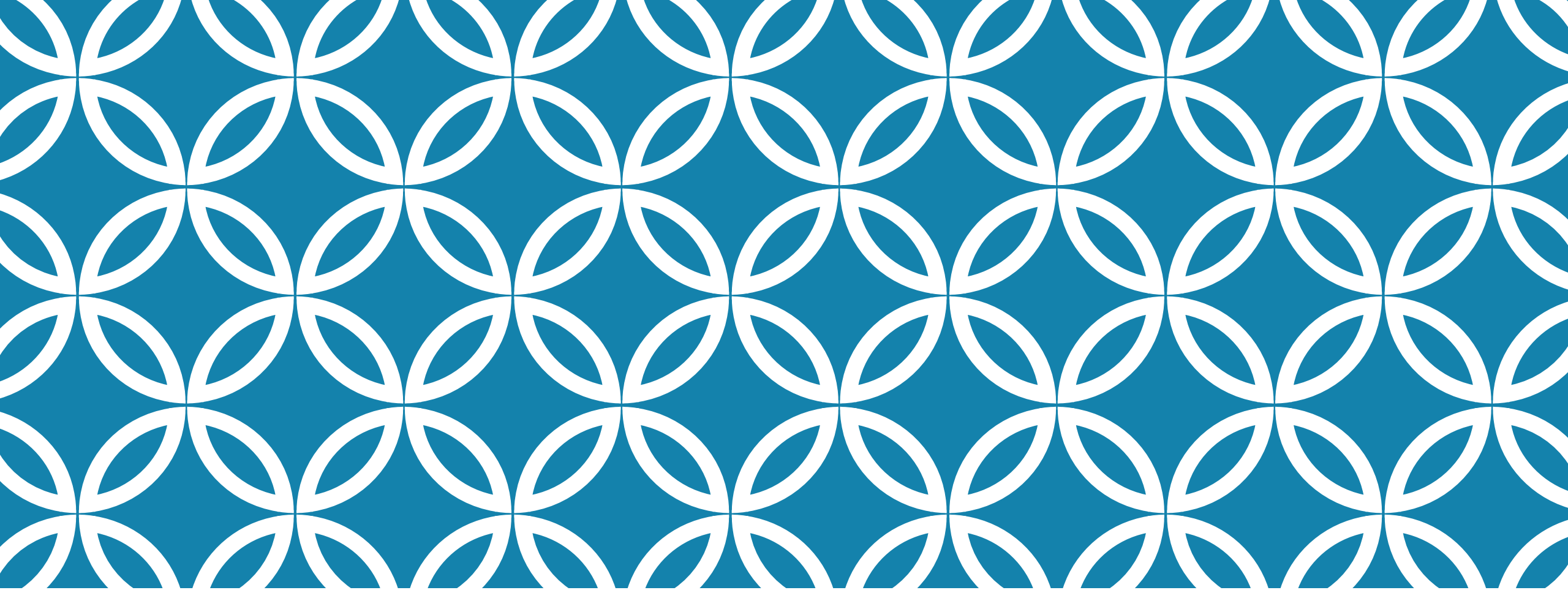


LECTURE 11: INTRODUCTION TO GRAPHS

Harold Soh
harold@comp.nus.edu.sg

ADMINISTRATIVE ISSUES

Problem Set 3 (Speed Demon) is out on Piazza

Due in **2 weeks**

NOT on Kattis.

Submission instructions are on Piazza.

QUIZ 2: ALMOST SORTED

Homer has designed an algorithm called `HomerSort` that sort arrays in increasing order, but it isn't perfect. It produces *almost-sorted* arrays. In particular, each element is at most k positions away from its correctly sorted position. As an example, given an array $A = [7, 3, 1, 4, 2, 6, 5]$, `HomerSort` outputs a result $R = [1, 4, 3, 2, 6, 5, 7]$. Notice that no element in R is more than $k = 2$ away from its correct sorted position.

Your task is to write an algorithm that correctly sorts the output of `HomerSort`. More precisely, write a function `fixHomerSort(R, k)` that accepts an almost-sorted array R and an integer parameter k, and sorts R in increasing order. For example, given $R = [1, 4, 3, 2, 6, 5, 7]$ and $k = 2$, your method sorts R into $[1, 2, 3, 4, 5, 6, 7]$.

Problem 2.a. [7 points] For this first sub-problem, consider the special case where $k = 1$. **Describe the most efficient algorithm you can think of to fix the output of `HomerSort` when $k = 1$.** *Prioritize time over space efficiency.* Write the time and space complexity of your method below.

QUIZ 2: ALMOST SORTED

Problem 2.b. [13 points] Now, consider the more general situation where k is a given parameter. **Describe the most efficient algorithm you can think of.** *Prioritize time over space efficiency.* Write the time and space complexity of your method below.
Hint: *A good solution uses $O(k)$ space.*

QUIZ 2: LCA

In CS2040S, we learnt about ancestors and descendants in Binary Trees. For this problem, we will focus on the *lowest common ancestor* (LCA). The LCA of two nodes a and b is the node furthest from the root that is an ancestor of both a and b. For example, given the Binary Tree shown in Fig 1, the LCA for 2 and 6 is 5. The LCA of 12 and 2 is 10. Your task is to design algorithms that compute the LCA given a Binary Tree T with n nodes and has height h.

Problem 3.a. [15 points] **Describe the most efficient algorithm to find the LCA given two nodes a and b in a Binary Tree T.** Prioritize time over space efficiency. **Each node does *not* have a parent pointer. Your method should work on any binary tree and not just binary search trees.**

Write the time and space complexity of your method below. Recall that T has n nodes and height h.

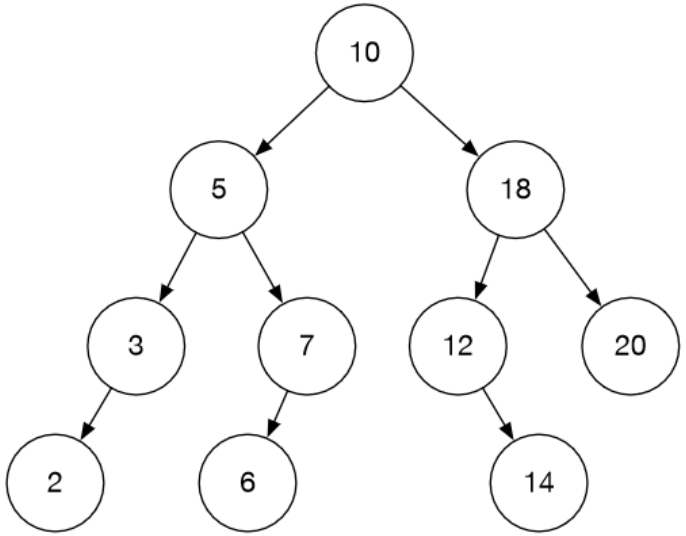


Figure 1: An example Binary Tree.

QUIZ 2: LCA

Problem 3.b. [10 points] Consider now that T is a **Balanced Binary Search Tree** (BBST) with *unique* keys. Can you design a more efficient algorithm to find the LCA? Prioritize time over space efficiency. Again, **nodes do *not* have parent pointers**.

Hint: *Try traversing to two different nodes from the root in Fig 1 or your own BBST.*

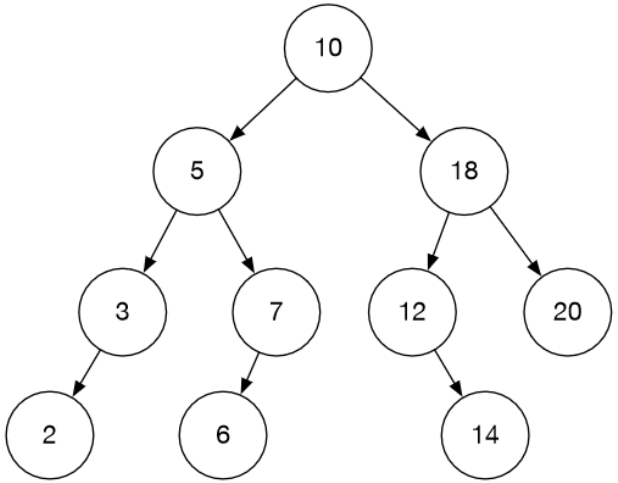


Figure 1: An example Binary Tree.

MID-SEMESTER SURVEY

Your feedback is important!

https://forms.gle/S59oACKSXcK2D6367

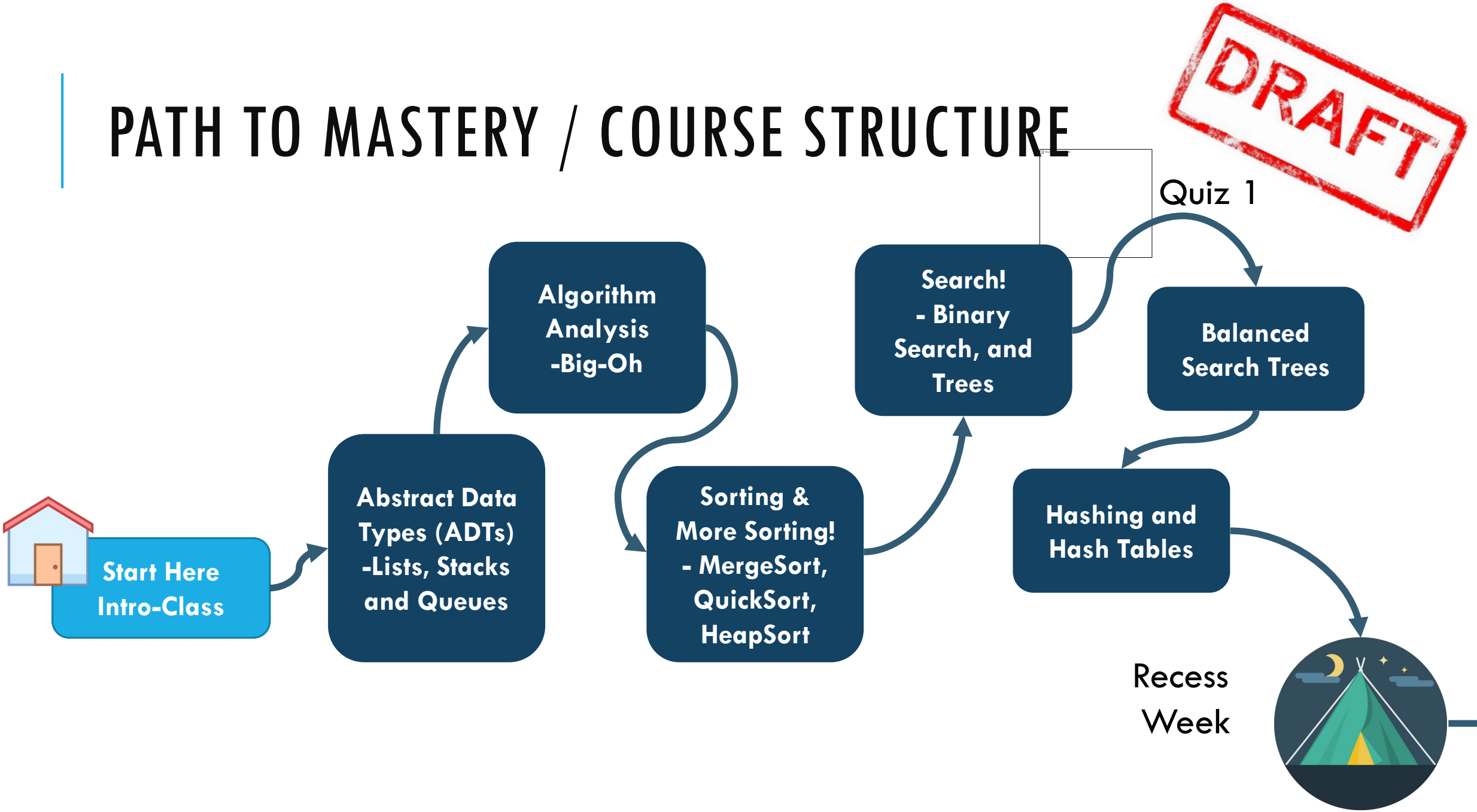
Completely Anonymous

Will take less than 5 minutes.

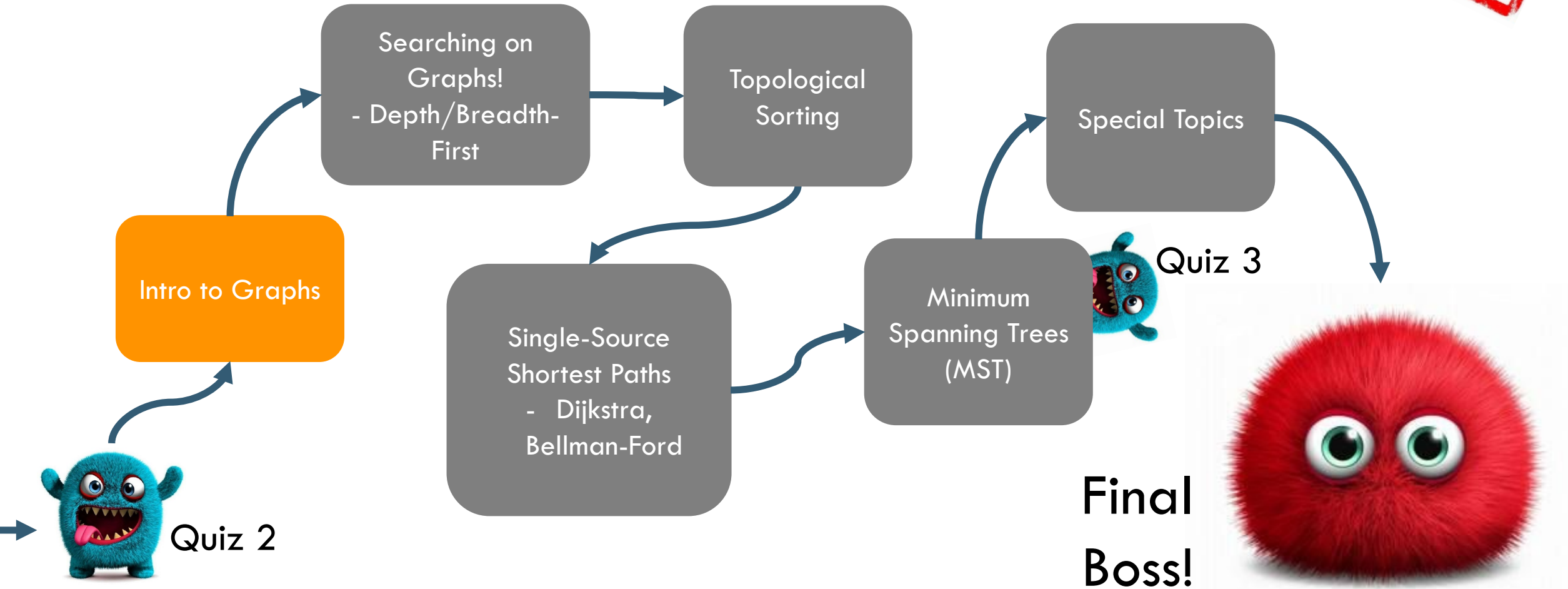
- Please just submit 1 form

QUESTIONS?

PATH TO MASTERY / COURSE STRUCTURE



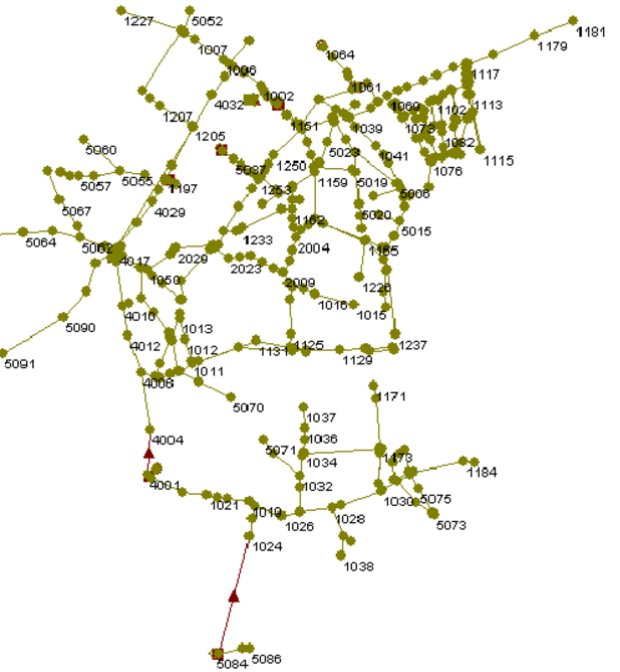
PATH TO MASTERY / COURSE STRUCTURE

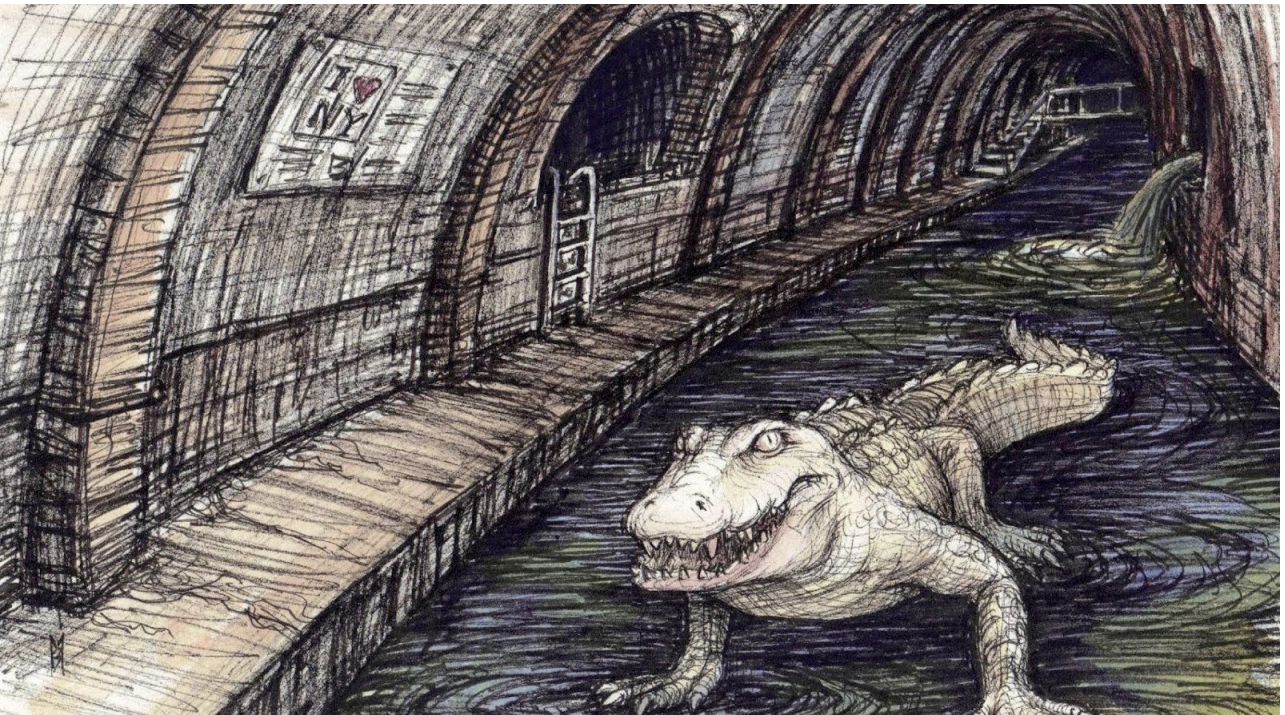


LEARNING OUTCOMES

By the end of this session, students should be able to:

- Describe the **graph** as a mathematical and data structure.
- Explain the **two primary methods to represent graphs** and the **key differences.**
- Give **examples** of where **graphs can be applied.**





PROBLEM: FINDING HERBERT.

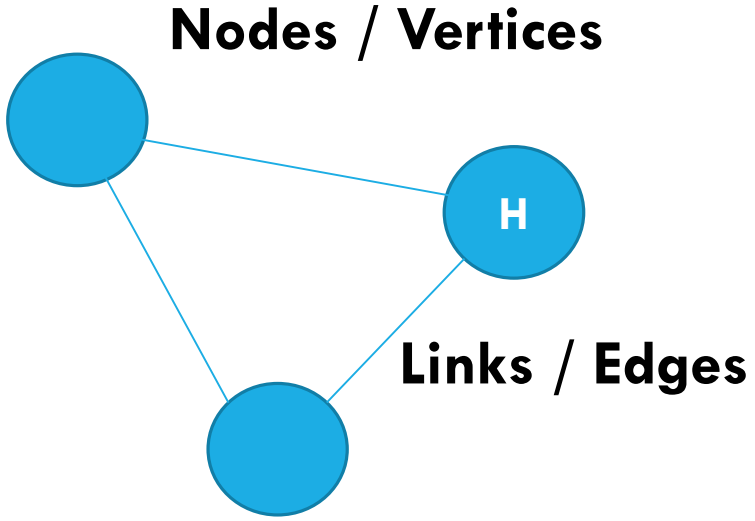
Herbert has gone missing!

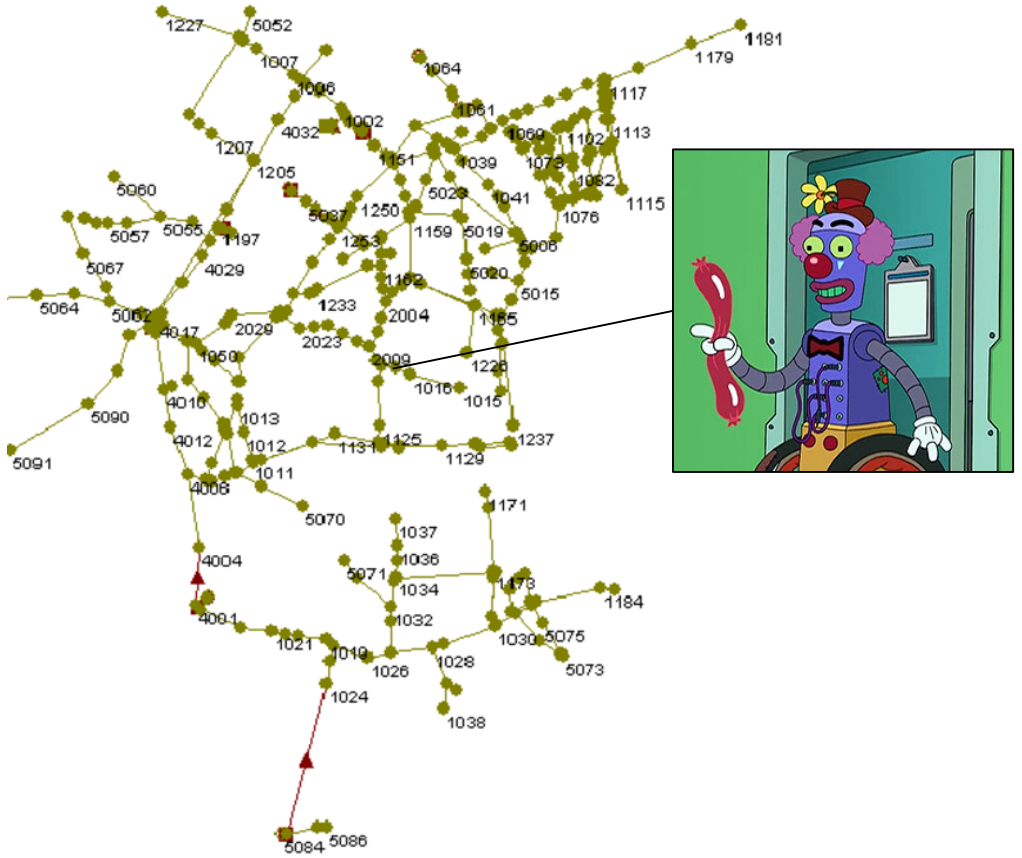
Last sighting: in the sewer system.

How can we *systematically* search for Herbert… before he gets destroyed by an alligator?

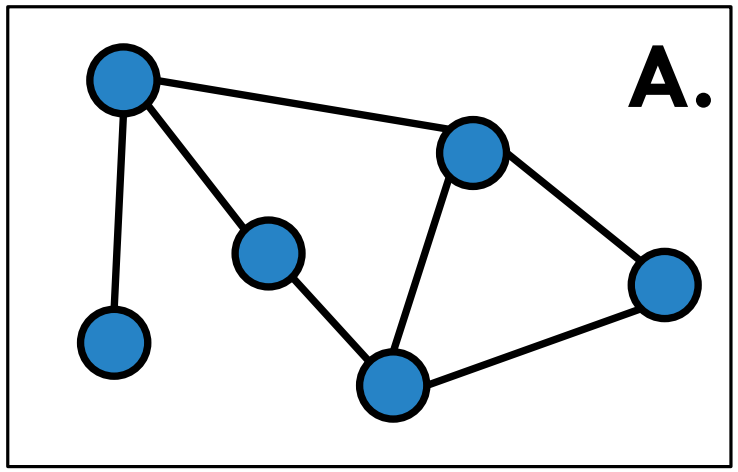
Lots of problems can be modeled and solved as graph problems

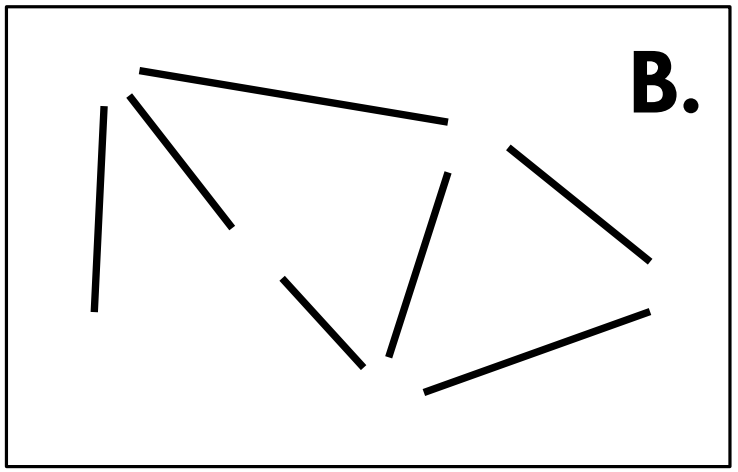
MODEL THE SEWER AS A GRAPH

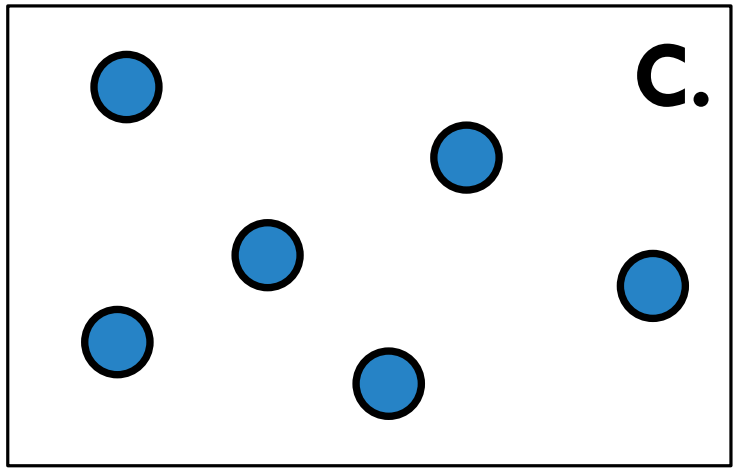


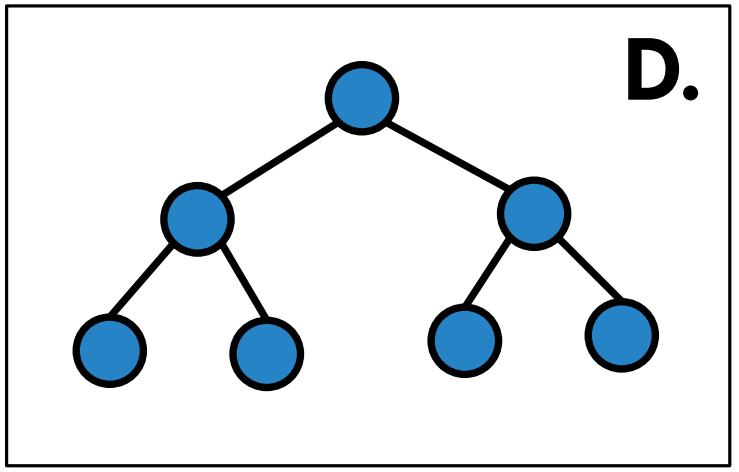


WHICH OF THESE IS **NOT** A GRAPH?





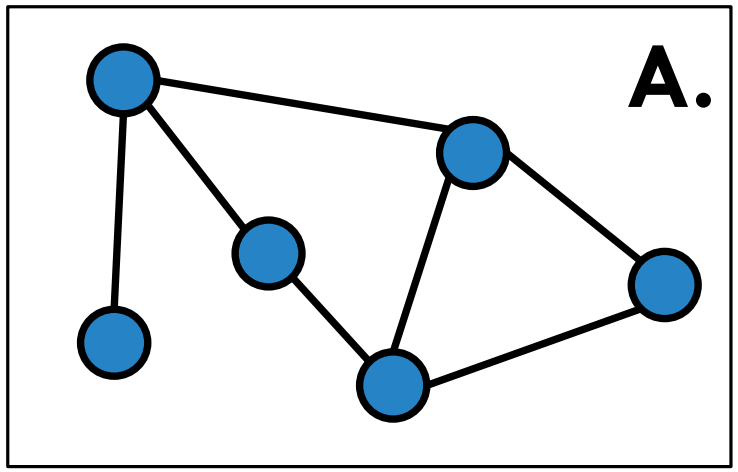


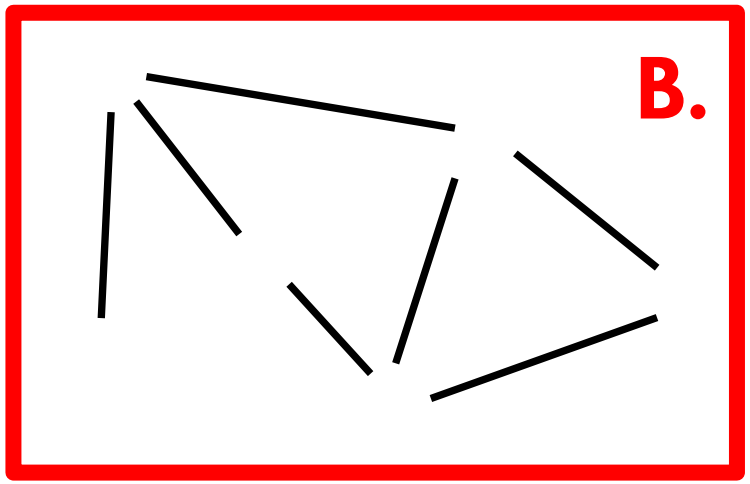


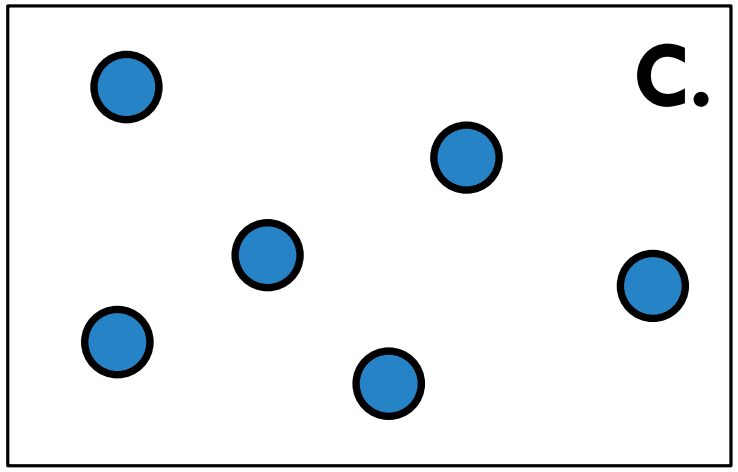
Which is **not** a graph?

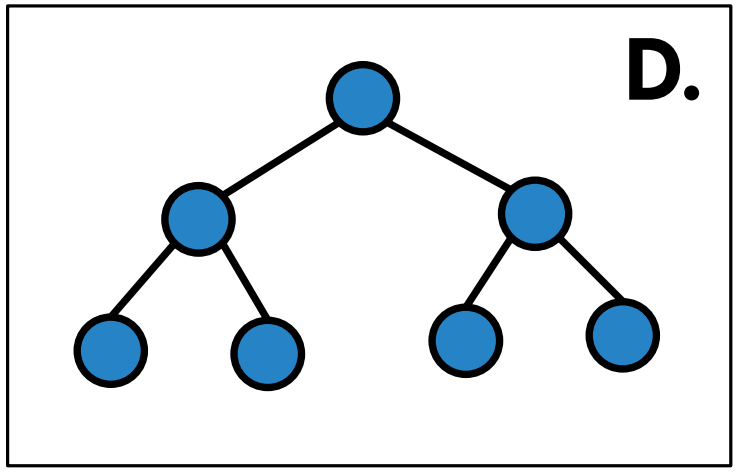
A. A
B. B
C. C
D. D
E. C & B
F. All are graphs
G. None are graphs

WHICH OF THESE IS **NOT** A GRAPH?









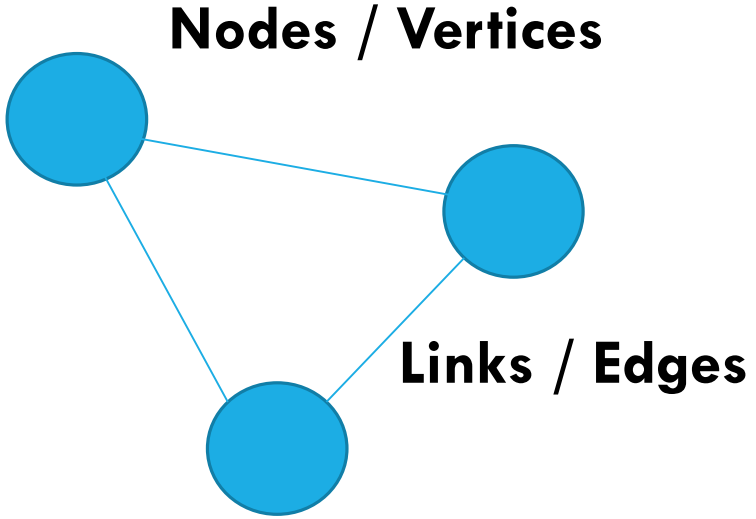
Which is **not** a graph?

A. A
B. B
C. C
D. D
E. C & B
F. All are graphs
G. None are graphs

UNDIRECTED GRAPHS: A FORMAL DEFINITION

Graph $G = \langle V, E \rangle$ ("a tuple of two sets")

- V is a set of nodes
- E is a set of edges
  - $E \subseteq \{ (v, w) : v, w \in V \}$



Simple Graph:

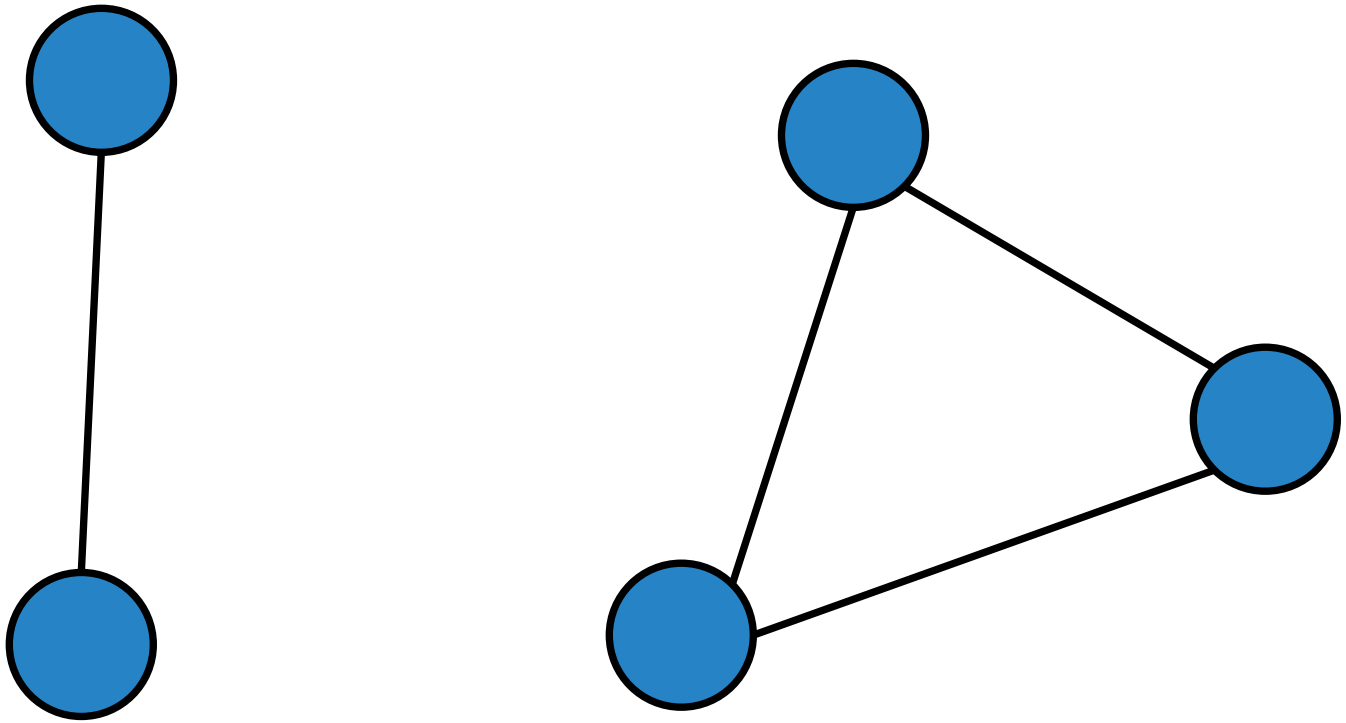
- $e = (v, w)$ for $v \neq w$ ("no self loops")
- $\forall e_1, e_2 \in E : e_1 \neq e_2$ ("only one edge per pair of nodes")

IS THIS A SIMPLE GRAPH?

Simple Graph:

$e = (v, w)$ for $v \neq w$ ("no self loops")

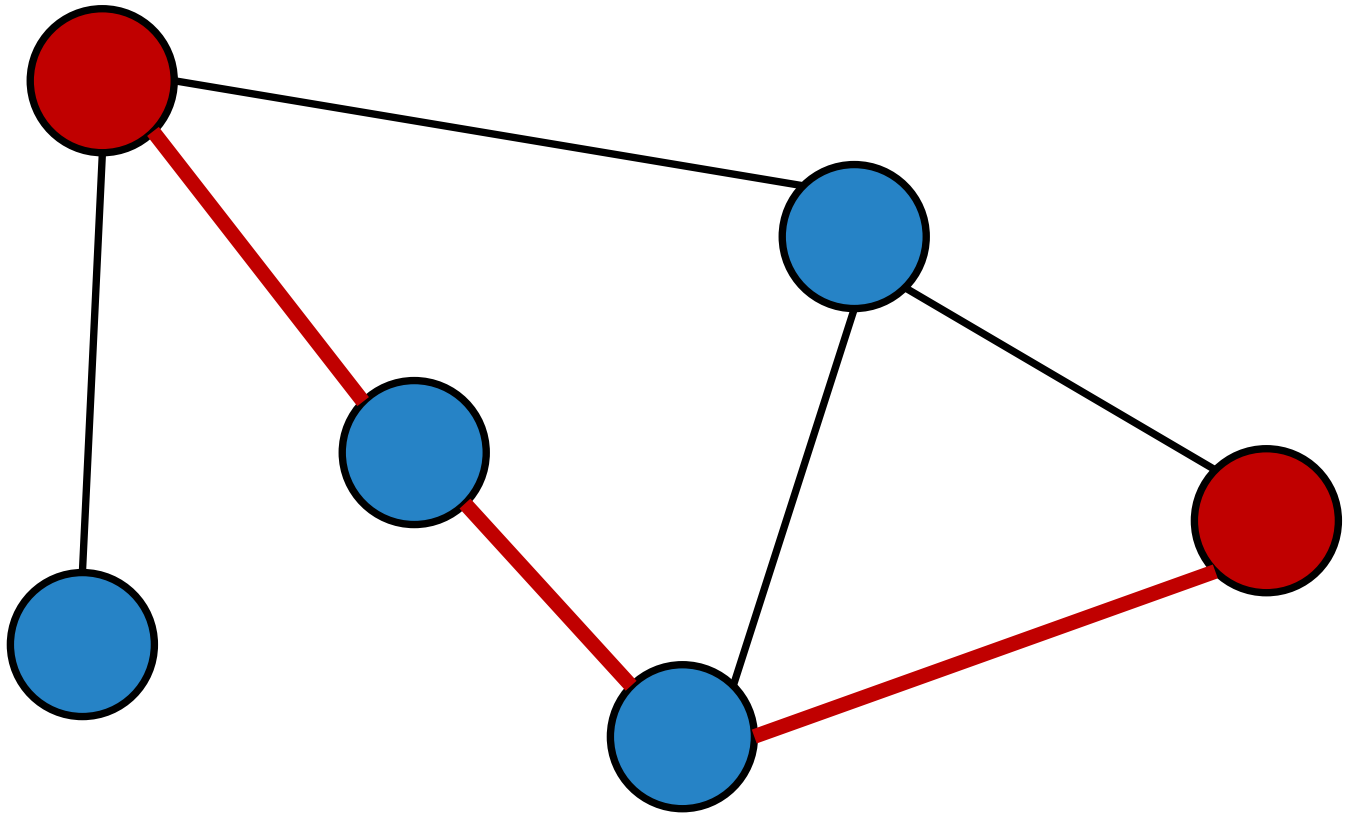
$\forall e_1, e_2 \in E : e_1 \neq e_2$ ("only one edge per pair of nodes")



Yes, it is. But it is not **connected**.

CONNECTED GRAPHS

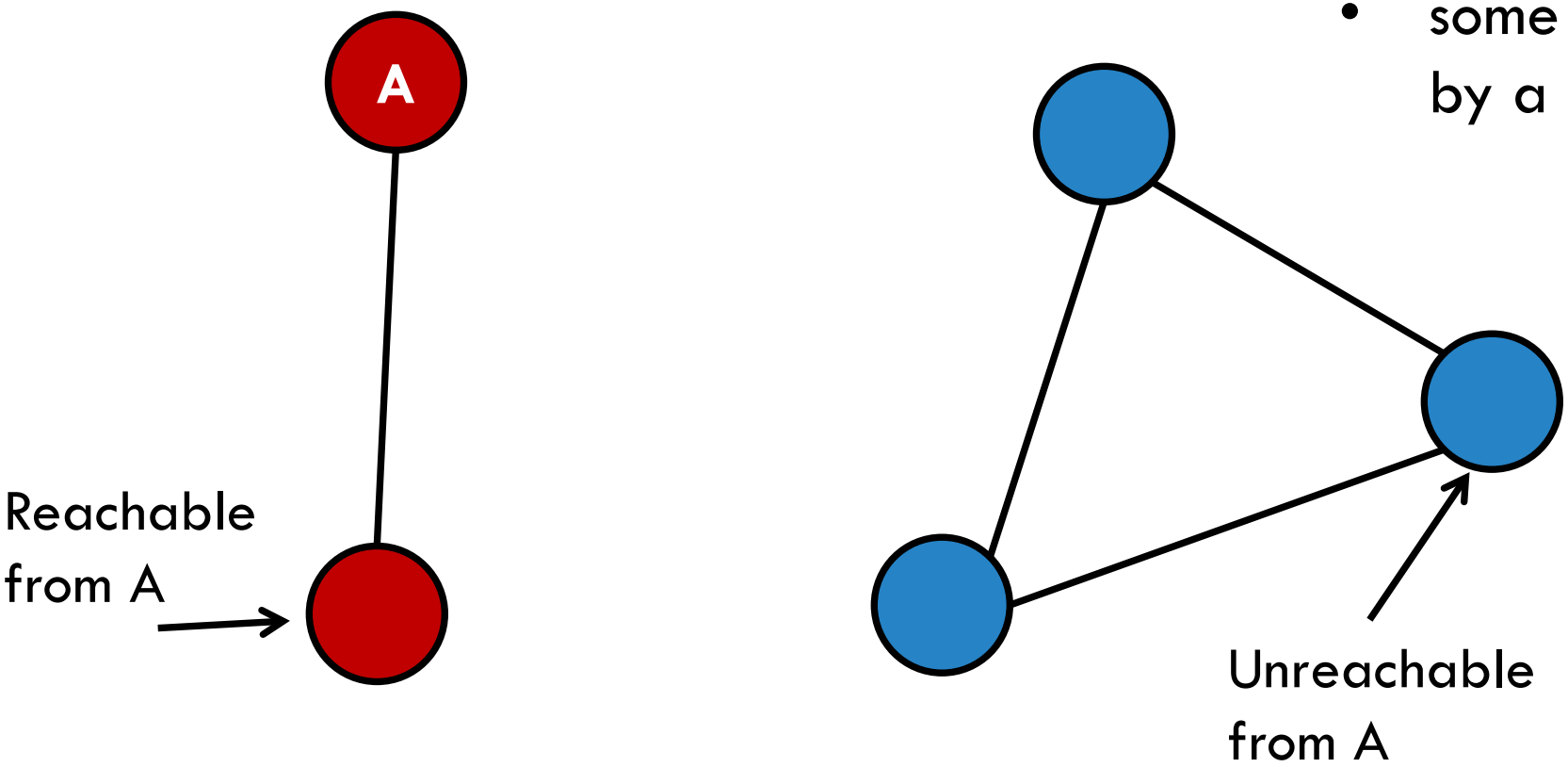
Connected Graph: Every pair of nodes is connected by a **path**.



DISCONNECTED GRAPH

This graph has 2 **connected components**

- some pair is not connected by a path



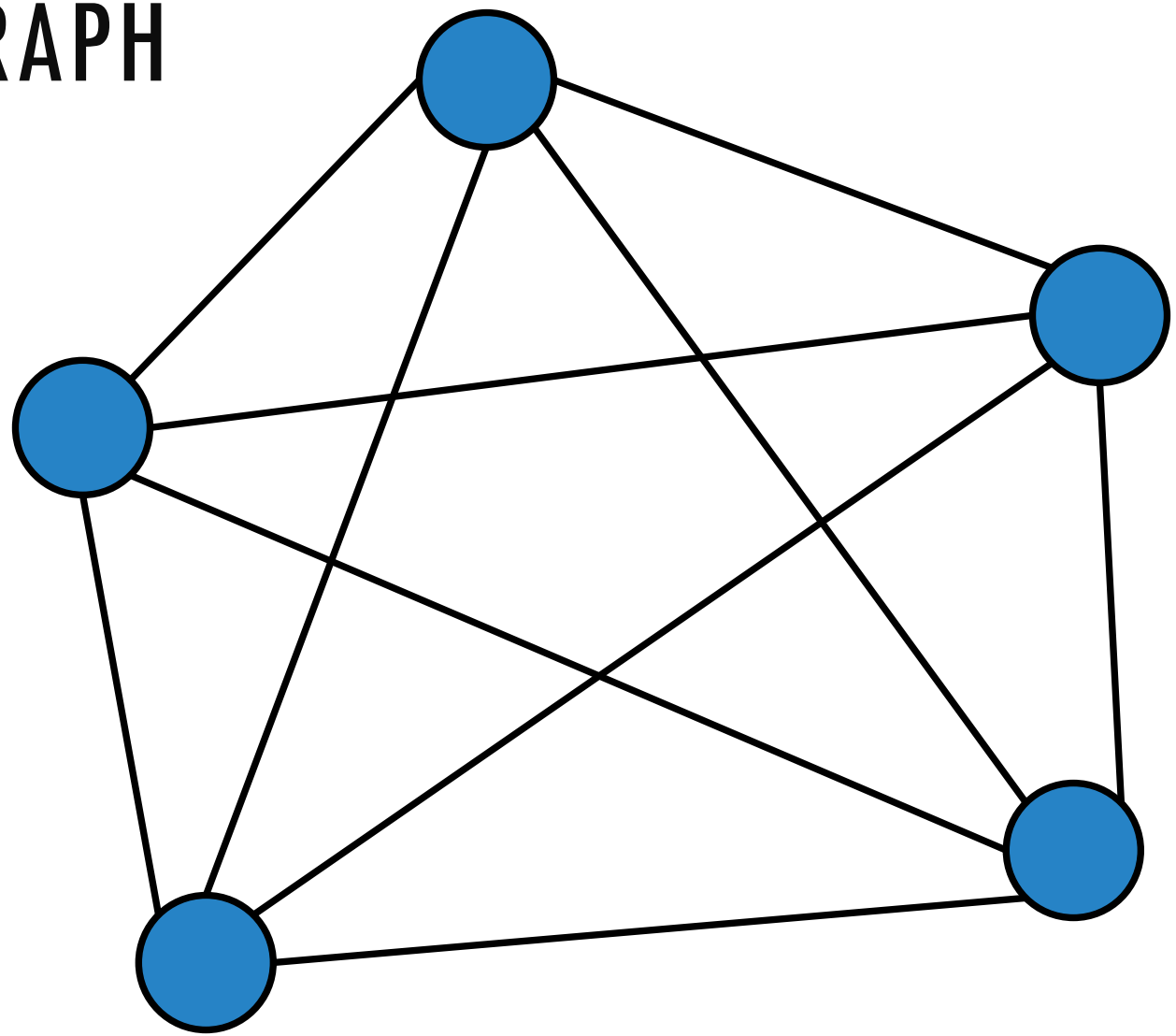
CLIQUE: A COMPLETE GRAPH

Fully connected

All pairs connected by edges

Question: In a clique, what is the longest path from one node to another?

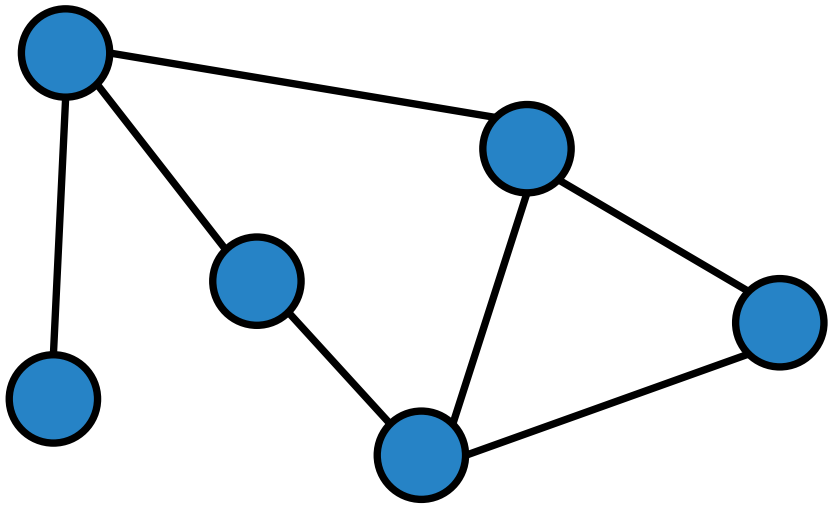
One. Every node is only 1 hop from any other node. The **diameter** of a clique = 1



DIAMETER OF A GRAPH

Diameter = maximum **distance** between any two (different) nodes

The **distance** between two nodes is the *length of the **shortest path** between the two nodes*



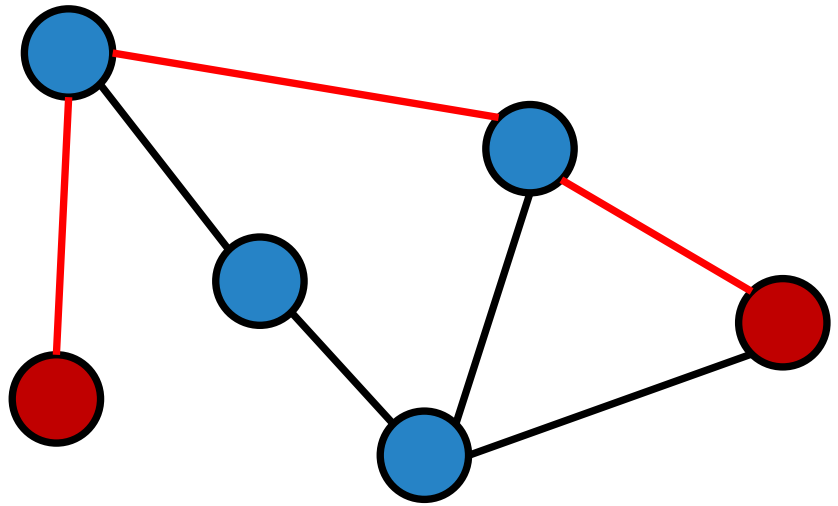
What is the diameter of the graph on the left?

A. 1
B. 2
C. 3
D. 4
E. 5
F. Infinity (and Beyond!)

DIAMETER OF A GRAPH

Diameter = maximum **distance** between any two (different) nodes

The **distance** between two nodes is the *length of the **shortest path** between the two nodes*

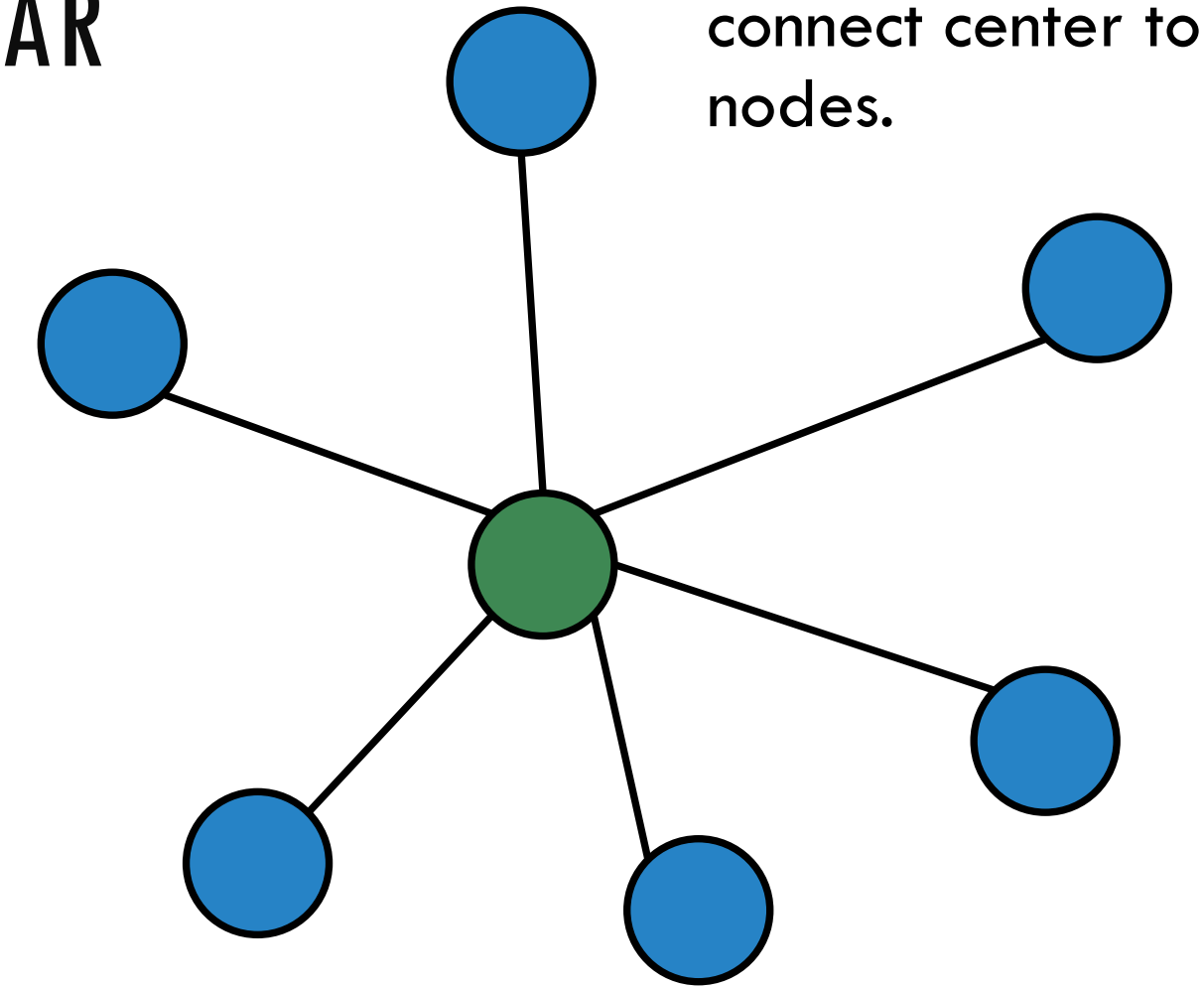


What is the diameter of the graph on the left?

A. 1
B. 2
C. 3
D. 4
E. 5
F. Infinity (and Beyond!)

STAR

One central node. All edges connect center to the other nodes.

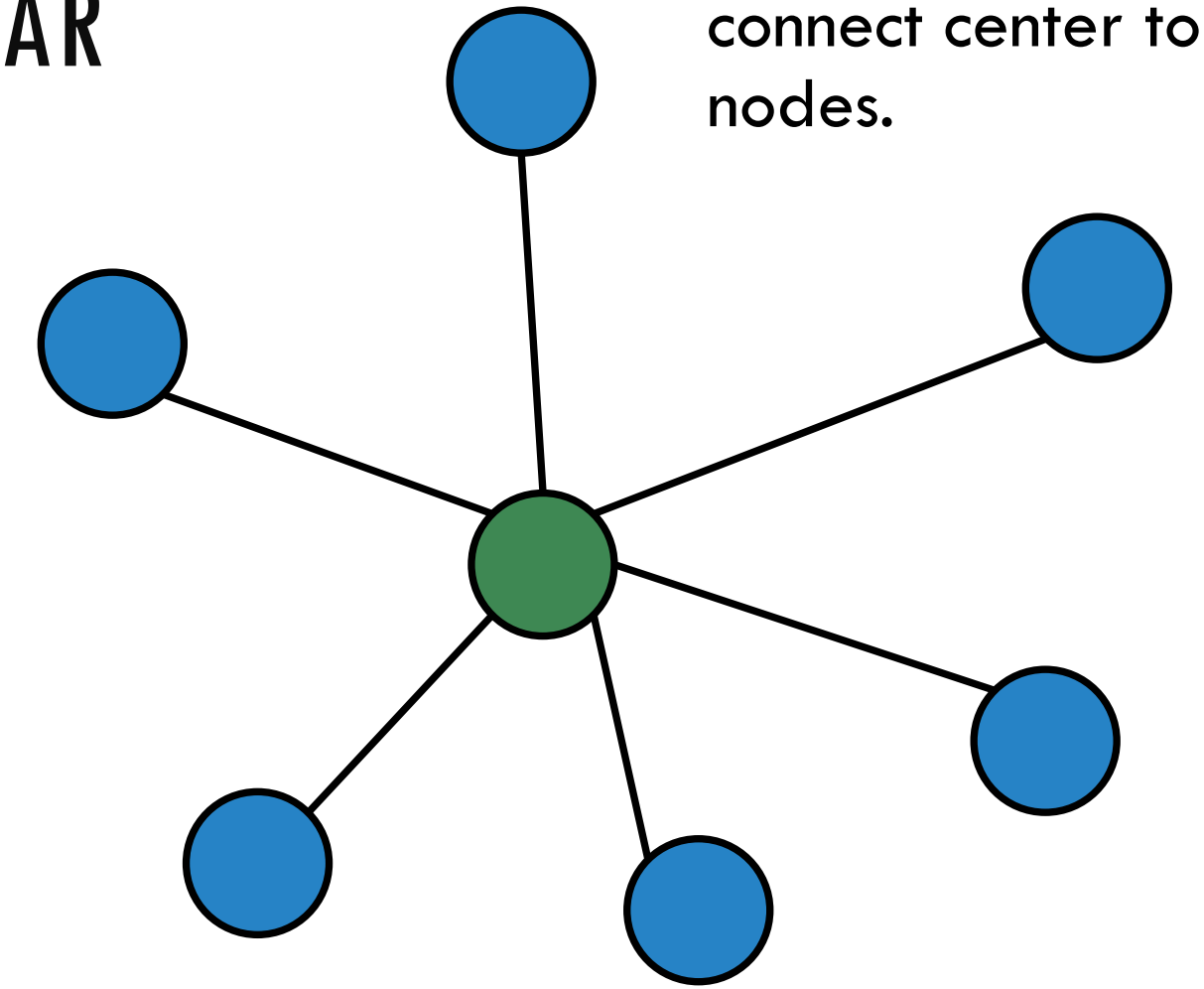


What is the diameter of a star?

A. 1
B. 2
C. 3
D. 4
E. Which star? the sun? 1.4×10^6 km

STAR

One central node. All edges connect center to the other nodes.



What is the diameter of a star?

A. 1

B. 2

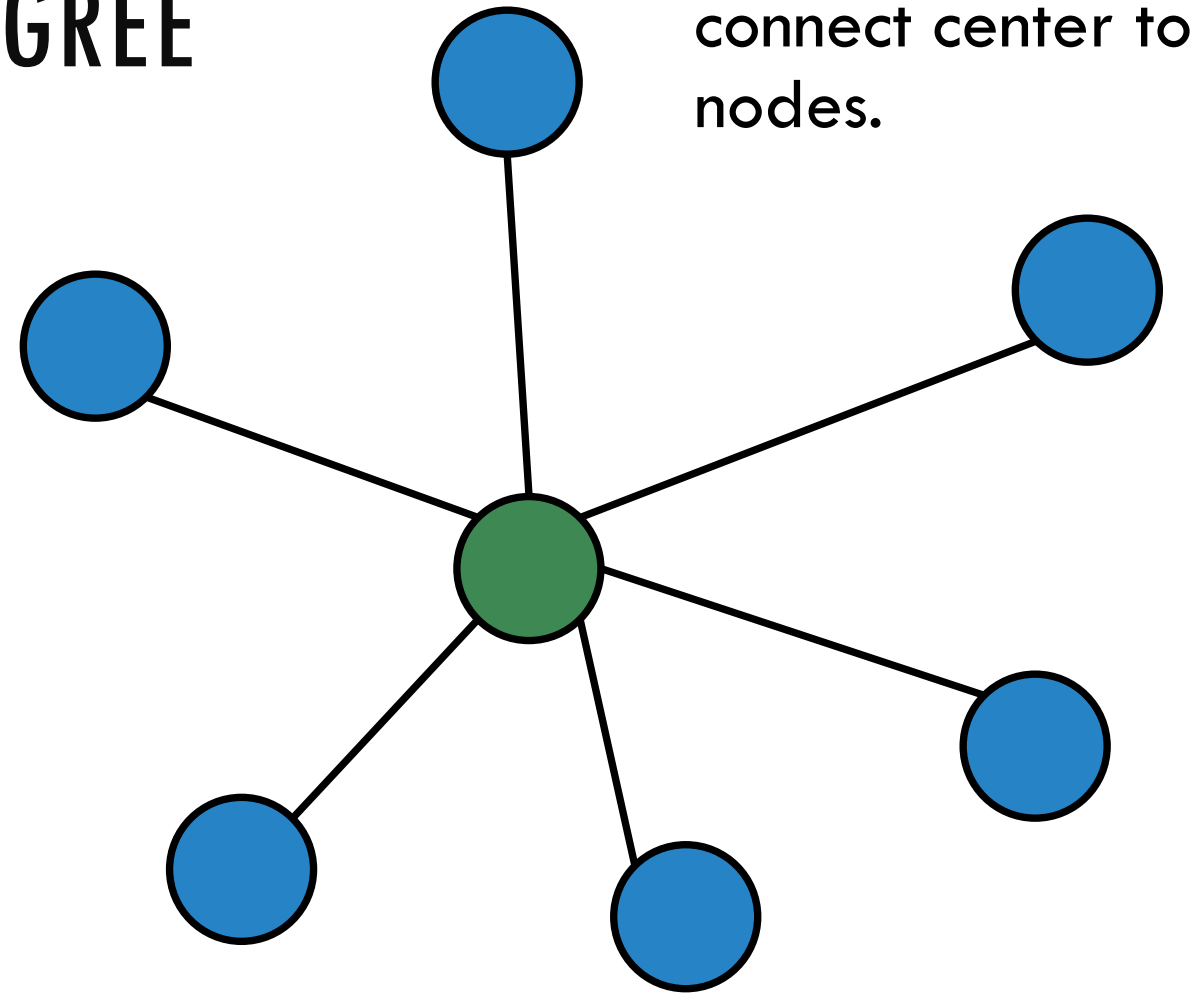
C. 3

D. 4

E. Which star? the sun? 1.4 × 10^6 km

DEGREE

One central node. All edges connect center to the other nodes.



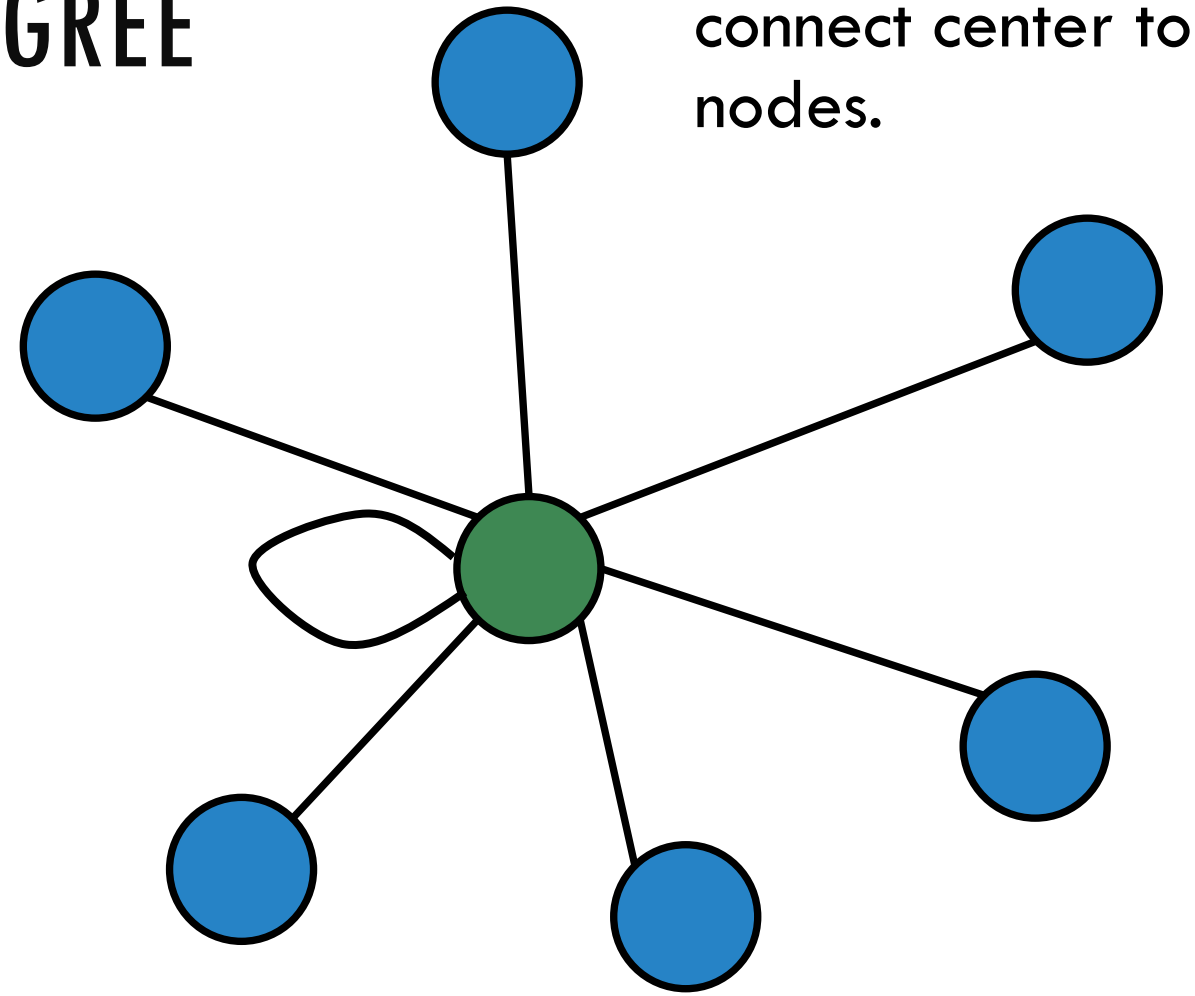
The **Degree** of a node is the number of edges incident upon it.

Question: What is the degree of the central (green) node?

Six edges are incident or "attached" to the node.

DEGREE

One central node. All edges connect center to the other nodes.



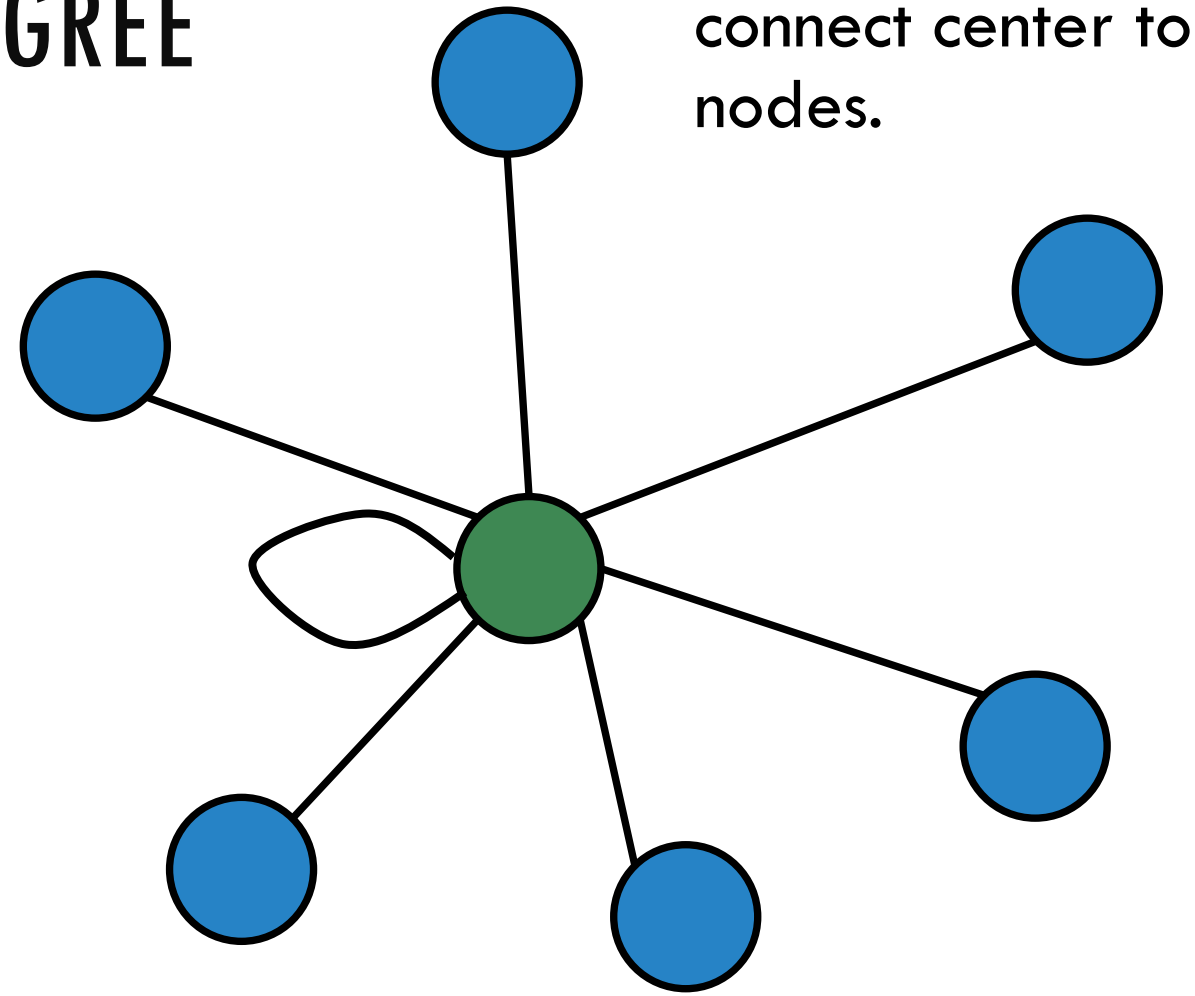
The **Degree** of a node is the number of edges incident upon it.

Question: What is the degree of the central (green) node?

Loops are counted twice.

DEGREE

One central node. All edges connect center to the other nodes.



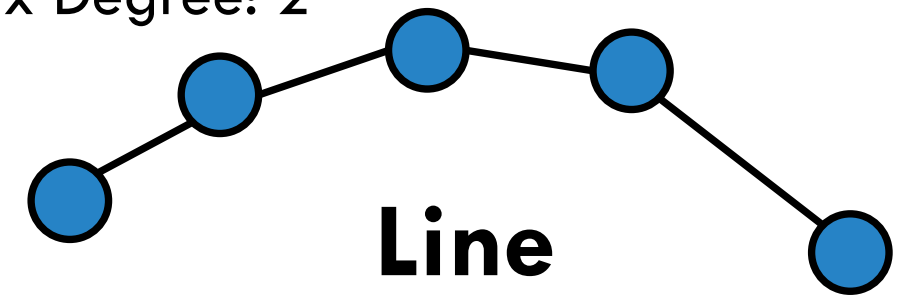
The **Degree** of a node is the number of edges incident upon it.

The **maximum degree** of a graph $\Delta(G)$ is the maximum degree of its vertices.

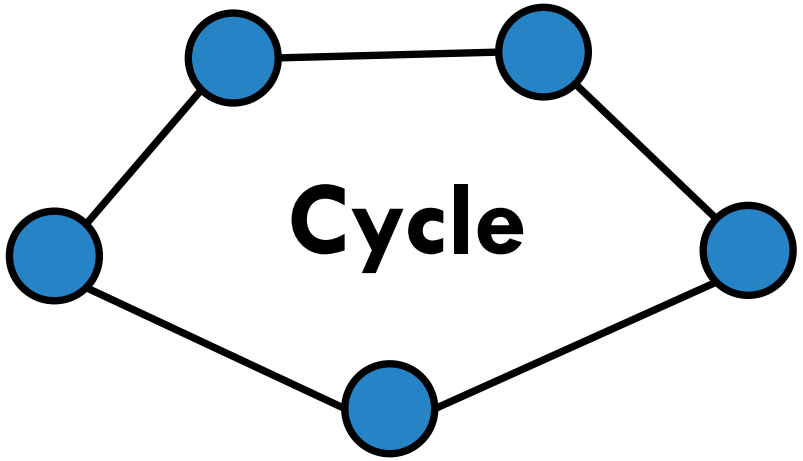
The **minimum degree** of a graph $\delta(G)$ is the minimum degree of its vertices.

LINES AND CYCLES

- Diameter: $n - 1$
- Max Degree: 2



- Diameter: $\frac{n}{2}$
- Max Degree: 2



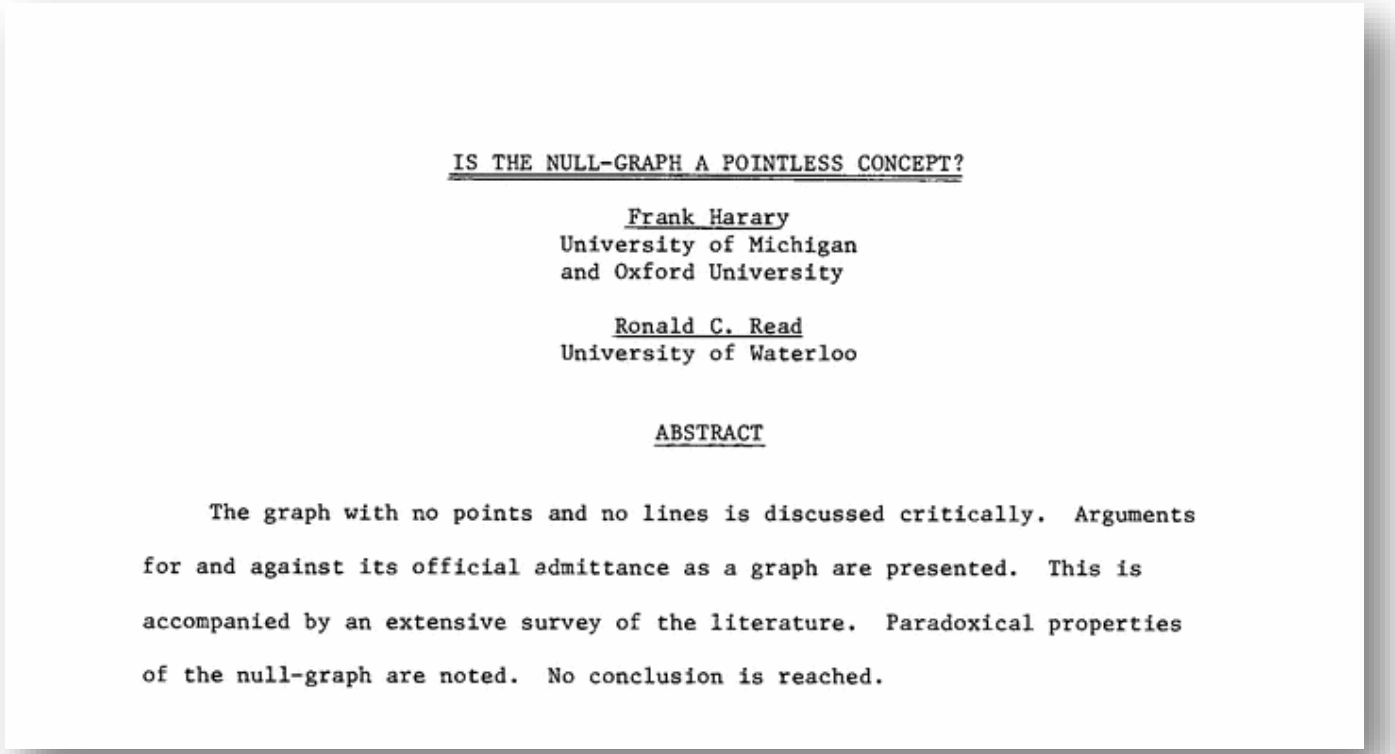
What are the <u>diameter</u> and <u>maximum degree</u> in terms of the number of nodes n?

IS THIS A VALID GRAPH?

Is $G = \langle V, E \rangle$ $V = \{\}$ and $E = \{\}$ a graph?

A. Yes
B. No
C. I don't know!

IS THIS A VALID GRAPH?



IS THE NULL-GRAPH A POINTLESS CONCEPT?

Frank Harary
University of Michigan
and Oxford University

Ronald C. Read
University of Waterloo

ABSTRACT

The graph with no points and no lines is discussed critically. Arguments for and against its official admittance as a graph are presented. This is accompanied by an extensive survey of the literature. Paradoxical properties of the null-graph are noted. No conclusion is reached.

Is $G = \langle V, E \rangle$ $V = \{\}$ and $E = \{\}$ a graph?

A. Yes
B. No
C. I don't know!

Harary, F. and Read, R. "Is the Null Graph a Pointless Concept?" In Graphs and Combinatorics Conference, George Washington University. New York: Springer-Verlag, 1973.

TERMINOLOGY SUMMARY

Graph: $G = \langle V, E \rangle$

Degree of a node: number of edges connected to it

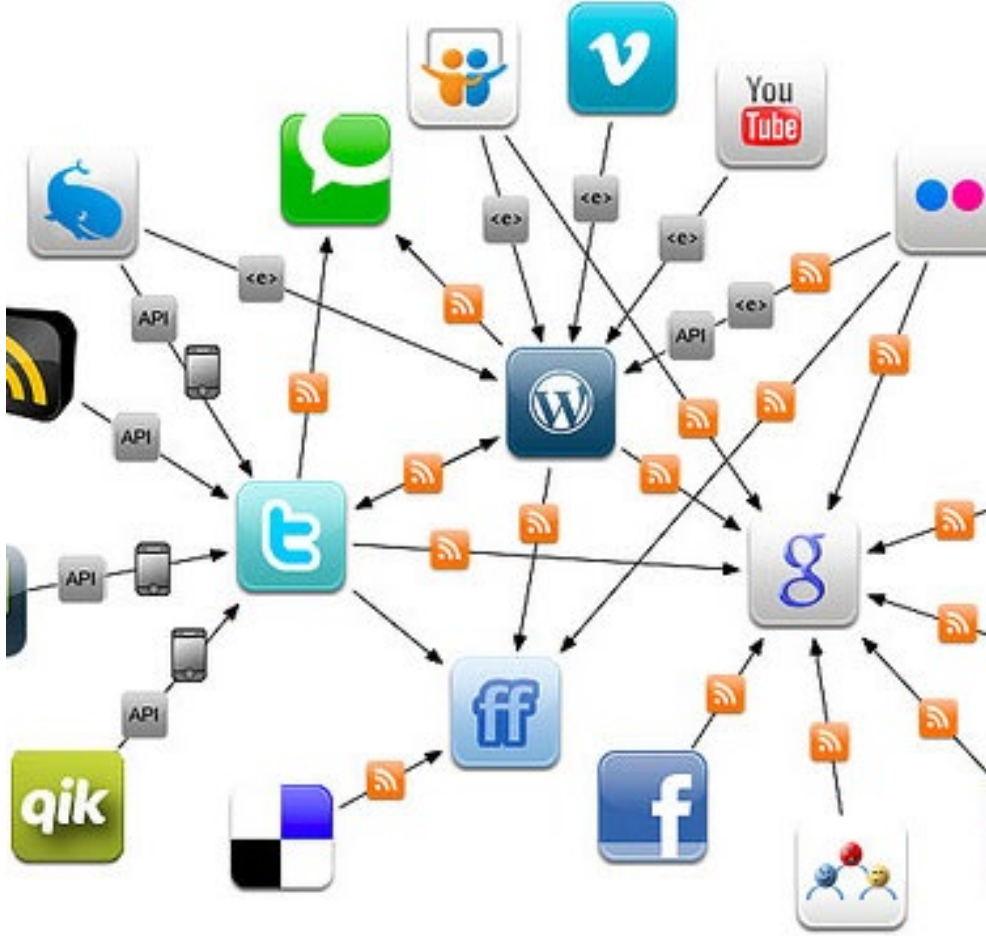
Diameter: longest shortest path between two different nodes

Connected Graph: path between any two nodes

Clique: fully connected graph

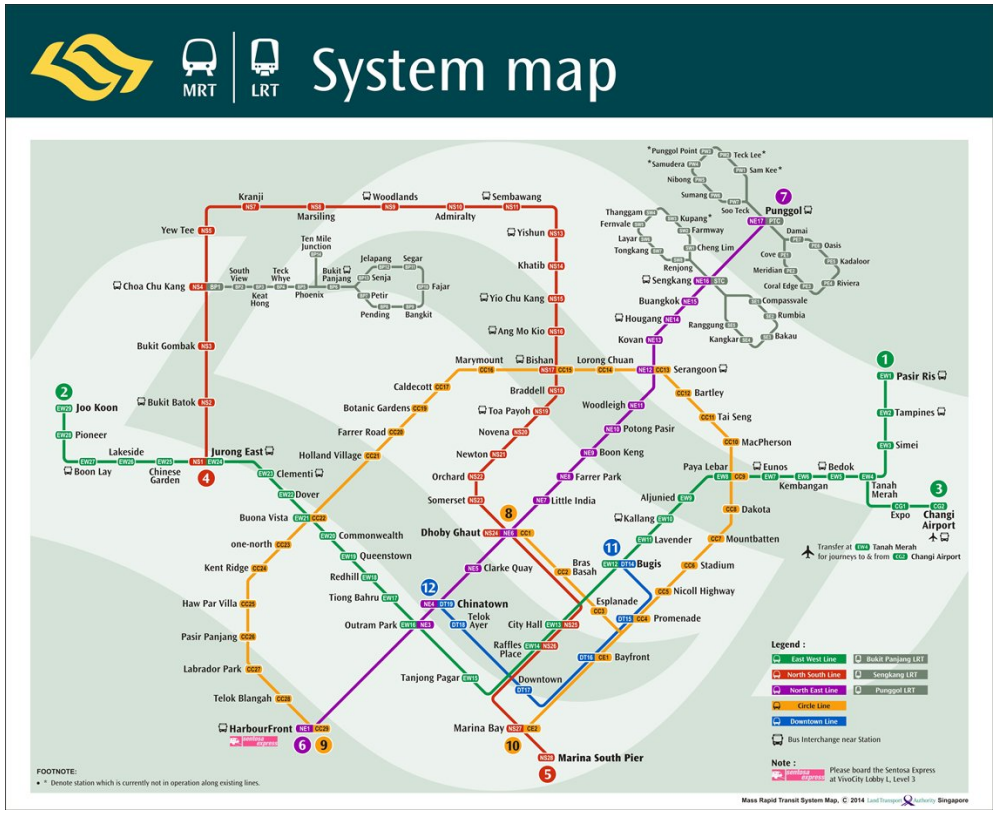
Line Graph: a line (duh!)

Star: central node connected to all other nodes.



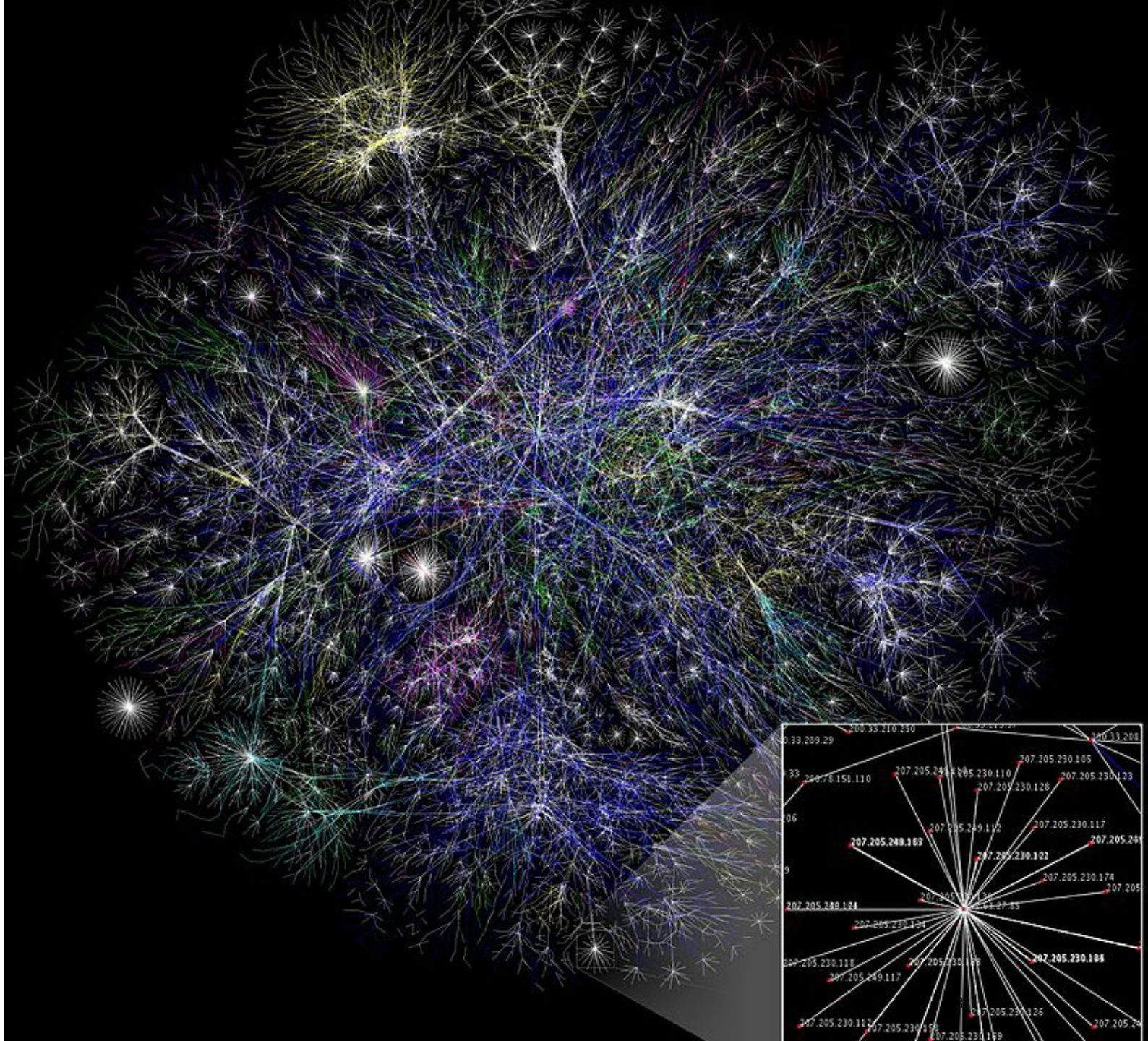
EXAMPLES OF REAL WORLD GRAPHS

EXAMPLES OF REAL WORLD GRAPHS



COMMUNICATION NETWORKS

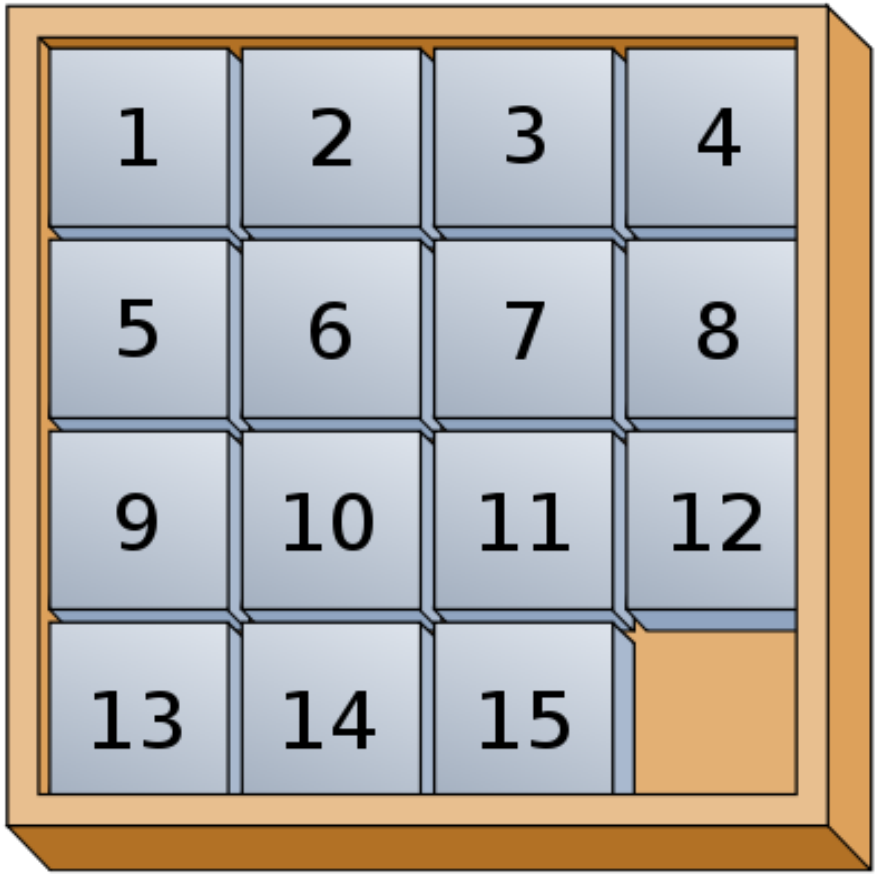
EXAMPLES: COMPUTER NETWORKS



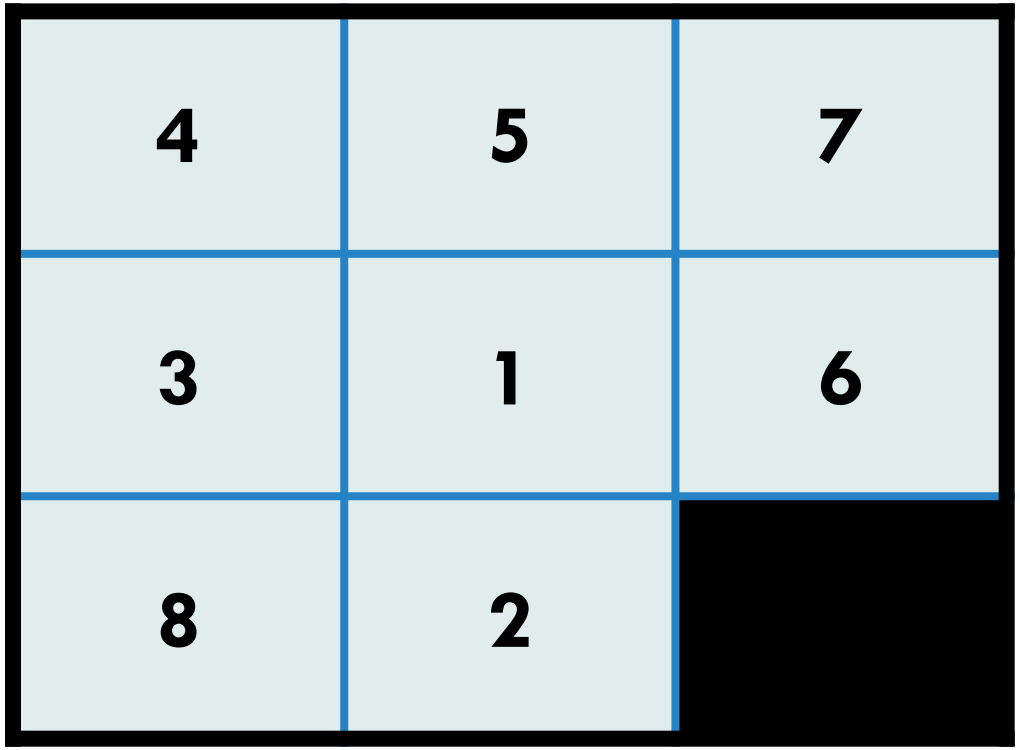
GAME GRAPHS


ALPHAGO
00:05:30
Google DeepMind
Challenge Match
LEE SEDOL
00:29:59

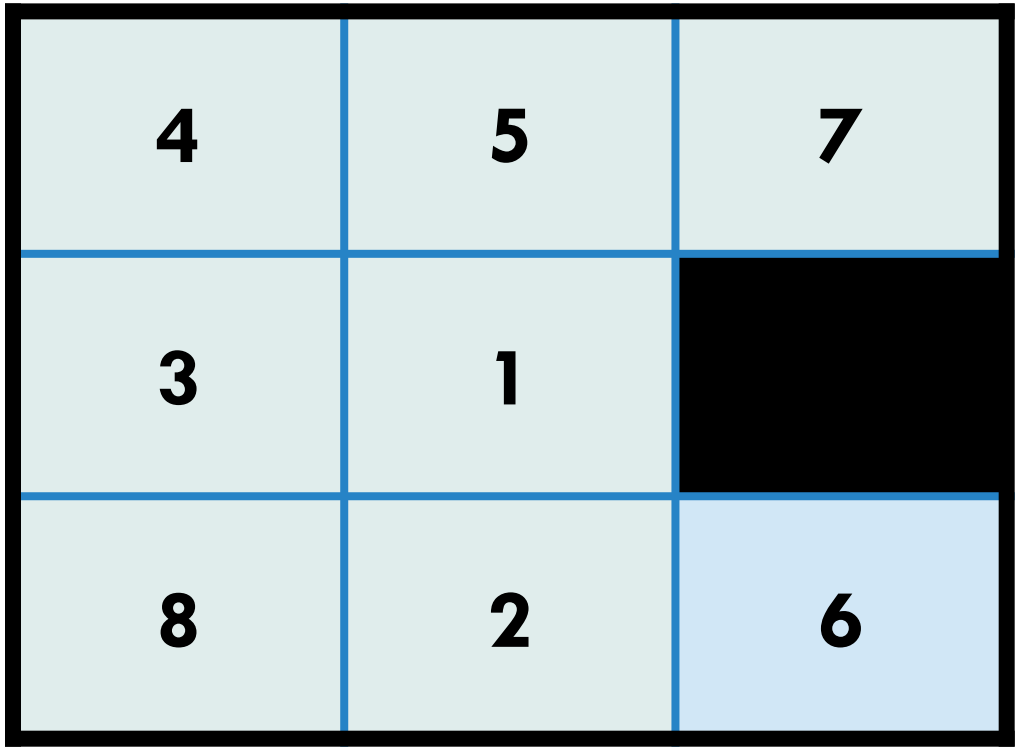
SLIDING PUZZLE



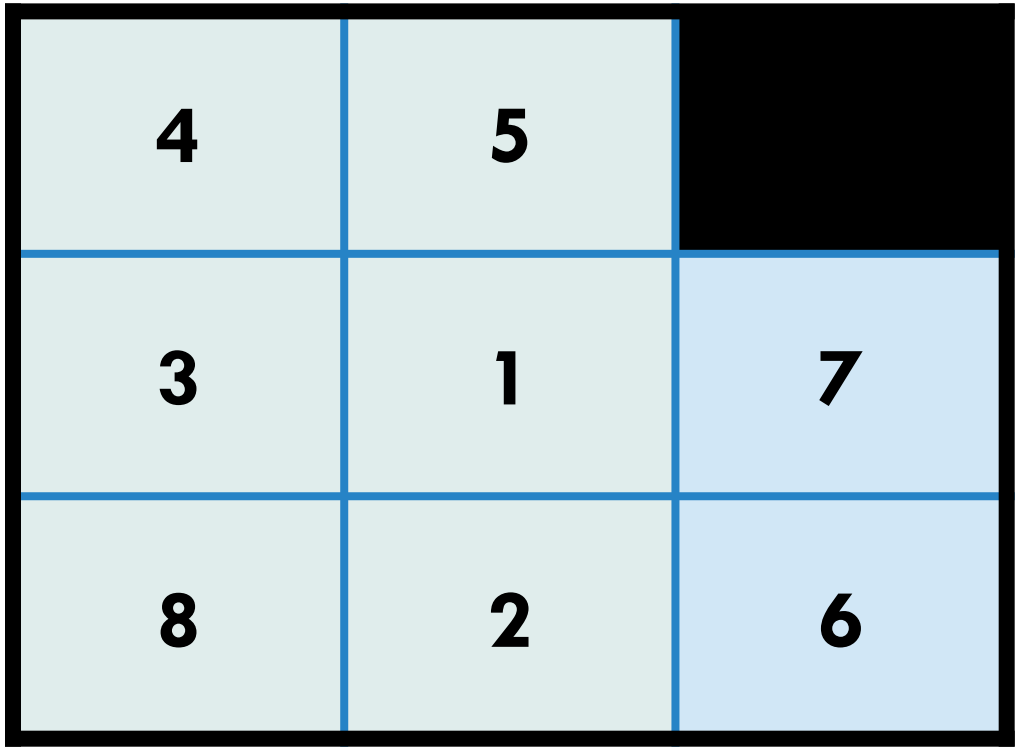
SLIDING PUZZLE



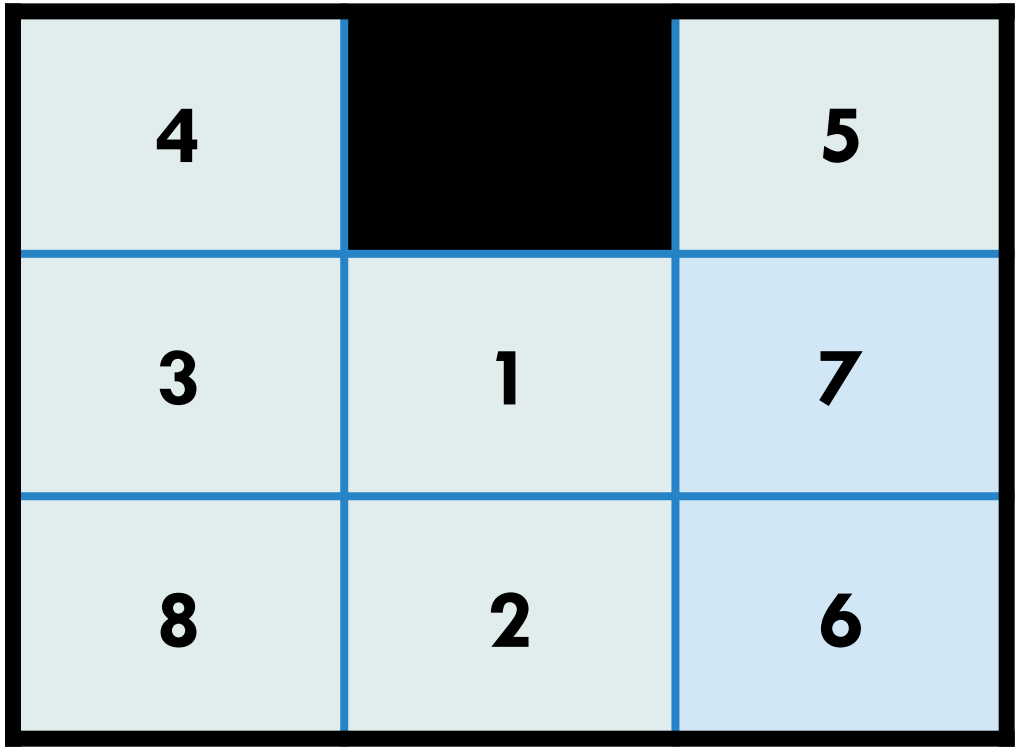
SLIDING PUZZLE



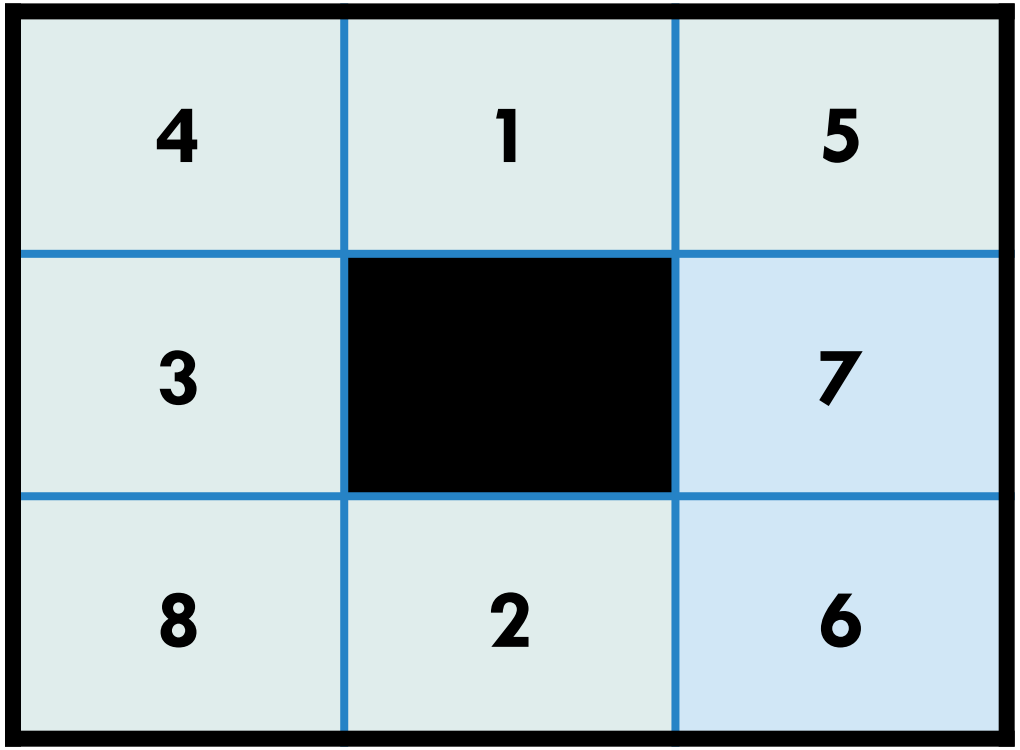
SLIDING PUZZLE



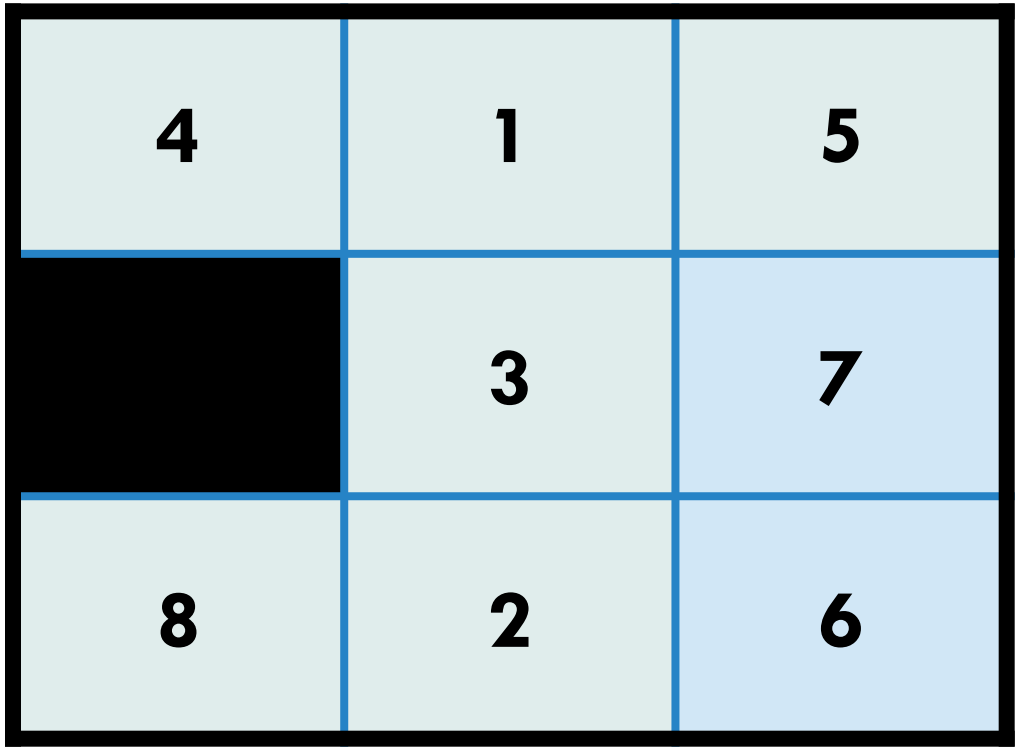
SLIDING PUZZLE



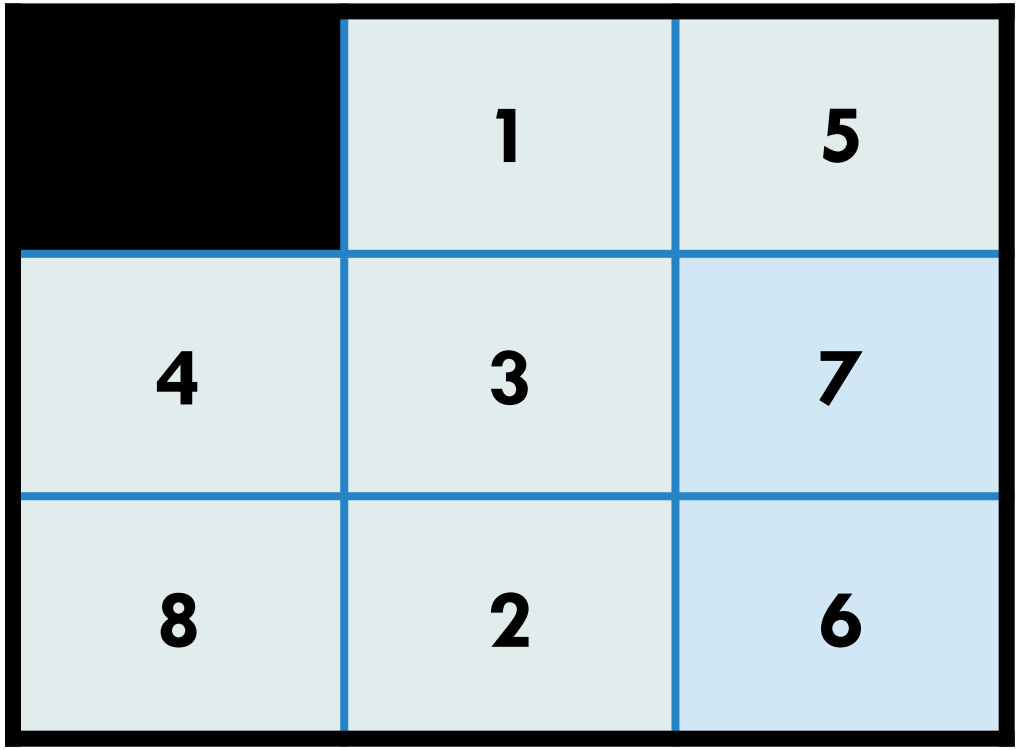
SLIDING PUZZLE



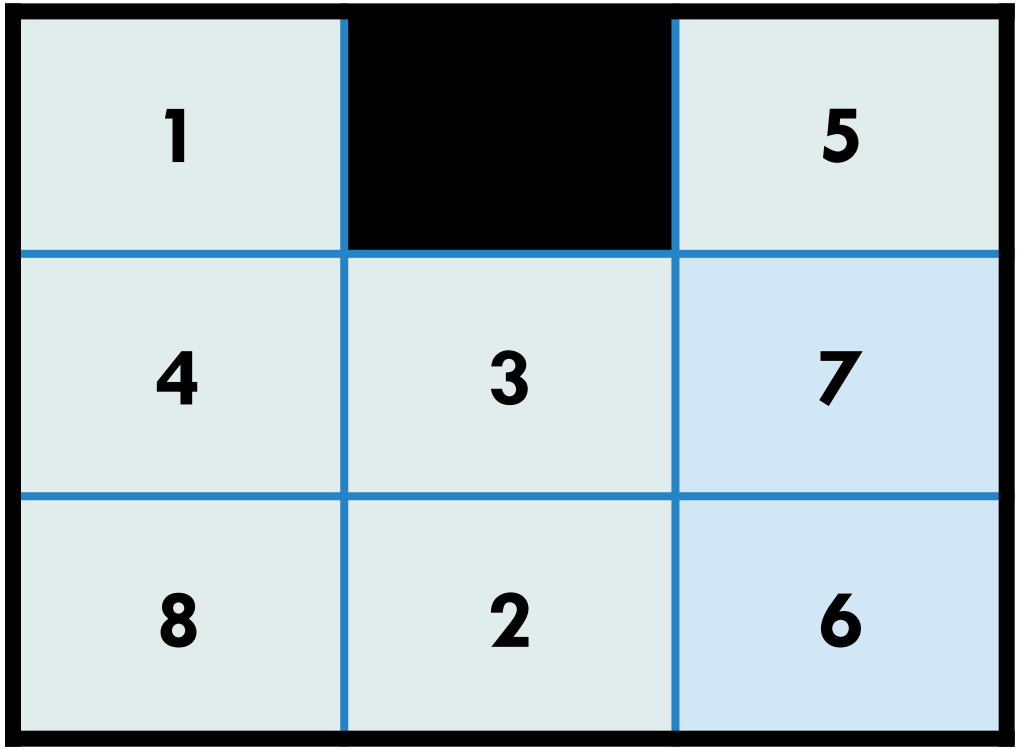
SLIDING PUZZLE

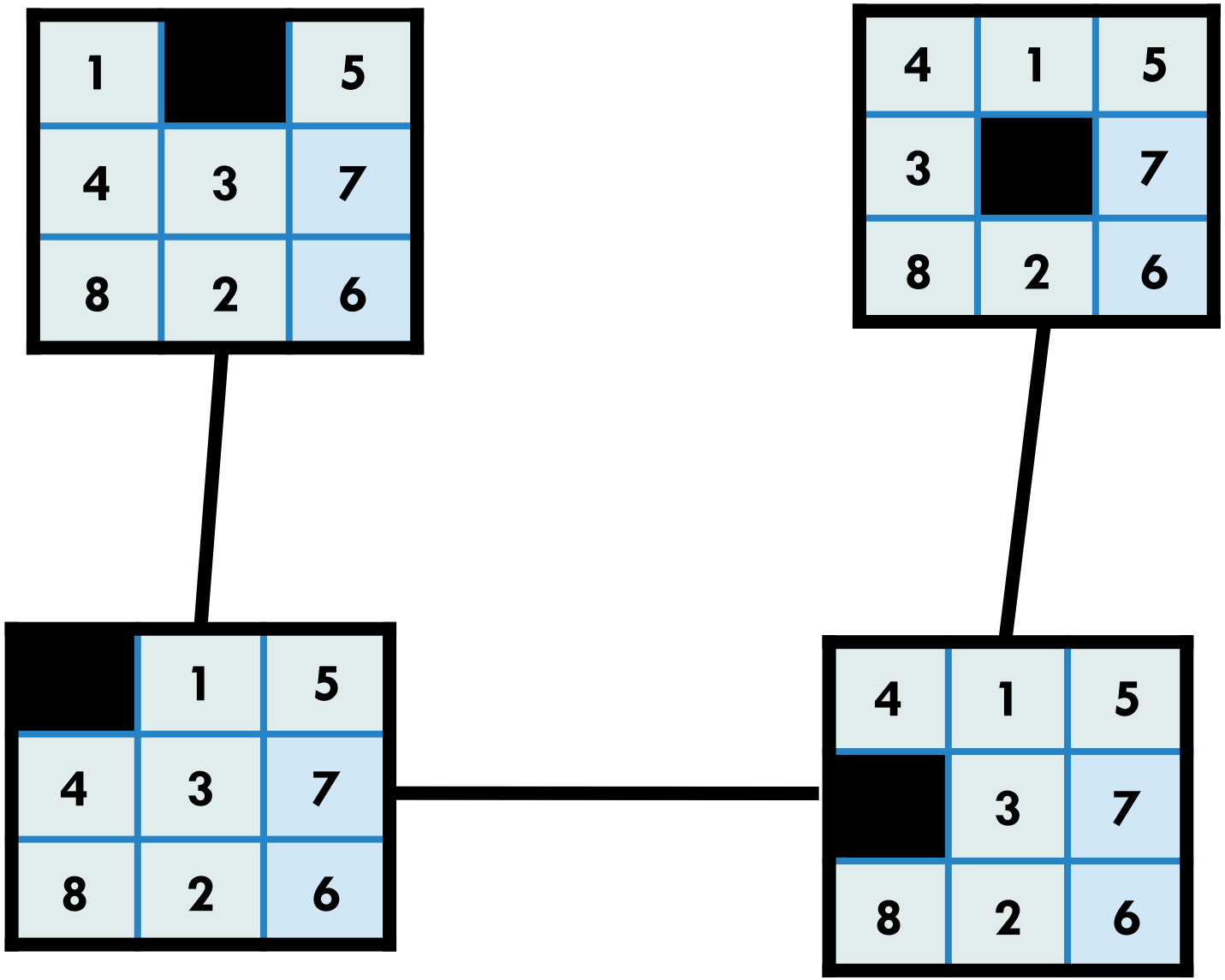


SLIDING PUZZLE



SLIDING PUZZLE




1 5
4 3 7
8 2 6
4 1 5
3 7
8 2 6
1 5
4 3 7
8 2 6
4 1 5
3 7
8 2 6

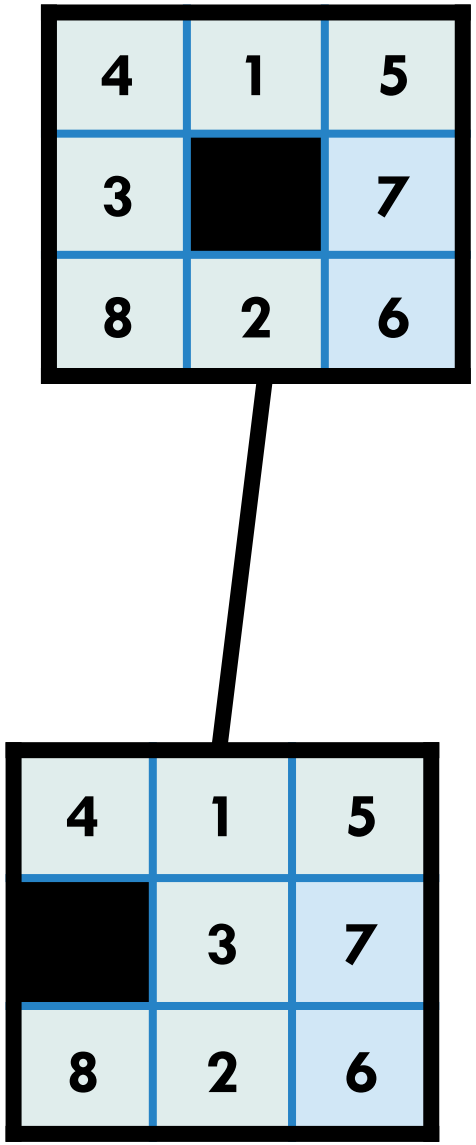
SLIDING PUZZLE

Nodes:

- State of the puzzle
- Permutation of nine tiles

Edges:

- Two states are edges if they differ by only one move.



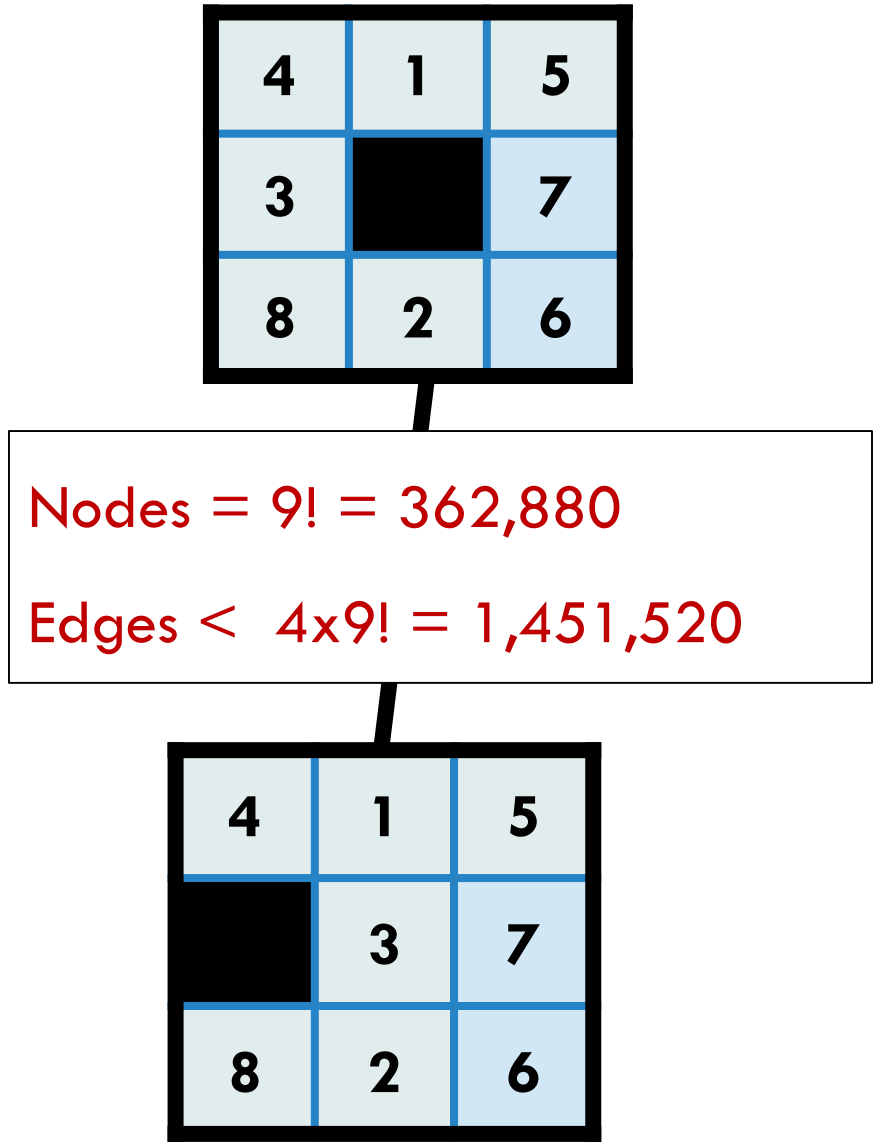
SLIDING PUZZLE

Nodes:

- State of the puzzle
- Permutation of nine tiles

Edges:

- Two states are edges if they differ by only one move.



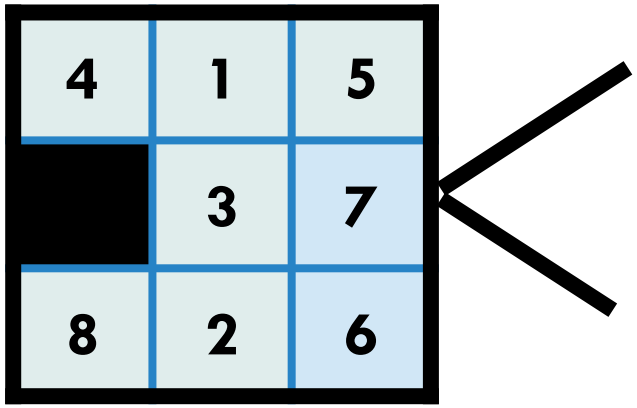
SLIDING PUZZLE

Most related to:

A. Diameter
B. Degree
C. Clique
D. Connectedness
E. Ummm.. I guess B?

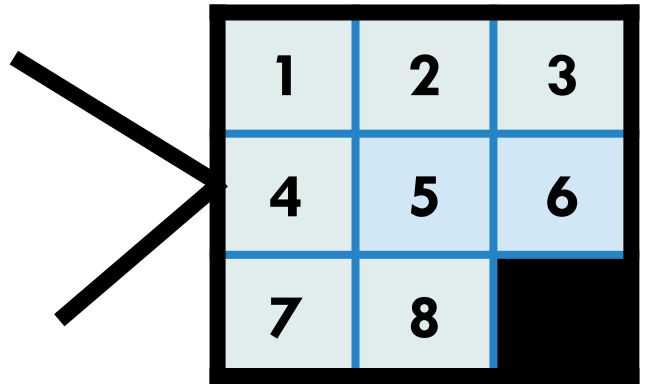
Question: Number of moves to solve the puzzle?

Initial, scrambled state:



?

Final, unscrambled state:



SLIDING PUZZLE

Most related to:

A. Diameter

B. Degree

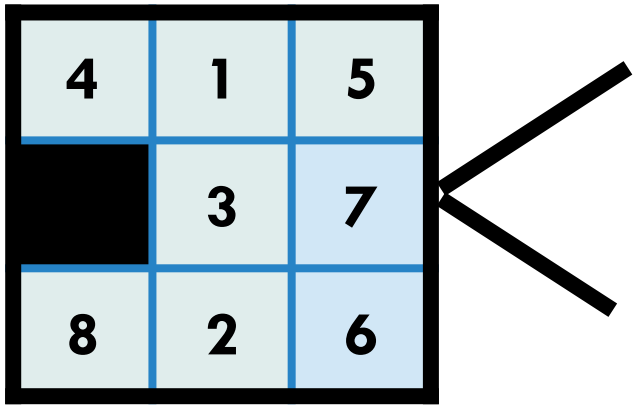
C. Clique

D. Connectedness

E. Ummm.. I guess B?

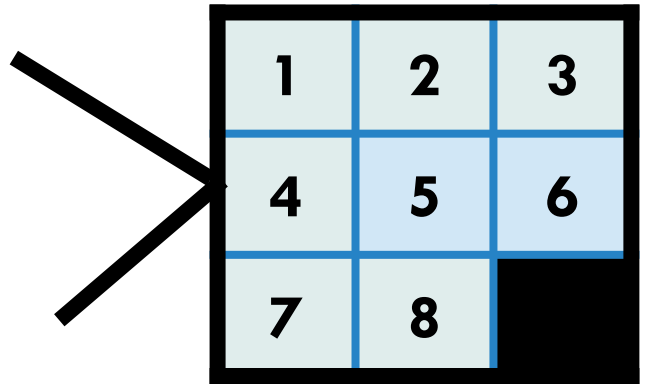
Question: Number of moves to solve the puzzle?

Initial, scrambled state:



?

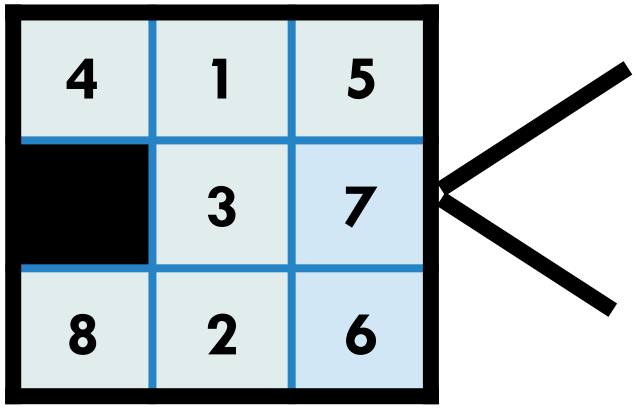
Final, unscrambled state:



SLIDING PUZZLE

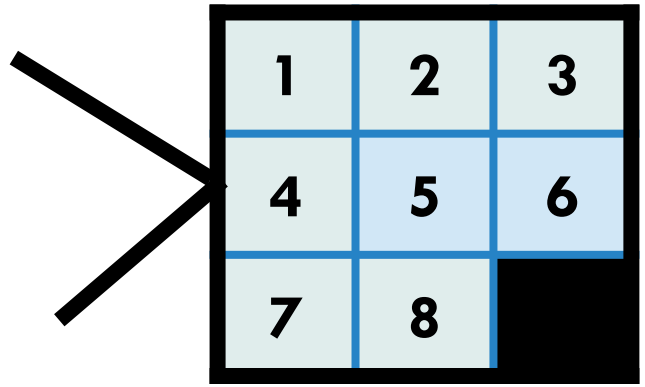
Number of moves to solve the puzzle ≤ diameter. **Why?**

Initial, scrambled state:

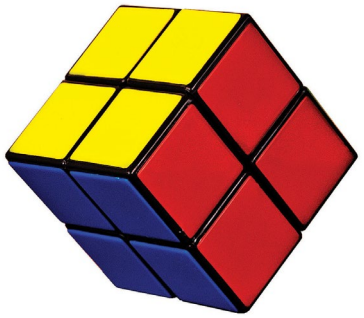


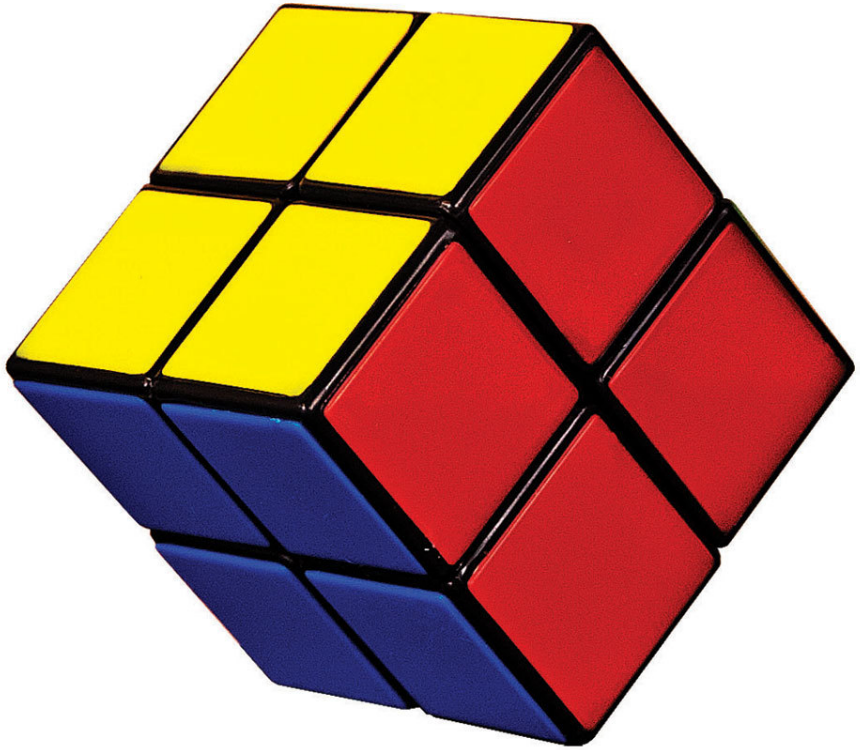
?

Final, unscrambled state:

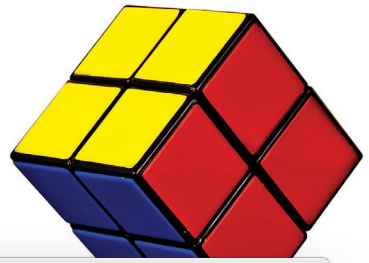


2X2X2 RUBIK'S CUBE





2X2X2 RUBIK'S CUBE



Record solve time:
0.49 seconds

worldcubeassociation.org

WORLD CUBE ASSOCIATION

Information | Competitions | Results | Regulations

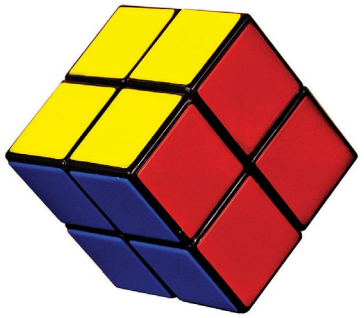
Event: 2x2x2 Cube | Region: World | Years: All | Show: 100 Persons

Single | Average

2x2x2 Cube **100 Persons**

Rank	Person	Result	Citizen of	Competition
1	Maciej Czapiewski	0.49	Poland	Grudziądz Open 2016
2	Michał Rzewuski	0.52	Poland	Grudziądz Open 2016
3	Mats Valk	0.56	Netherlands	Kaohsiung Open 2016
4	Kim Roger Høyland Larsen	0.58	Norway	Sandnes Open 2016
	Rami Sbahi	0.58	United States	Canadian Open 2015
6	Justin He	0.60	Australia	Sydney Championship 2016
7	Yu-Kai Chen (陳宇楷)	0.62	Taiwan	Kaohsiung Open 2016
8	Igor Ośmiałowski	0.63	Poland	Grudziądz Open 2016
9	Lucas O'Connell	0.64	New Zealand	Techweek NZ 2017

2X2X2 RUBIK'S CUBE



Configuration Graph

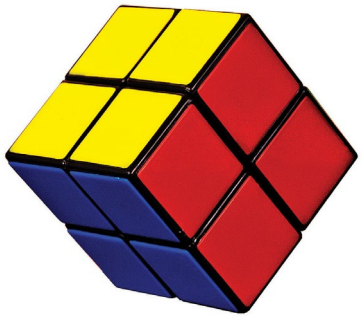
Vertex for each possible state

Edge for each basic move

- 90 degree turn
- 180 degree turn

Puzzle: given initial state, find a path to the solved state.

2X2X2 RUBIK'S CUBE



How many vertices?

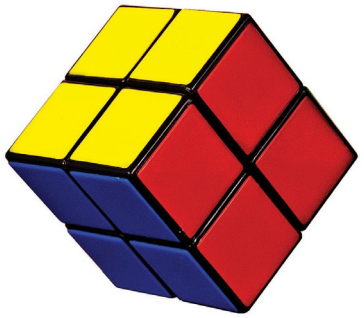
$$8! \cdot 3^8 = 264{,}539{,}520$$

cubelets

Each cubelet is
in one of 8 positions.

Each of the 8 cubelets
can be in one of three
orientations

2X2X2 RUBIK'S CUBE



How many vertices?

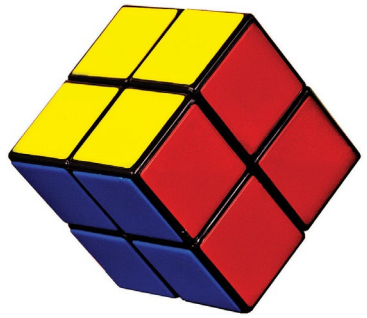
$$7! \cdot 3^7 = 11{,}022{,}480$$

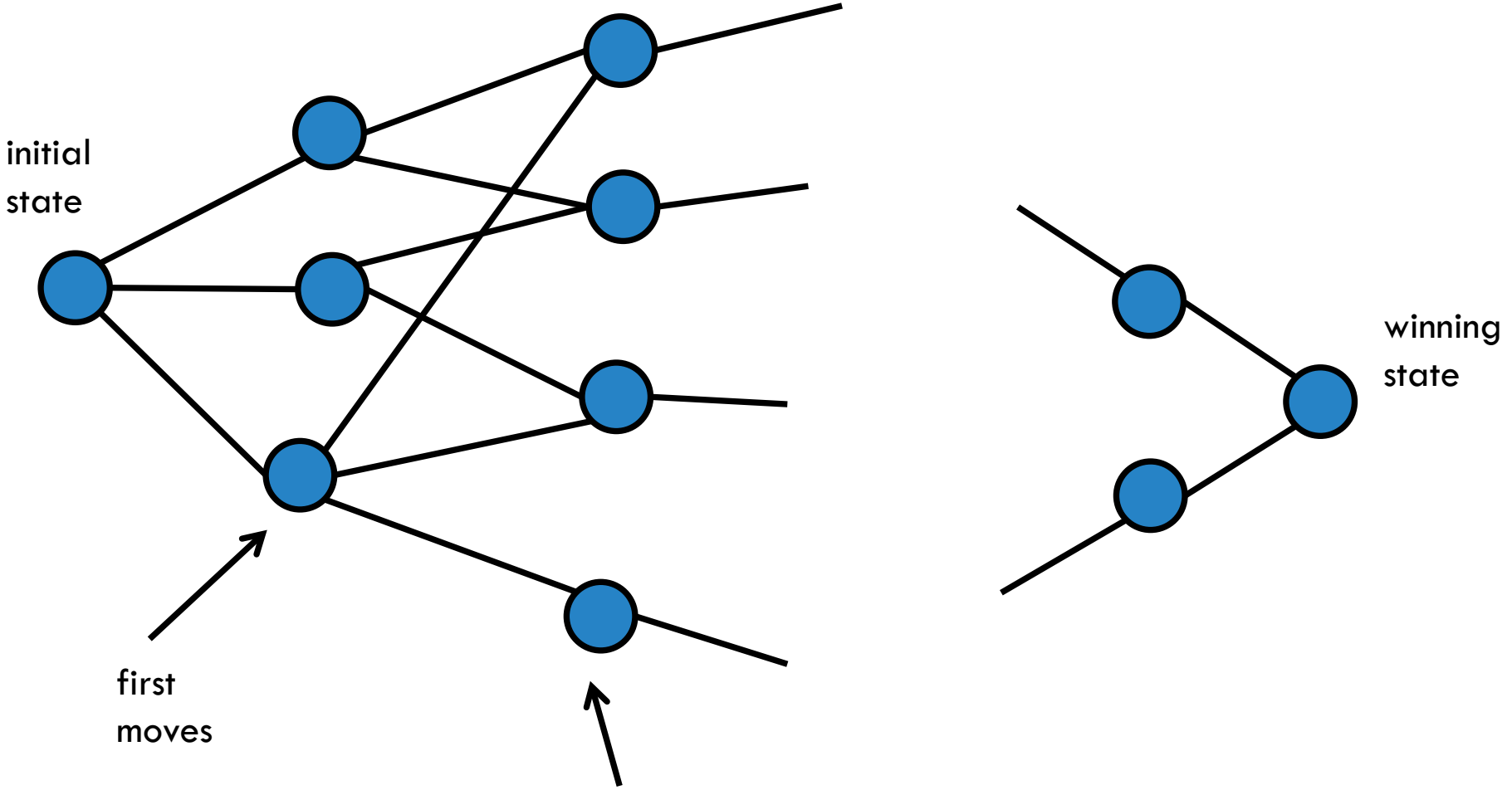
Symmetry:

Fix one cubelet.

Each of the 8 cubelets can be in one of three orientations

2X2X2 RUBIK'S CUBE: CONFIGURATION GRAPH





3X3X3 RUBIK'S CUBE

Configuration Graph

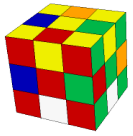
43 quintillion vertices (approximately)

Diameter: **20**

- 1995: require at least 20 moves.
- 2008: 20 moves is enough from every position.
- Using Google server farm.
- 35 CPU-years of computation.
- 20 seconds / set of 19.5 billion positions.
- Lots of mathematical and programming tricks.

Link: http://cube20.org/

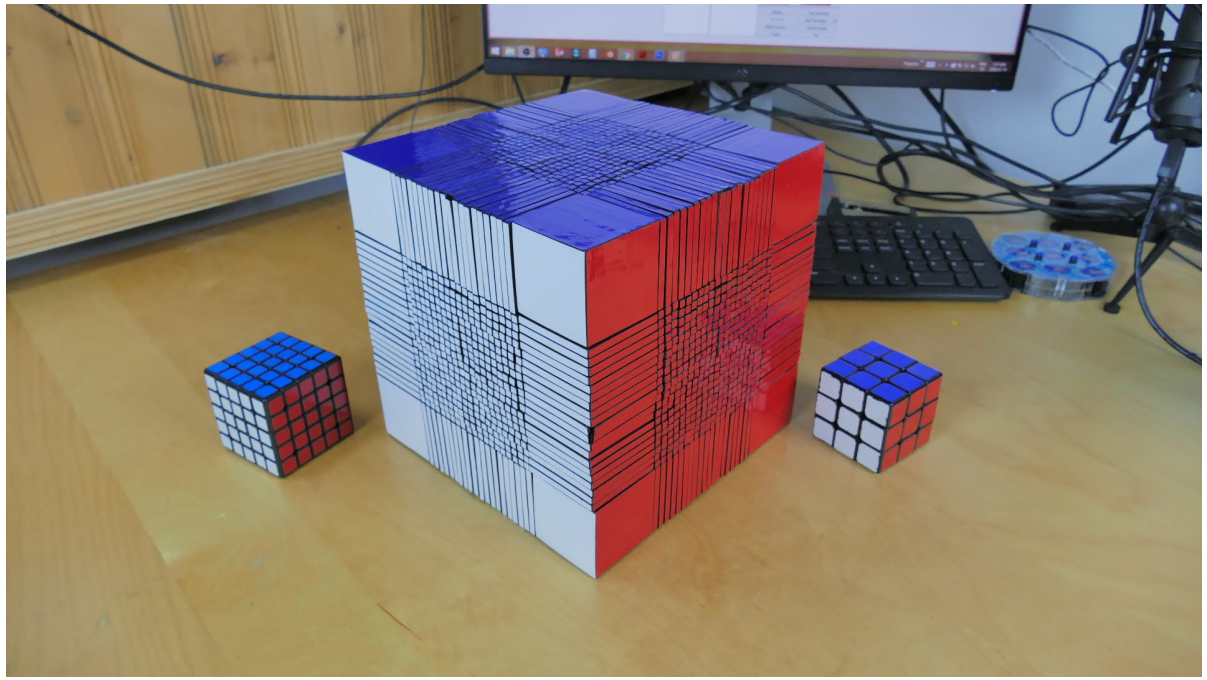
Date	Lower bound	Upper bound	Gap	Notes and Links
July, 1981	18	52	34	Morwen Thistlethwaite proves 52 moves suffice.
December, 1990	18	42	24	Hans Kloosterman improves this to 42 moves.
May, 1992	18	39	21	Michael Reid shows 39 moves is always sufficient.
May, 1992	18	37	19	Dik Winter lowers this to 37 moves just one day later!
January, 1995	18	29	11	Michael Reid cuts the upper bound to 29 moves by analyzing Kociemba's two-phase algorithm.
January, 1995	20	29	9	Michael Reid proves that the "superflip" position (corners correct, edges placed but flipped) requires 20 moves.
December, 2005	20	28	8	Silviu Radu shows that 28 moves is always enough.
April, 2006	20	27	7	Silviu Radu improves his bound to 27 moves.
May, 2007	20	26	6	Dan Kunkle and Gene Cooperman prove 26 moves suffice.
March, 2008	20	25	5	Tomas Rokicki cuts the upper bound to 25 moves.
April, 2008	20	23	3	Tomas Rokicki and John Welborn reduce it to only 23 moves.
August, 2008	20	22	2	Tomas Rokicki and John Welborn continue down to 22 moves.
July, 2010	20	20	0	Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge prove that God's Number for the Cube is exactly 20.



nxnxn RUBIK'S CUBE

What is the diameter of an (n x n x n) cube?

E.g.: 22 x 22 x 22 Rubik's Cube



nxnxn RUBIK'S CUBE

What is the diameter of an (n x n x n) cube?

E.g.: 22 x 22 x 22 Rubik's Cube

$$O(n^2 / \log n)$$

Algorithms for Solving Rubik's Cubes

Erik D. Demaine[1], Martin L. Demaine[1], Sarah Eisenstat[1],
Anna Lubiw[2], and Andrew Winslow[3]

[1] MIT Computer Science and Artificial Intelligence Laboratory,
Cambridge, MA 02139, USA, {edemaine,mdemaine,seisenst}@mit.edu
[2] David R. Cheriton School of Computer Science,
University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, alubiw@uwaterloo.ca
[3] Department of Computer Science, Tufts University,
Medford, MA 02155, USA, awinslow@cs.tufts.edu

Abstract. The Rubik's Cube is perhaps the world's most famous and iconic puzzle, well-known to have a rich underlying mathematical structure (group theory). In this paper, we show that the Rubik's Cube also has a rich underlying algorithmic structure. Specifically, we show that the $n \times n \times n$ Rubik's Cube, as well as the $n \times n \times 1$ variant, has a "God's Number" (diameter of the configuration space) of $\Theta(n^2 / \log n)$. The upper bound comes from effectively parallelizing standard $\Theta(n^2)$ solution algorithms, while the lower bound follows from a counting argument. The upper bound gives an asymptotically optimal algorithm for solving a general Rubik's Cube in the worst case. Given a specific starting state, we show how to find the shortest solution in an $n \times O(1) \times O(1)$ Rubik's Cube. Finally, we show that finding this optimal solution becomes NP-hard in an $n \times n \times 1$ Rubik's Cube when the positions and colors of some of the cubies are ignored (not used in determining whether the cube is solved).

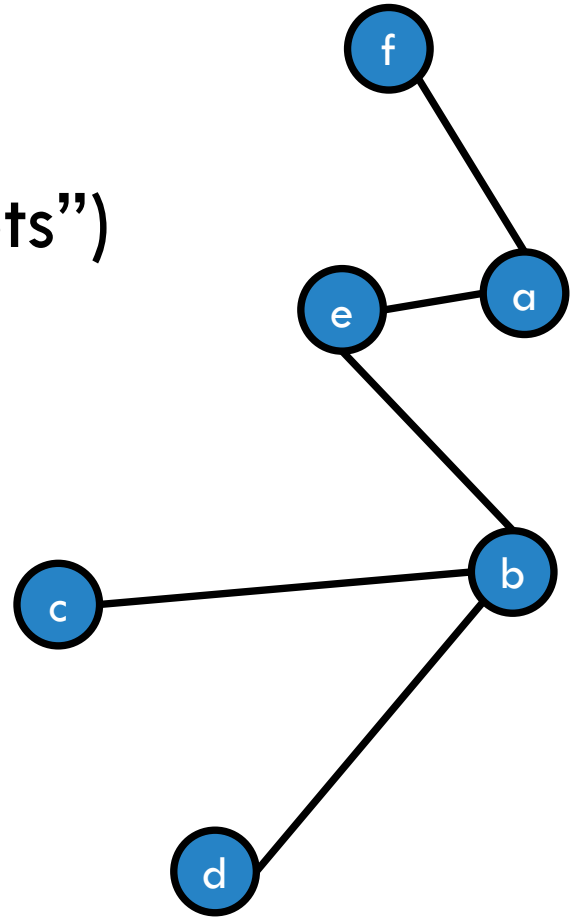
Keywords: combinatorial puzzles, diameter, God's number, combinatorial optimization

REPRESENTING A GRAPH

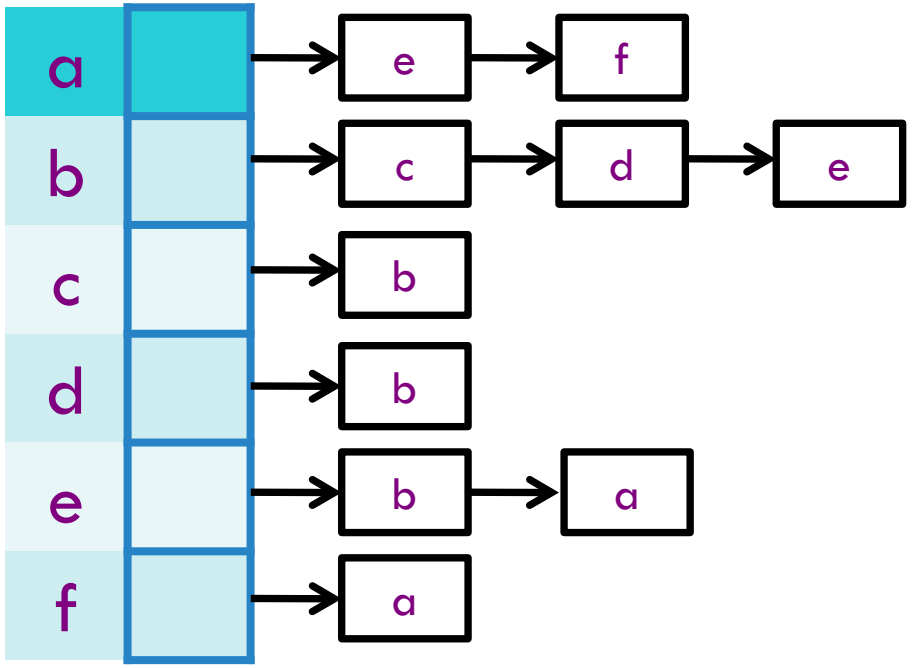
Graph $G = \langle V, E \rangle$ ("a tuple of two sets")

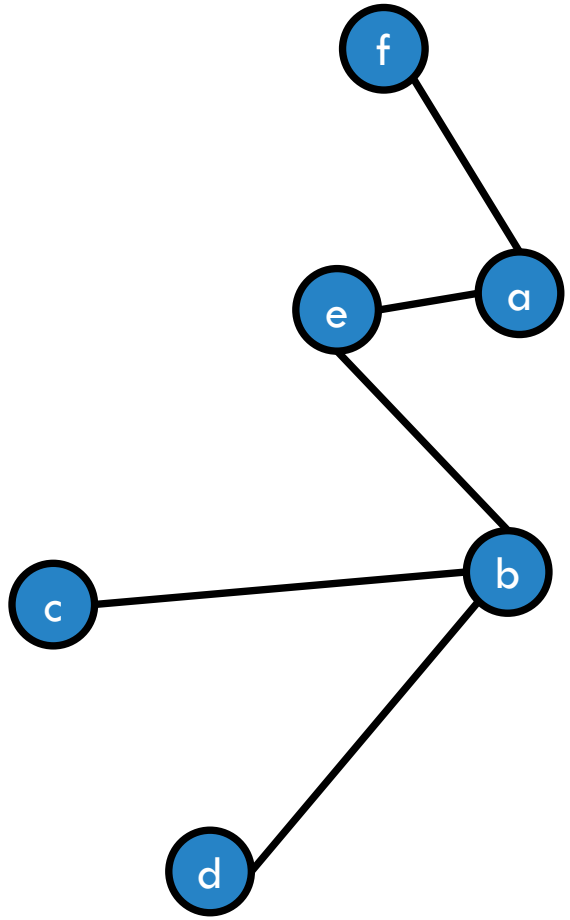
- V is a set of nodes
- E is a set of edges
  - $E \subseteq \{ (v, w) : v, w \in V \}$

How about in a computer?



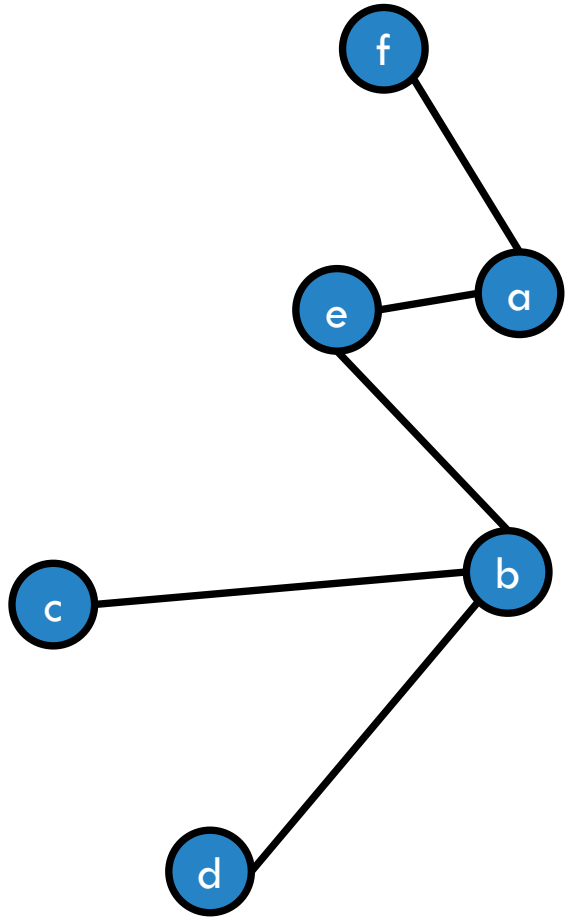
ADJACENCY LIST





ADJACENCY MATRIX

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



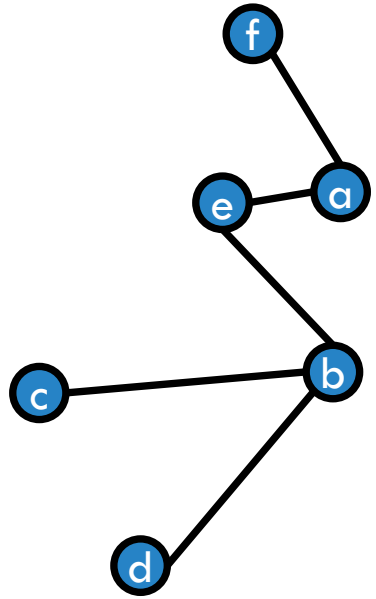
ADJACENCY MATRIX

Graph represented as:

$A[v][w] = 1$ iff $(v, w) \in E$

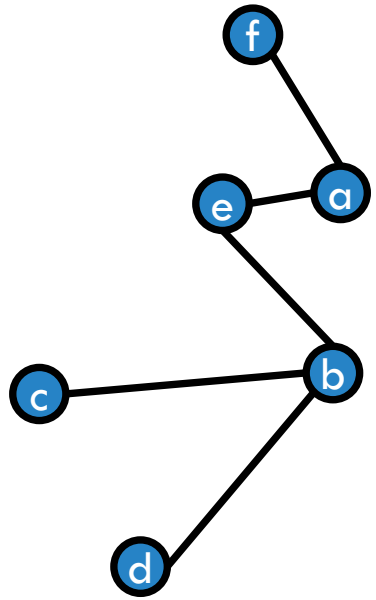
Neat property:

A^2 = length 2 paths



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

ADJACENCY MATRIX



To find out if c and d are 2-hop neighbors:

Let $B = A^2$.

- $B[c, d] = A[c, .] \cdot A[., d]$

$B[c, d] = 1$ iff

$$A[c, x] = A[x, d]$$

for some x.

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

ADJACENCY MATRIX

Graph represented as:

$$A[v][w] = 1 \text{ iff } (v, w) \in E$$

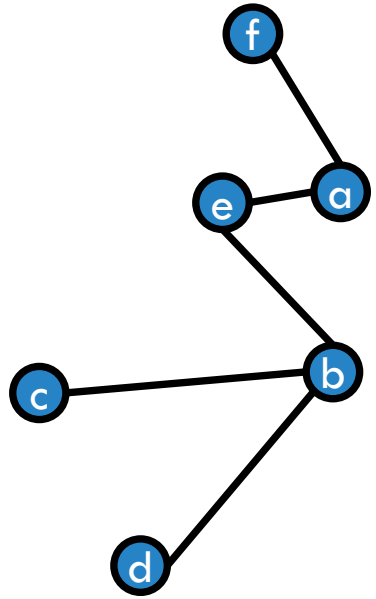
Neat properties:

A^2 = length 2 paths

A^3 = length 3 paths

A^k = length k paths

A^∞ = Google Pagerank



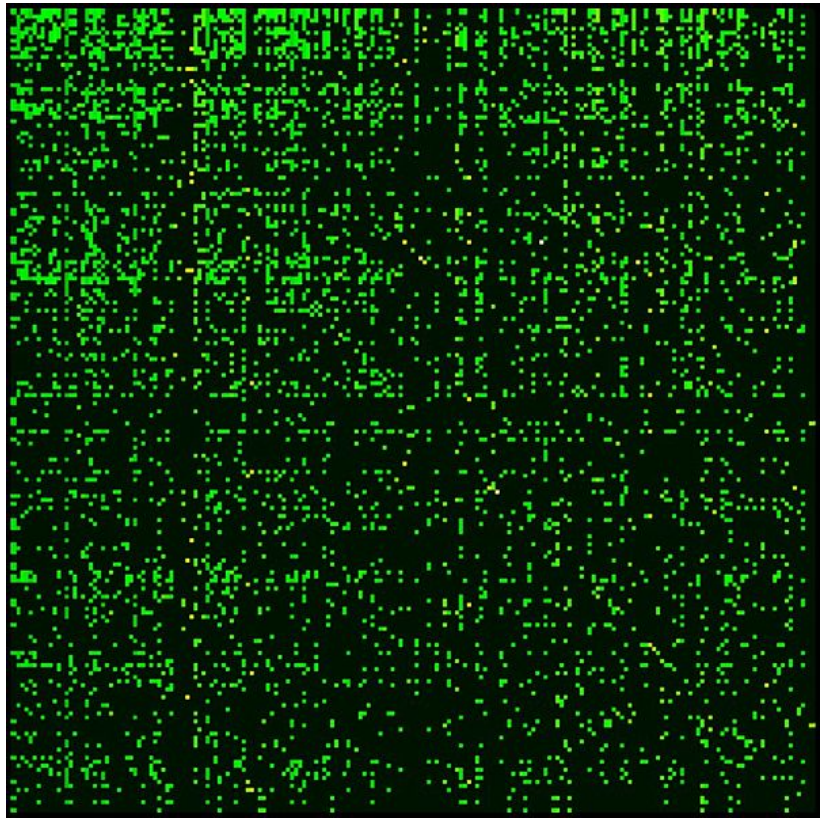
	a	b	c	d	e	f
a	0	0	0	0	1	1
b	0	0	1	1	1	0
c	0	1	0	0	0	0
d	0	1	0	0	0	0
e	1	1	0	0	0	0
f	1	0	0	0	0	0

GOOGLE MATRIX

Used by Google's PageRank algorithm.

A graph with edges representing links between pages.

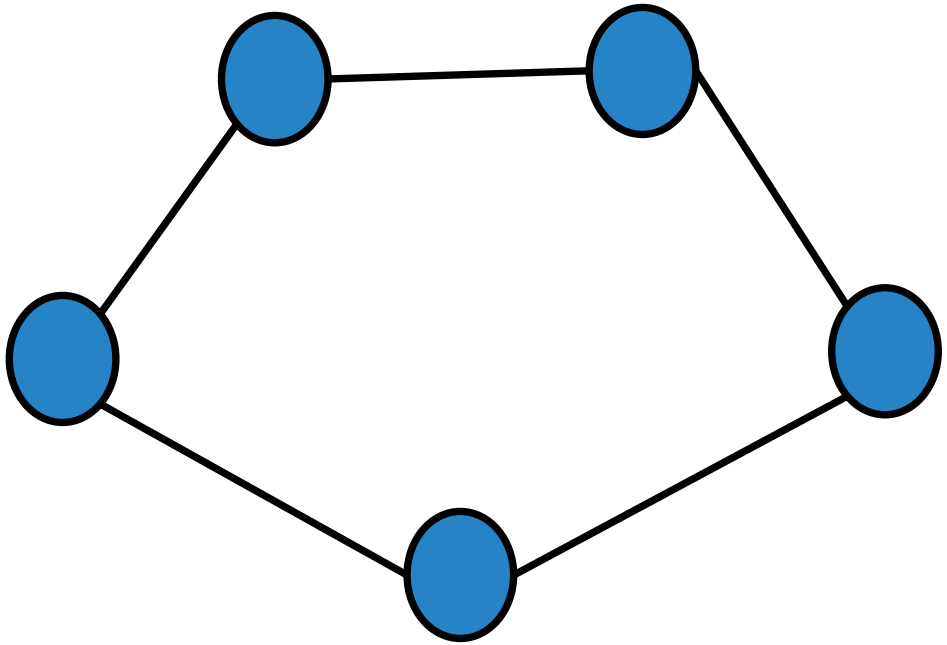
The **rank** of each page can be generated iteratively from the Google matrix using the **power method**. (Beyond scope of CS2040S, but not difficult to implement)



FOR A CYCLE, WHICH REPRESENTATION IS MORE MEMORY EFFICIENT? (asymptotically… $O(g(n))$)

Poll Everywhere

https://bit.ly/2LvG9bq



I think the answer is:

A. Adjacency List
B. Adjacency Matrix
C. They are the same
D. I'm still thinking about the Rubik's Cube

ADJACENCY LIST

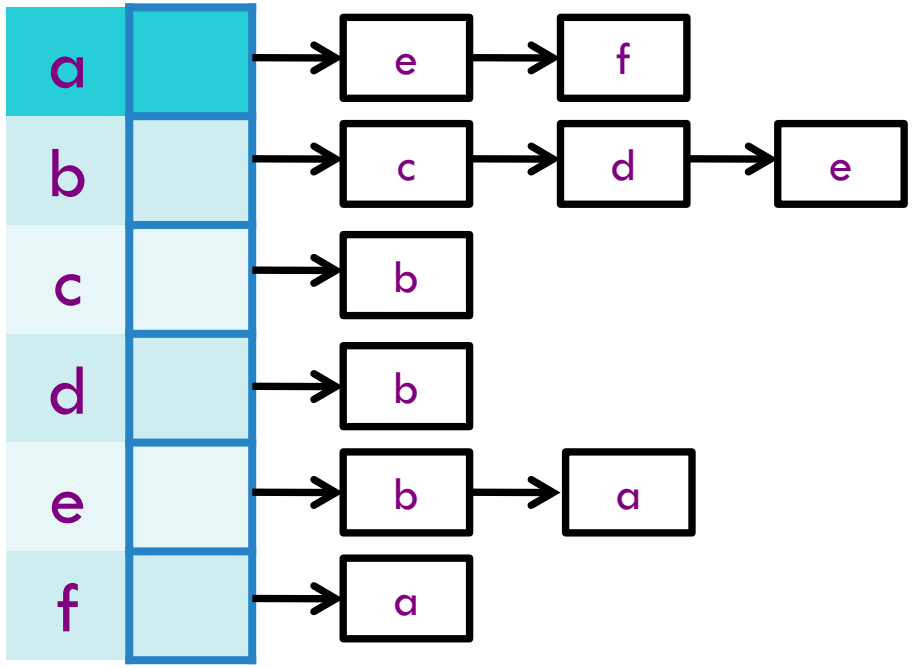
Memory usage for graph G:

array of size $|V|$

linked lists of size $|E|$

Total: $O(V + E)$

For a cycle: $O(V)$



ADJACENCY MATRIX

Memory usage for graph G:

array of size $|V| \times |V|$

Total: $O(V^2)$

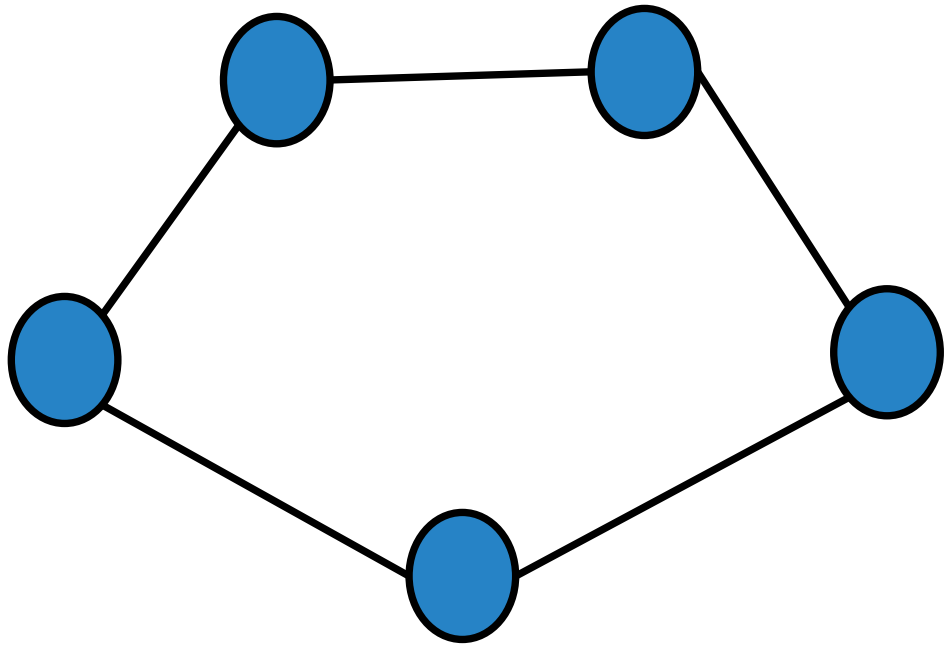
For a cycle: $O(V^2)$

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

FOR A CYCLE, WHICH REPRESENTATION IS MORE MEMORY EFFICIENT? (asymptotically… $O(g(n))$)

Poll Everywhere

https://bit.ly/2LvG9bq



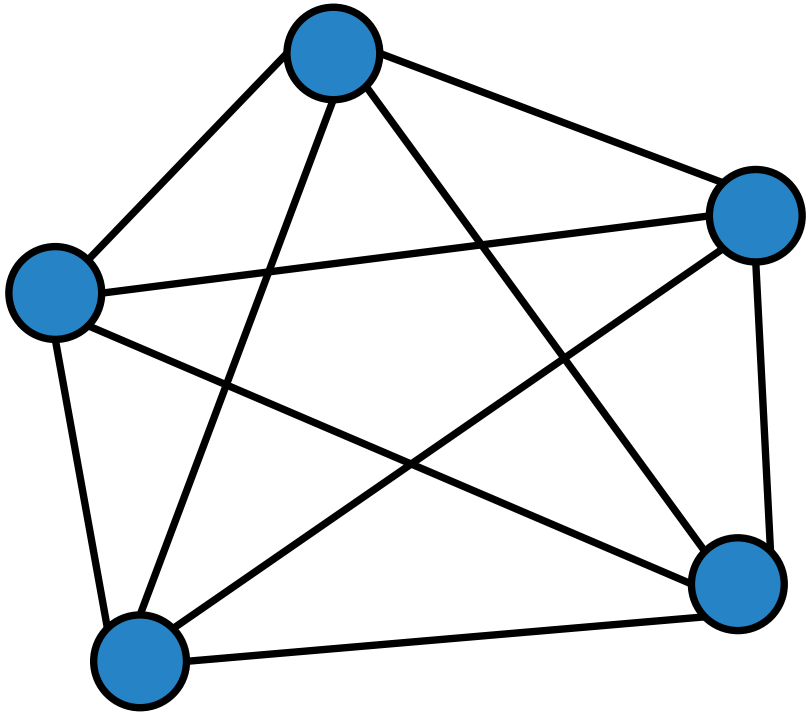
I think the answer is:

A. **Adjacency List**
B. Adjacency Matrix
C. They are the same
D. I'm still thinking about the Rubik's Cube

FOR A CLIQUE, WHICH REPRESENTATION IS MORE MEMORY EFFICIENT? (asymptotically… $O(g(n))$)

Poll Everywhere

https://bit.ly/2LvG9bq



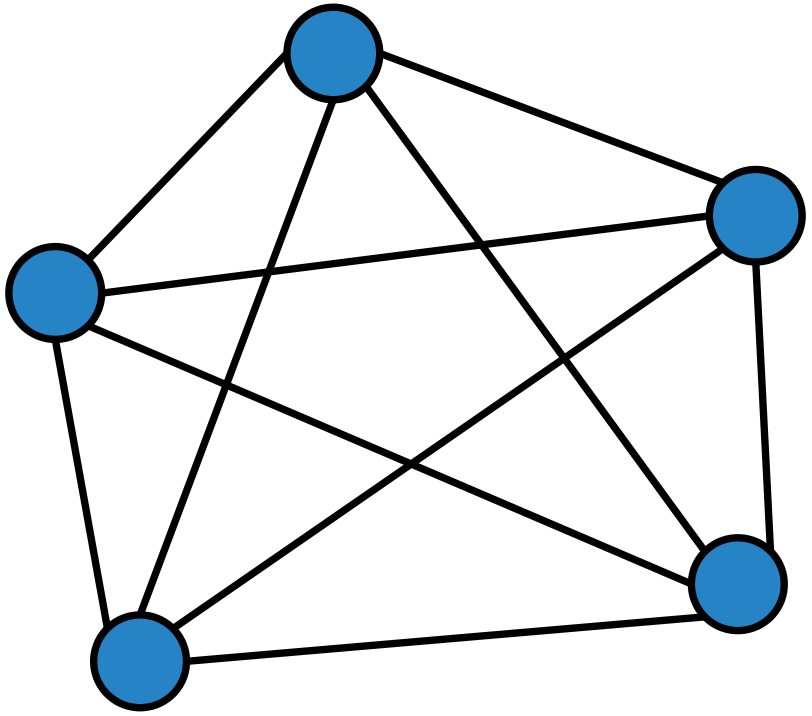
I think the answer is:

A. Adjacency List
B. Adjacency Matrix
C. Same Same
D. I'm **really** still thinking about the Rubik's Cube…

FOR A CLIQUE, WHICH REPRESENTATION IS MORE MEMORY EFFICIENT? (asymptotically… $O(g(n))$)

Poll Everywhere

https://bit.ly/2LvG9bq



I think the answer is:

A. Adjacency List
B. Adjacency Matrix
C. Same Same
D. I'm **really** still thinking about the Rubik's Cube…

ADJACENCY **LIST**

Memory usage for graph G:

array of size $|V|$

linked lists of size $|E|$

Total: $O(V + E)$

For a cycle: $O(V)$

For a clique: $O(V^2)$

ADJACENCY **MATRIX**

Memory usage for graph G:

array of size $|V| \times |V|$

Total: $O(V^2)$

For a cycle: $O(V^2)$

For a clique: $O(V^2)$

ADJACENCY **LIST**

Mem

arr

link

Tota

For a cycle: $O(V)$

For a clique: $O(V^2)$

ADJACENCY **MATRIX**

For a cycle: $O(V^2)$

For a clique: $O(V^2)$

General Guideline: If a graph is **dense** $|E| = \theta(V^2)$, use an adjacency matrix.

Otherwise, use an adjacency list.

GRAPH REPRESENTATIONS

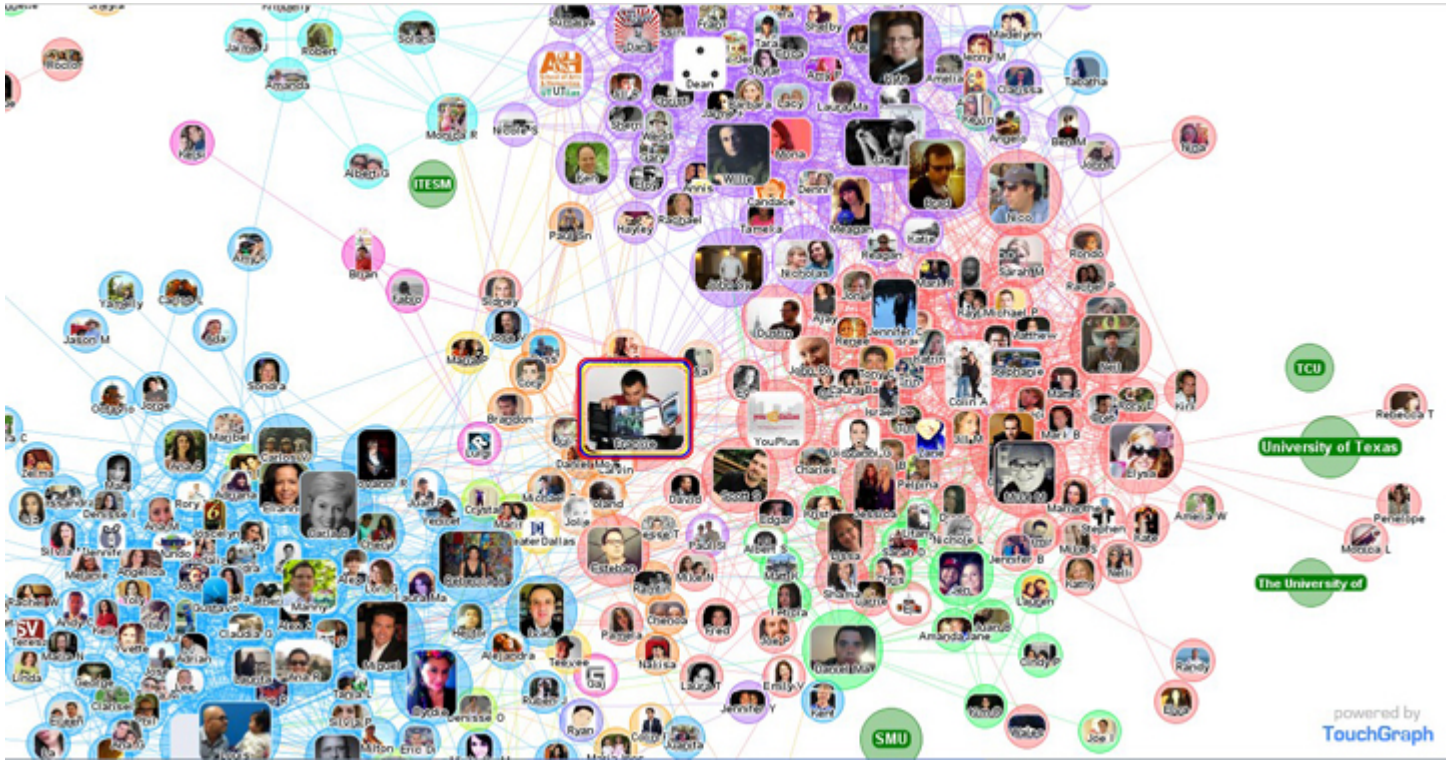
Key questions to ask:

- Space usage: is graph dense or sparse?
- Queries: what type of queries do I need?
  - Enumerate neighbors?
  - Query relationship?

QUERY: ARE ALICE & BOB FRIENDS?

Poll Everywhere

https://bit.ly/2LvG9bq



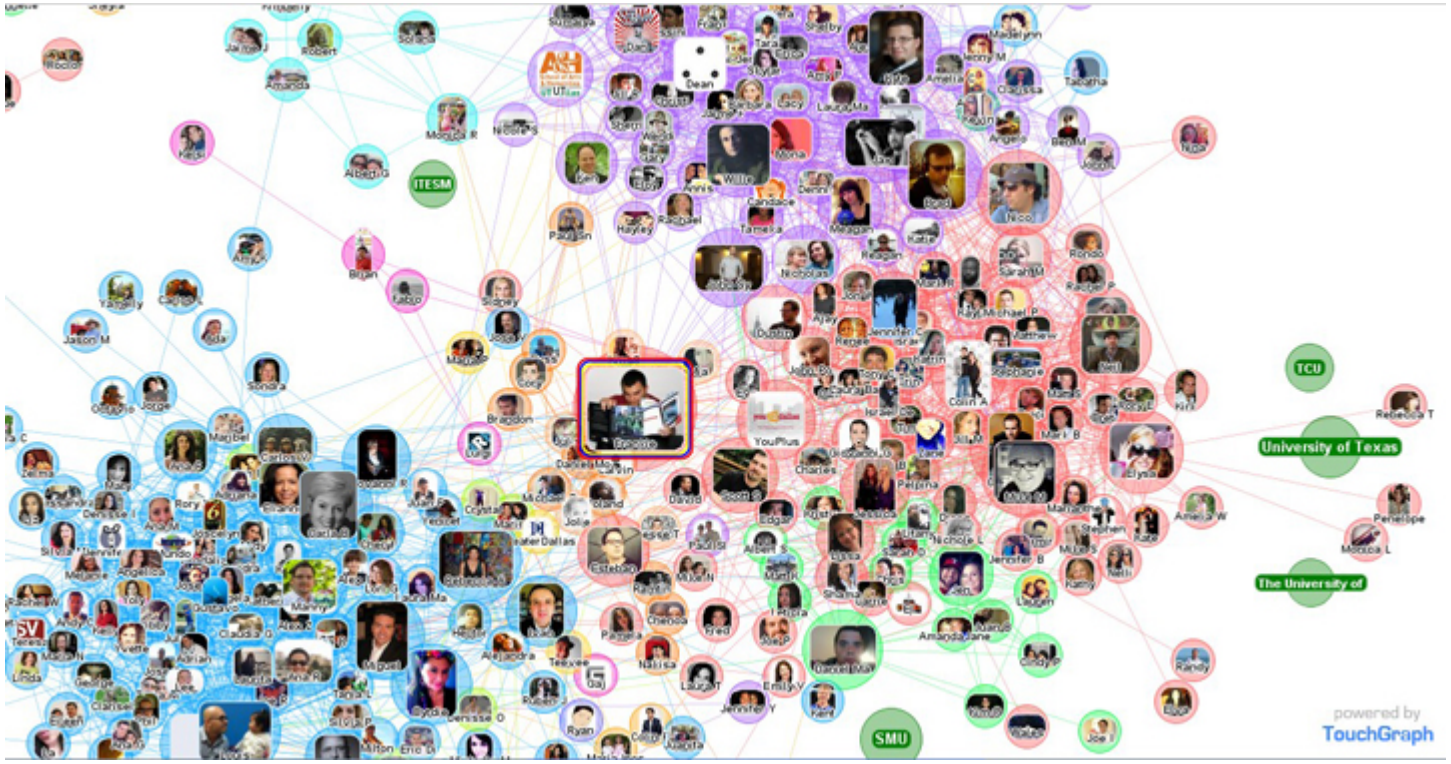
Faster representation for friend check?

A. Adjacency List
B. Adjacency Matrix
C. Same Same
D. I have no friends. ☹

QUERY: ARE ALICE & BOB FRIENDS?

Poll Everywhere

https://bit.ly/2LvG9bq



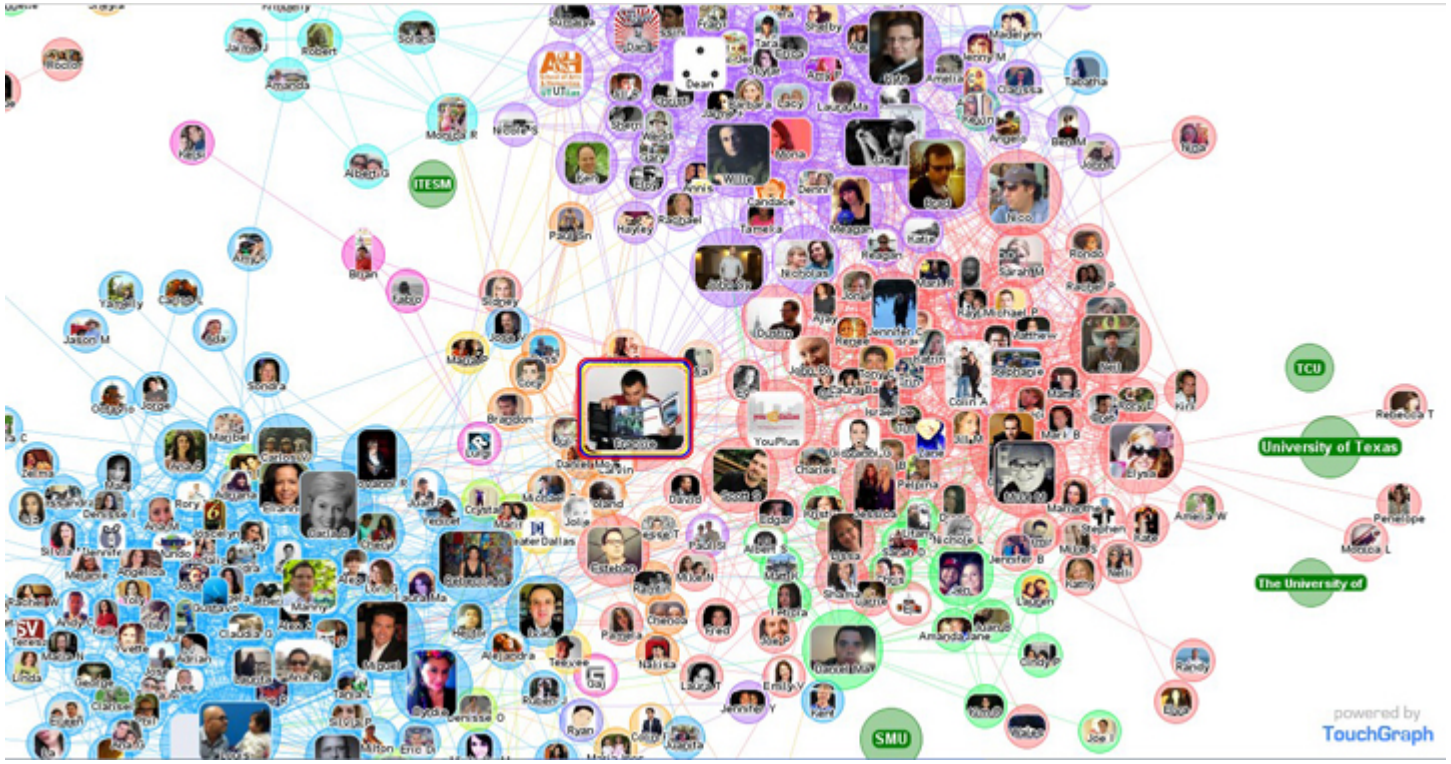
Faster representation for friend check?

A. Adjacency List
B. **Adjacency Matrix**
C. Same Same
D. I have no friends. ☹

Poll Everywhere

https://bit.ly/2LvG9bq

QUERY: WHO ARE ALICE'S FRIENDS?



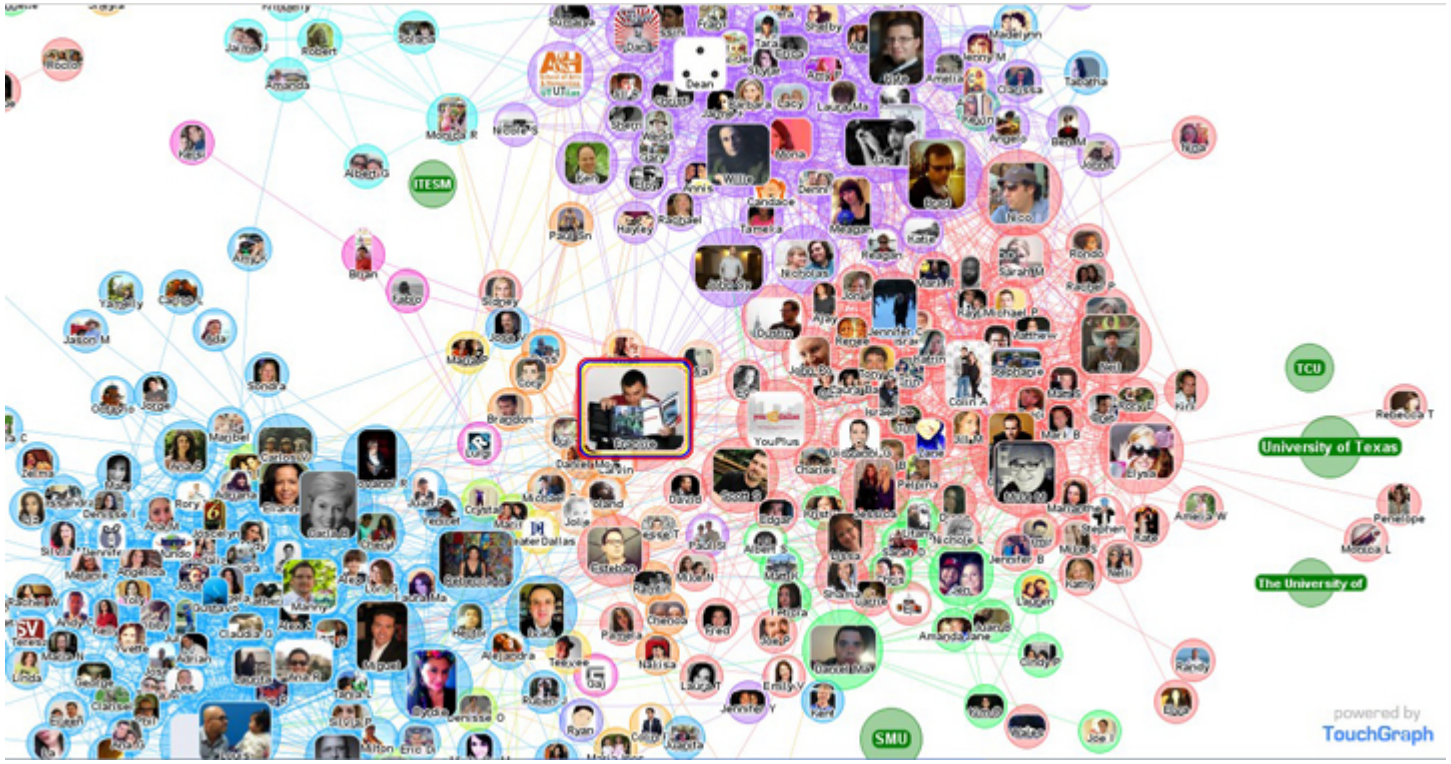
Faster representation for friend enumeration?

A. Adjacency List
B. Adjacency Matrix
C. Same Same
D. Not sure about this!

Poll Everywhere

https://bit.ly/2LvG9bq

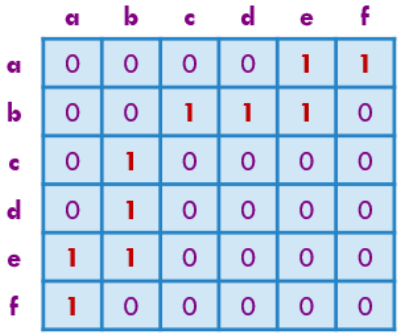
QUERY: WHO ARE ALICE'S FRIENDS?



Faster representation for friend enumeration?

A. **Adjacency List**
B. Adjacency Matrix
C. Same Same
D. Not sure about this!

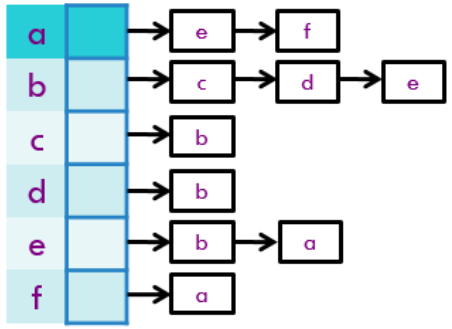
TRADE-OFFS



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

Adjacency Matrix:

- Fast query: are v and w neighbors?
- Slow query: find me any neighbor of v.
- Slow query: enumerate all neighbors.



Adjacency List:

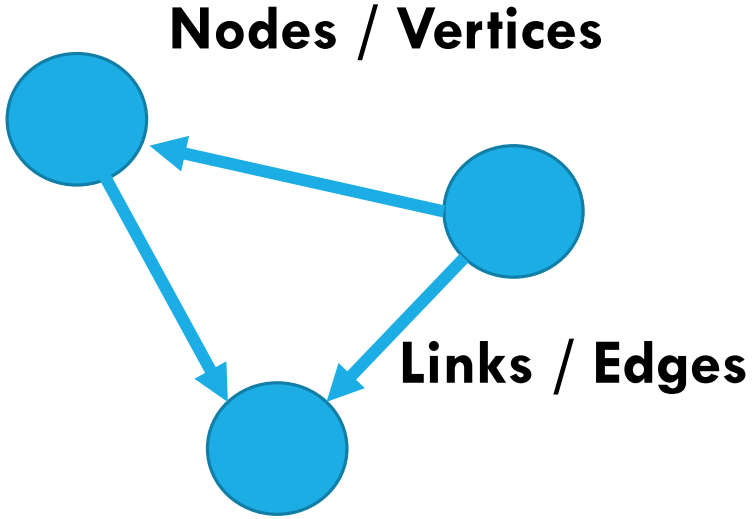
- Fast query: find me any neighbor.
- Fast query: enumerate all neighbors.
- Slower query: are v and w neighbors?

DIRECTED GRAPHS

Graph $G = \langle V, E \rangle$ ("a tuple of two sets")

- V is a set of nodes
- E is a set of edges
  - $E \subseteq \{ (v, w) : v, w \in V \}$

Order matters!

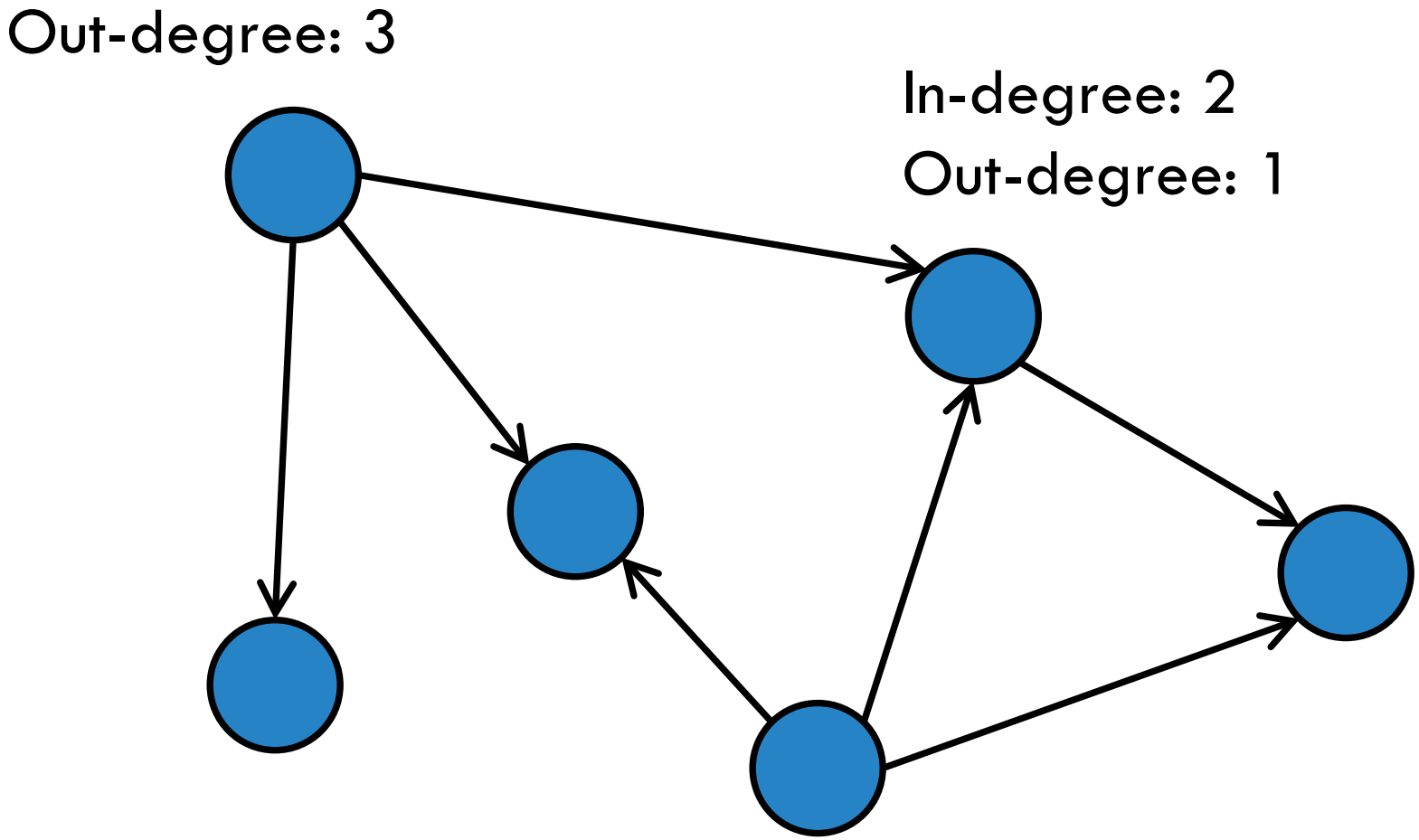


(v, w) means an edge pointing from $v \rightarrow w$

DEGREE OF NODES

In-degree: number of incoming edges

Out-degree: number of outgoing edges



REPRESENTING A DIRECTED GRAPH

Adjacency List:

- Array of nodes
- Each node maintains a list of neighbors
- Space: $O(V + E)$

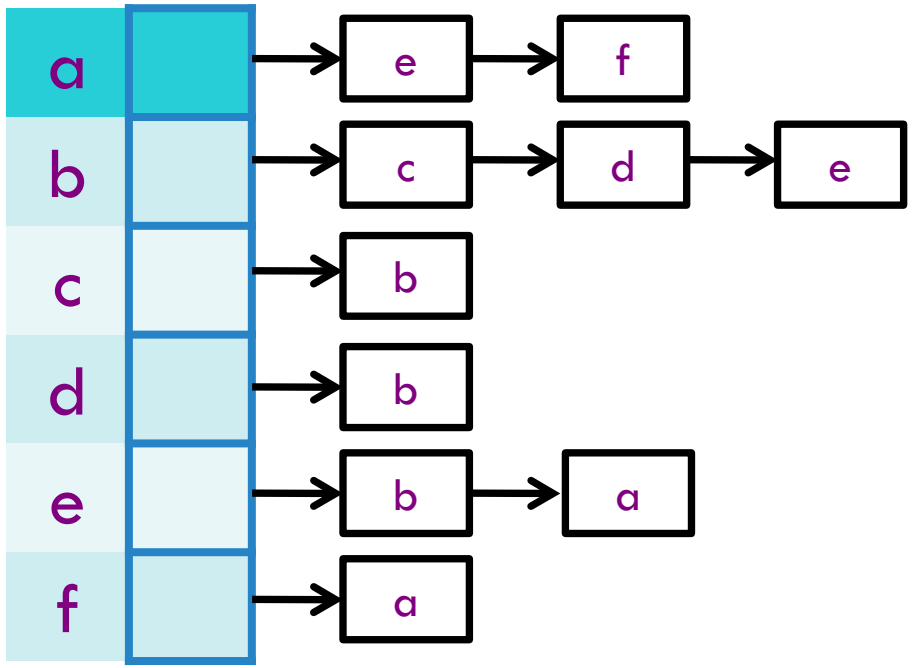
Adjacency Matrix:

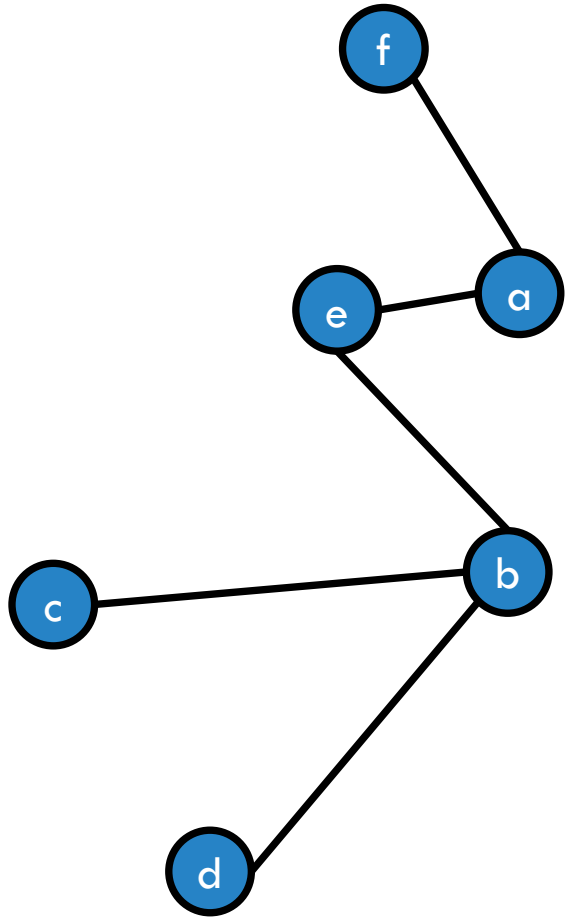
- Matrix $A[v, w]$ represents edge (v, w)
- Space: $O(V^2)$

ADJACENCY LIST

Undirected Graph consists of:

- Nodes: stored in an array
- Edges: linked list per node

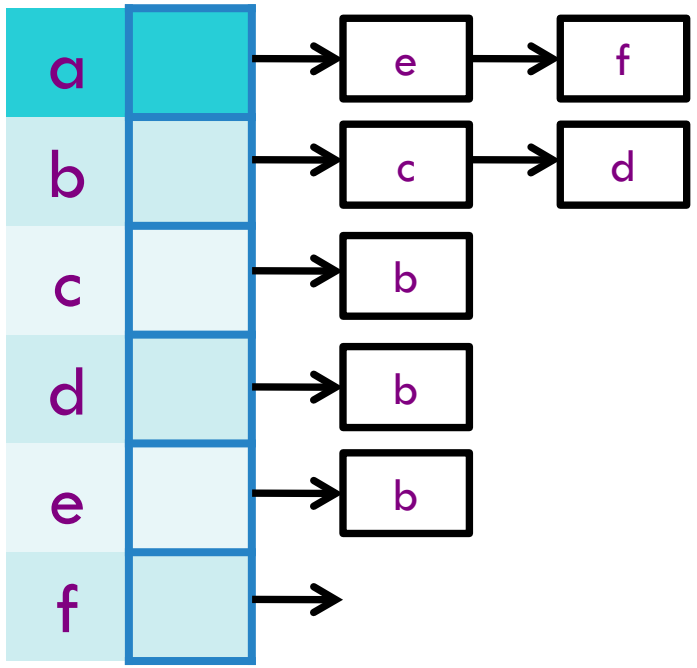


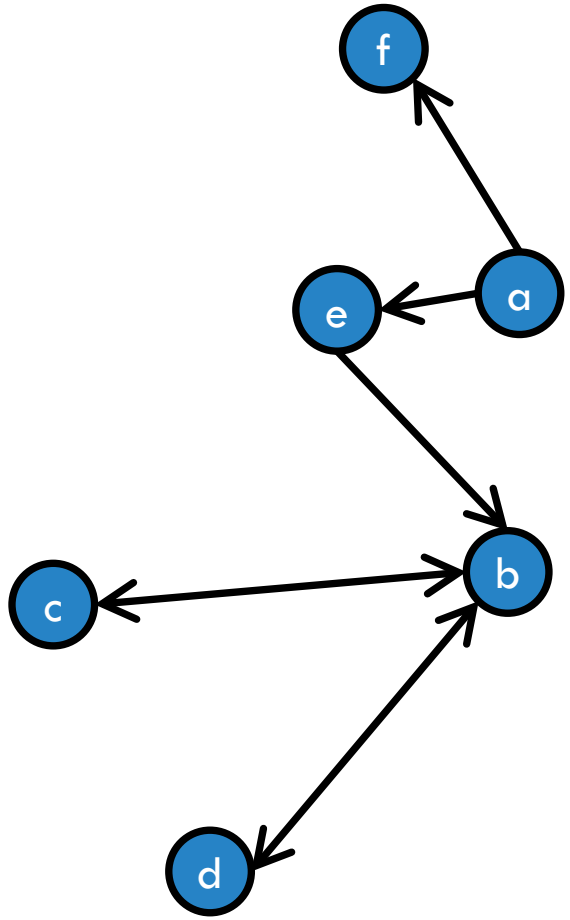


ADJACENCY LIST

Directed Graph consists of:

- Nodes: stored in an array
- Outgoing Edges: linked list per node





REPRESENTING A DIRECTED GRAPH

Adjacency List:

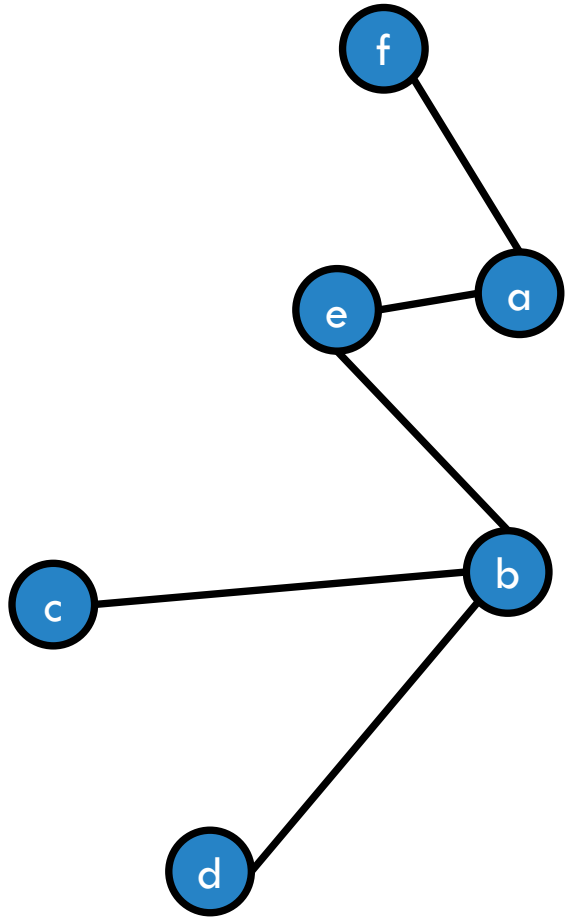
- Array of nodes
- Each node maintains a list of neighbors
- Space: $O(V + E)$

Adjacency Matrix:

- Matrix $A[v, w]$ represents edge (v, w)
- Space: $O(V^2)$

ADJACENCY MATRIX

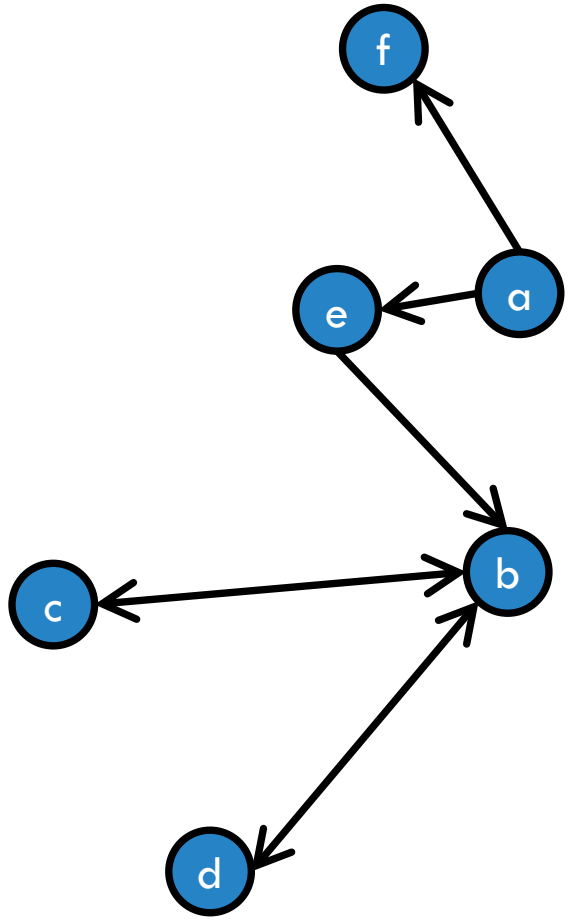
	a	b	c	d	e	f
a	0	0	0	0	1	1
b	0	0	1	1	1	0
c	0	1	0	0	0	0
d	0	1	0	0	0	0
e	1	1	0	0	0	0
f	1	0	0	0	0	0



ADJACENCY MATRIX

$$A[v][w] = 1 \text{ iff } (v, w) \in E$$

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



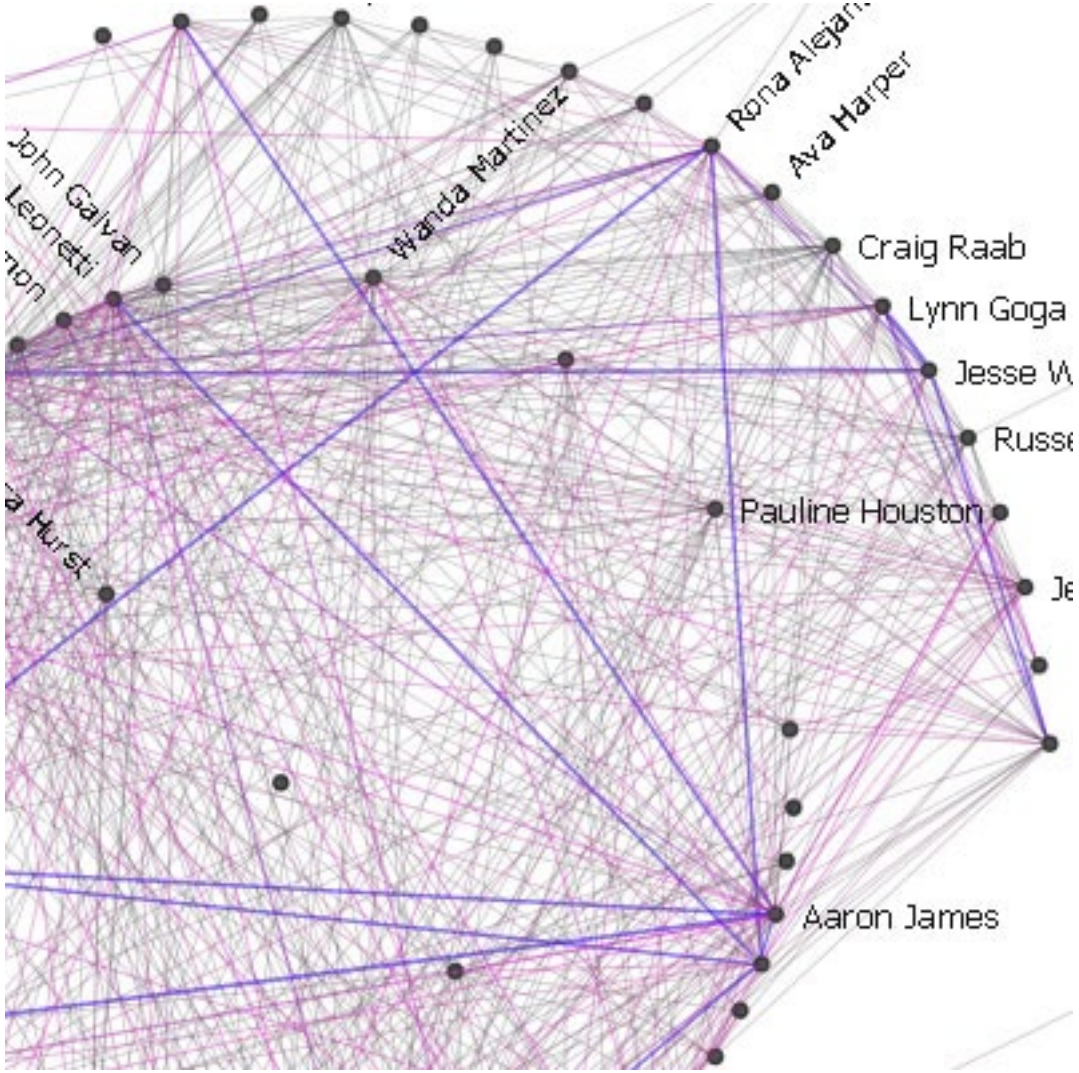
FRIENDSHIPS

Is friendship always bidirectional?:

- Nodes are people
- Edge = friendship

Facebook: yes

Google+: no



FRIENDSHIPS

Is friendship always bidirectional?:

- Nodes are people
- Edge = friendship

Facebook: yes

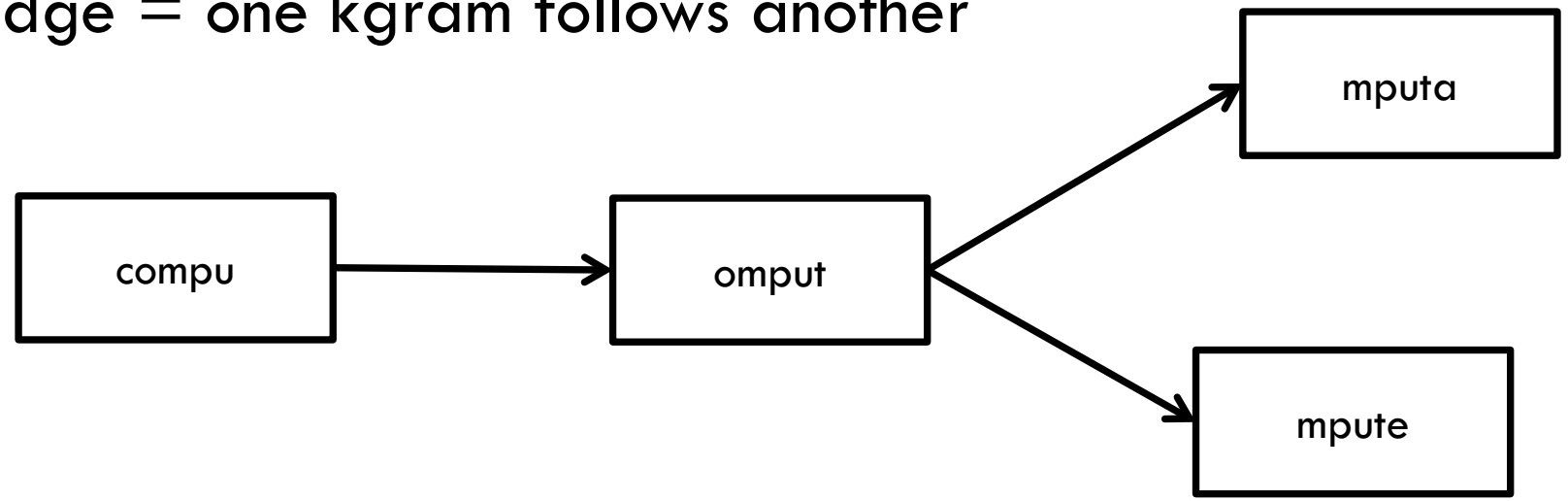
Google+: no

Google is Shutting Down Google+, Admits Low Consumer Adoption

TEXT GENERATION

Markov text generation:

- Nodes are kgrams
  - A k-gram is a contiguous sequence of k items e.g. syllables, letters, words, etc.
- Edge = one kgram follows another



SCIgen - An Automatic CS Paper Generator

About Generate Examples Talks Code Donations Related People Blog

About

SCIgen is a program that generates random Computer Science research papers, including graphs, figures, and citations. It uses a hand-written **context-free grammar** to form all elements of the papers. Our aim here is to maximize amusement, rather than coherence.

One useful purpose for such a program is to auto-generate submissions to conferences that you suspect might have very low submission standards. A prime example, which you may recognize from spam in your inbox, is SCI/IIIS and its dozens of co-located conferences (check out the very broad conference description on the **WMSCI 2005** website). There's also a list of **known bogus conferences**. Using SCIgen to generate submissions for conferences like this gives us pleasure to no end. In fact, one of our papers was accepted to SCI 2005! See **Examples** for more details.

We went to WMSCI 2005. Check out the **talks and video**. You can find more details in our **blog**.

Also, check out our 10th anniversary celebration project: **SCIpher**!

https://pdos.csail.mit.edu/archive/scigen/

A CONFERENCE ACCEPTED IT!

Rooter: A Methodology for the Typical Unification of Access Points and Redundancy

Jeremy Stribling, Daniel Aguayo and Maxwell Krohn

ABSTRACT

Many physicists would agree that, had it not been for congestion control, the evaluation of web browsers might never have occurred. In fact, few hackers worldwide would disagree with the essential unification of voice-over-IP and public-private key pair. In order to solve this riddle, we confirm that SMPs can be made stochastic, cacheable, and interposable.

I. INTRODUCTION

Many scholars would agree that, had it not been for active networks, the simulation of Lamport clocks might never have occurred. The notion that end-users synchronize with the investigation of Markov models is rarely outdated. A theoretical grand challenge in theory is the important unification

The rest of this paper is organized as follows. For starters, we motivate the need for fiber-optic cables. We place our work in context with the prior work in this area. To address this obstacle, we disprove that even though the much-tauted autonomous algorithm for the construction of digital-to-analog converters by Jones [10] is NP-complete, object-oriented languages can be made signed, decentralized, and signed. Along these same lines, to accomplish this mission, we concentrate our efforts on showing that the famous ubiquitous algorithm for the exploration of robots by Sato et al. runs in $\Omega((n + \log n))$ time [22]. In the end, we conclude.

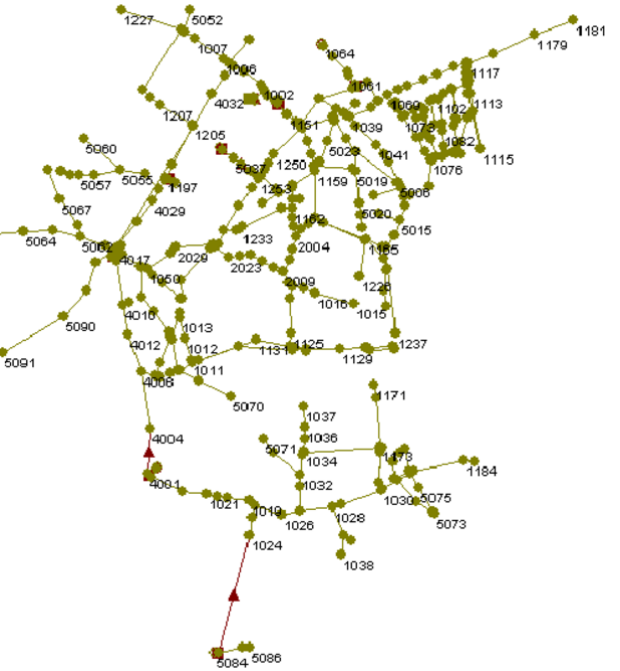
II. ARCHITECTURE

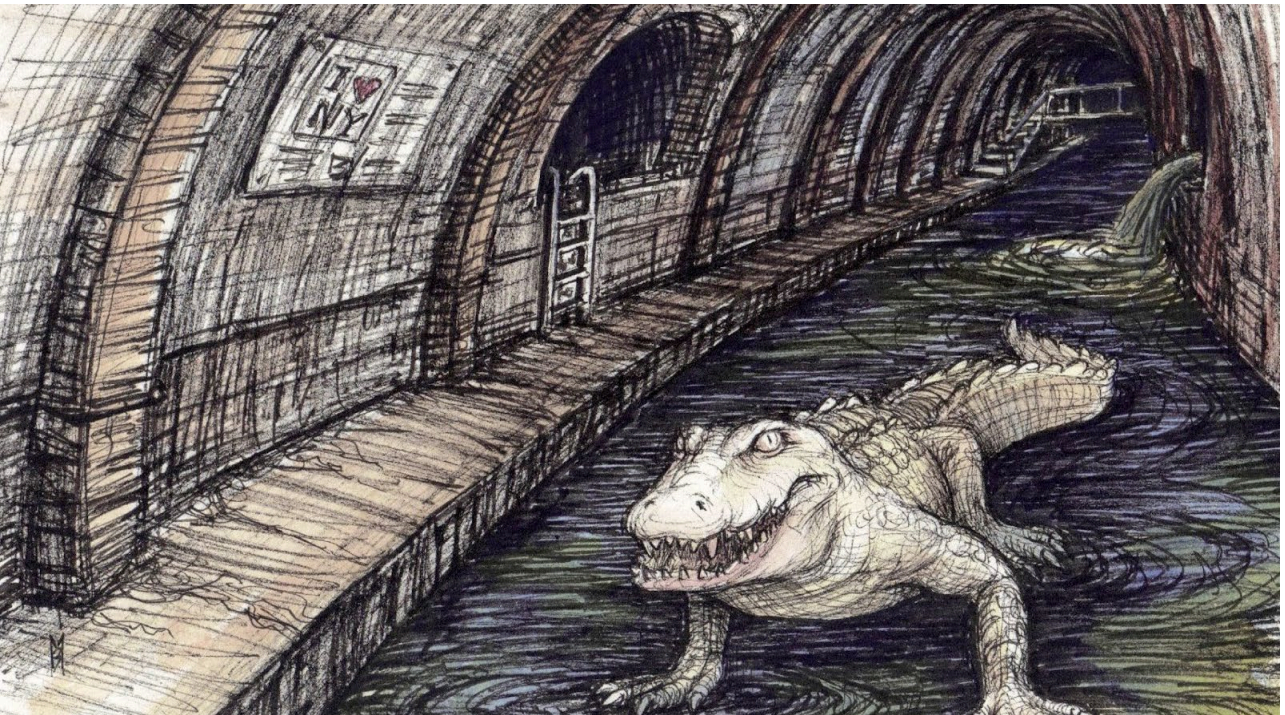
Our research is principled. Consider the early methodology by Martin and Smith; our model is similar, but will actually

SUMMARY: LEARNING OUTCOMES

By the end of this session, students should be able to:

- Describe the **graph** as a mathematical and data structure.
- Explain the **two primary methods to represent graphs** and the **key differences.**
- Give **examples** of where **graphs can be applied.**





PROBLEM: FINDING HERBERT.

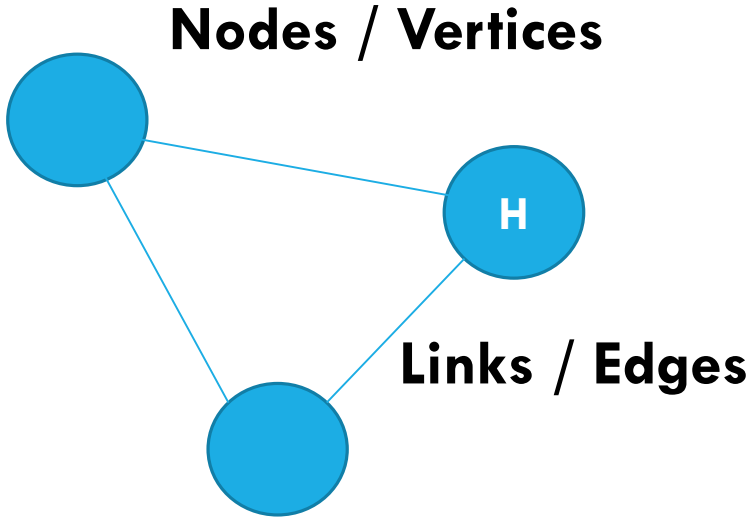
Herbert has gone missing!

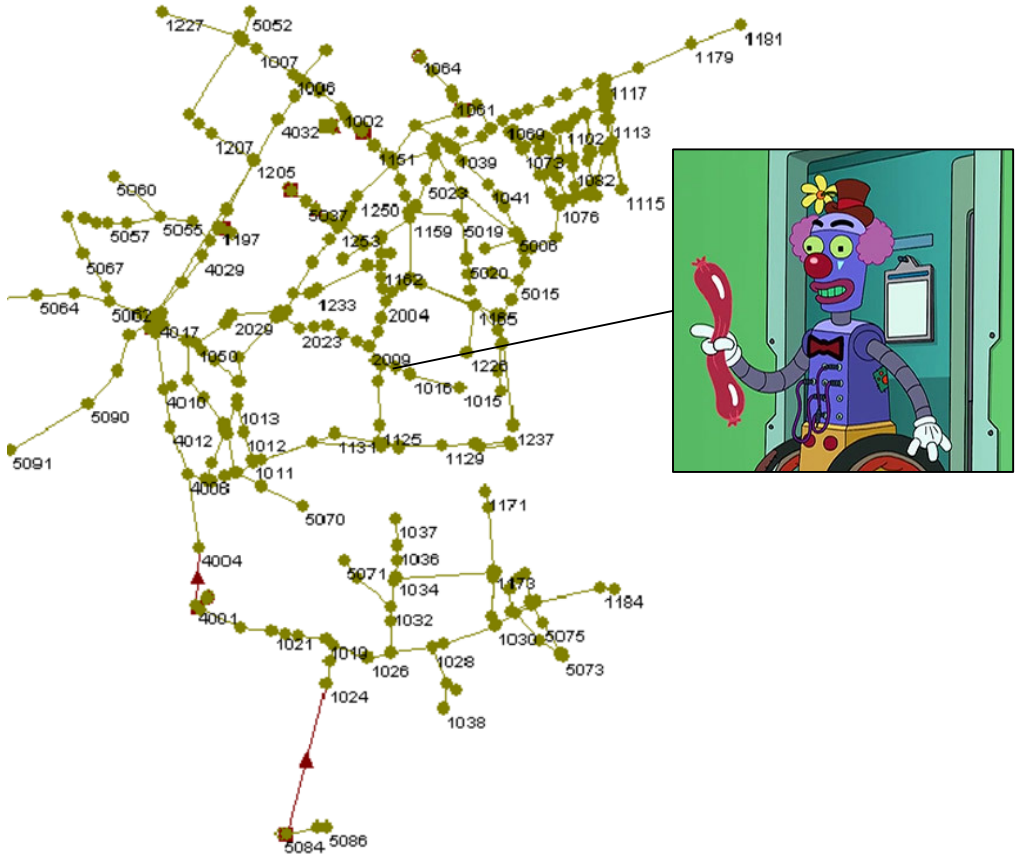
Last sighting: in the sewer system.

How can we *systematically* search for Herbert… before he gets destroyed by an alligator?

We'll save Herbert next week!

MODEL THE SEWER AS A GRAPH





SEARCHING A GRAPH

Goal:

- Start at some vertex **s** = start.
- Find some other vertex **f** = finish.

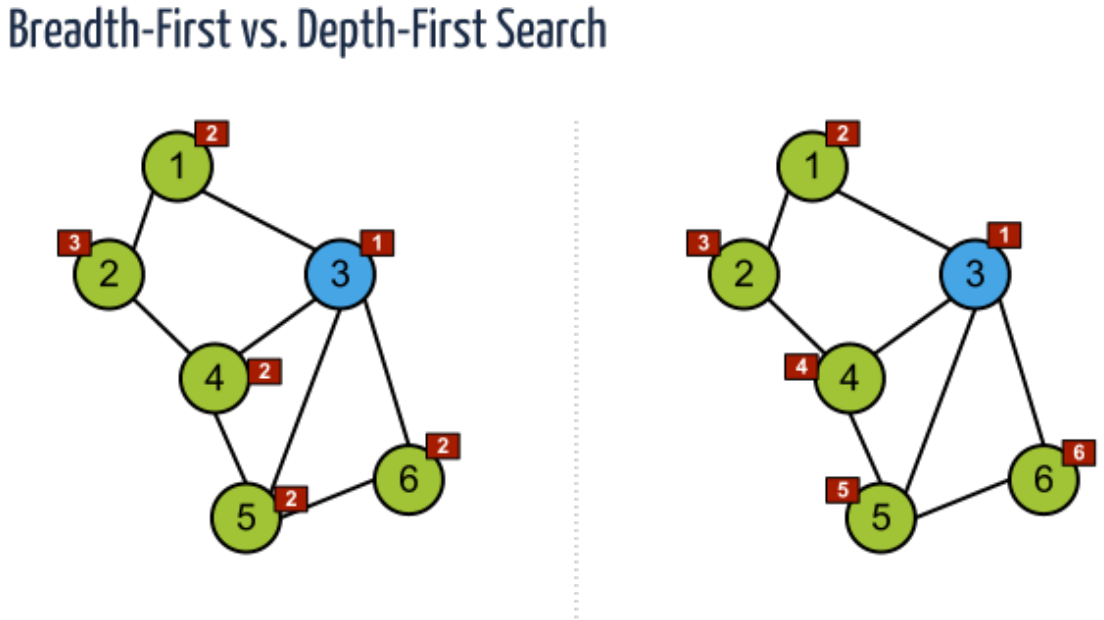
  Or: visit **all** the nodes in the graph

Two basic techniques:

- Breadth-First Search (BFS)
- Depth-First Search (DFS)

Graph representation:

- Adjacency list



QUESTIONS?