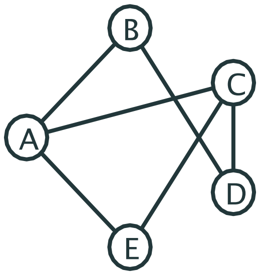
Modeling using Graph ADT

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|

Graphs can model many things

Trivial:

- Mobile networks
- Computer networks
- Social networks

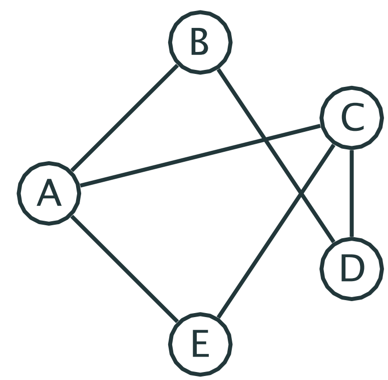
Non-trivial:

- Web pages
- States of the game
- …

Graph: airlines

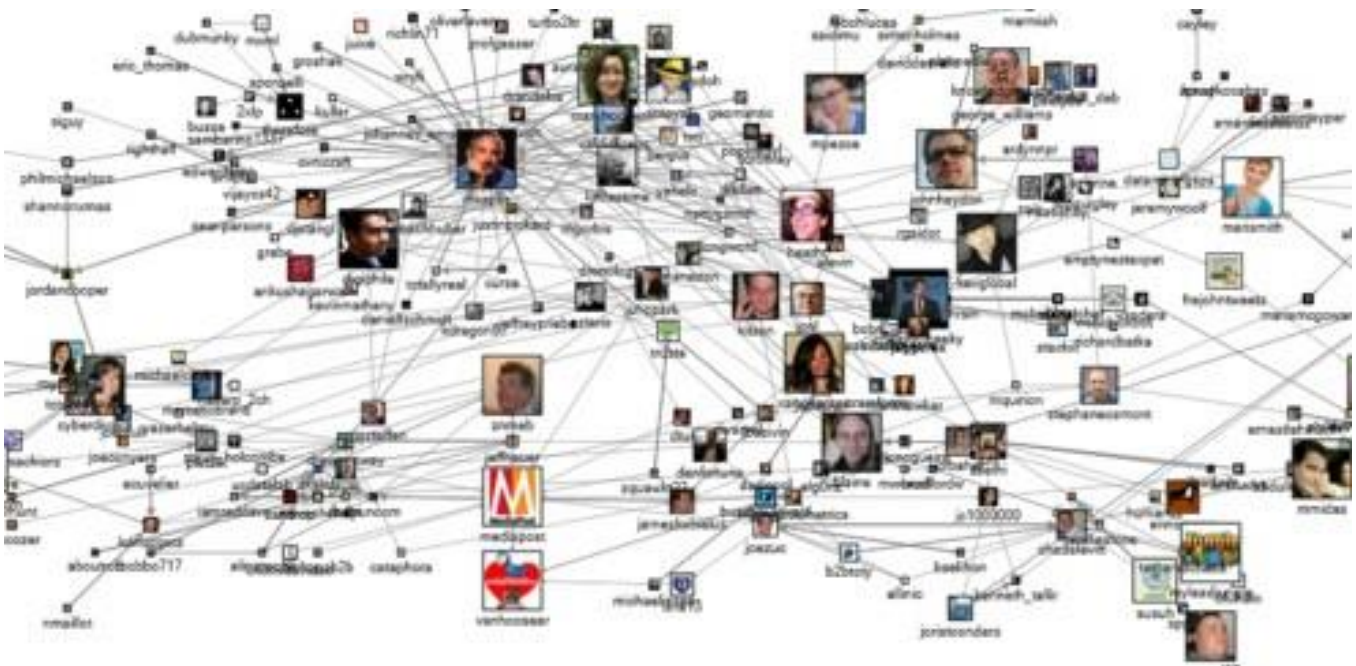
Graph: airlines

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

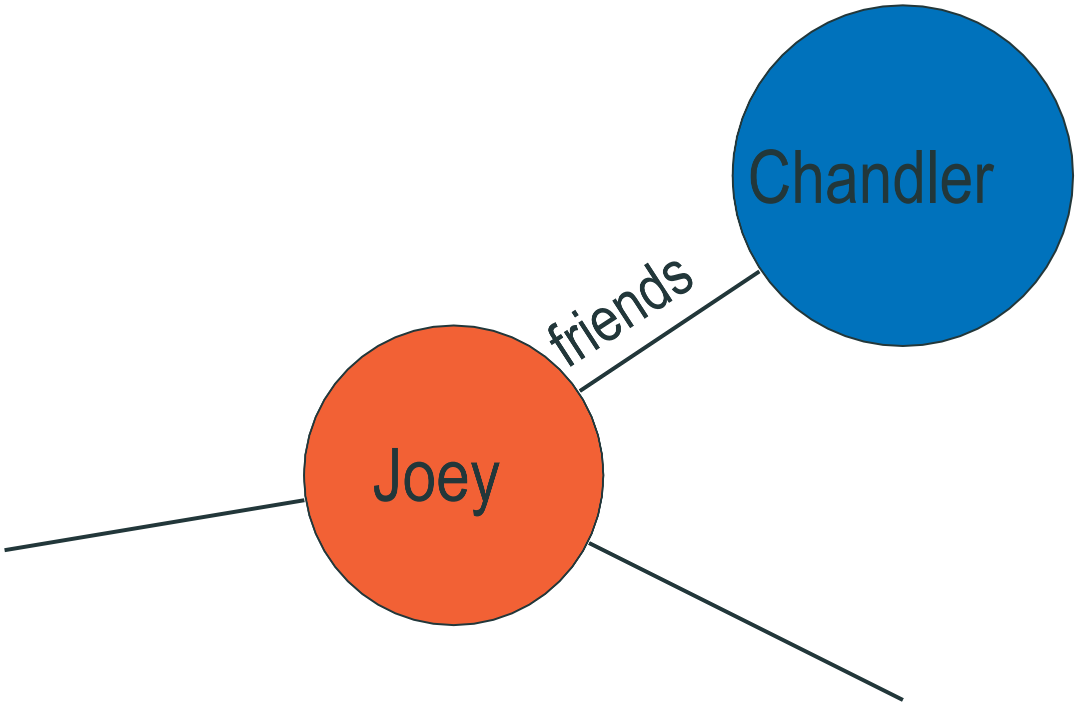


Graph of flights between 5 cities

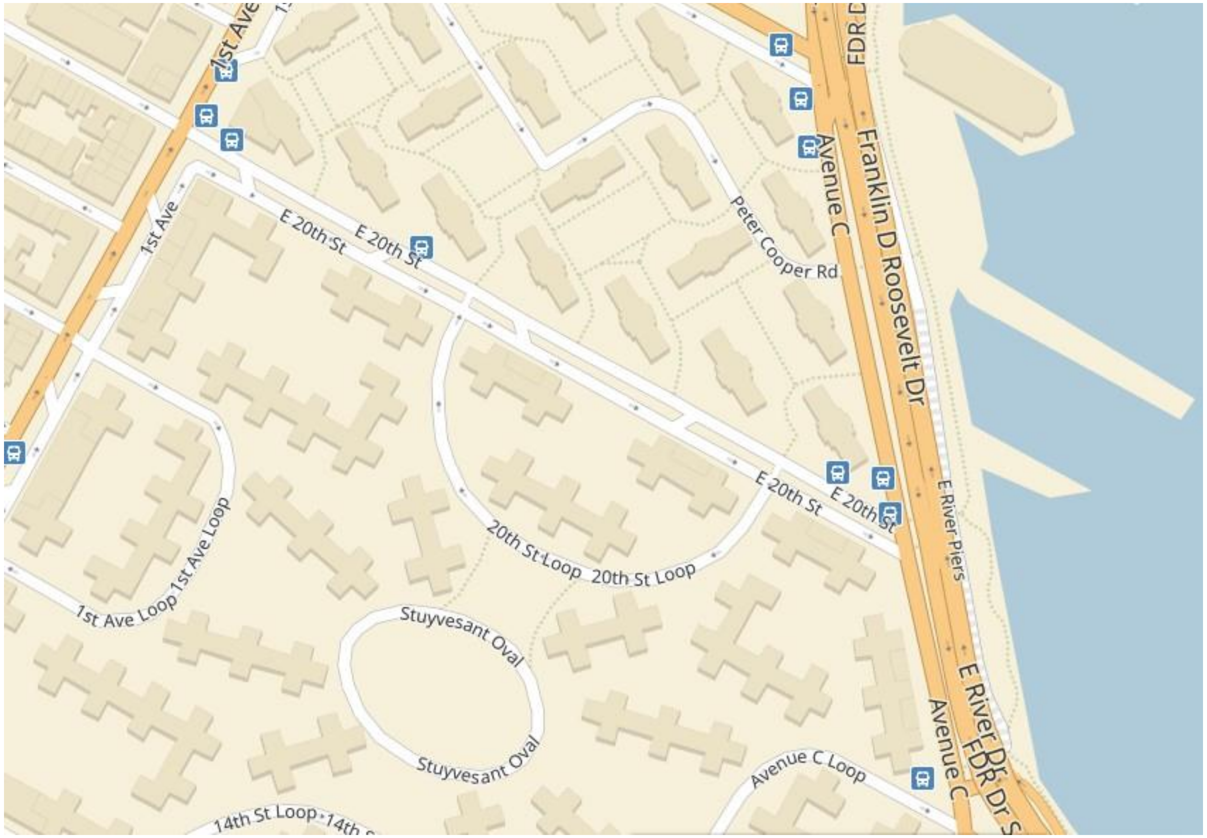
Facebook graph



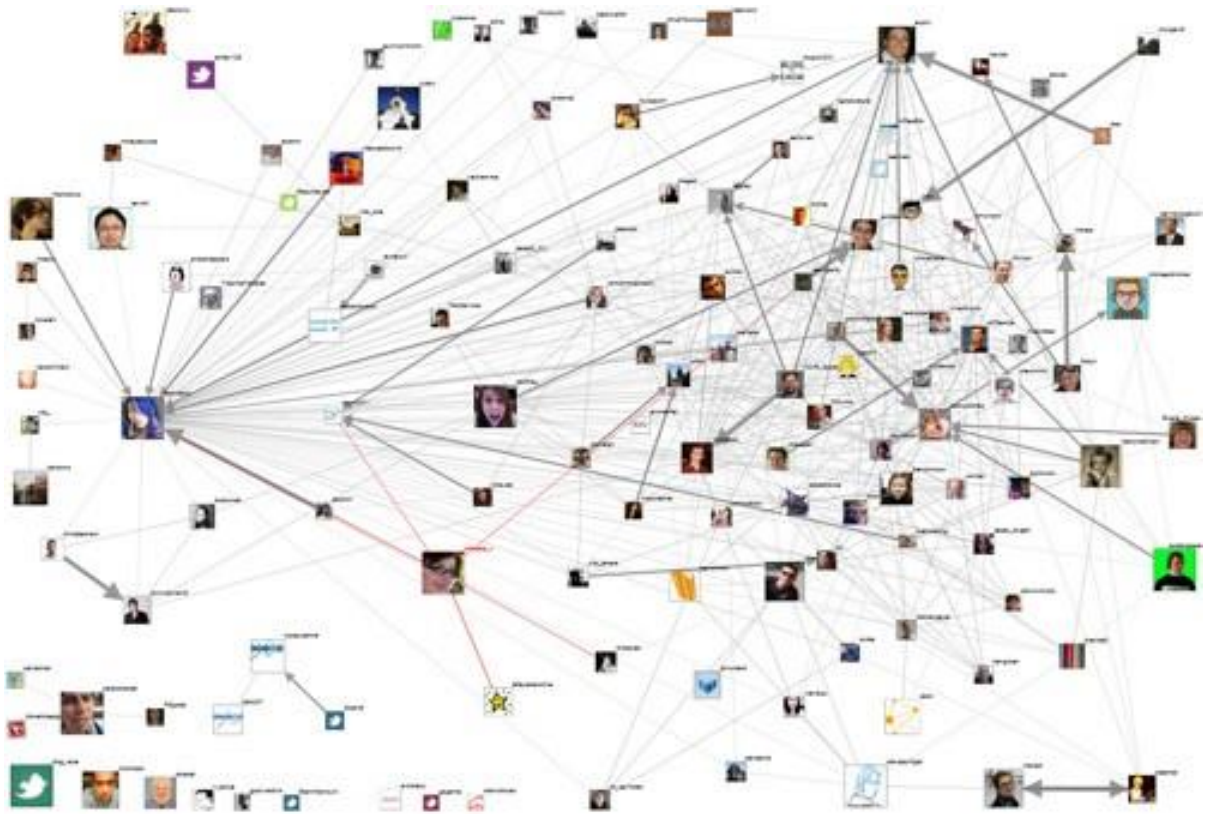
Facebook graph



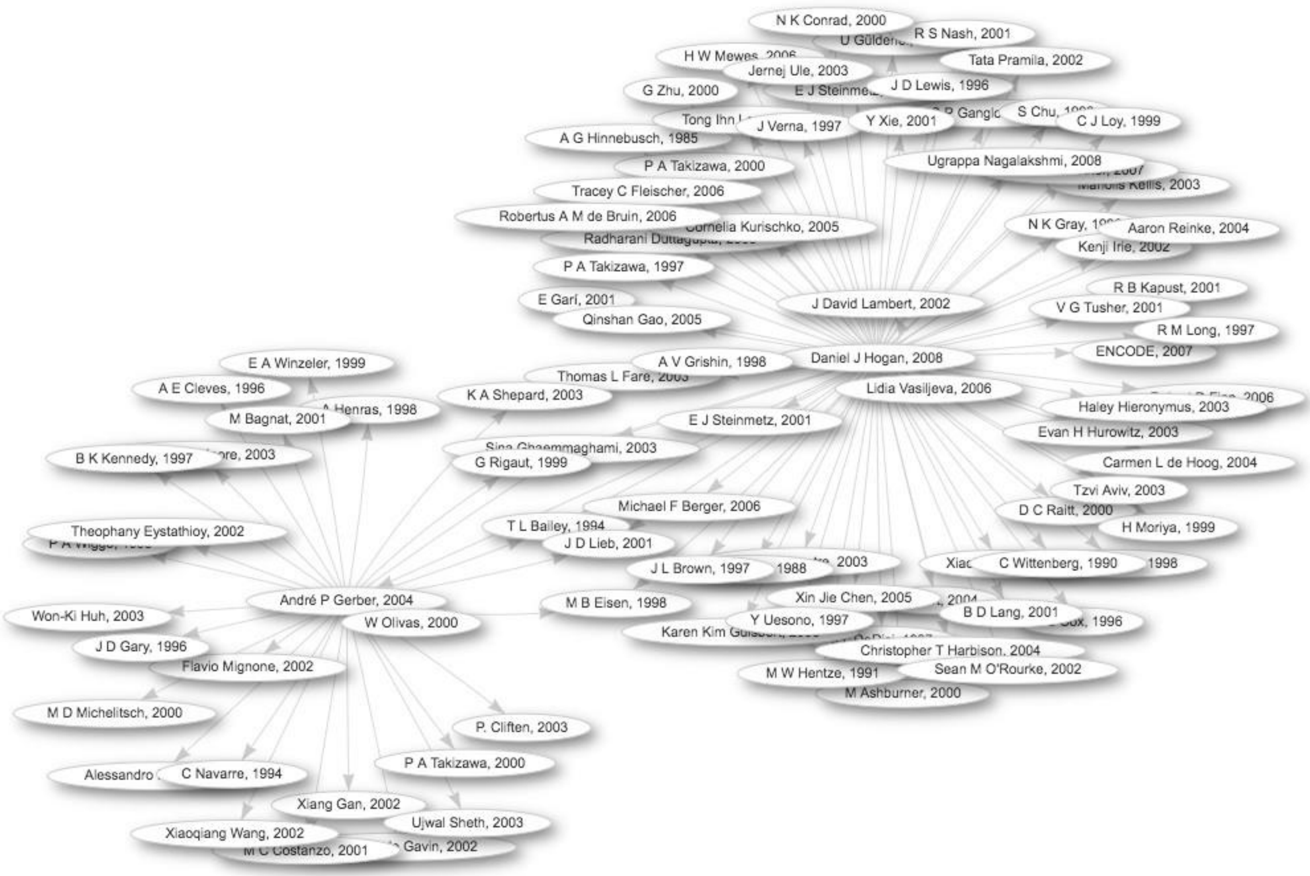
Directed graph: one-way streets



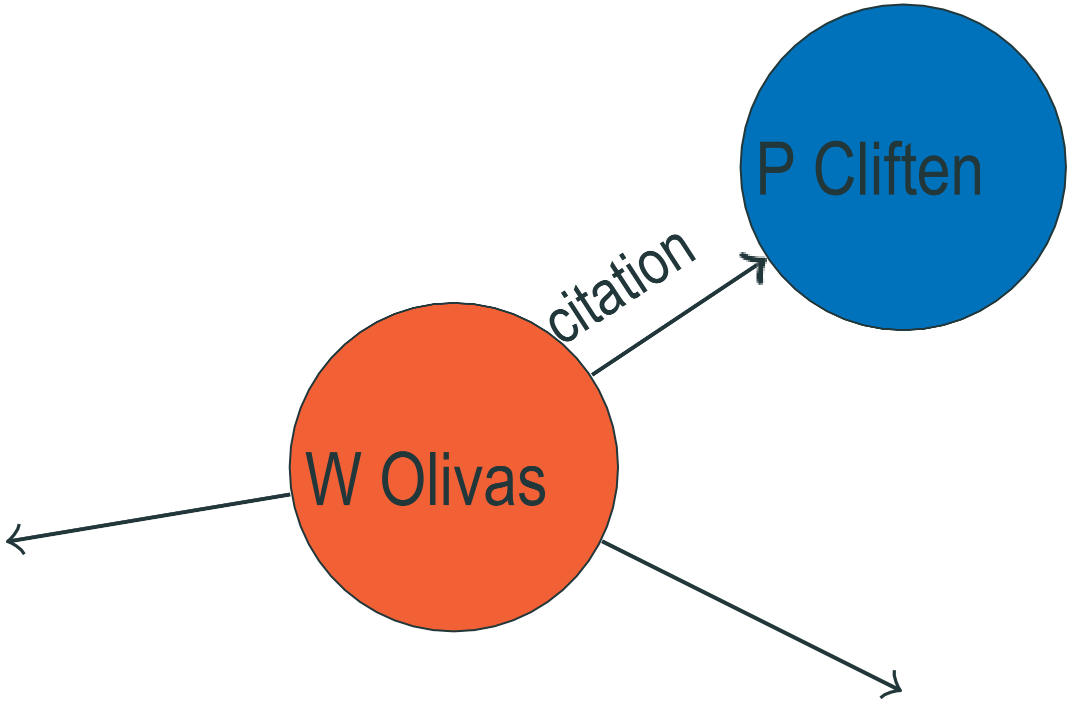
Directed graph: followers



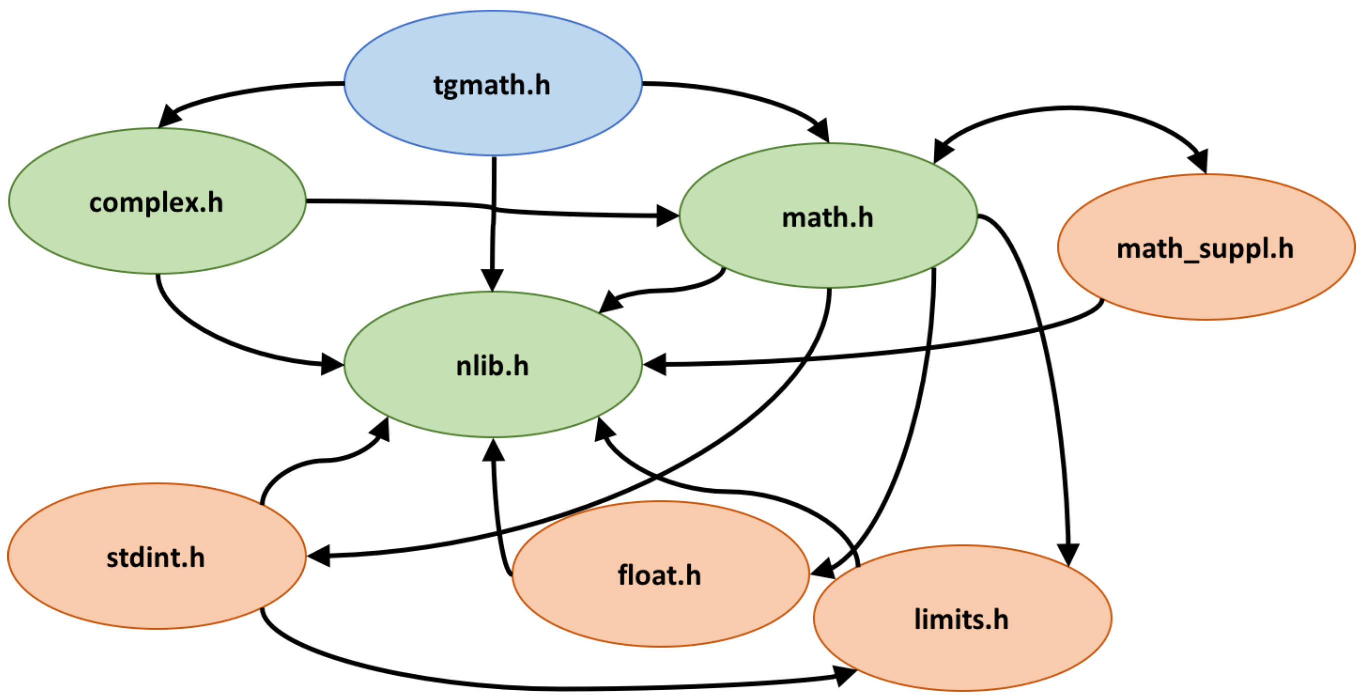
Directed graph: citations



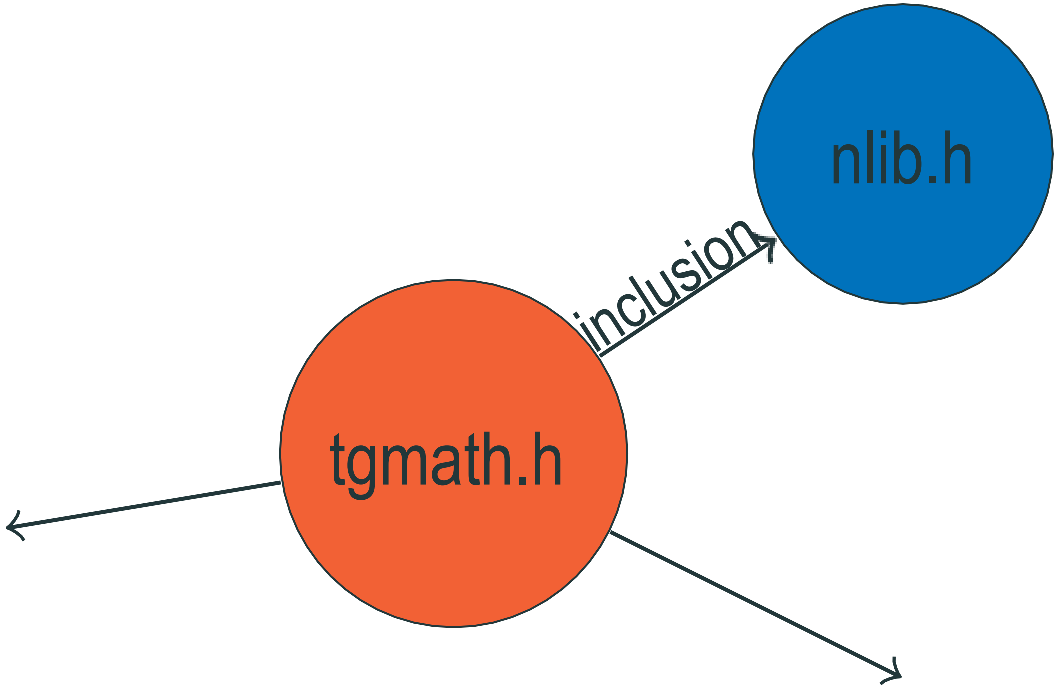
Directed graph: citations



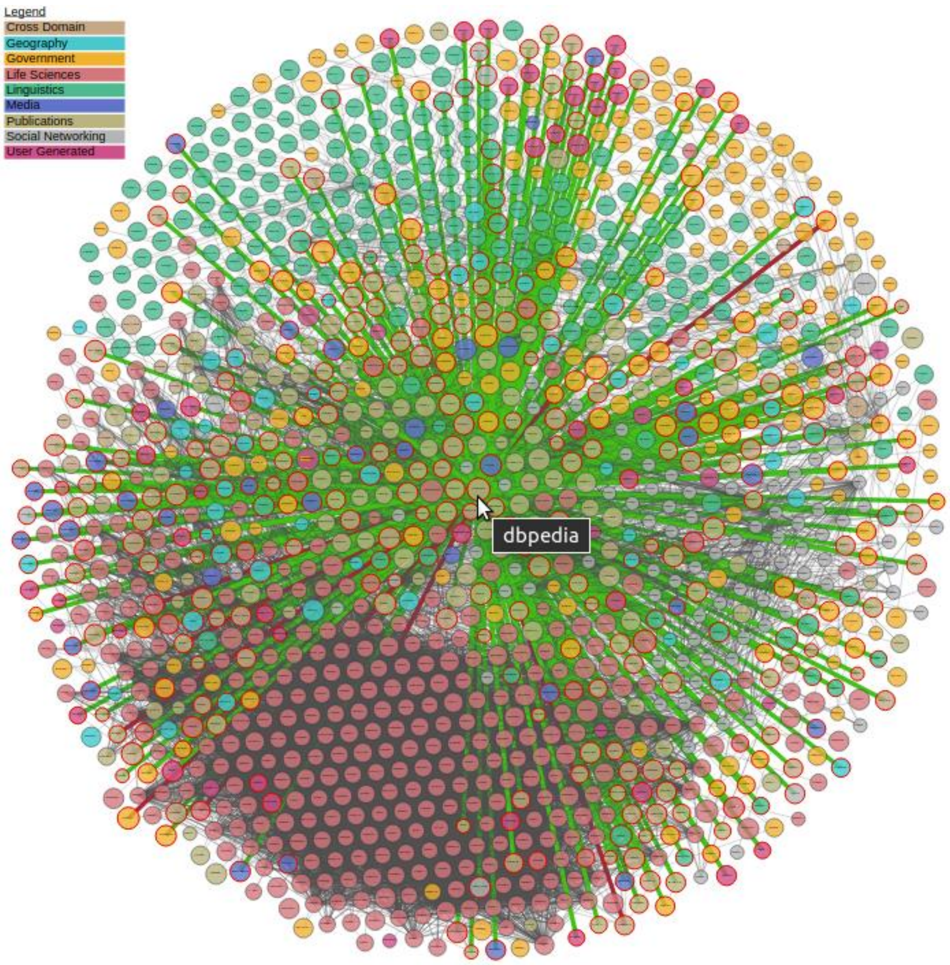
Directed graph: dependencies



Directed graph: dependencies

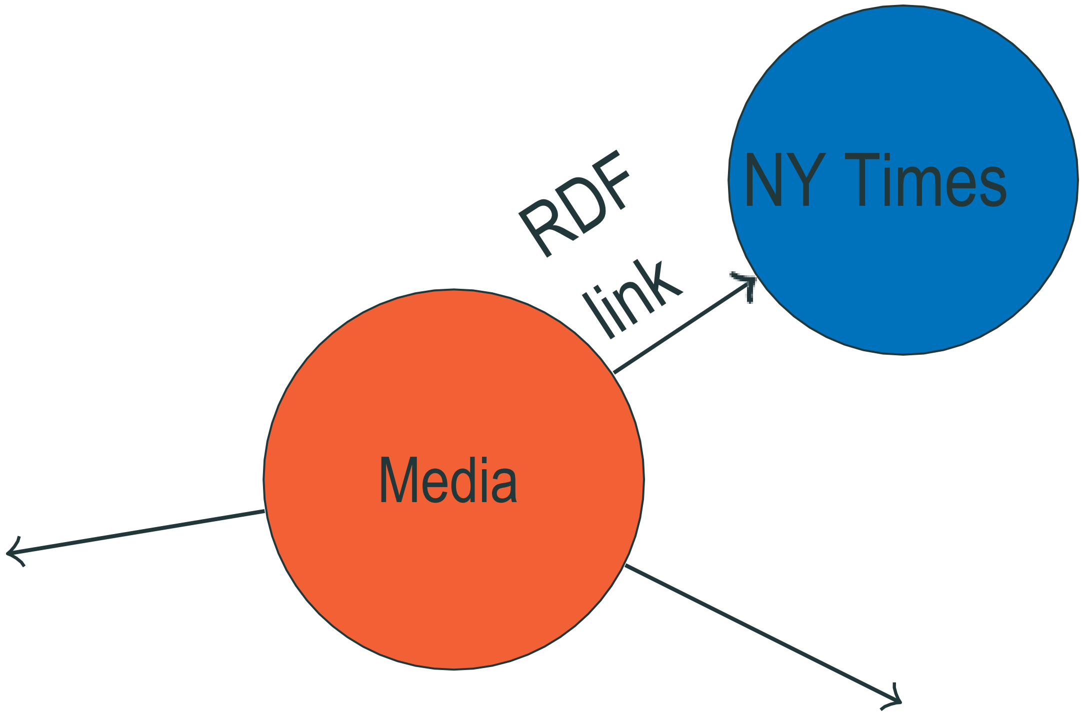


Linked Open Data Diagram

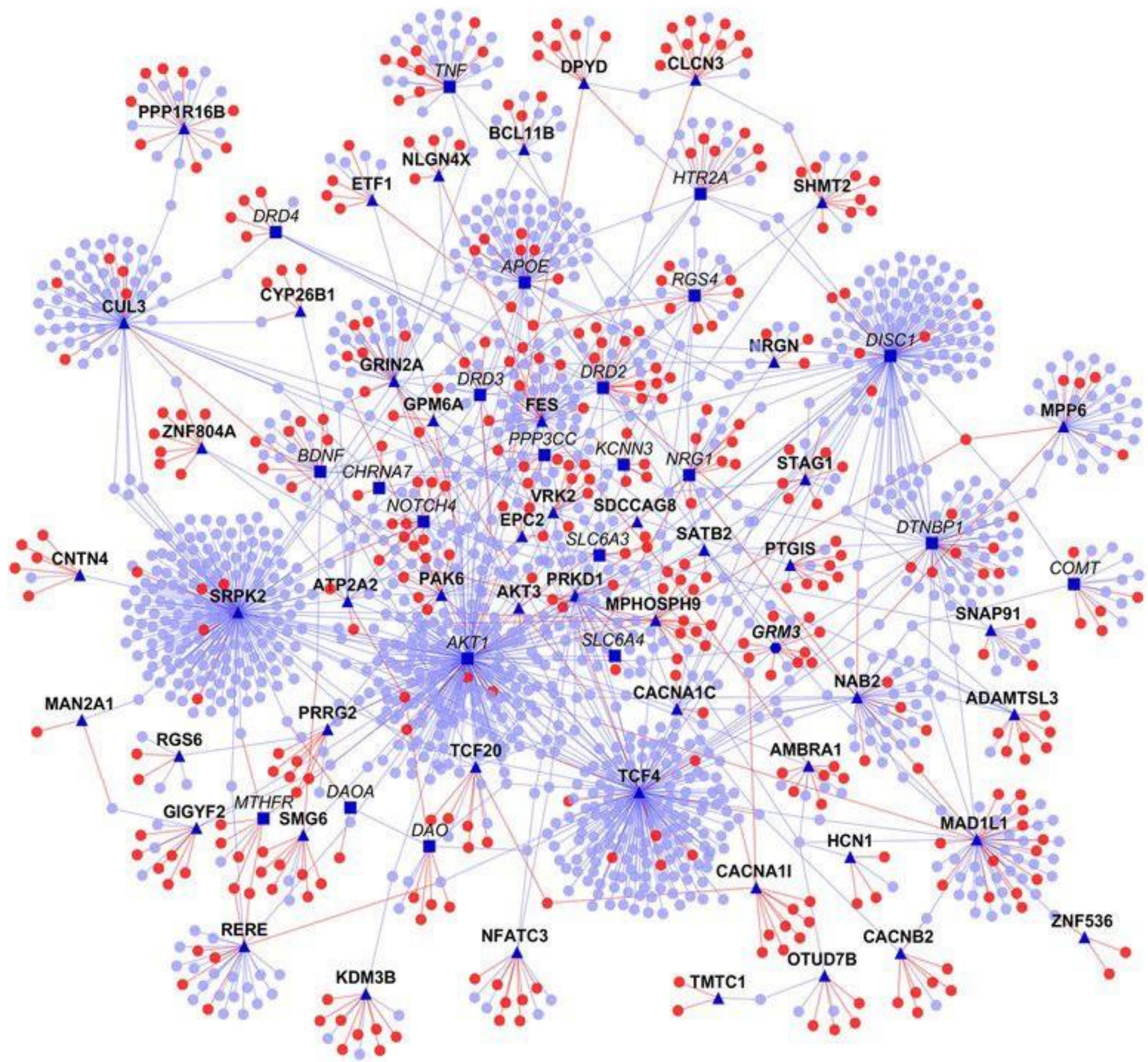


DBpedia: structured cross-domain knowledge

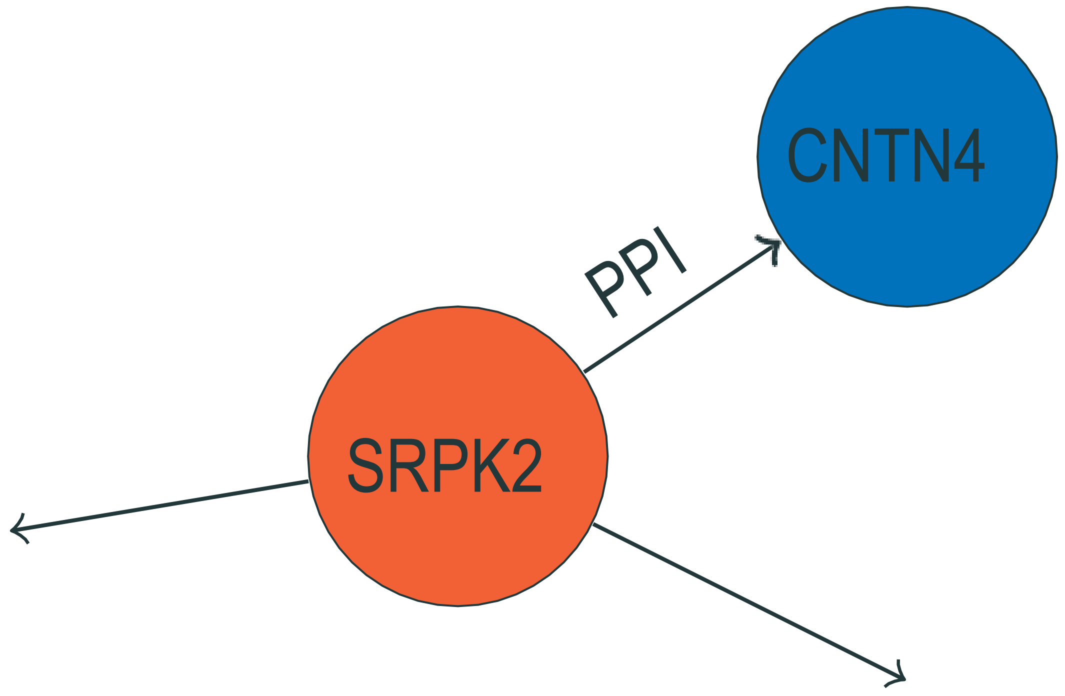
Linked Open Data Diagram



Schizophrenia Protein-Protein Interaction (PPI)

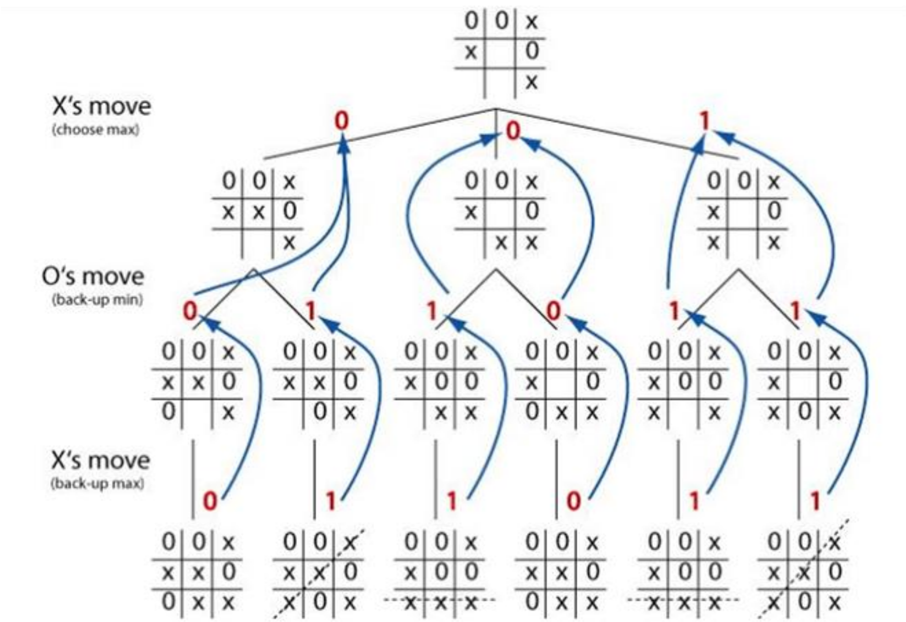


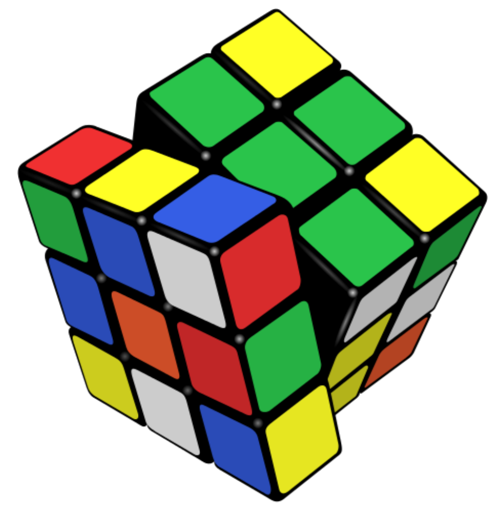
Schizophrenia Protein-Protein Interaction (PPI)



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)

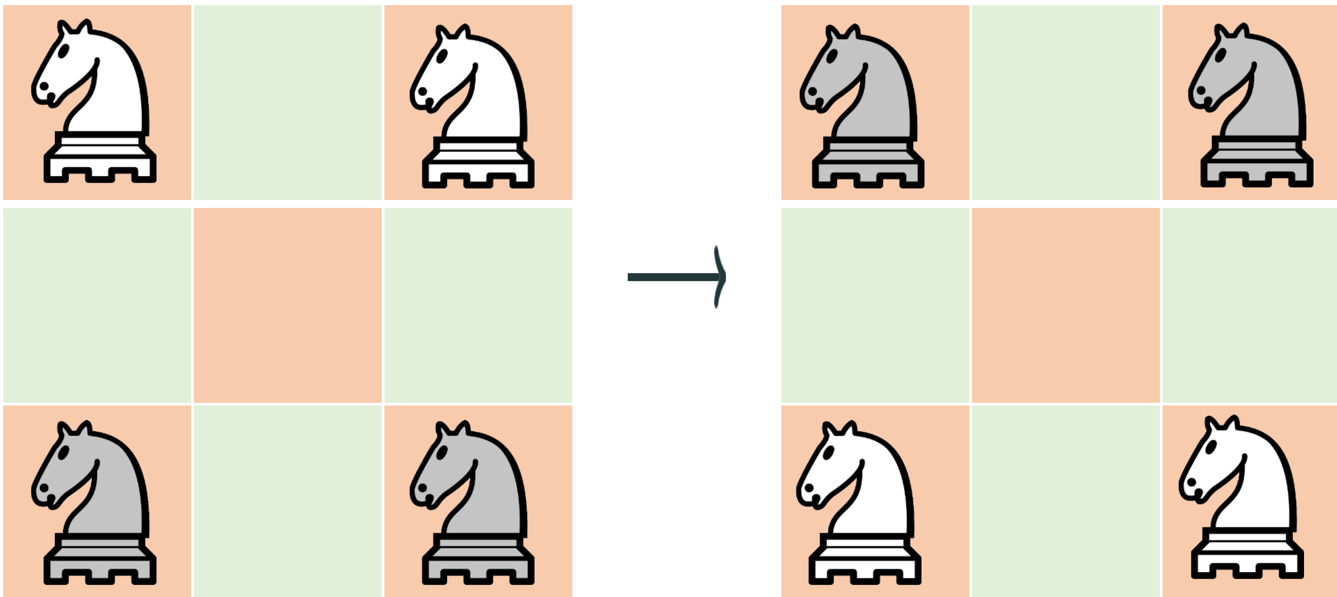
Paolo Guarini di Forli, Italy
15th - 16th Century

Use case 1:
Solving puzzles

With graphs!

Guarini's Puzzle

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

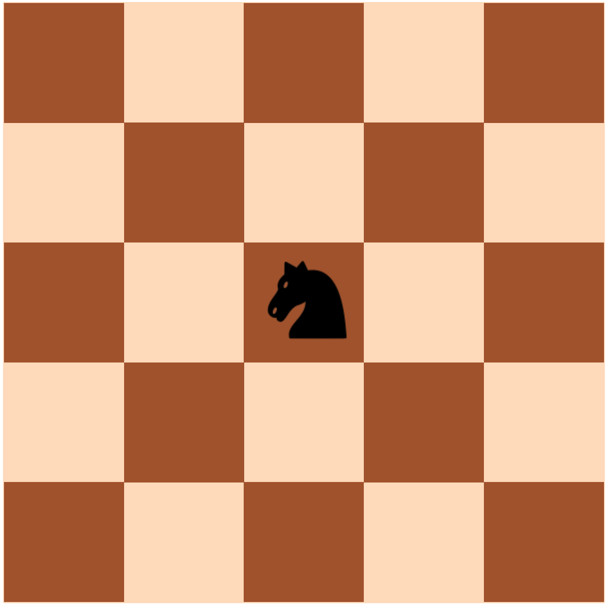


There are four knights on the 3×3 chessboard: the two white knights are at the two upper corners, and the two black knights are at the two bottom corners of the board.

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.

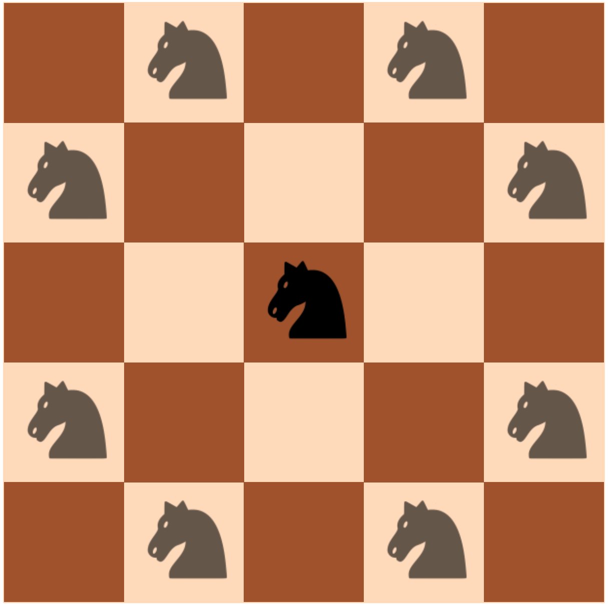
Chess Knight

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

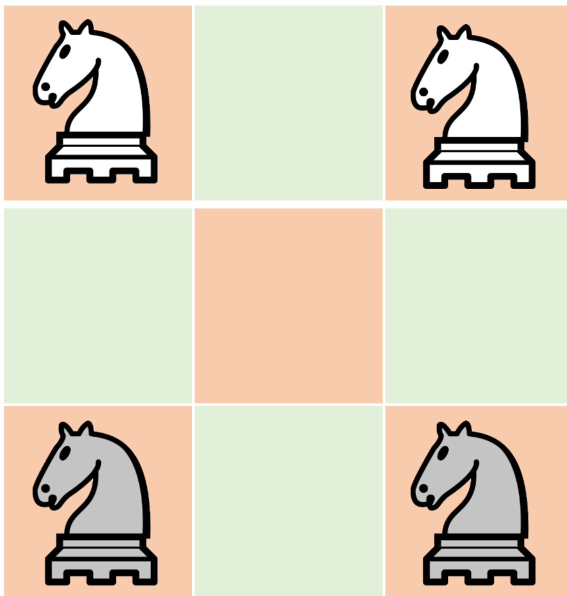


Chess Knight

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



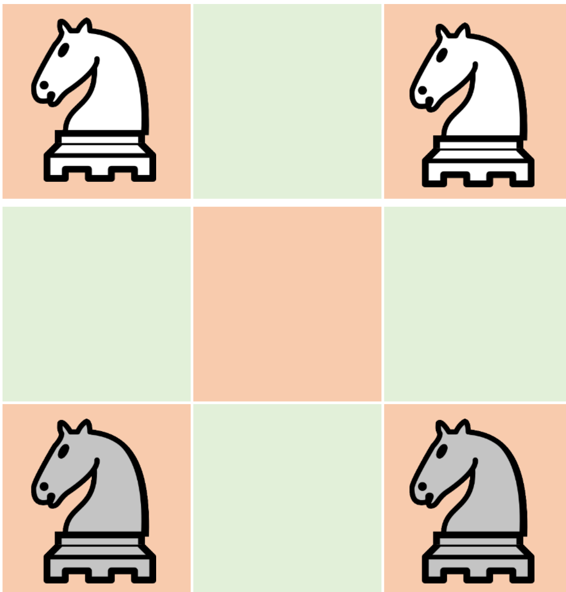
Graph: nodes

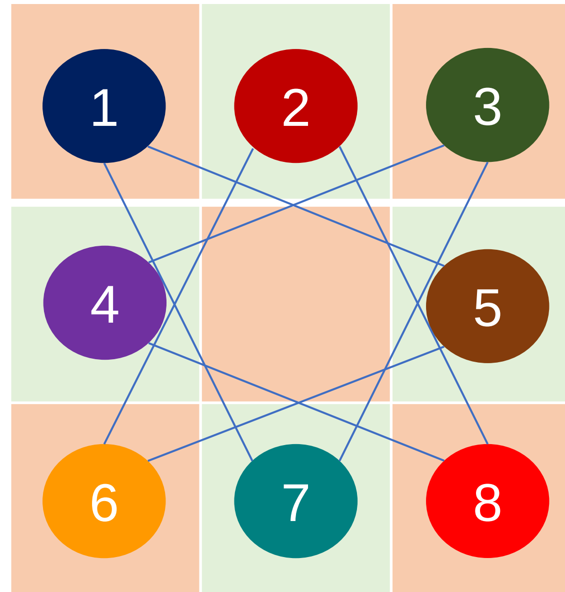


1	2	3
4		5
6	7	8

Each position is a node in a graph

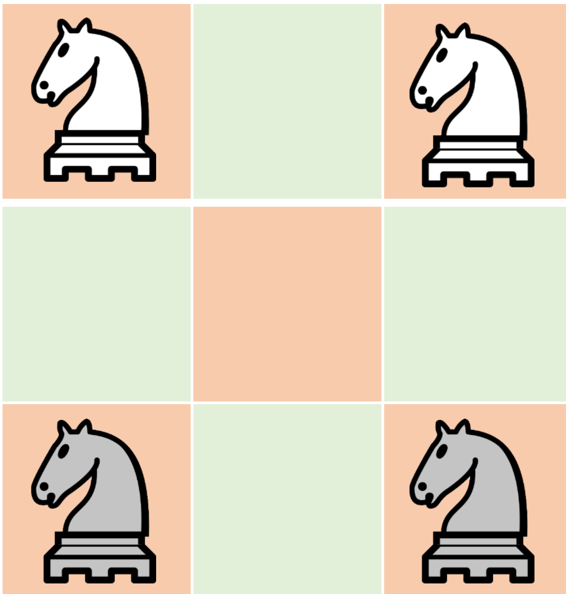
Graph: edges

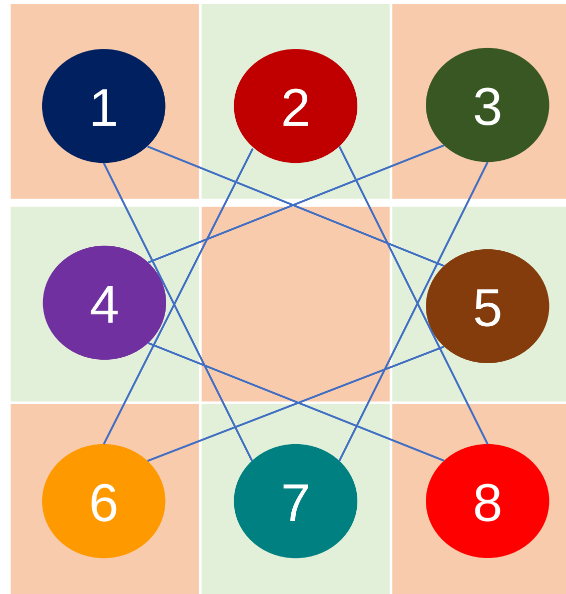




There is an edge between the nodes if you can go from 1 node to another by 1 knight move

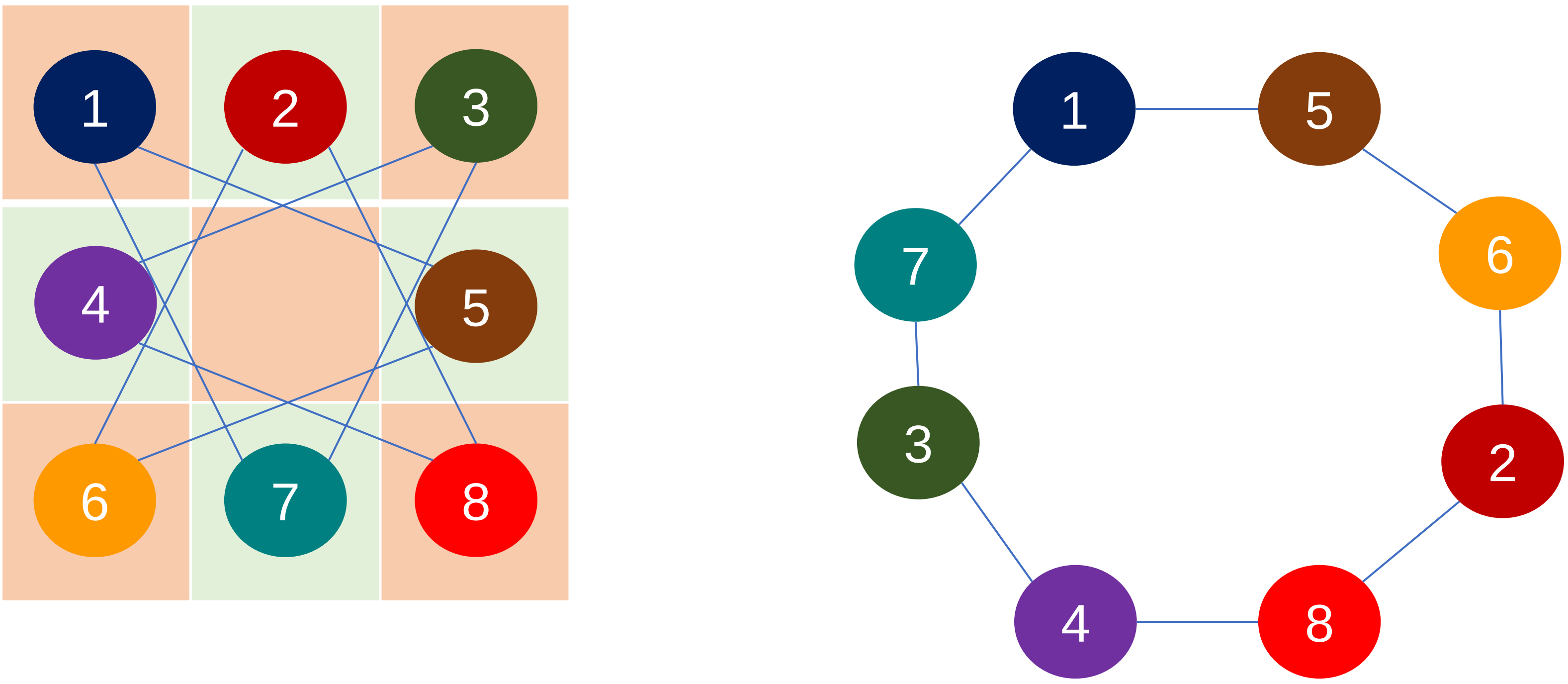
Graph: edges





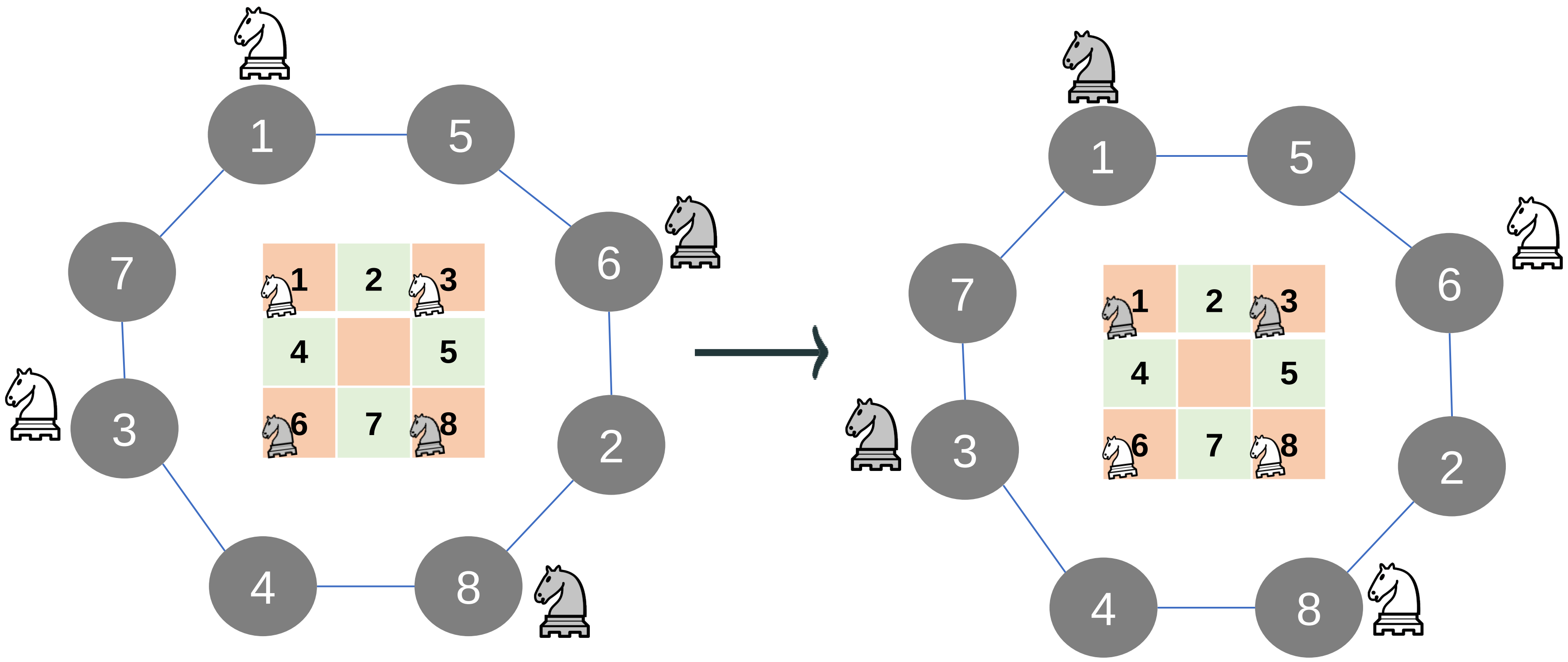
Does it help to solve the puzzle?

Unfold the graph!



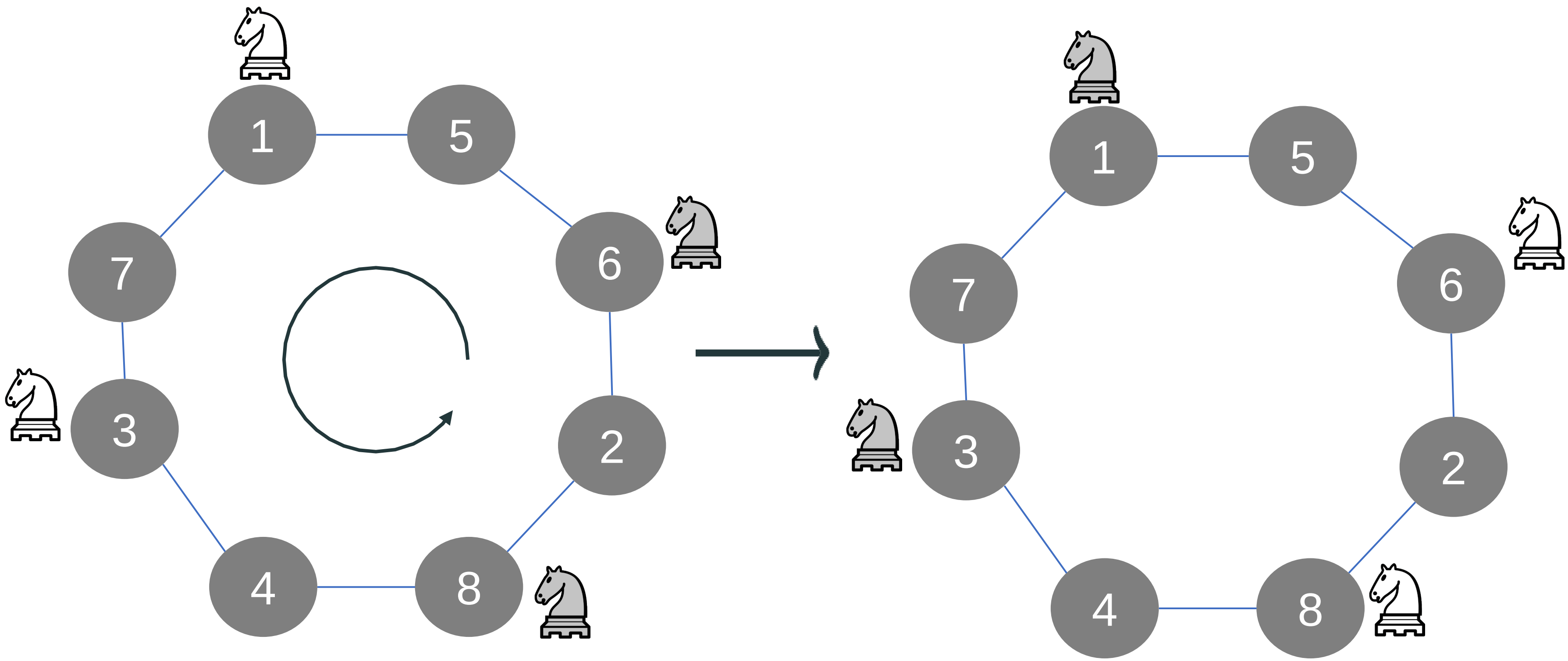
All the nodes are on a circle

Solution



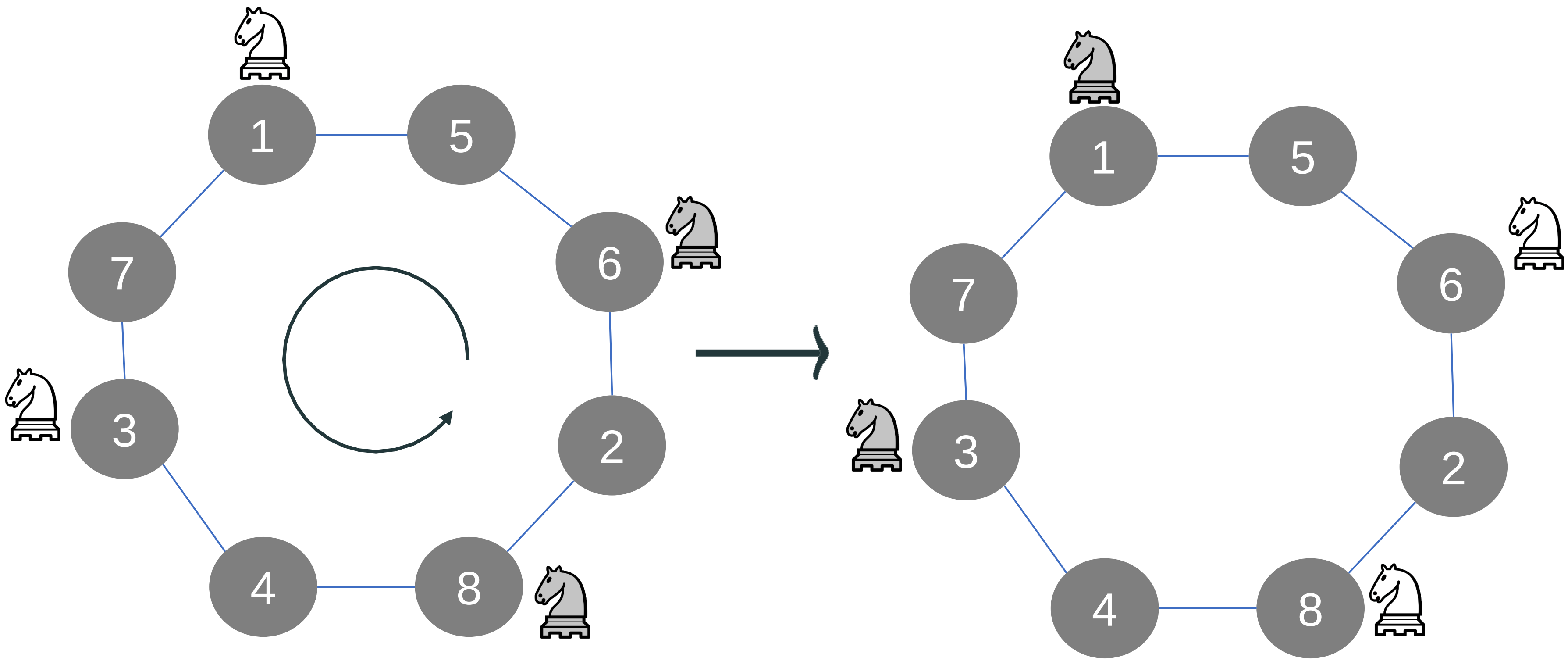
Do you see it now?

Solution



Move around the circle following legal edges

Solution

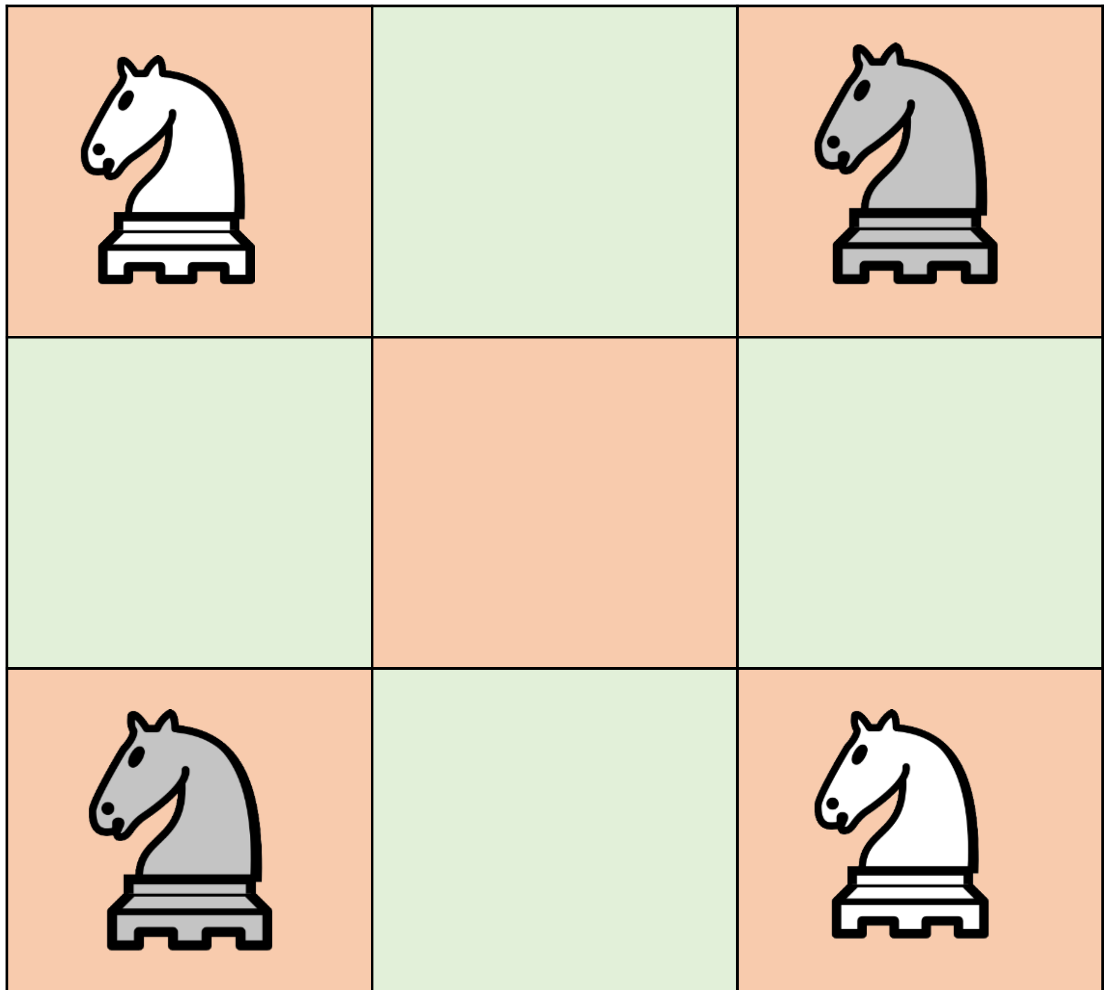


Until knights are in desired positions

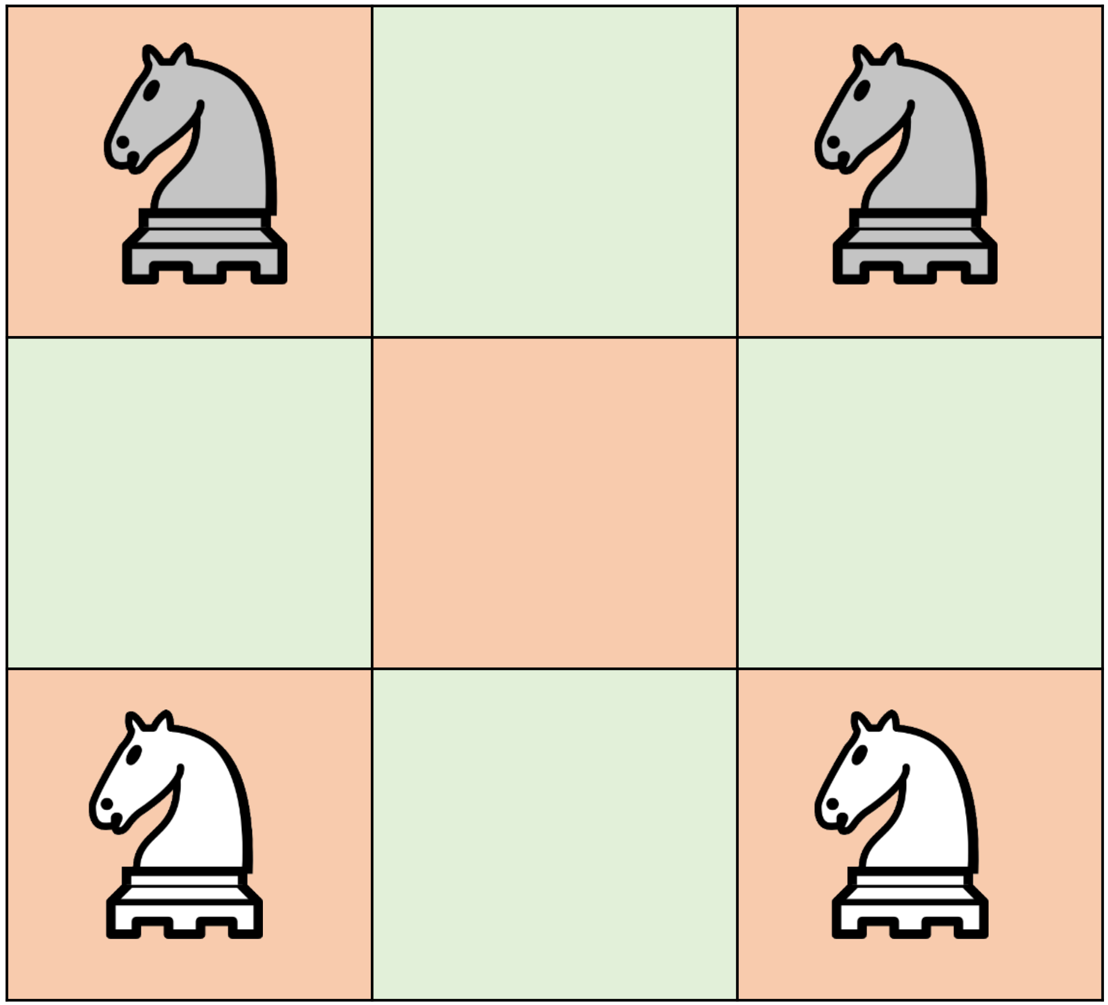
Try it out

http://barsky.ca/knights/

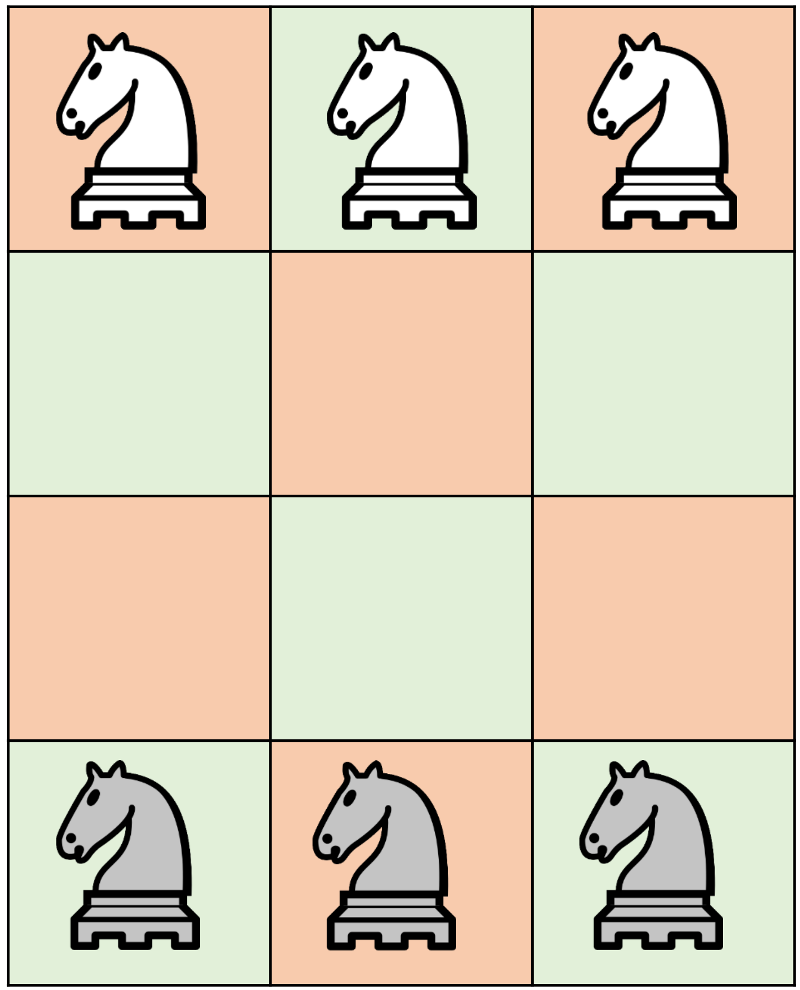
Puzzle 2. Start configuration



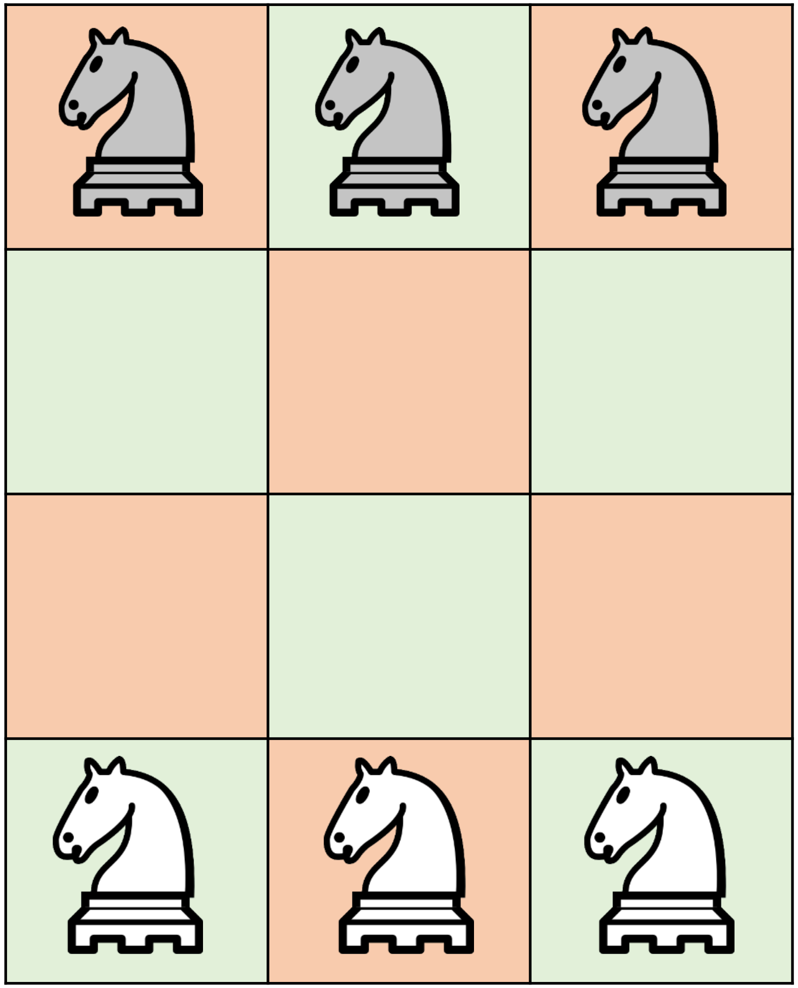
Puzzle 2. End configuration



Puzzle 3. Start configuration



Puzzle 3. End configuration



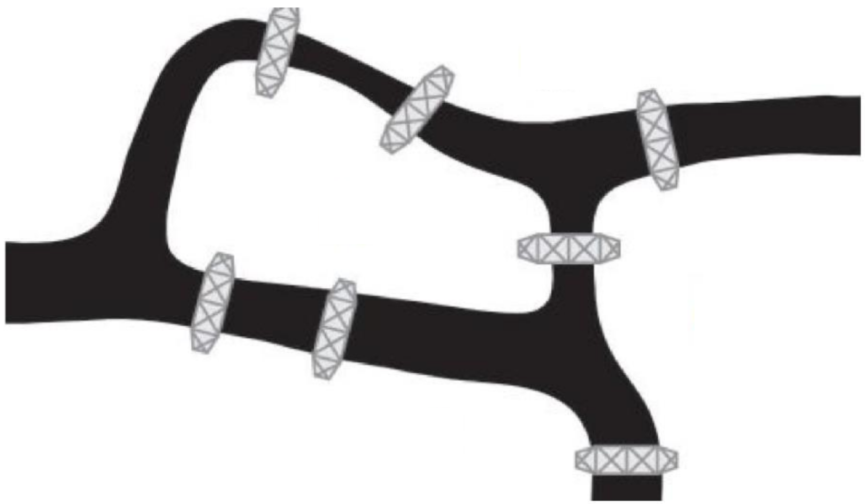
Use case 2:
Genome assembly

With graphs!

Euler's dilemma:

Leonhard Euler
1707 - 1783

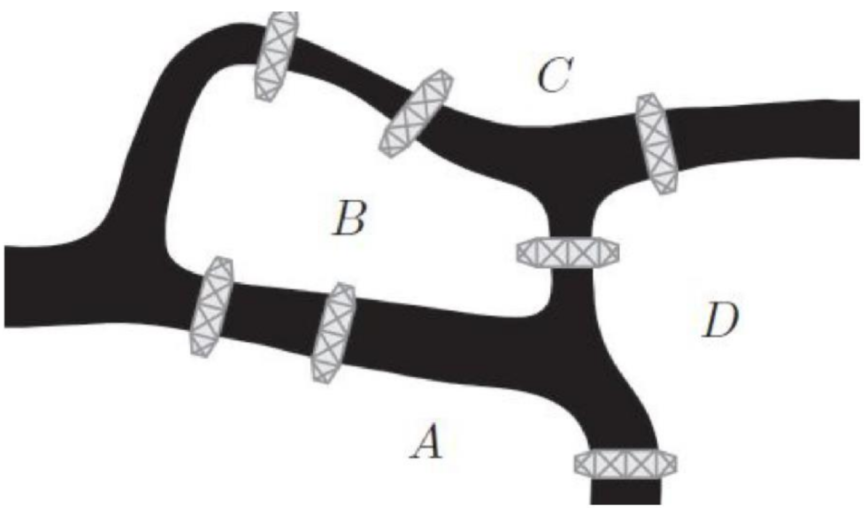
Can I take a walk and visit each bridge exactly once?



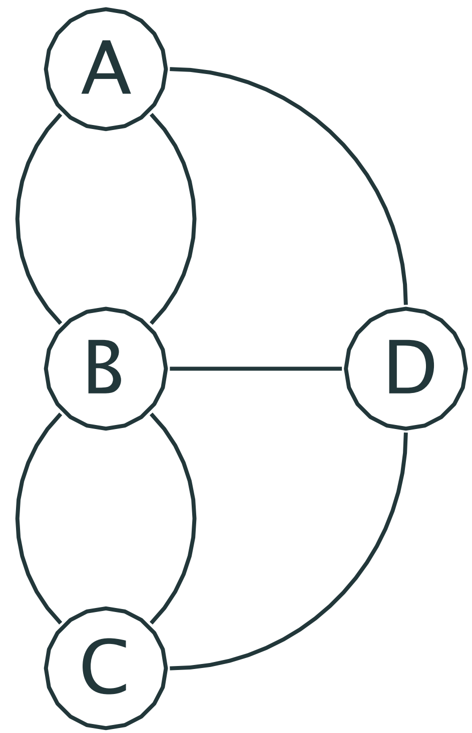
Seven bridges of Königsberg

Euler’s path problem

Is there a path which visits **every edge** of the graph exactly once?

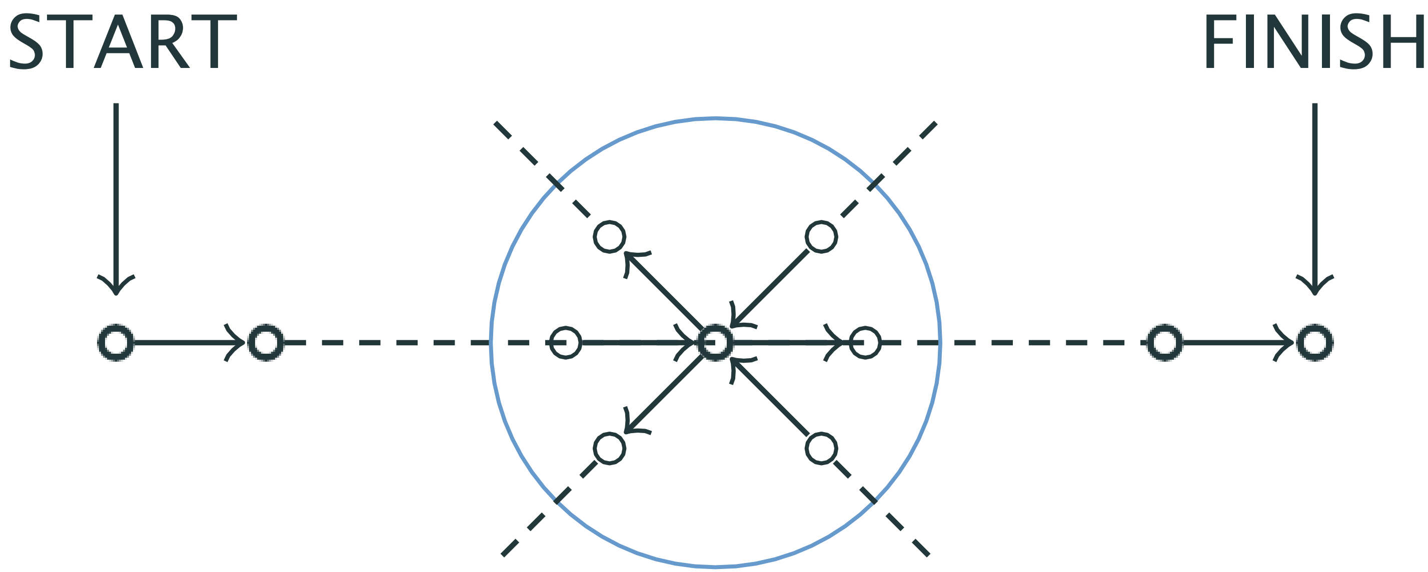


Seven bridges of Königsberg



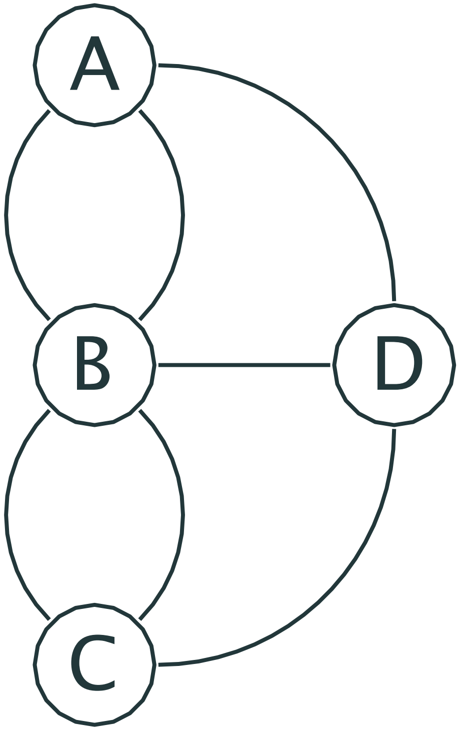
Modeled as Graph

Eulerian Path

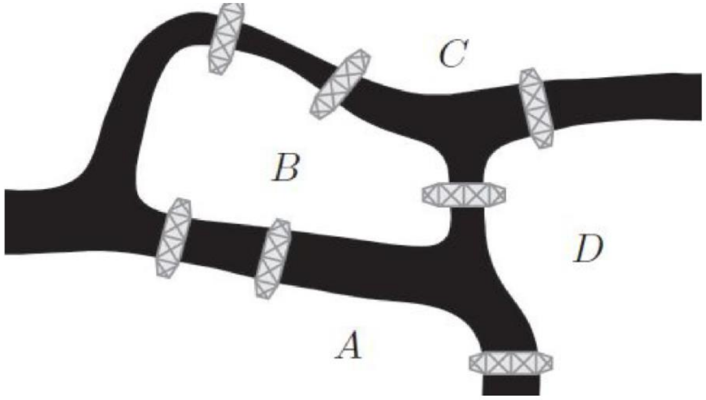


Necessary condition: all but START and FINISH vertices must have **even** degrees. Why?

Seven bridges of Königsberg



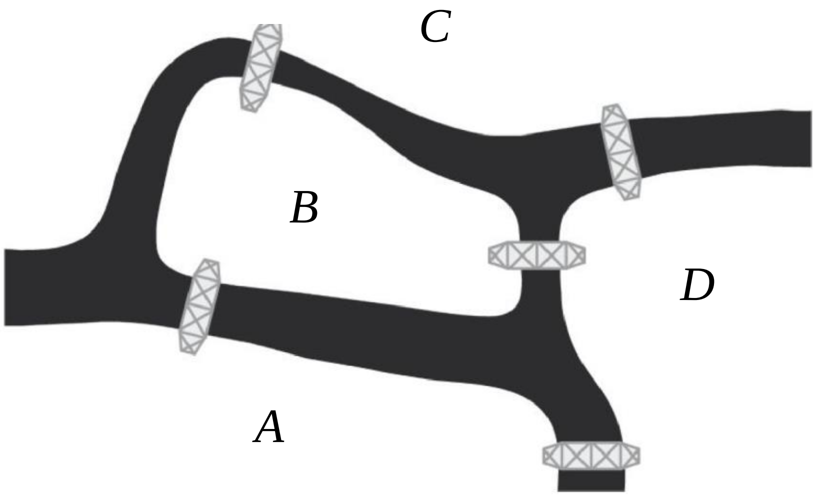
Is there an Eulerian Path through these seven bridges?



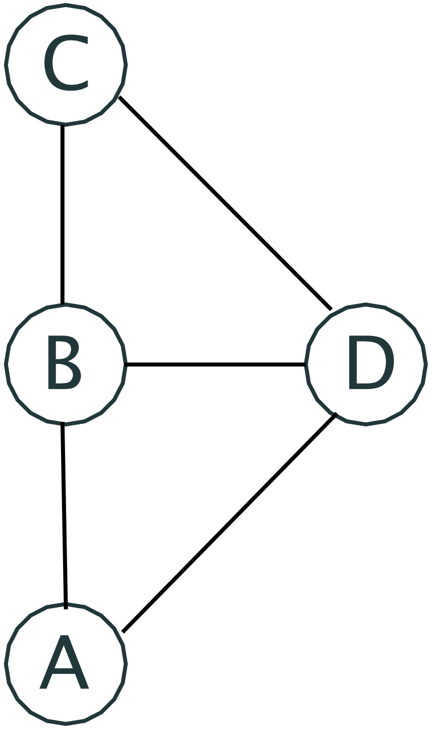
Königsberg, 17-th century

Five Bridges of Kaliningrad

Is there an Eulerian Path through these five bridges?



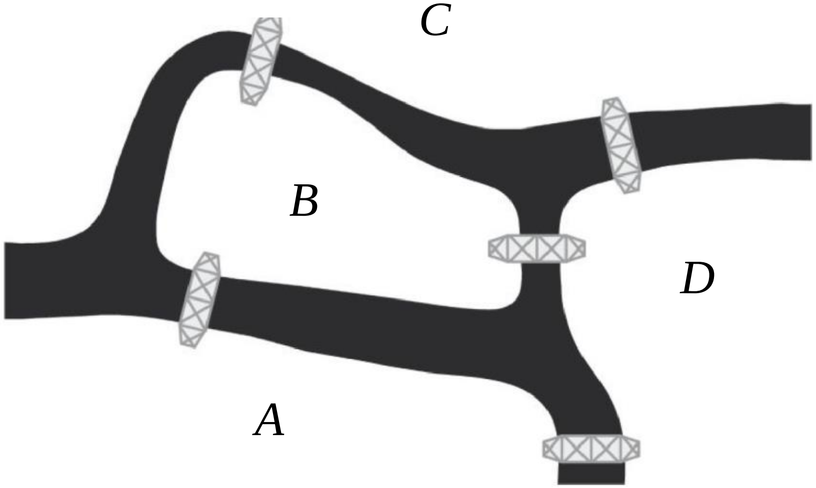
Königsberg (Kaliningrad), 21-th century



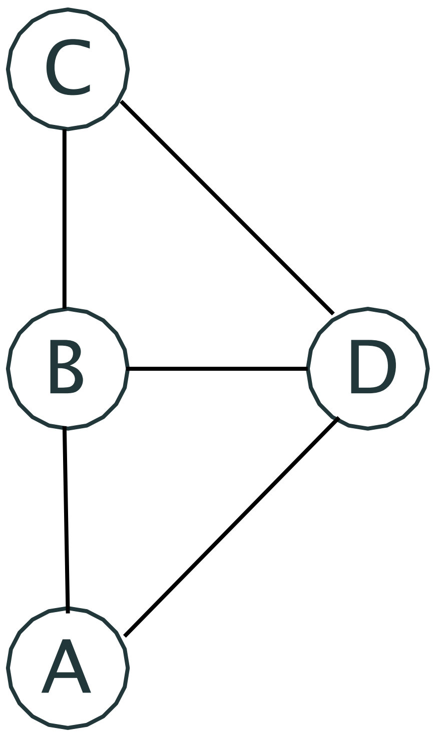
Five Bridges of Kaliningrad

B and D have odd degree

If there exists an Eulerian path, B and D must be START and FINISH



Königsberg (Kaliningrad), 21-th century



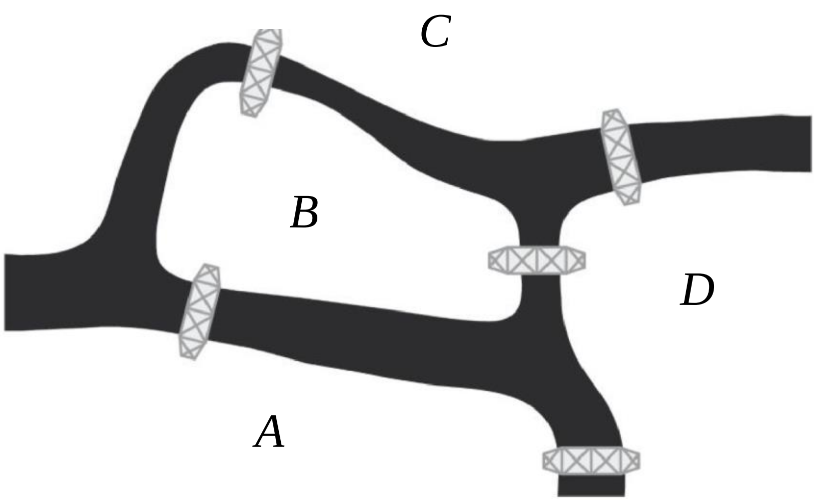
Eulerian Cycle

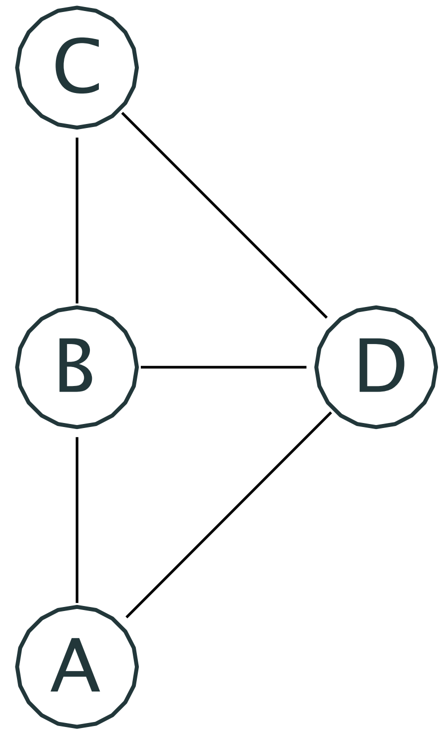
An Eulerian cycle (circuit) visits every edge exactly once and returns to the starting vertex.

- The definition works for both directed and undirected graphs
- A cycle must have the same starting and ending vertex
- While in a path the starting and ending node should not necessarily be the same (but they might be the same). So the cycle is a special case of a path.

Eulerian cycle

If there exists an Eulerian cycle, all vertices must have even degrees





Is there an Eulerian cycle here?

Criteria for Eulerian Cycle (Path)

Theorem

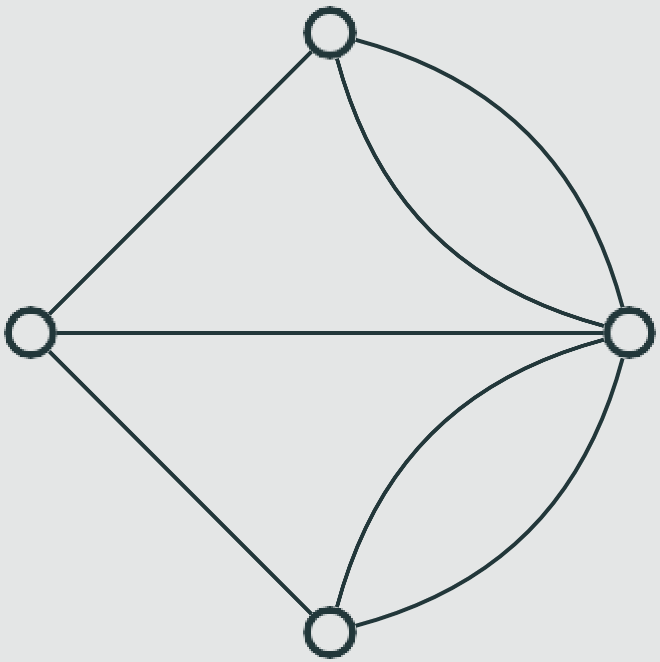
A **connected** *undirected* graph contains an Eulerian cycle, **if and only if** the degree of every node is even.

Note: every cycle is also a path, so if we have an Eulerian cycle, we also have an Eulerian path

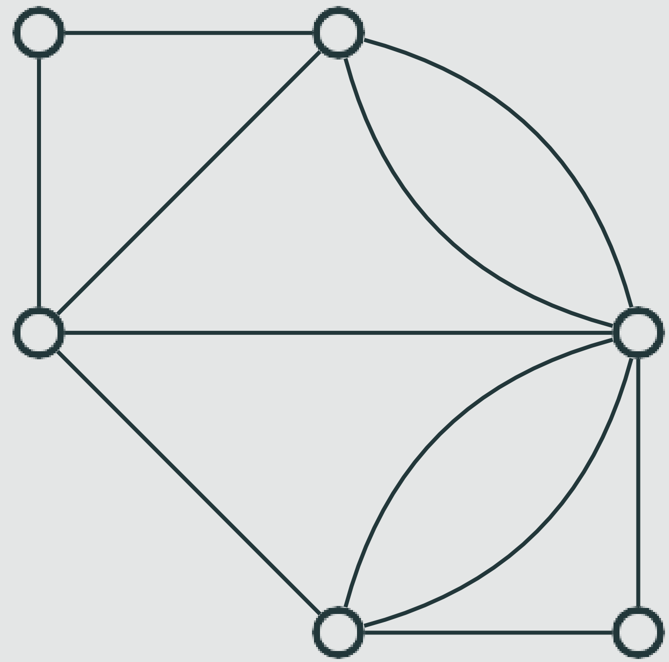
But if we only want a path which is not a cycle, then exactly 2 vertices (namely start and end) are allowed to have odd degrees.

Example

Non-Eulerian graph

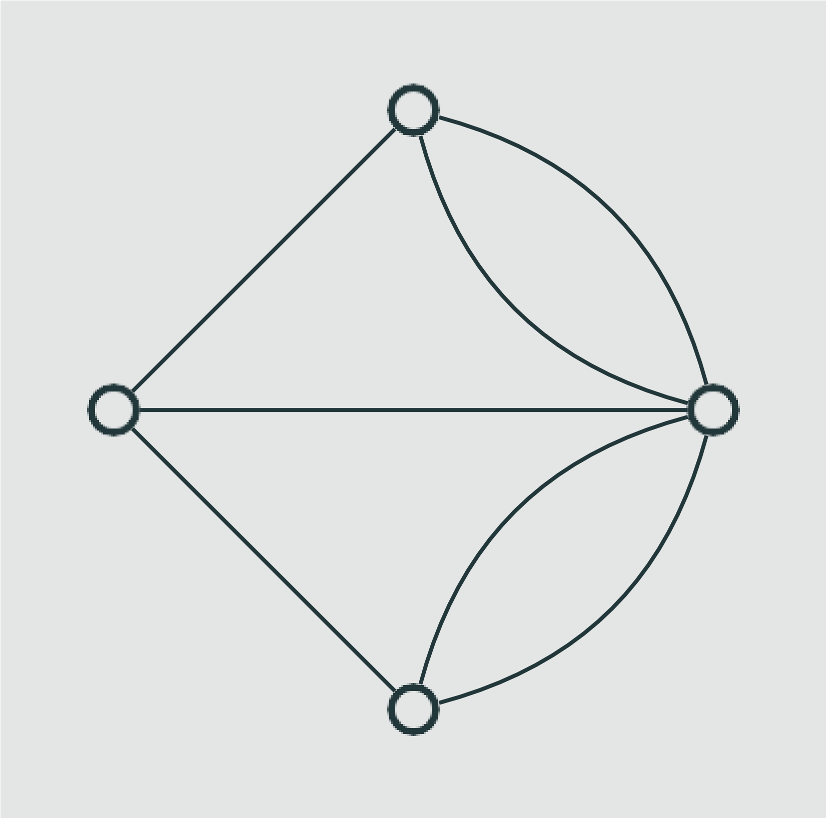


Eulerian graph

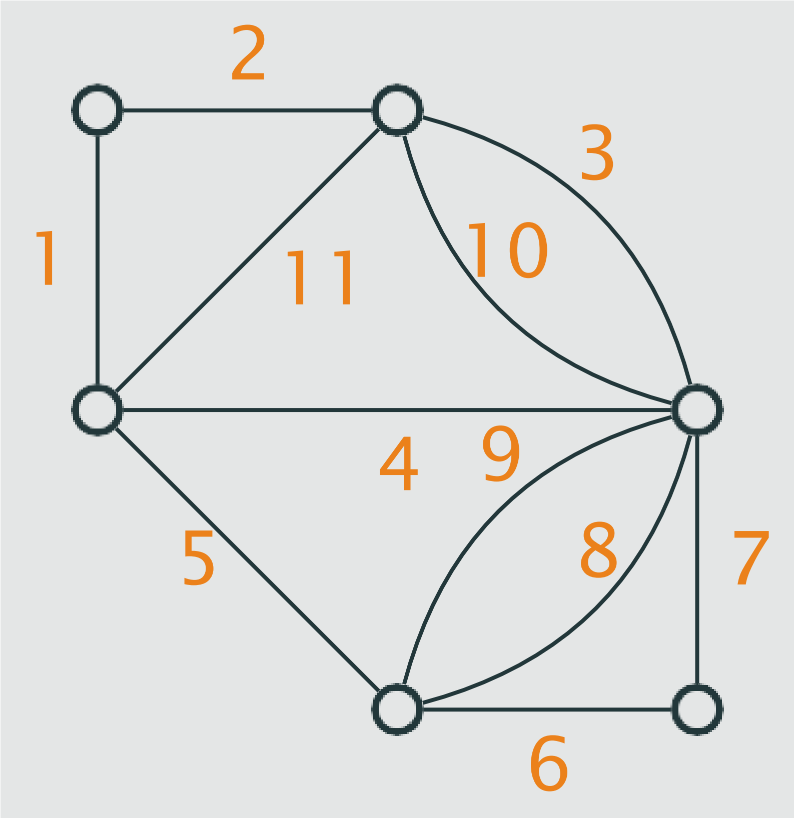


Example

Non-Eulerian graph



Eulerian graph



Criteria for Eulerian cycle (path)

Theorem

A **connected** *undirected* graph contains an Eulerian cycle, **if and only if** the degree of every node is even.

Theorem

A strongly connected *directed* graph contains an Eulerian cycle, **if and only if**, for every node, its in-degree is equal to its out-degree.

Balanced directed graph

Algorithm for finding Eulerian Cycle (Path)

The proof of existence of an Eulerian cycle can be transformed into an efficient algorithm for constructing it

Finding Eulerian Path: algorithm

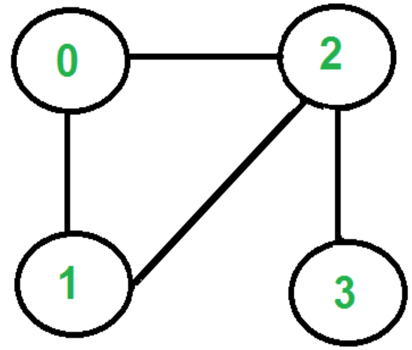
- If there are 0 odd vertices, start anywhere
  If there are 2 odd vertices, start at one of them.

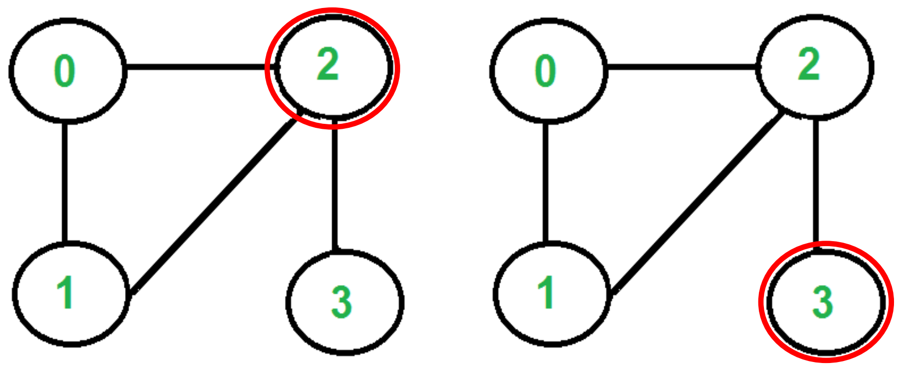
- Follow edges one at a time
  If you have a choice between a bridge and a non-bridge, always choose the non-bridge: “don’t burn bridges“ so that you can come back to a vertex and traverse remaining edges

- Remove followed edge (or mark as processed)

- Stop when you run out of edges

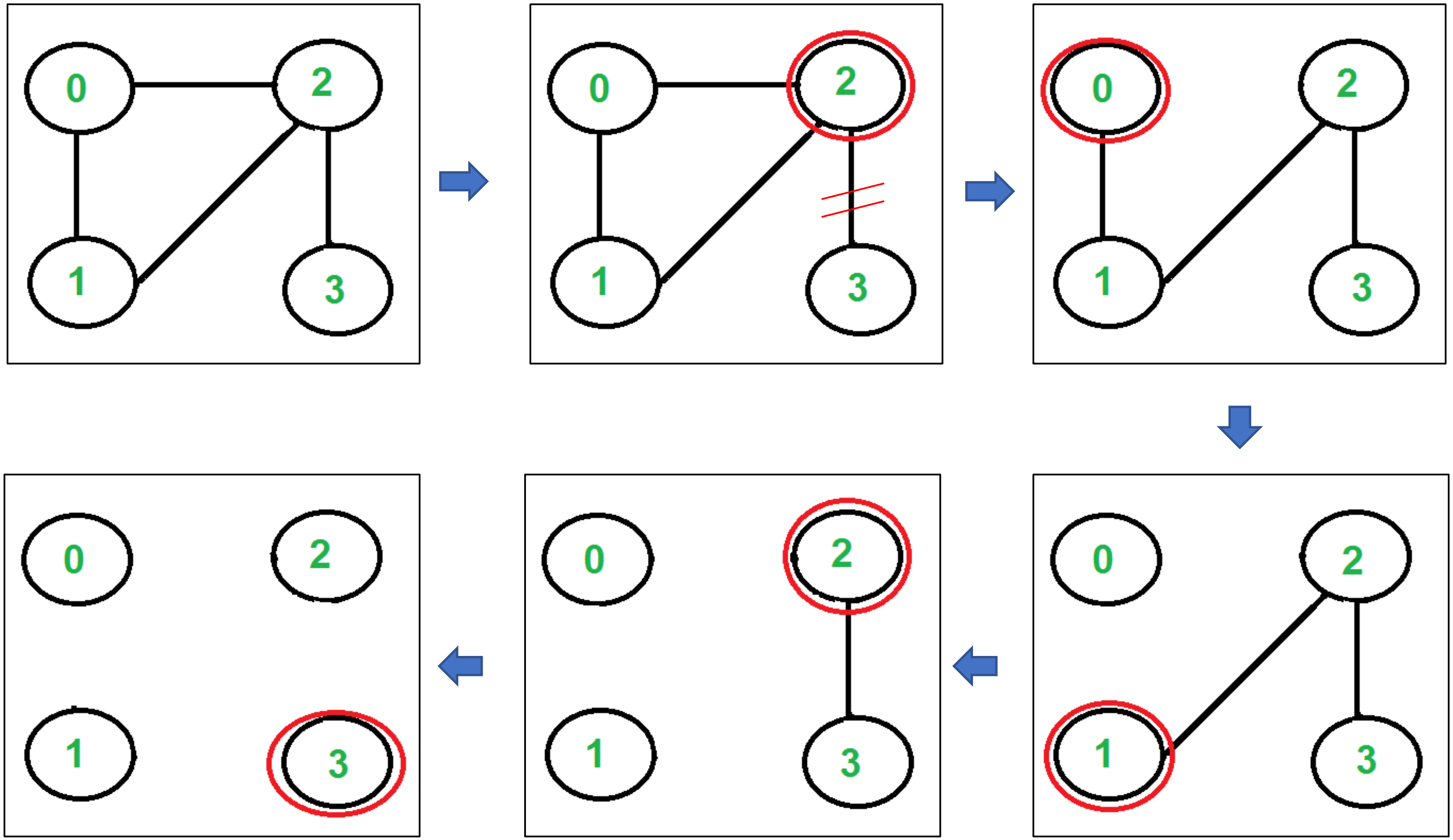
Example: undirected graph





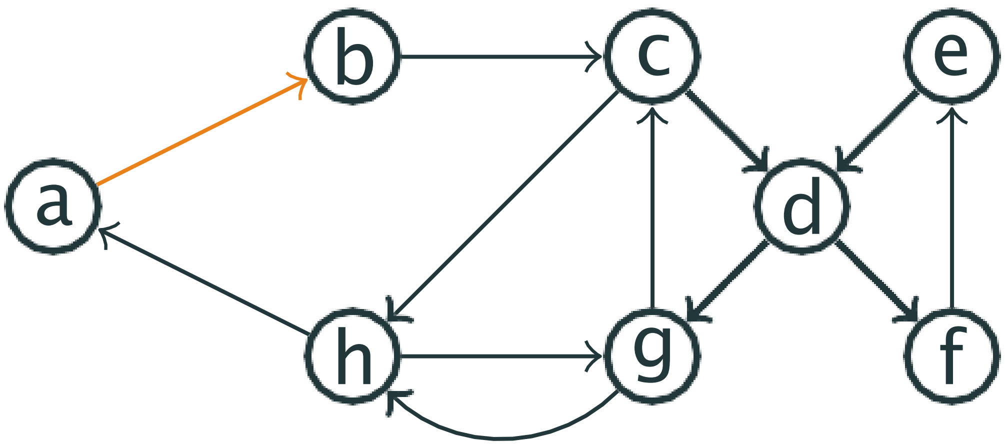
Two vertices with odd degree – choose any of them to start

Example



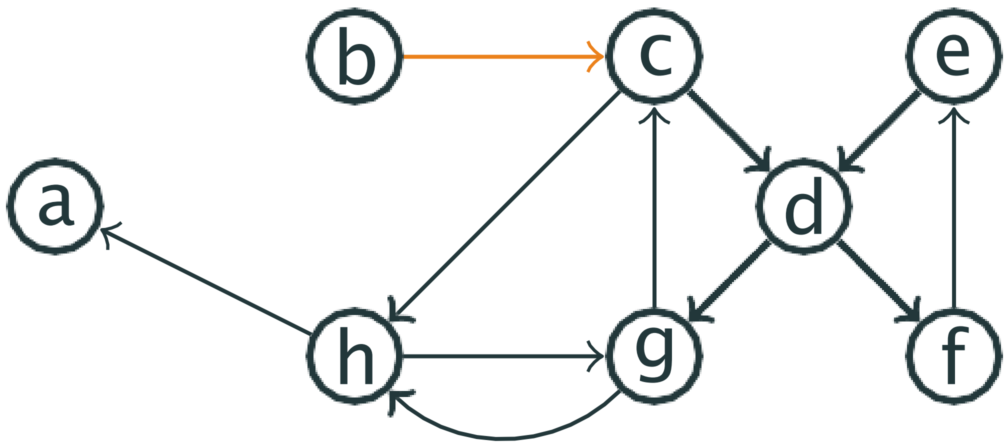
Eulerian Path: (2,0), (0,1), (1,2), (2,3)

Example: Directed Graph



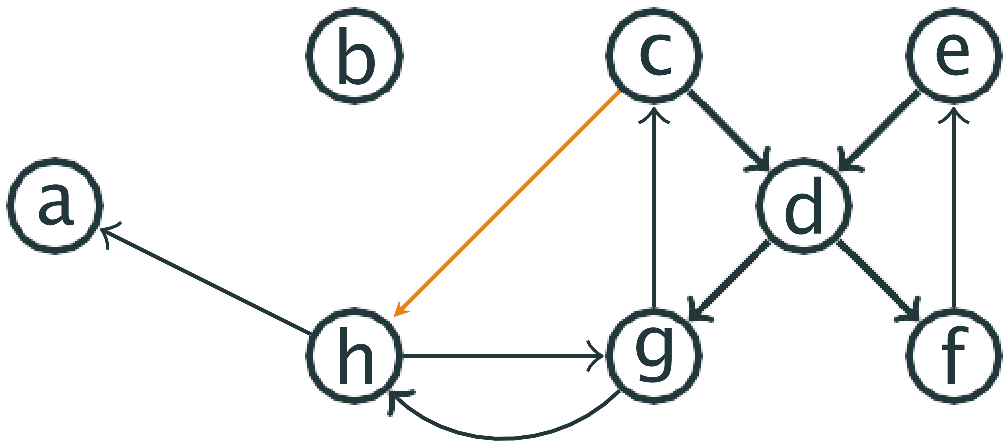
start walking from some node
(all are balanced)

Example: Directed Graph



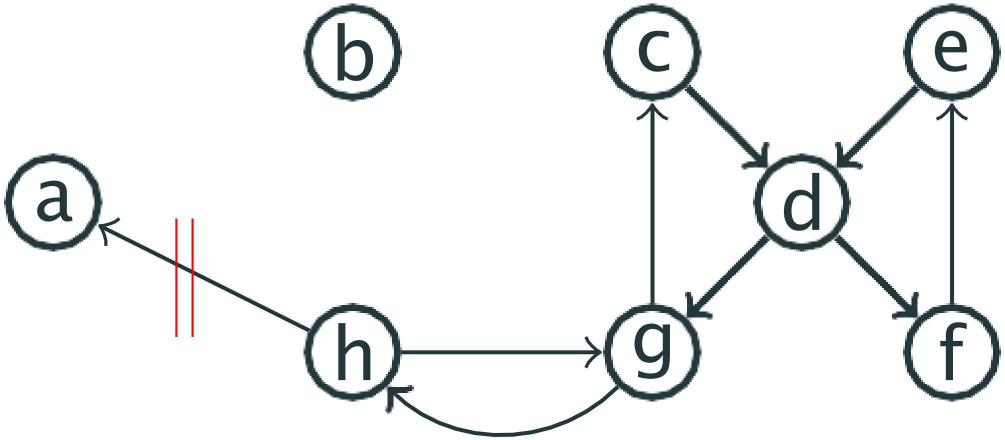
Path so far: a→b

Example: Directed Graph



Path so far: a→b→c

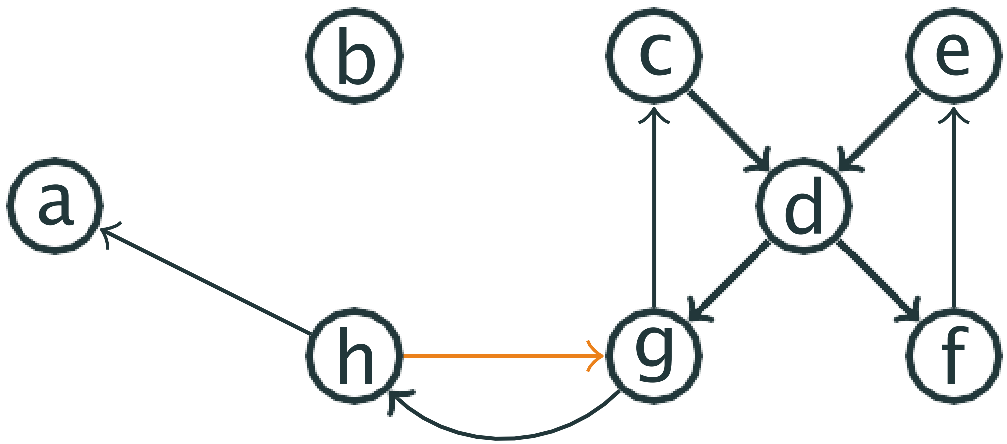
Example: Directed Graph



Path so far: a→b→c→h

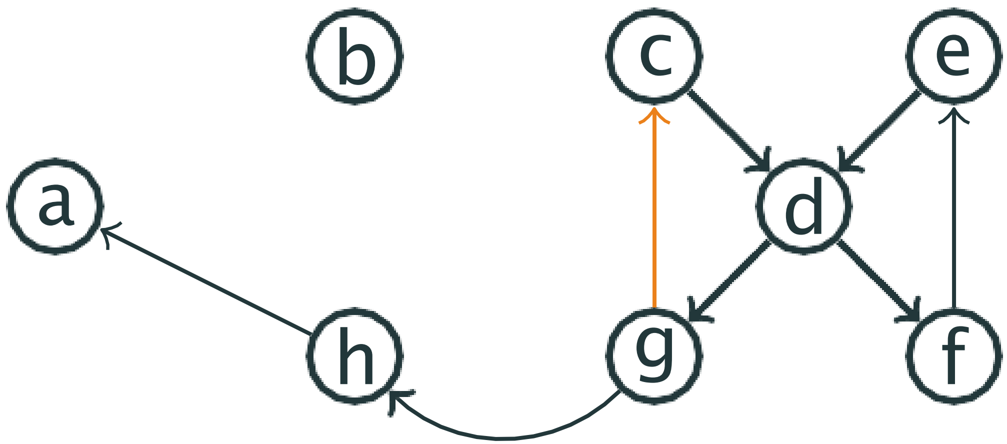
We cannot go to a: h-a is a bridge!

Example: Directed Graph



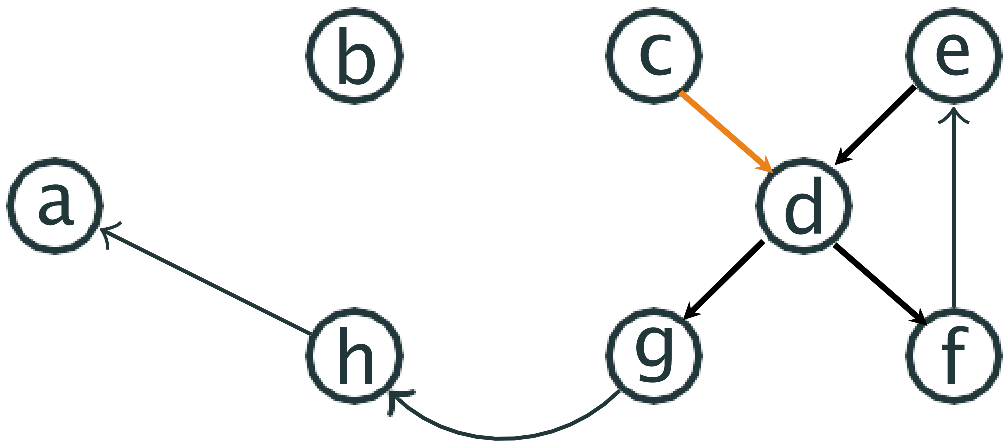
Path so far: a→b→c→h

Example: Directed Graph



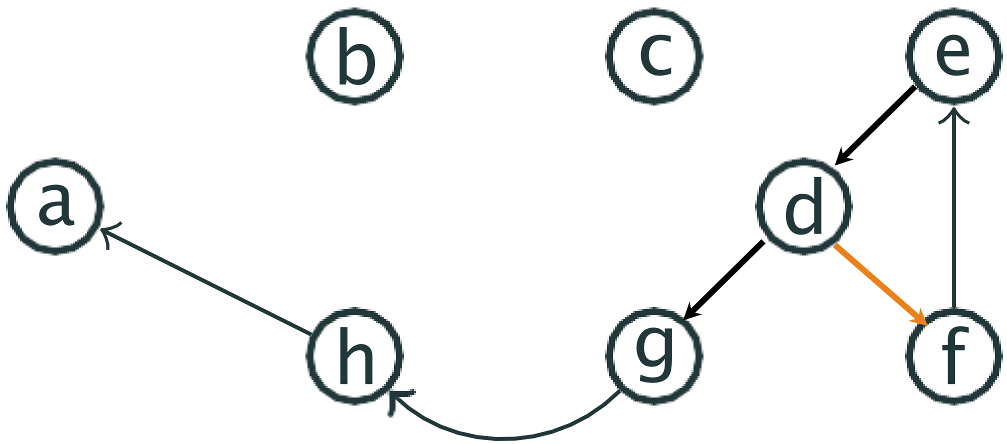
Path so far: a→b→c→h→g

Example: Directed Graph



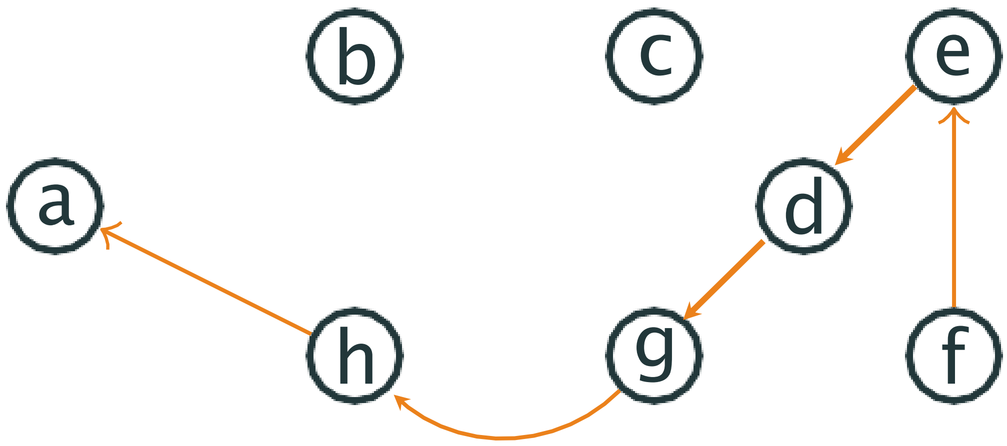
Path so far: a→b→c→h→g→c

Example: Directed Graph



Path so far: a→b→c→h→g→c →d

Example: Directed Graph



Path so far: a→b→c→h→g→c →d …

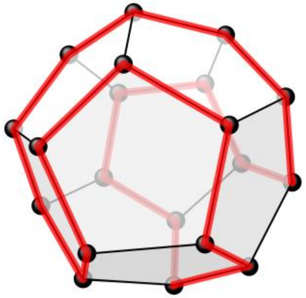
Hamiltonian Cycle

Definition

A Hamiltonian cycle visits every **node** of a graph exactly once and returns to the start node.

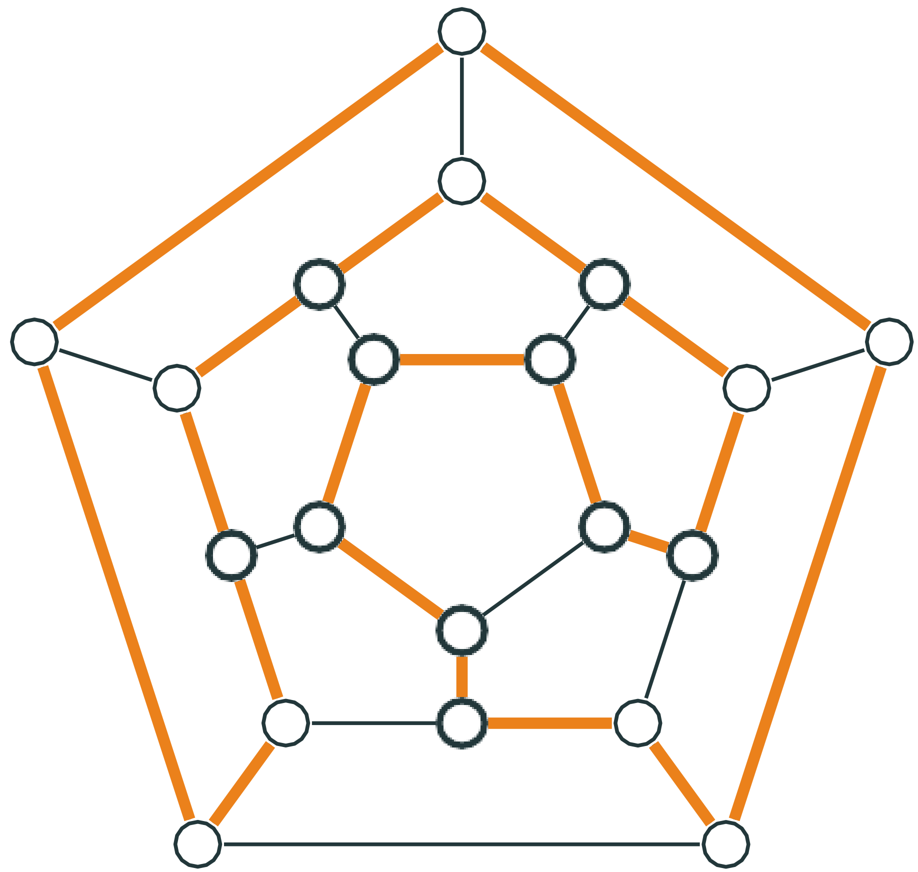
Sir William Rowan Hamilton, 1805–1865

As before, the Hamiltonian **path** is not required to end at the start node, and a cycle is a special case of a path.

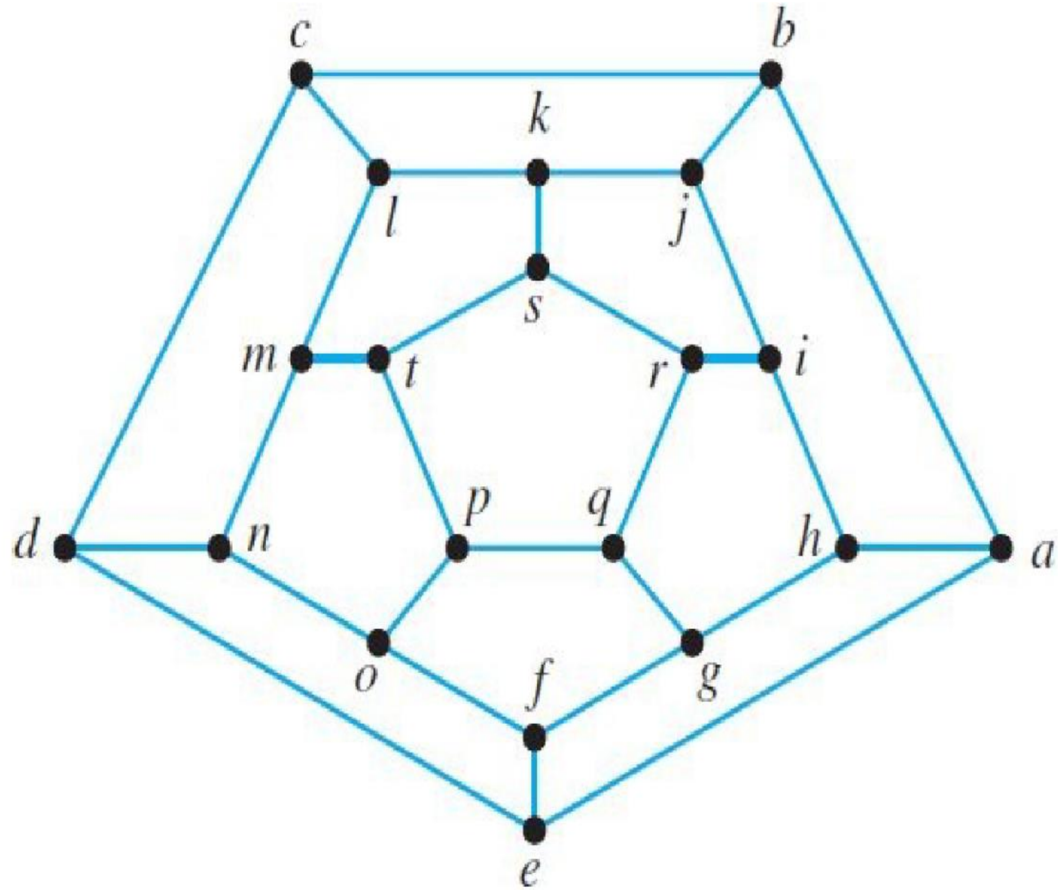


The icosian game

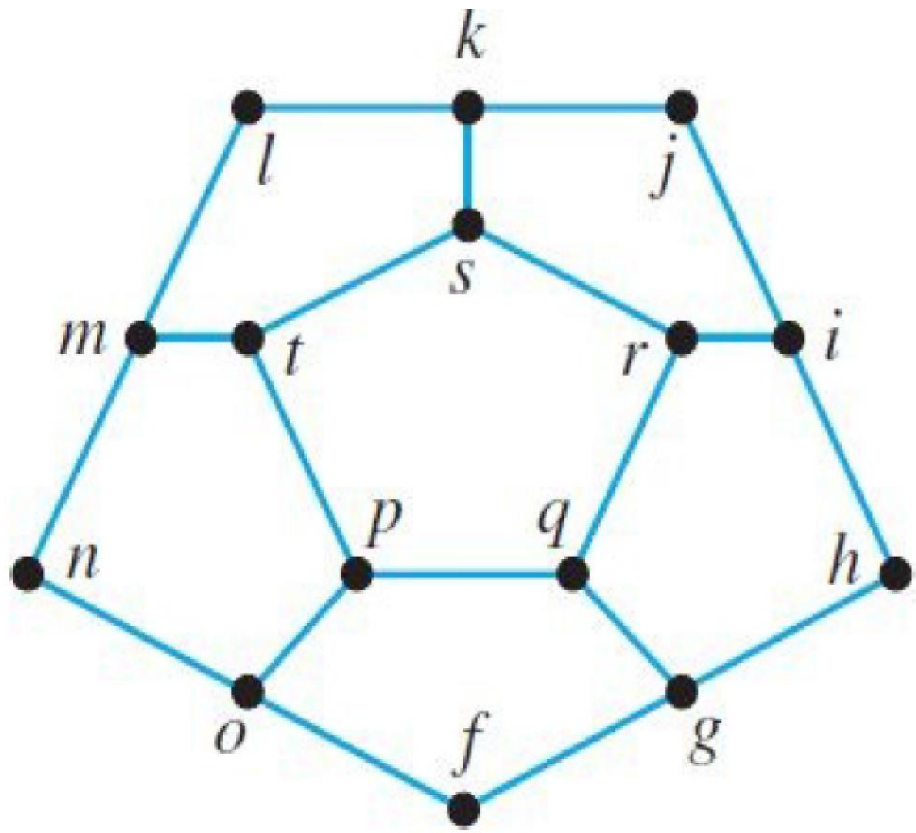
Example: Hamiltonian Path (and Cycle)



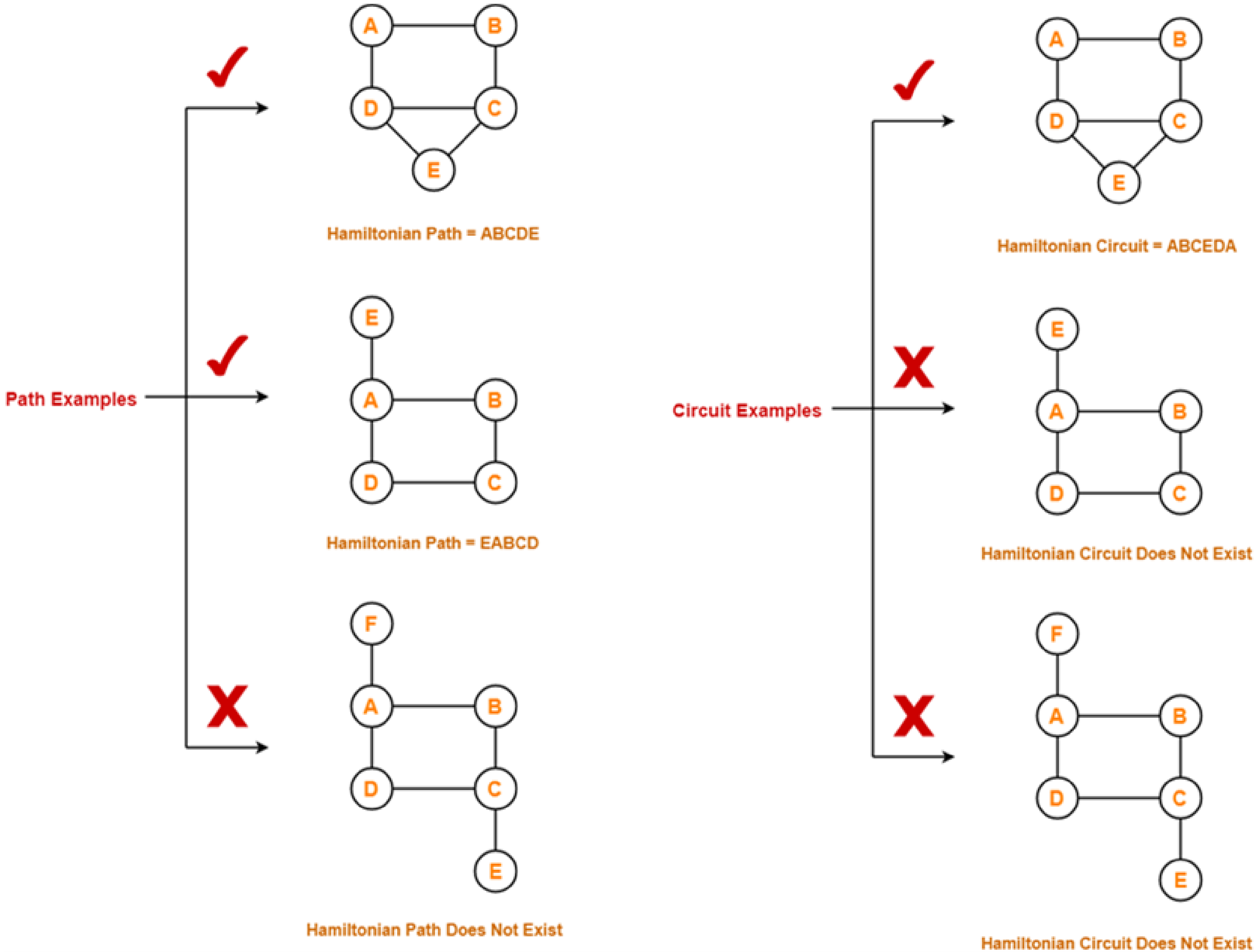
Find Hamiltonian Cycle 1



Find Hamiltonian Cycle 2



The Hamiltonian **cycle** does not exist for this graph: can you see why?

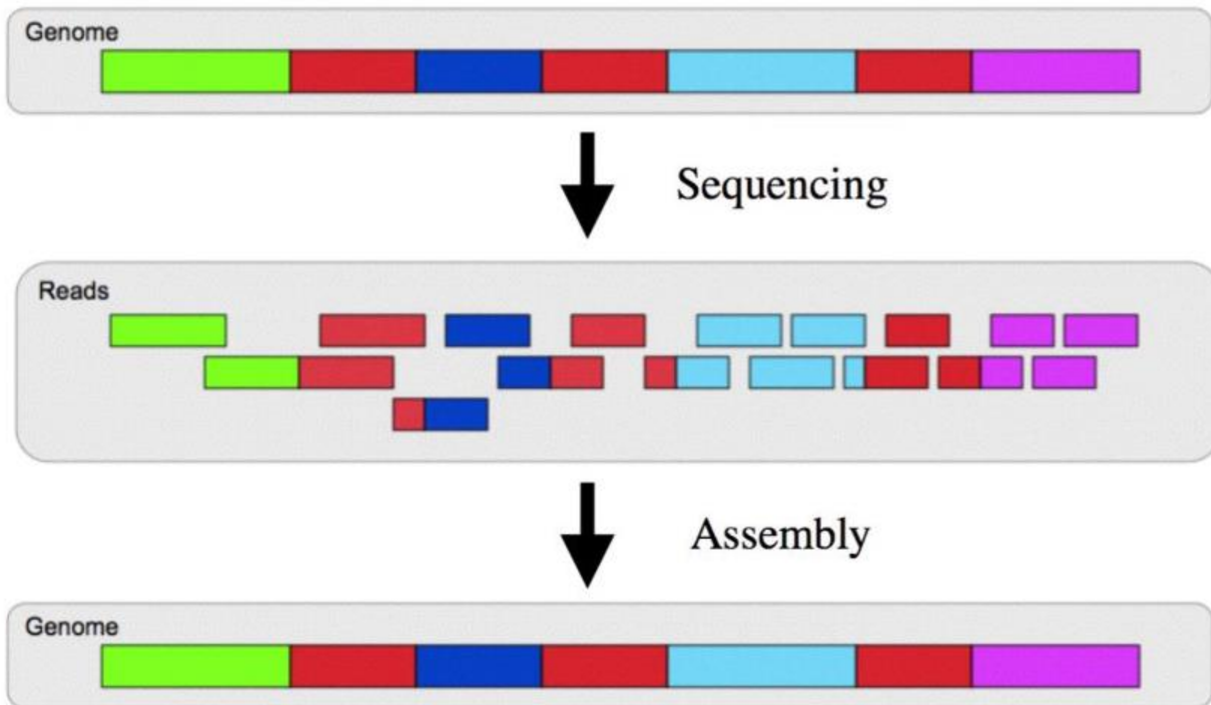

Path Examples
A
B
D
C
E
Hamiltonian Path = ABCDE
E
A
B
D
C
Hamiltonian Path = EABCD
F
A
B
D
C
E
Hamiltonian Path Does Not Exist
Circuit Examples
A
B
D
C
E
Hamiltonian Circuit = ABCEDA
E
A
B
D
C
Hamiltonian Circuit Does Not Exist
F
A
B
D
C
E
Hamiltonian Circuit Does Not Exist

Hamiltonian path: simple criteria?

- There are some existence theorems about Hamiltonian paths, but they don't give a complete characterization of graphs containing Hamiltonian paths (cycles)

- As a result, no polynomial-time algorithm is known for finding Hamiltonian paths!

Genome Assembly problem



Genome Assembly problem: toy example

Find a string whose all substrings of length 3 are:

AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC.

How is this related to paths in graphs?..

All Substrings of Length 3

DISCRETE

DIS

I**SC**

SCR

CRE

RET

ETE

Every two neighbor 3-substrings have
a common part of length 2, called an overlap

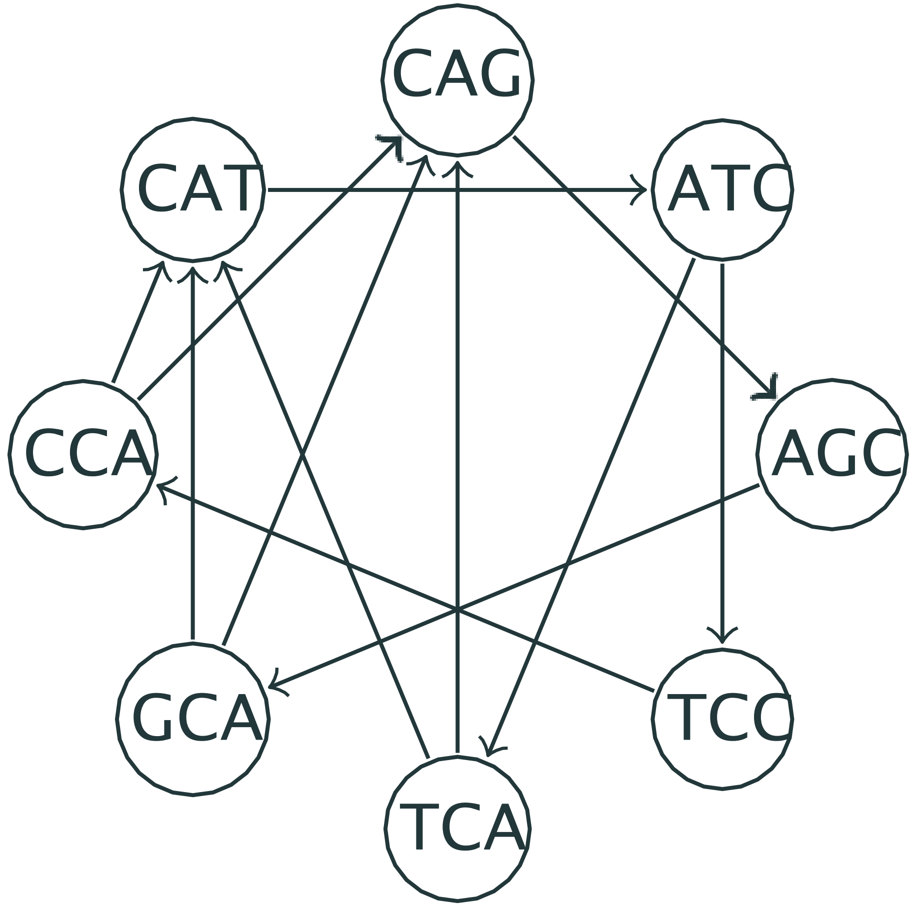
Finding a Permutation

- Goal: Find a string whose all substrings of length 3 are AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC

- Hence, we need to order these 3-substrings such that the overlap between any two consecutive substrings is equal to 2

Overlap Graph

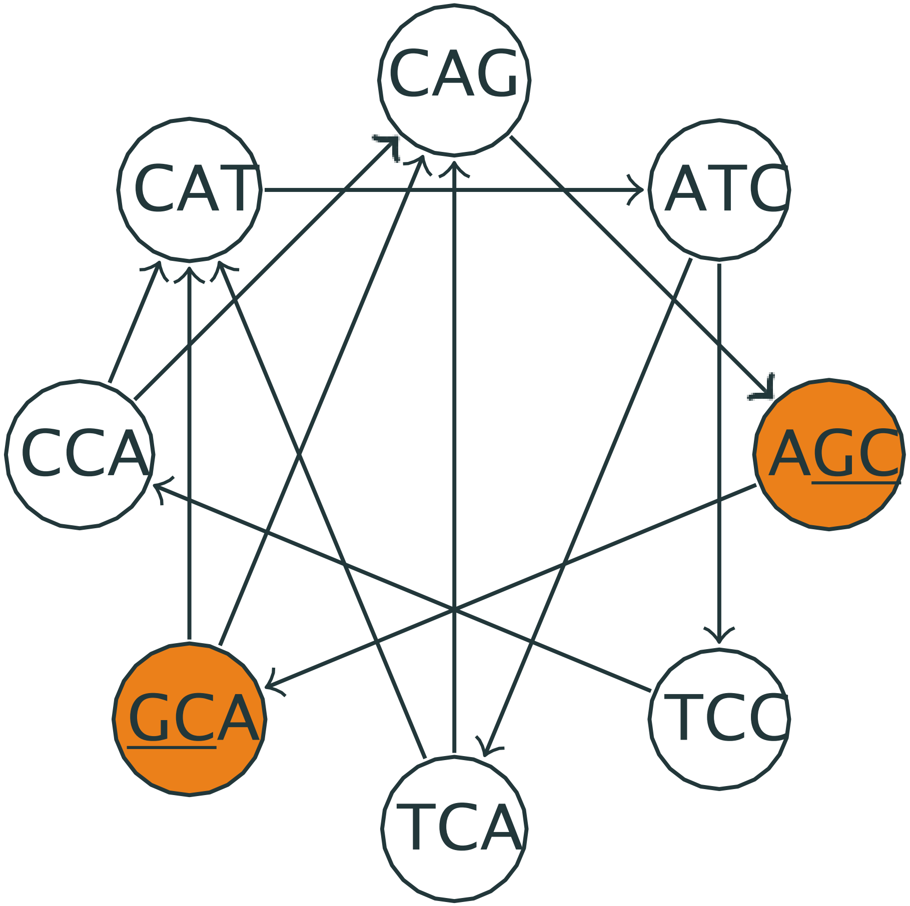
AGC
ATC
CAG
CAT
CCA
GCA
TCA
TCC



Nodes are substrings: short DNA sequence reads

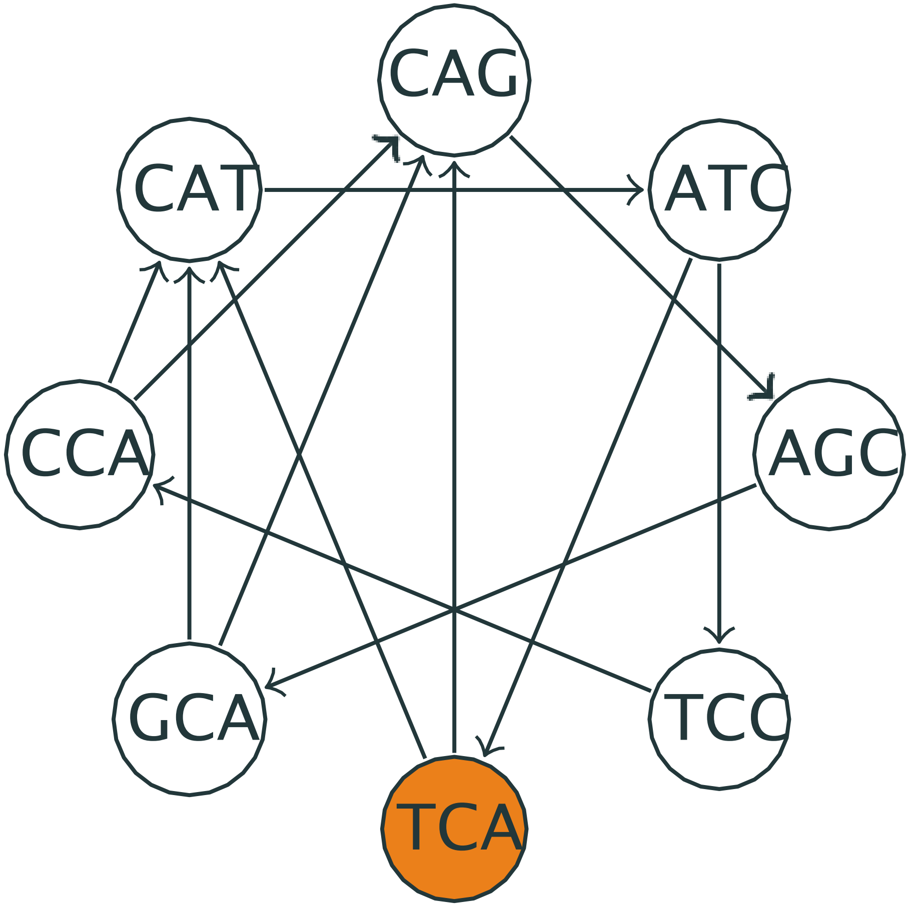
Overlap Graph

AGC
ATC
CAG
CAT
CCA
GCA
TCA
TCC



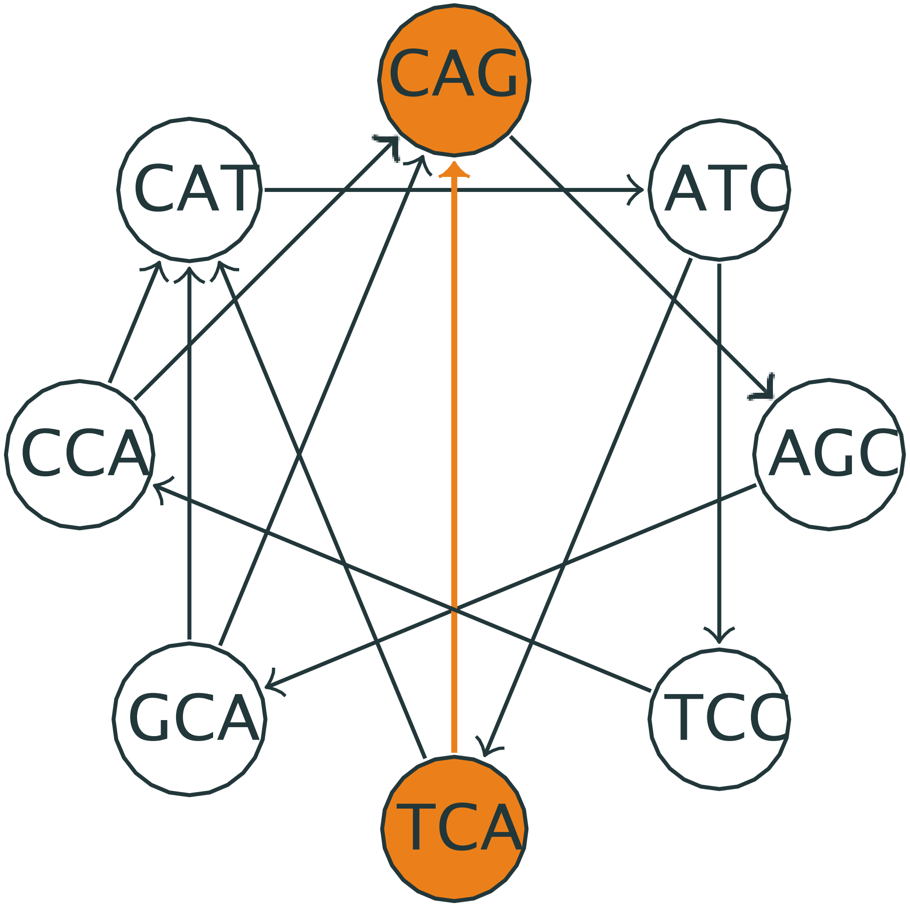
There is an edge from s_1 to s_2 if $s_1[2:3]=s_2[1:2]$

Hamiltonian path in the Overlap Graph



TCA

Hamiltonian path in the Overlap Graph



TCAG ...

We solved Genome Assembly Problem!

- We modeled the problem of genome assembly as Hamiltonian path problem in the overlap graph!

We solved Genome Assembly Problem!

- We modeled the problem of genome assembly as Hamiltonian path problem in the overlap graph!

- But unfortunately we don't have efficient algorithms for solving the Hamiltonian path problem!
- The approach is useless for the case when there are thousands or millions of input sub-strings

Different approach

(De Bruijn; Pevzner, Tang, Waterman)

State-of-the-art genome assemblers

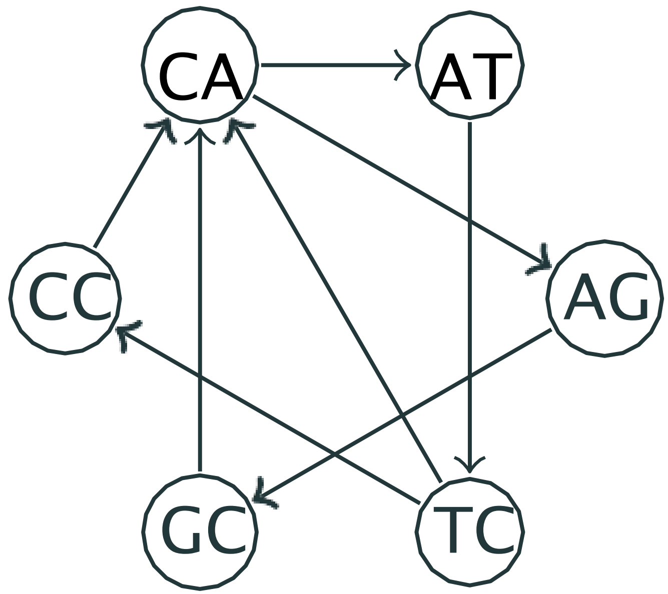
- In the overlap graph, each node corresponds to the input string
- Let's instead represent each edge by the same string, broken into 2 nodes (overlaps):

  E.g., represent the string CAT as an edge

  CA $\rightarrow$ AT

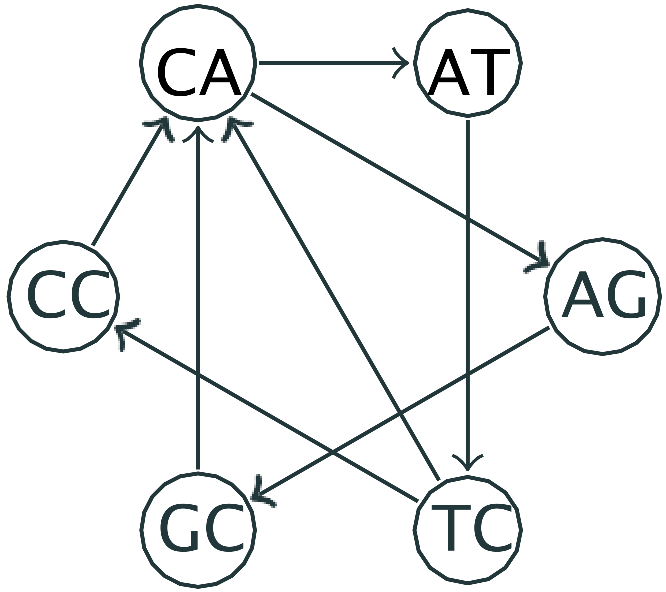
De Bruijn Graph

AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



De Bruijn Graph

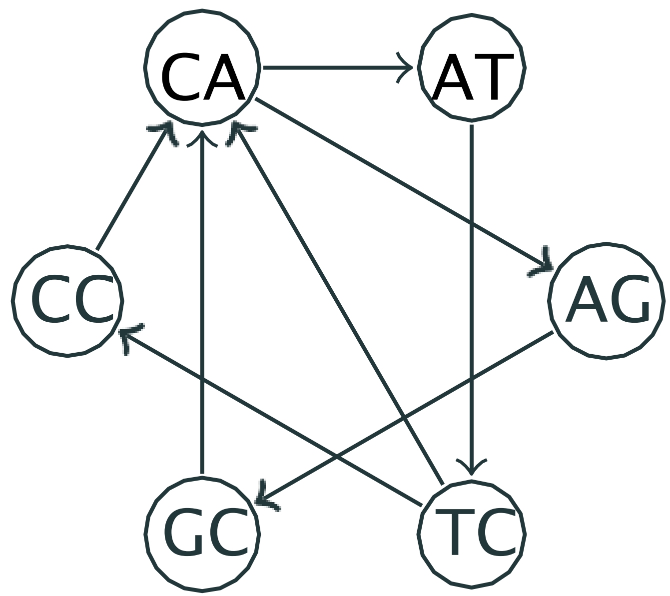
AGC, ATC, CAG, CAT, CCA, GCA,
TCA, TCC



now, we need to find an order of edges

De Bruijn Graph

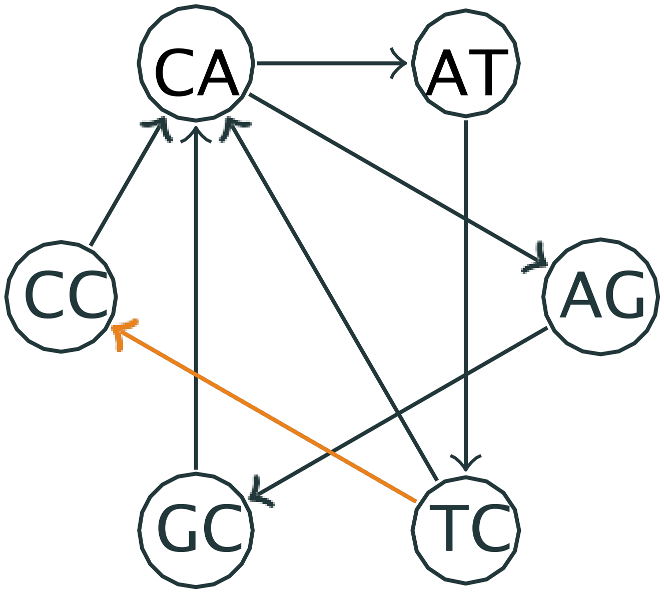
AGC, ATC, CAG, CAT, CCA, GCA,
TCA, TCC



that is, an Eulerian path

De Bruijn Graph

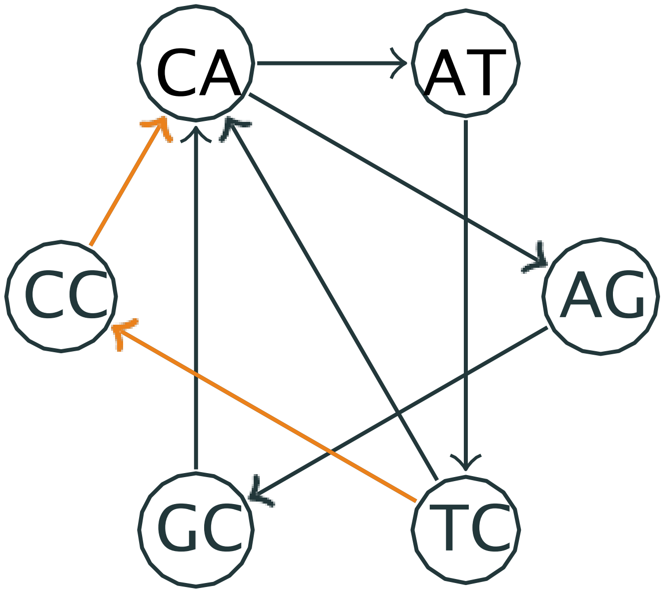
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCC

De Bruijn Graph

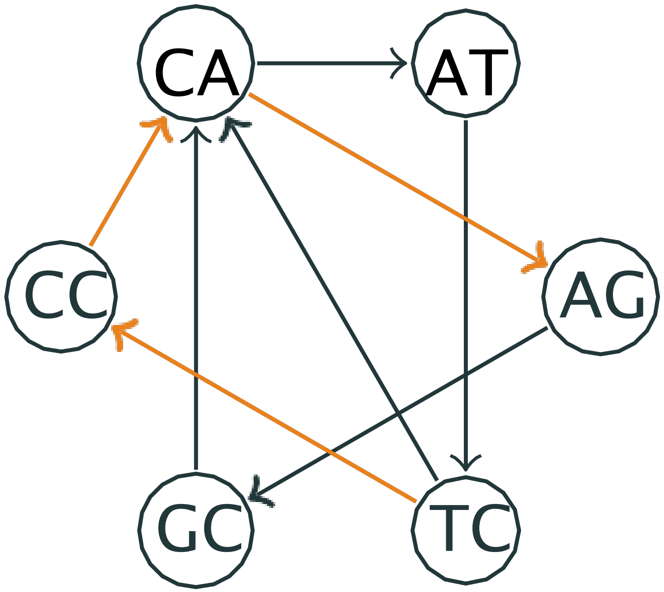
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCCA

De Bruijn Graph

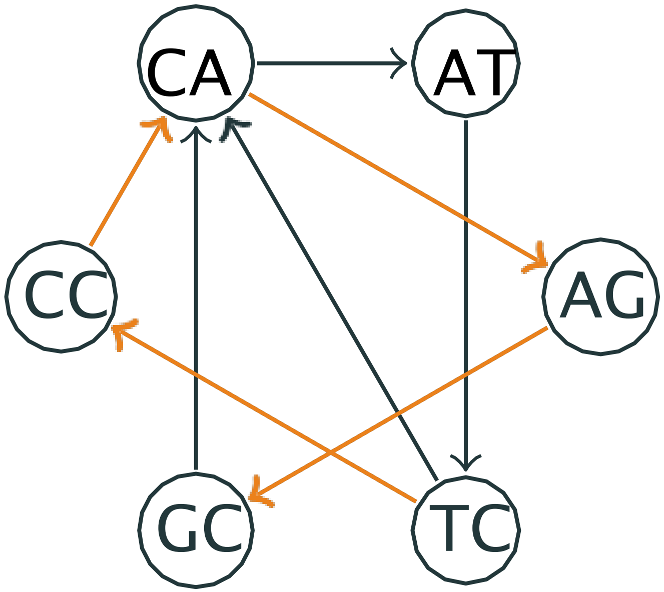
AGC, ATC, CAG, CAT, CCA, GCA,
TCA, TCC



TCCAG

De Bruijn Graph

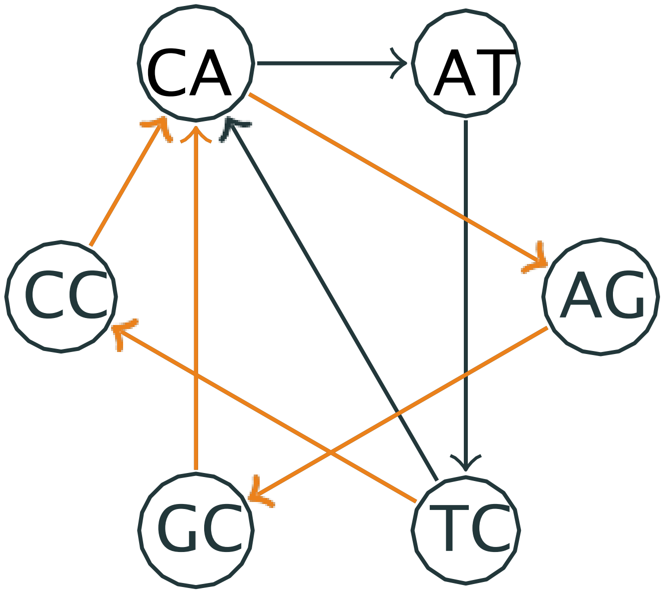
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCCAGC

De Bruijn Graph

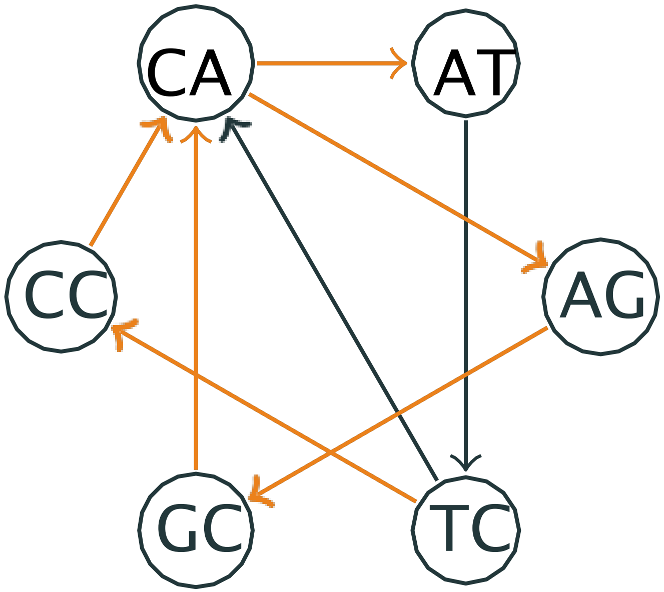
AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCCAGCA

De Bruijn Graph

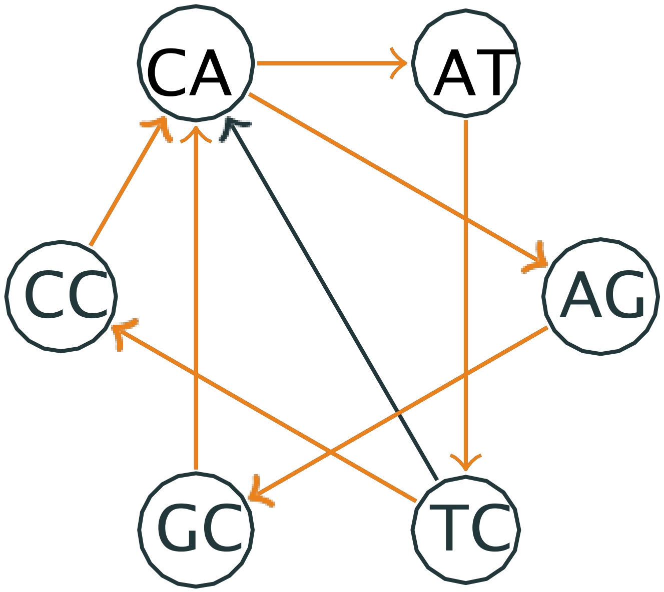
AGC, ATC, CAG, CAT, CCA, GCA,
TCA, TCC



TCCAGCAT

De Bruijn Graph

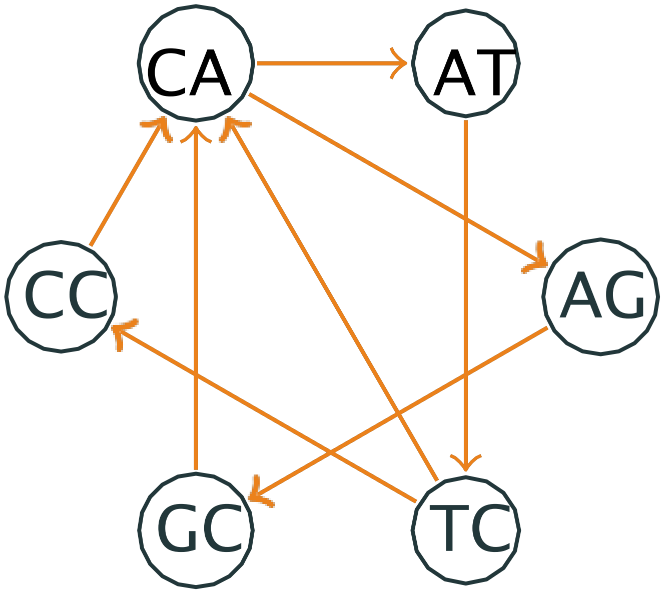
AGC, ATC, CAG, CAT, CCA, GCA,
TCA, TCC



TCCAGCATC

De Bruijn Graph

AGC, ATC, CAG, CAT, CCA, GCA, TCA, TCC



TCCAGCATCA

Genome Assembly: summary

- Eulerian cycle visits every edge exactly once (we have an efficient solution)
- Hamiltonian cycle visits every node exactly once (efficient solution is unknown)

- We were able to solve the problem of Genome Assembly just by changing the graph model!