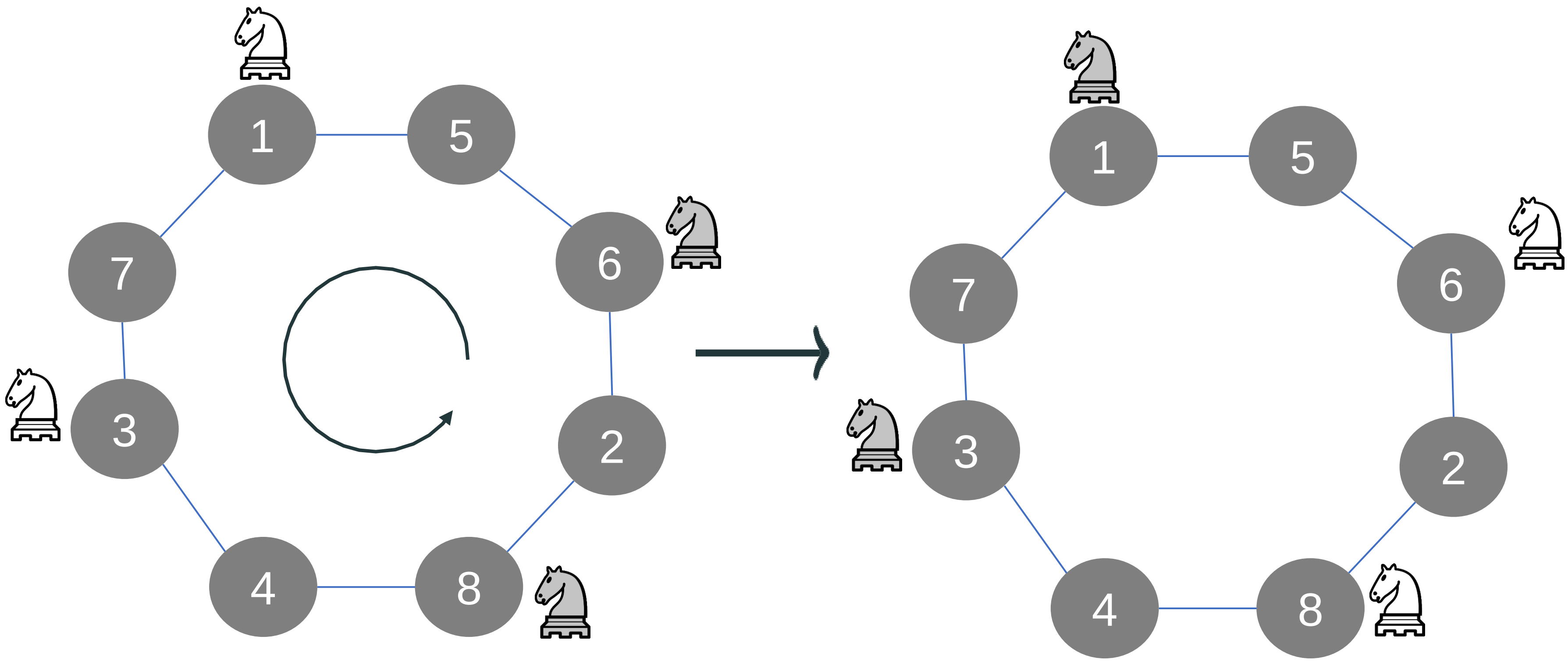
All the nodes are on a circle

Solution



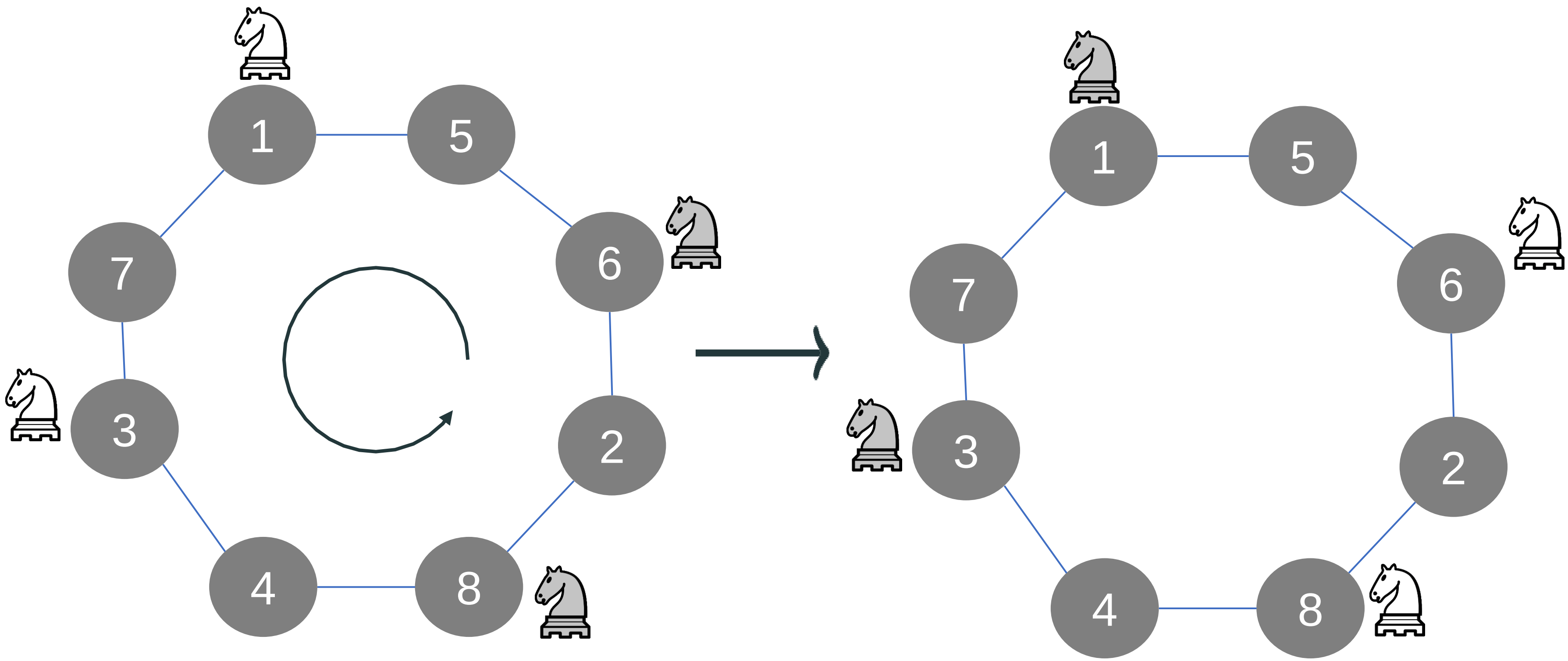
Do you see it now?

Solution



Move around the circle following legal edges

Solution

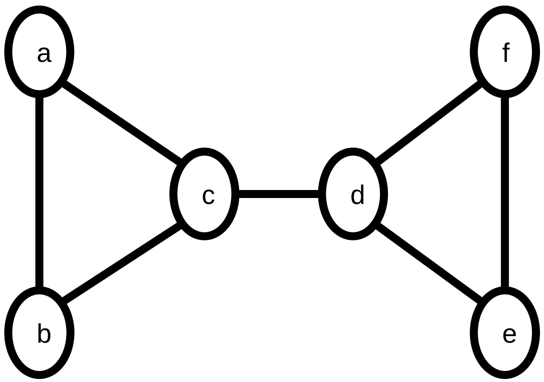


Until knights are in desired positions

Data Structures for Graph ADT

Representing Graph as Edge Set (**Edge List**)

The most straightforward way of storing graphs is to create a set of all graph vertices, and a set of all edges in form of tuples:



V = {a,b,c,d,e,f}
E = {(a,b), (a,c), (b,c), (c,d), (d,e), (d,f), (e,f)}

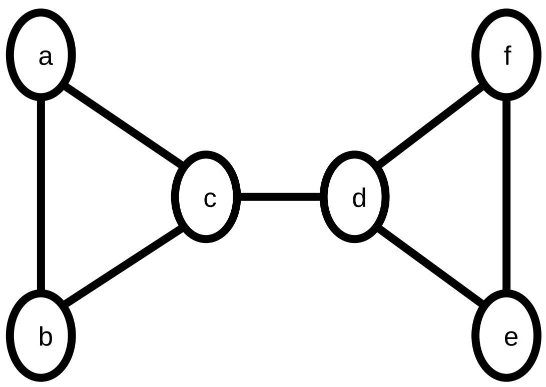
- Edge lists are simple, but if we want to find whether the graph contains a particular edge, we have to search through the edge list.
- If the edges appear in the edge list in no particular order, that's a linear search through *m* edges.

Question: How would you organize an edge list to make searching for a particular edge take O(log m) time?

Adjacency Lists and Adjacency Matrices

Graphs are commonly stored as ***adjacency lists*** or ***adjacency matrices***.

- In undirected graphs each edge is stored twice.
- Non-simple graphs use adjacency counts instead of 0/1 in the adjacency matrix.
- Non-simple graphs repeat vertices or use edge numbers in the adjacency list.



Graph

a	b, c
b	a, c
c	a, b, d
d	c, e, f
e	d, f
f	d, e

Adjacency List

	a	**b**	**c**	**d**	**e**	**f**
a	0	1	1	0	0	0
b	1	0	1	0	0	0
c	1	1	0	1	0	0
d	0	0	1	0	1	1
e	0	0	0	1	0	1
f	0	0	0	1	1	0

Adjacency Matrix

Efficient implementations

The data structure used to store a graph affects the efficiency of algorithms running on it.

Task	**Winner**
To test if (x,y) is in graph?	
Find a degree of a vertex	
Store a sparse graph: m = O(n)	
Store a dense graph: m = O(n^2)	
Insert/delete an edge	
Traverse the graph	
Most problems	

$n = |V|$, $m = |E|$

Efficient Representation

The data structure used to store a graph affects the efficiency of algorithms running on it.

Task	**Winner**
To test if (x,y) is in graph?	Adj. matrix O(1)
Find a degree of a vertex	Adj. list O(d) vs. O(n)
Store a sparse graph: m = O(n)	Adj. list (n + m) vs. n^2
Store a dense graph: m = O(n^2)	Adj. matrix (save on links)
Insert/delete an edge	Adj. matrix O(1) vs. O(d)
Traverse the graph	Adj. list (n + m) vs. n^2
Most problems	Adj. list