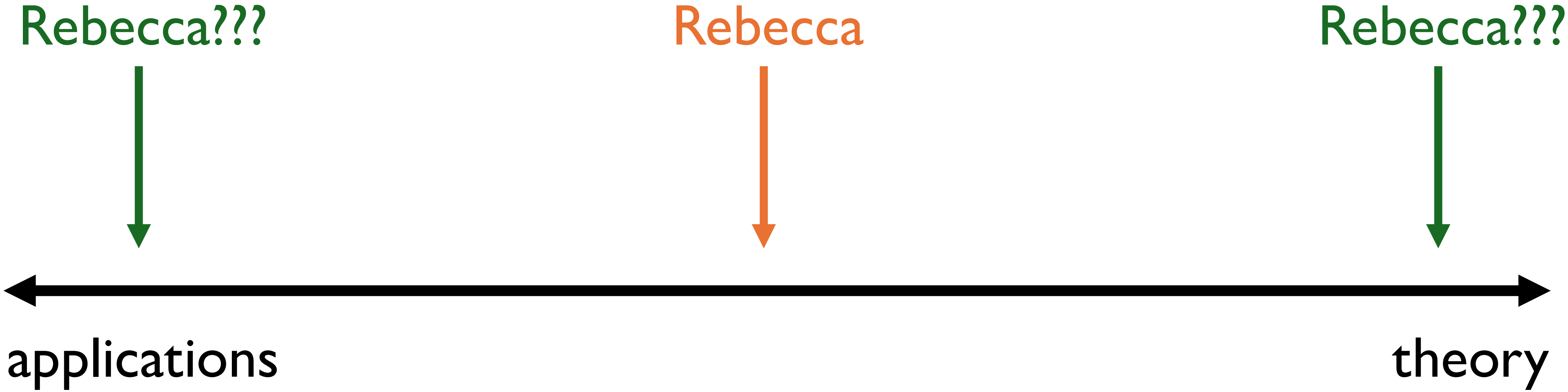
Why does theory matter in computer science?

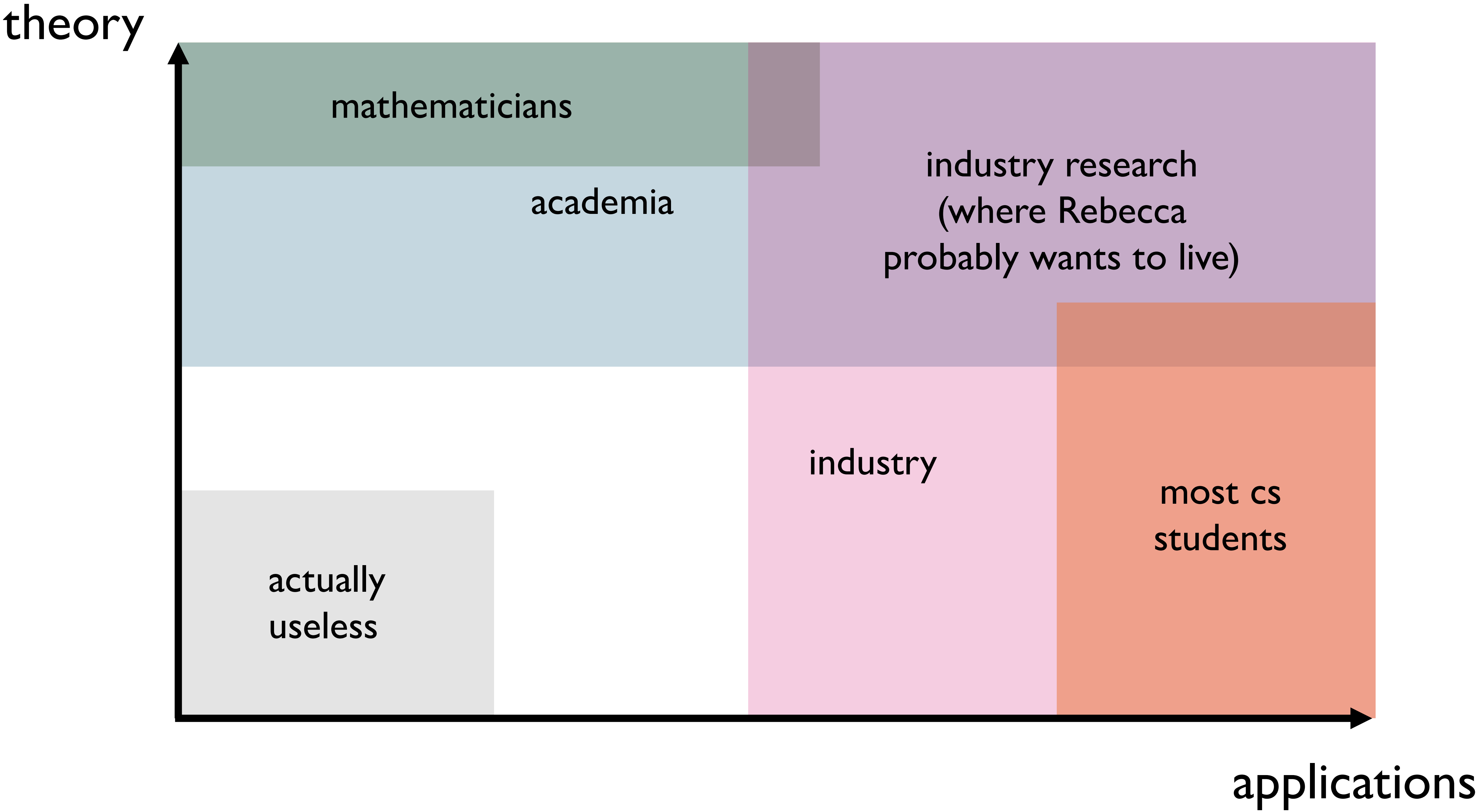
Rebecca Kempe

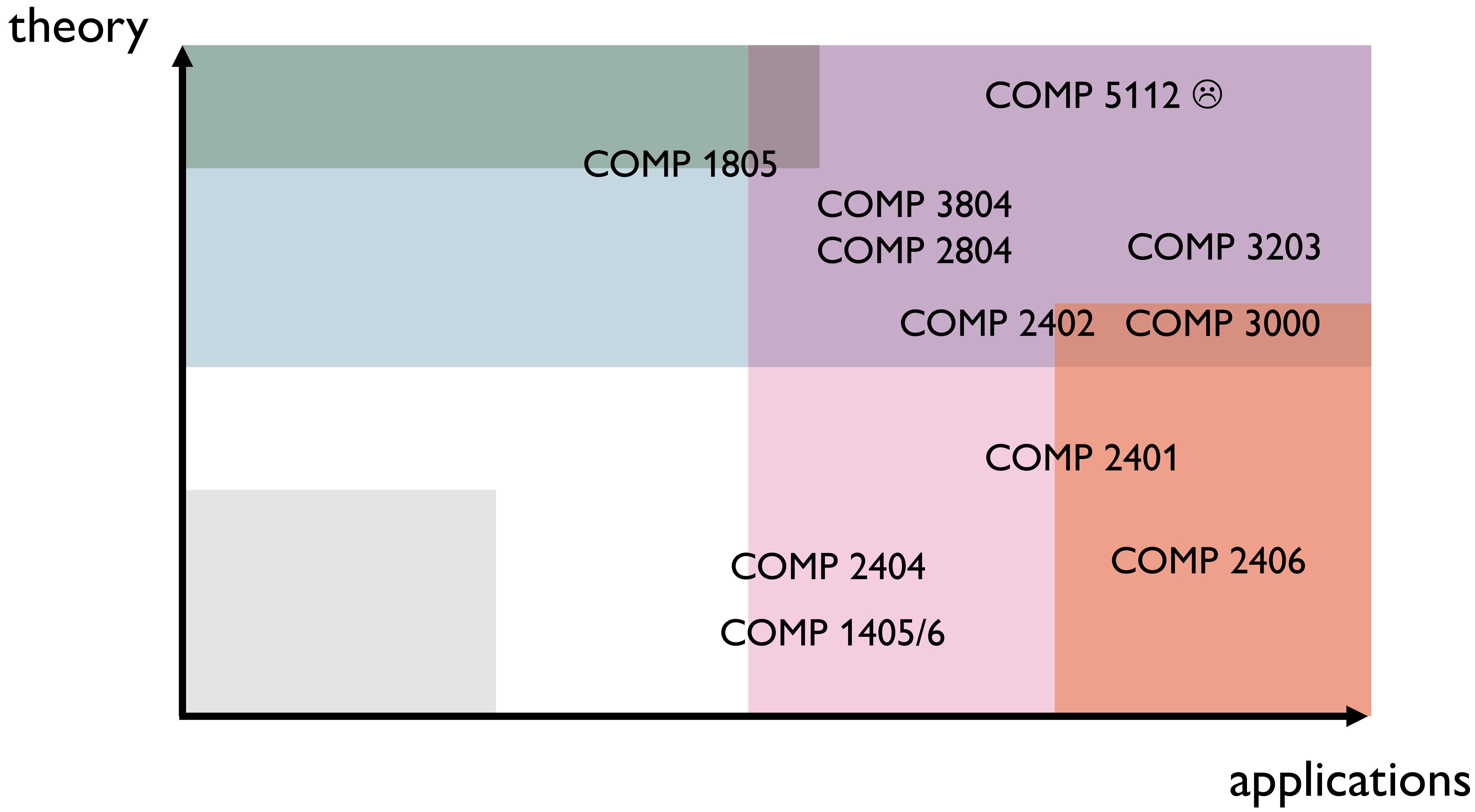
How many of you have taken COMP 1805?

How many of you liked it?


Rebecca
applications
theory


Rebecca???
Rebecca
Rebecca???
applications
theory


theory
mathematicians
academia
industry research
(where Rebecca
probably wants to live)
industry
most cs
students
actually
useless
applications


theory
COMP 5112 ☹
COMP 1805
COMP 3804
COMP 2804
COMP 3203
COMP 2402
COMP 3000
COMP 2401
COMP 2404
COMP 2406
COMP 1405/6
applications

Why does Rebecca think theory matter~~s~~ in computer science?* **

Rebecca Kempe

Why does Rebecca think theory matter~~s~~ in computer science?* **

Rebecca Kempe

* I am not an expert.

** also, I might "lie" to you.

“lies-to-children”

Lie-to-children

3 languages

Article Talk Read Edit View history Tools

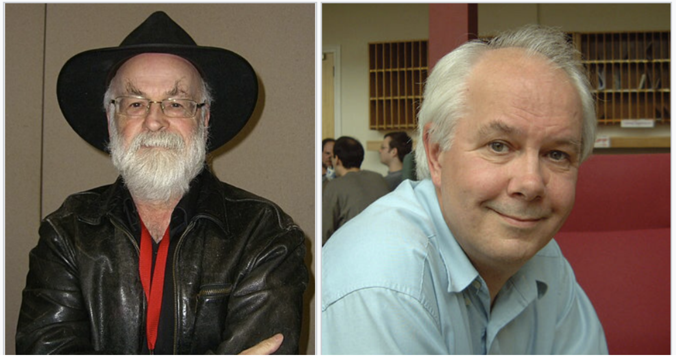
From Wikipedia, the free encyclopedia

A **lie-to-children** is a simplified, and often technically incorrect, explanation of technical or complex subjects employed as a teaching method. Educators who employ lies-to-children do not intend to deceive, but instead seek to 'meet the child/pupil/student where they are', in order to facilitate initial comprehension, which they build upon over time as the learner's intellectual capacity expands. The technique has been incorporated by academics within the fields of biology, evolution, bioinformatics and the social sciences.

Origin and development [edit]

The "lie-to-children" concept was first discussed by scientist Jack Cohen and mathematician Ian Stewart in the 1994 book *The Collapse of Chaos: Discovering Simplicity in a Complex World* as myths —a means of ensuring that accumulated cultural lore is passed on to future generations in a way that was sufficient but not completely true.[1][2][3]

The Science of Discworld **authors**



A lie-to-children is a simplified (and maybe also technically incorrect) explanation of a concept used for teaching purposes.

Pictures, for example, are *useful*, but ofted flawed.

Part 1:

WTF is theory???

“theory”

1. a plausible or scientifically acceptable general principle or body of principles offered to explain phenomena
2. a belief, policy, or procedure proposed or followed as the basis of action
3. the analysis of a set of facts in their relation to one another

(Merriam-Webster Dictionary)

“theory”

1. a formal set of ideas that is intended to explain why something happens or exists

(Oxford Learner’s Dictionary)

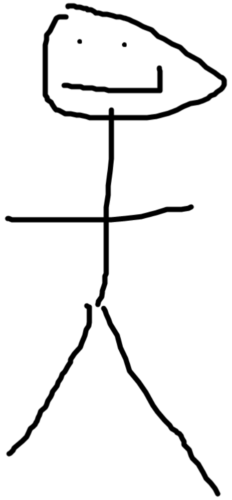
Theory gives us language, frameworks, and ideas that help us sort problems into different types and figure out common ways of solving them.

(Rebecca's definition)

abstraction and generalization

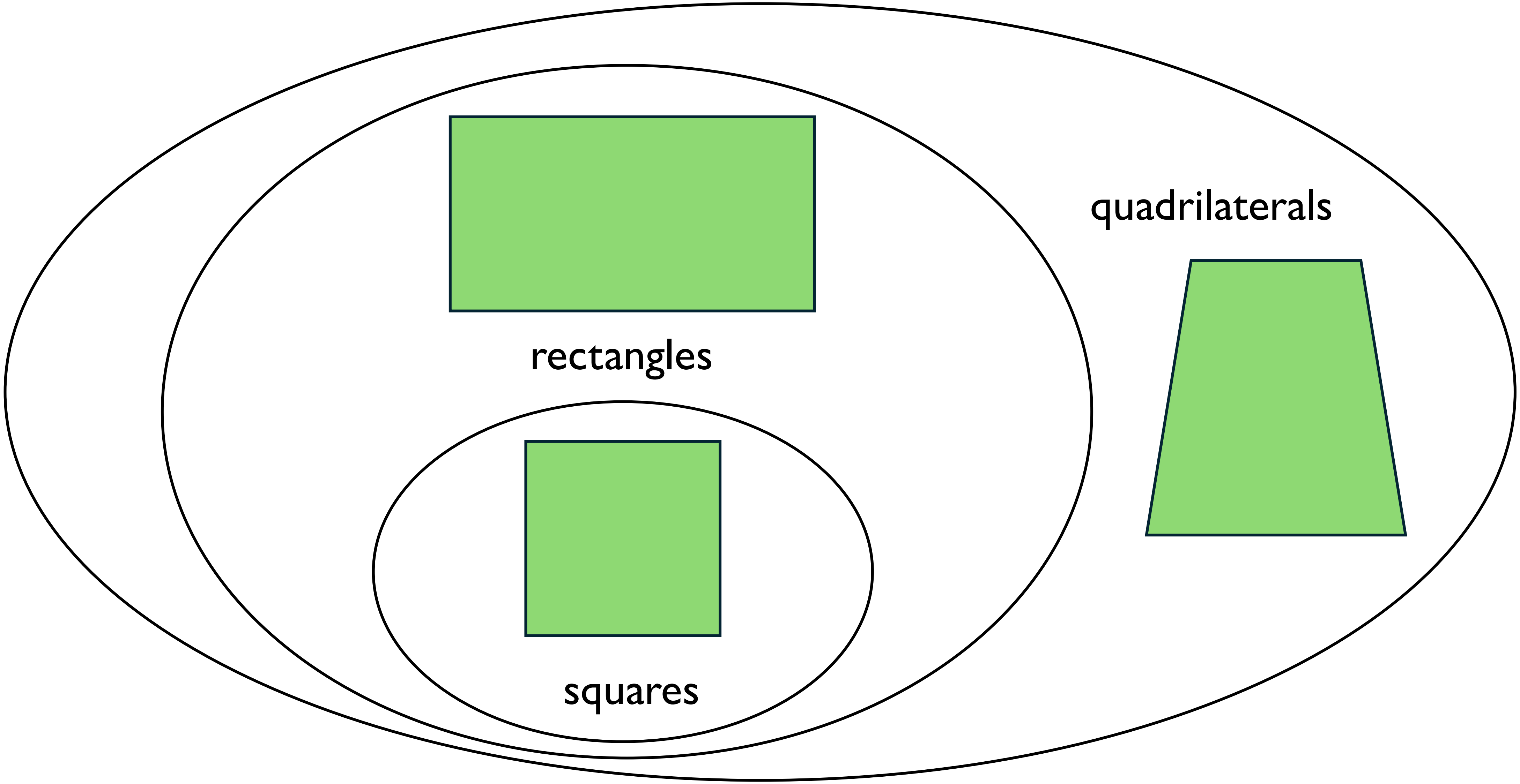
- abstraction: stripping away unnecessary details so that the bigger picture becomes more clear
- generalization: viewing objects in terms of ways in which they're the same

abstraction (examples)

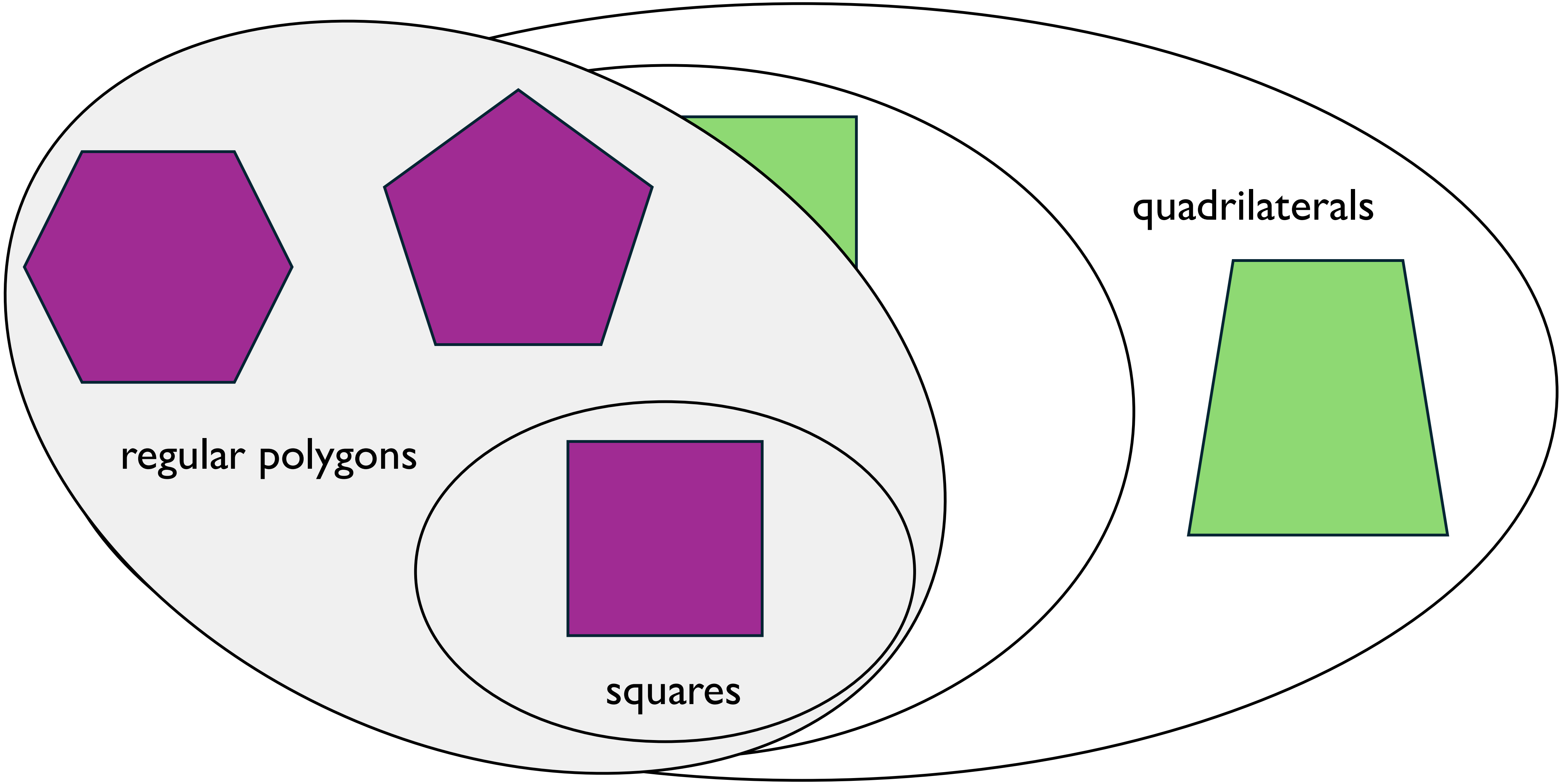
 →  → "person"

machine code → JVM → bytecode →
→ Java Code → pseudocode

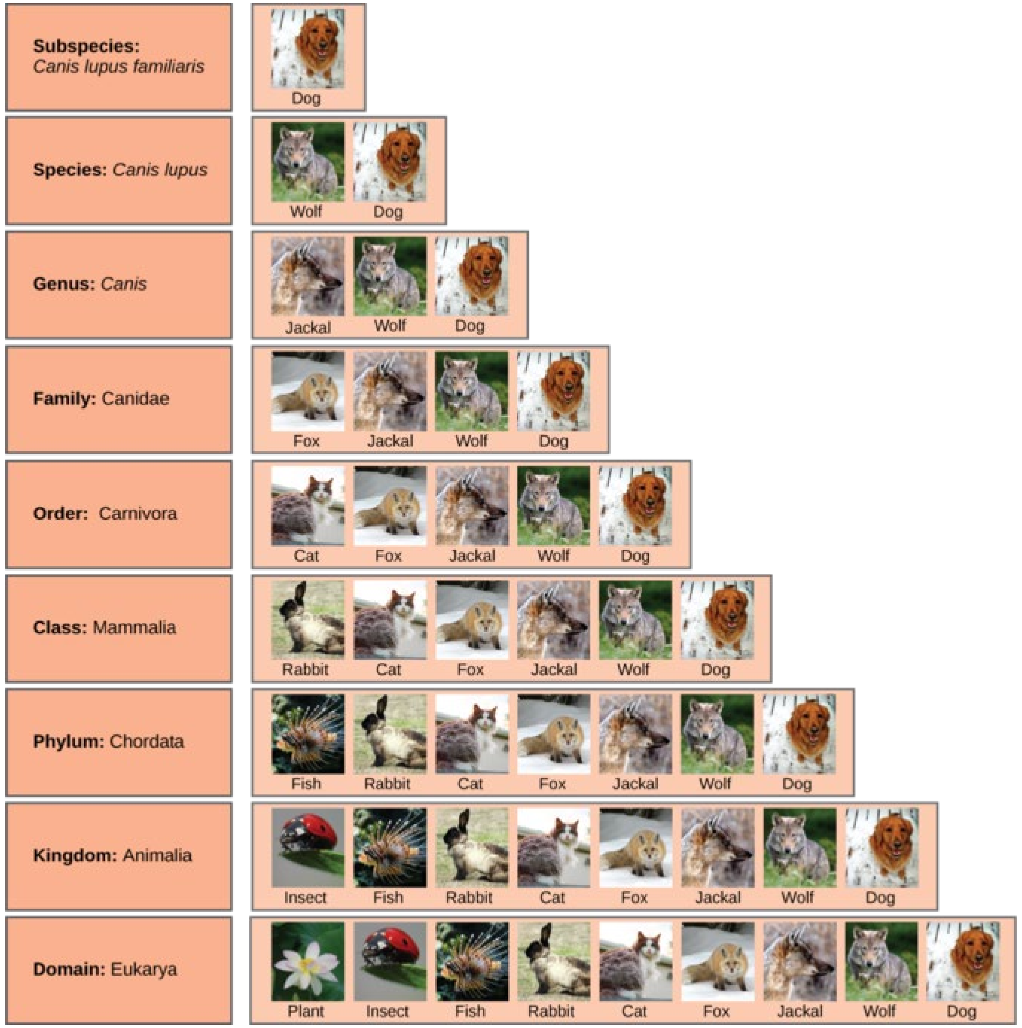
generalization (examples)



generalization (examples)



generalization (examples)



In biology, taxonomy is the classification of organisms into inclusive groupings.

Generalization and classification are tightly linked concepts.

theory $\neq$ math!

However, math gives us a *precise* language for describing problems abstractly

note: precise $\neq$ clear!

where are we going with this?

- we will look at some real-world problems
- we will abstractly describe them using math
- we will look at some solutions
- we will see how generalizing these solutions gives us solutions to related problems.

where are we going with this?

- we will look at some real-world problems
- we will abstractly describe them using math
- we will look at some solutions
- we will see how generalizing these solutions gives us solutions to related problems.

This is the power of theory!

Part 2:

Some interesting “real-world” problems

community detection

- how do we detect groups of web pages that are related to each other?
- how do we find the “authoritative” web pages?

(Kleinberg, 1999)

topic clustering

- how do we decide which content is similar?
- which groups of related content are the most popular?
- this is also known as “large near-clique extraction”

correlation mining

- which assets are strongly correlated?
- which assets are most influential on the overall market?
- correlation mining is also used in genetics, neuroscience, spam detection, etc.

How do we solve these problems?* **

How do we solve these problems?* **

* How do we solve them efficiently?

** wait... are these all the same problem?

Part 3:

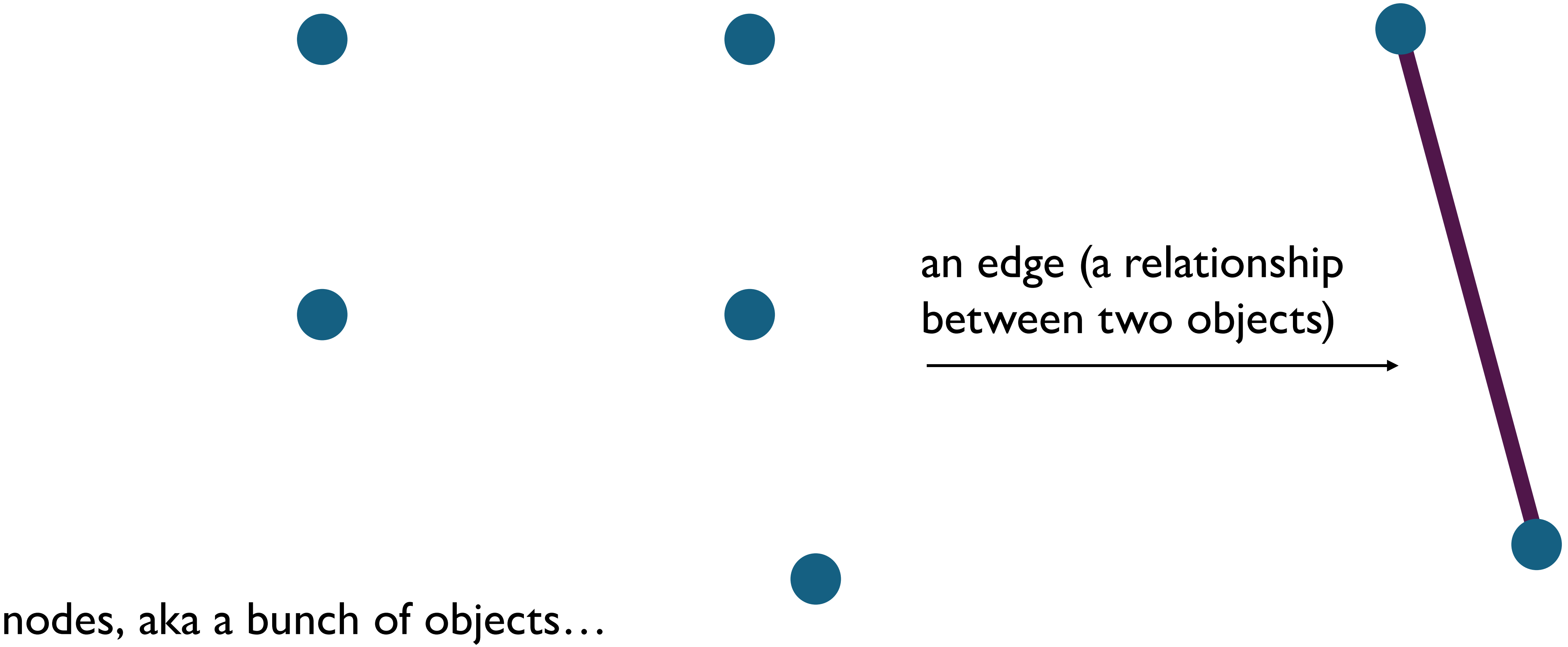
The Densest Subgraph Problem (DSP)

(an abstraction)

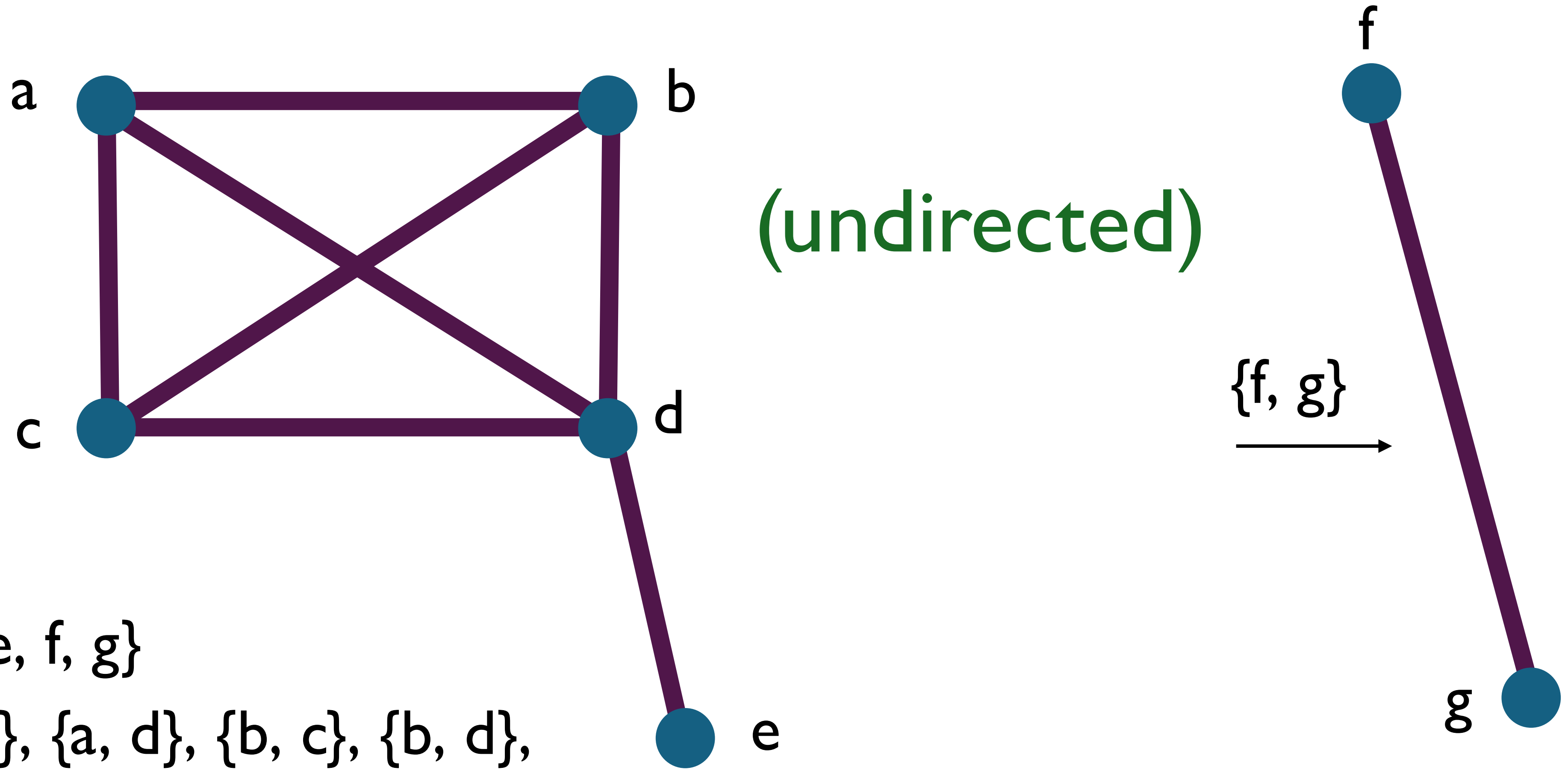
graph theory (a crash course)

A graph is a way of representing a set of objects (nodes/vertices) and the pairwise relationships between them (edges).

graph theory (a crash course)



graph theory (a crash course)

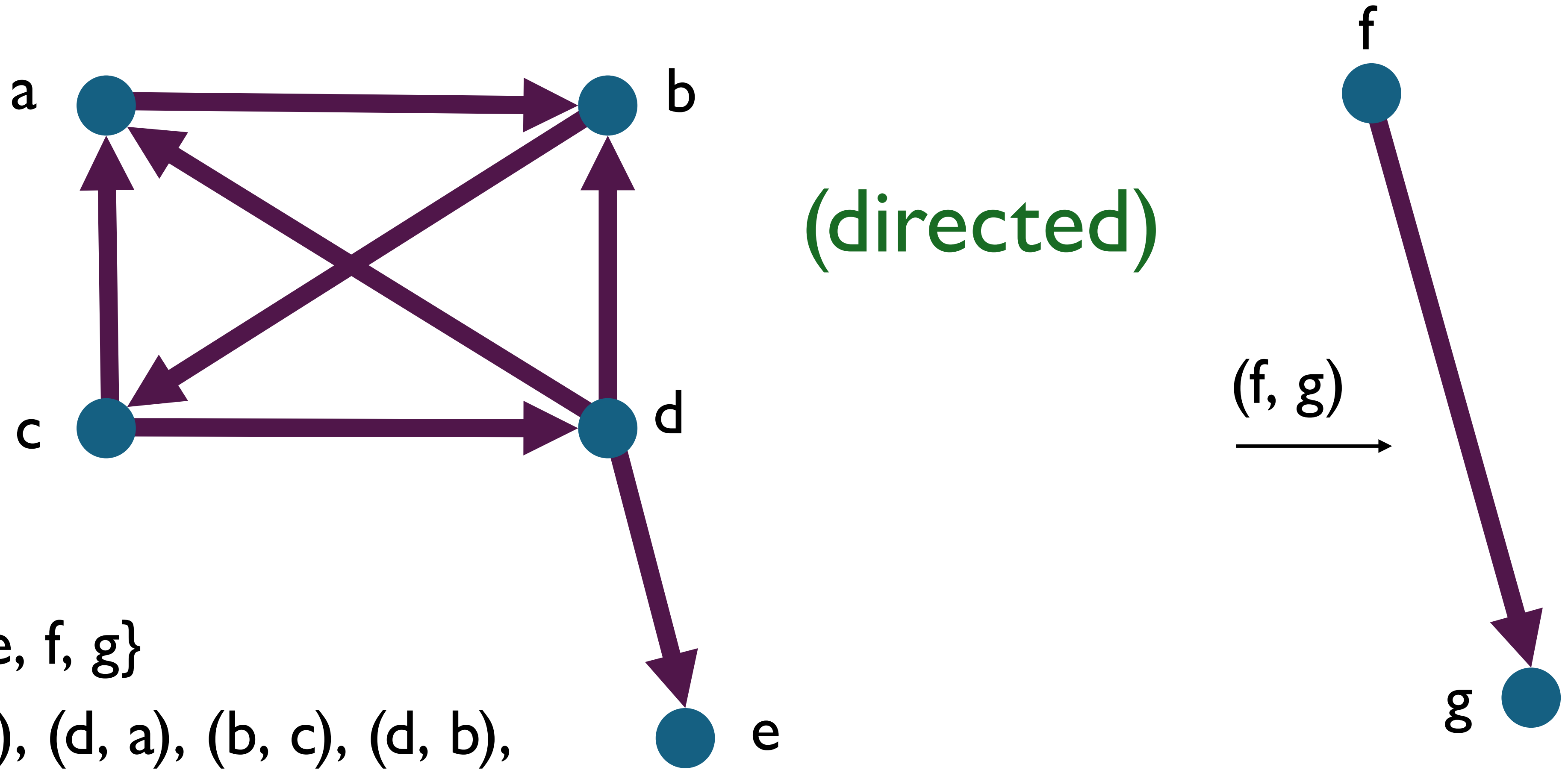


$G = (V, E)$

$V = \{a, b, c, d, e, f, g\}$

$E = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\},$
$\{c, d\}, \{d, e\}, \{f, g\}\}$

graph theory (a crash course)

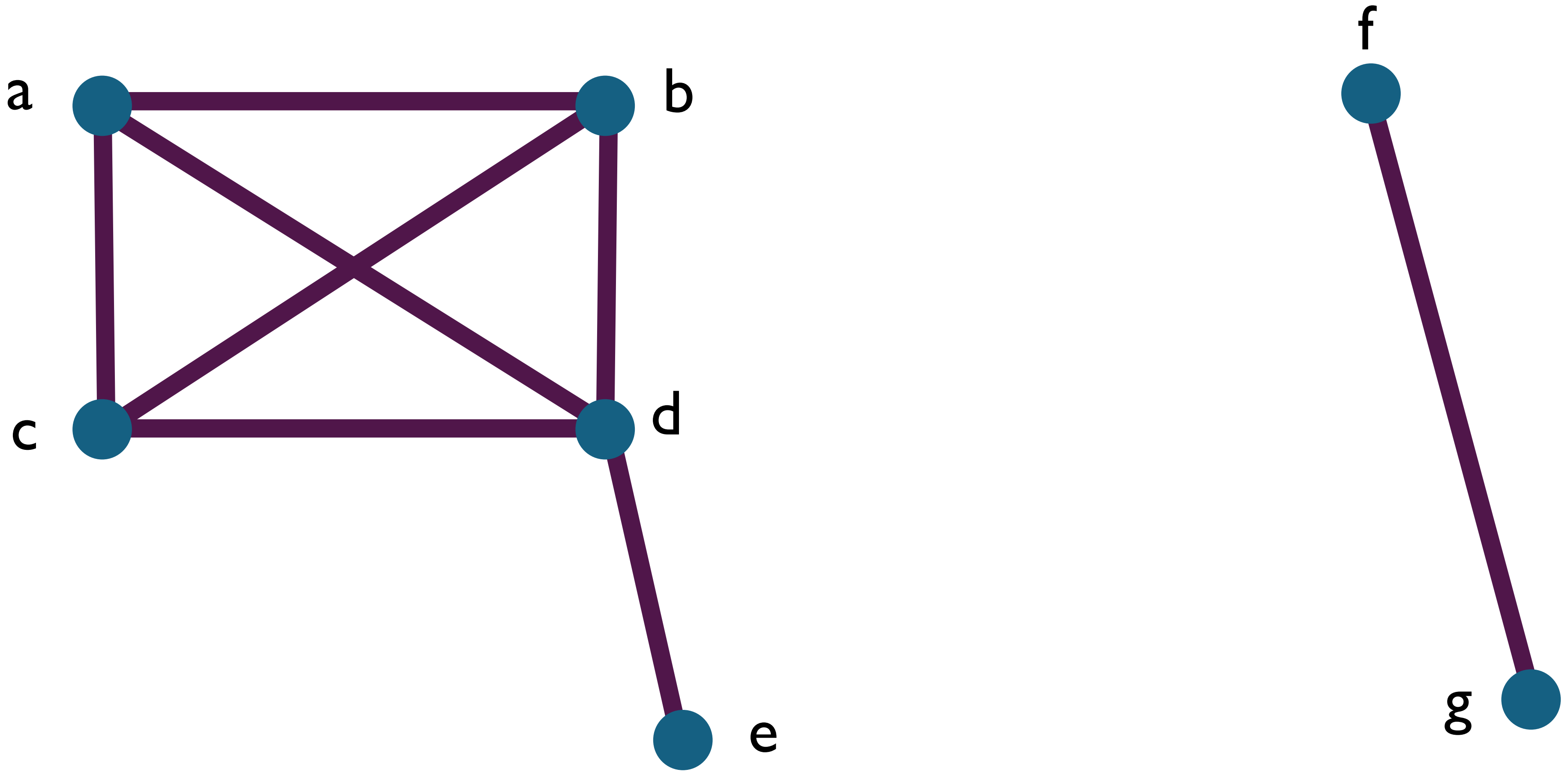


$G = (V, E)$

$V = \{a, b, c, d, e, f, g\}$

$E = \{(a, b), (c, a), (d, a), (b, c), (d, b),$
$(c, d), (d, e), (f, g)\}$

graph theory (a crash course)

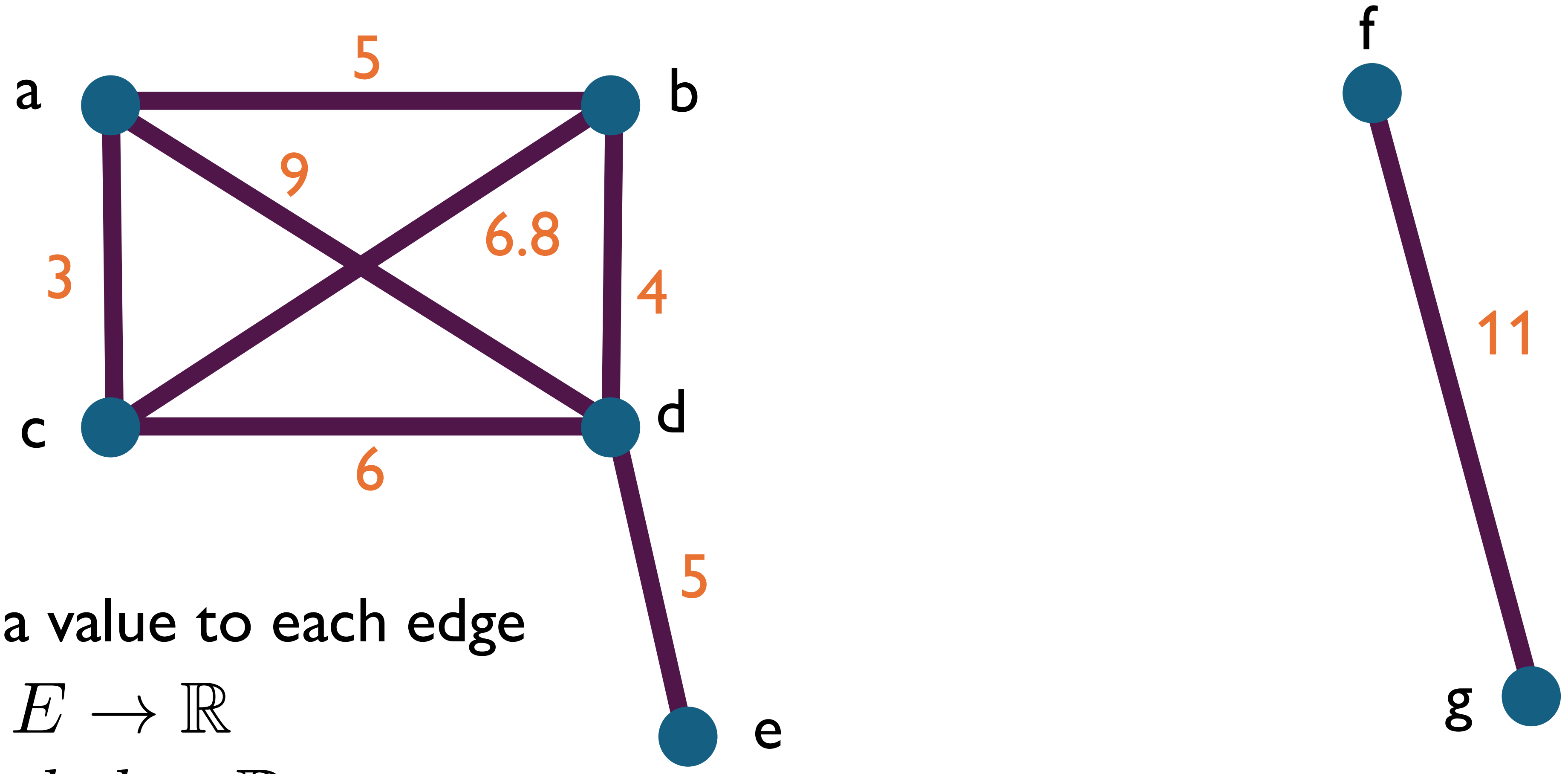


G = (V, E)

order = |V| = 7

size = |E| = 8

graph theory (a crash course)

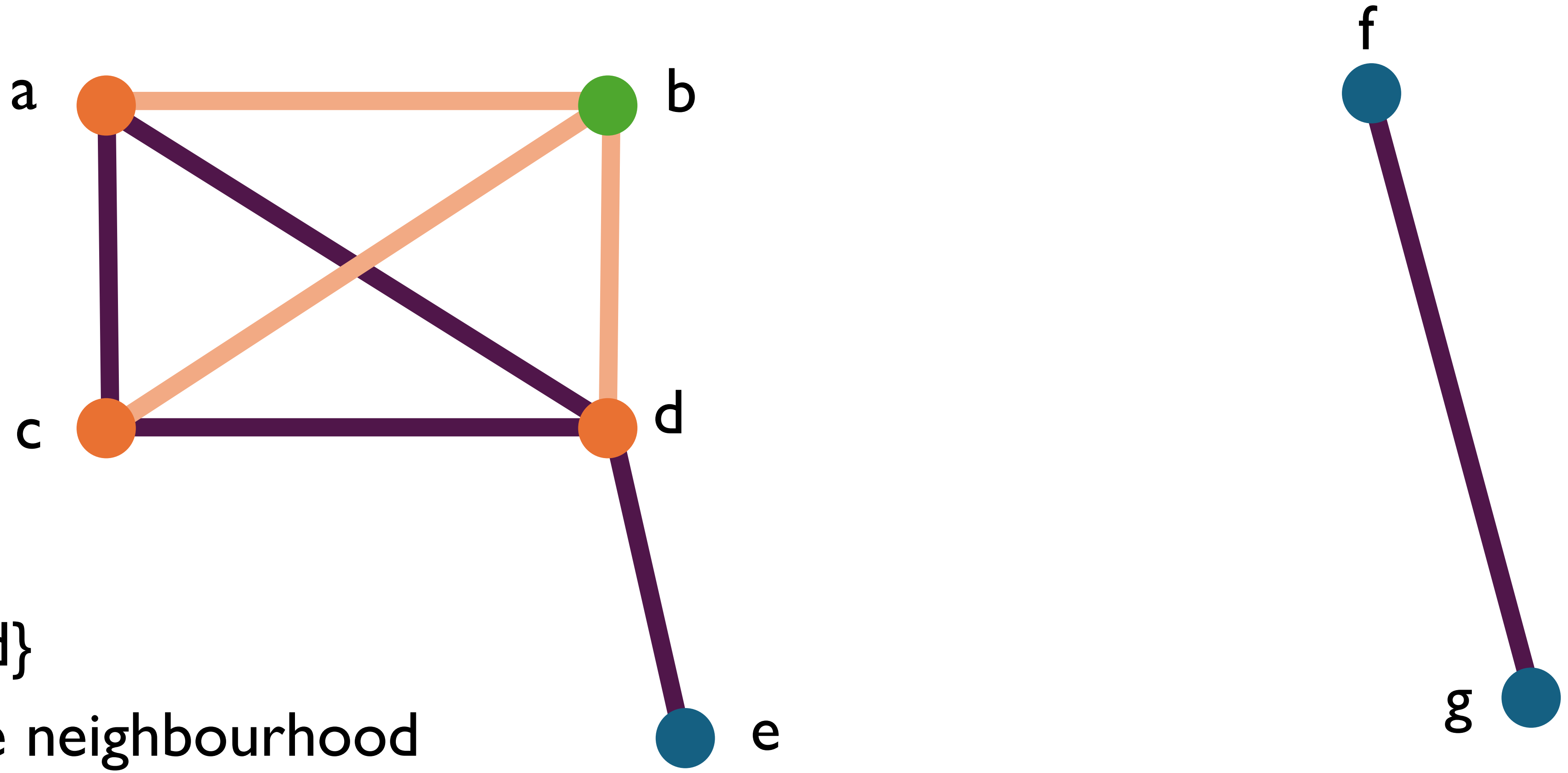


weights: assign a value to each edge

$$wt : E \to \mathbb{R}$$
$$wt(u) = k, k \in \mathbb{R}$$

common when distance is involved!

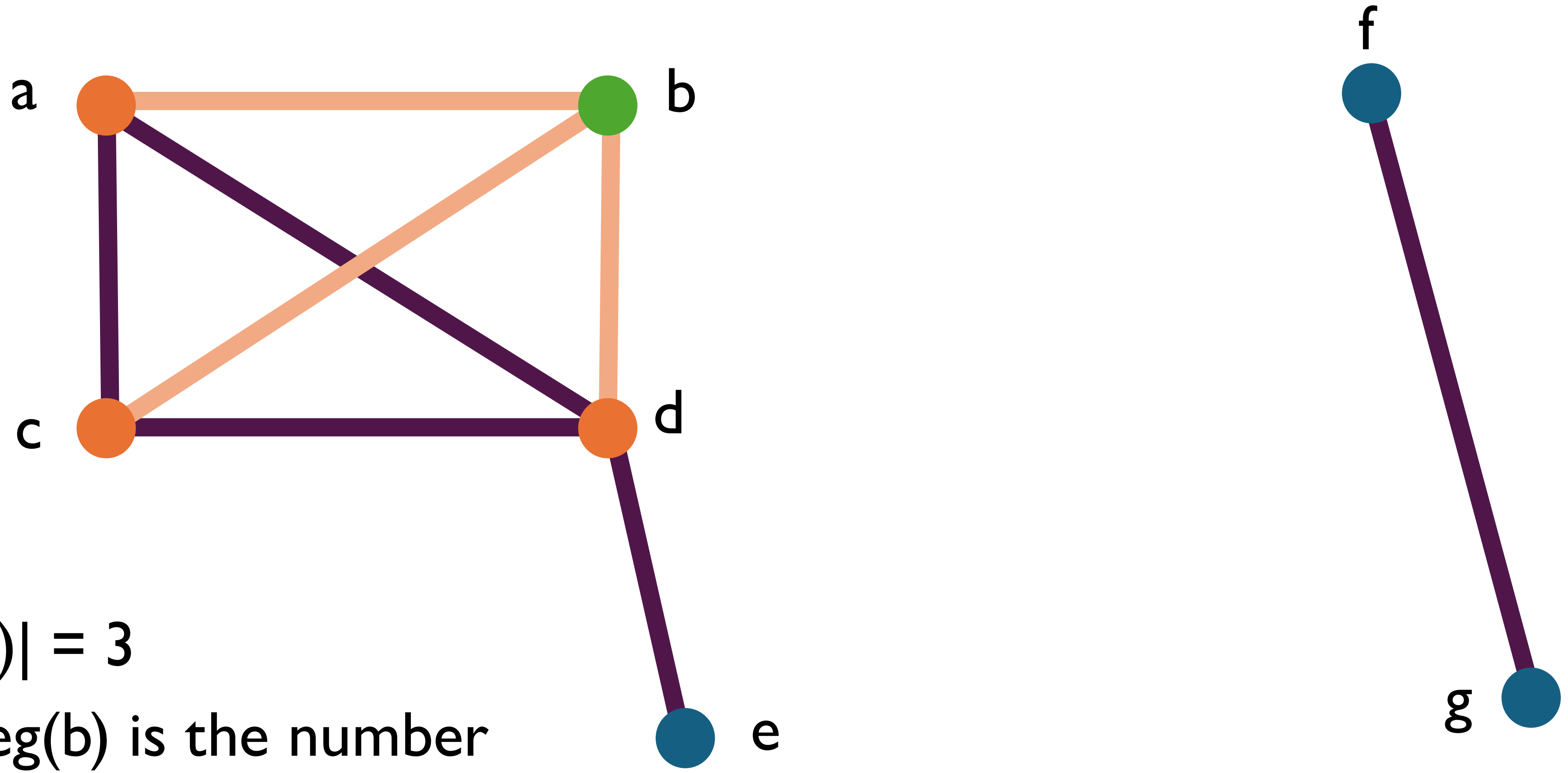
graph theory (a crash course)



nbr(b) = {a, c, d}

closed/inclusive neighbourhood includes b; open does not

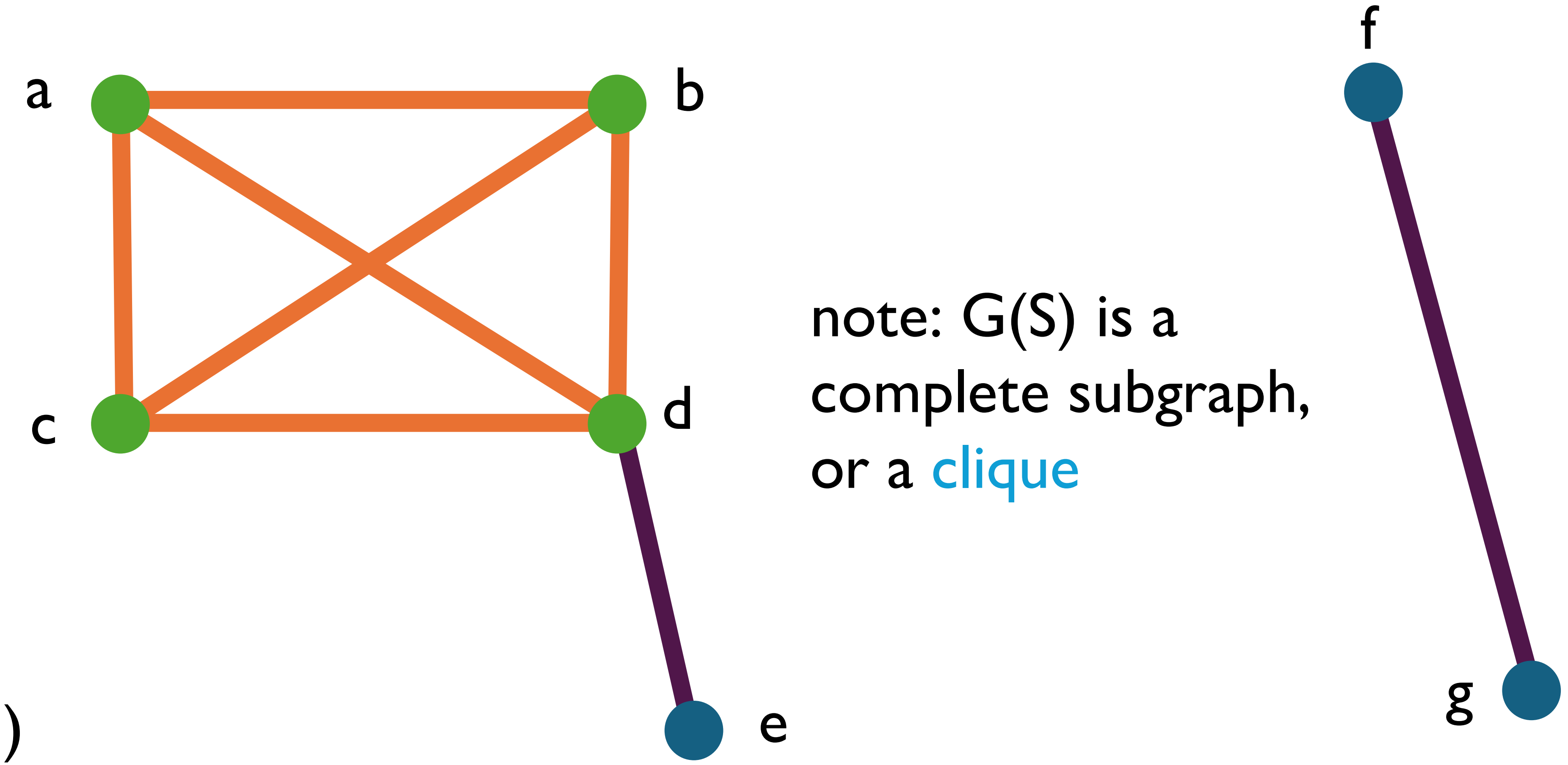
graph theory (a crash course)



deg(b) = |nbr(b)| = 3

alternatively, deg(b) is the number of edges connected to b

graph theory (a crash course)



note: G(S) is a complete subgraph, or a clique

$S = \{a, b, c, d\}$

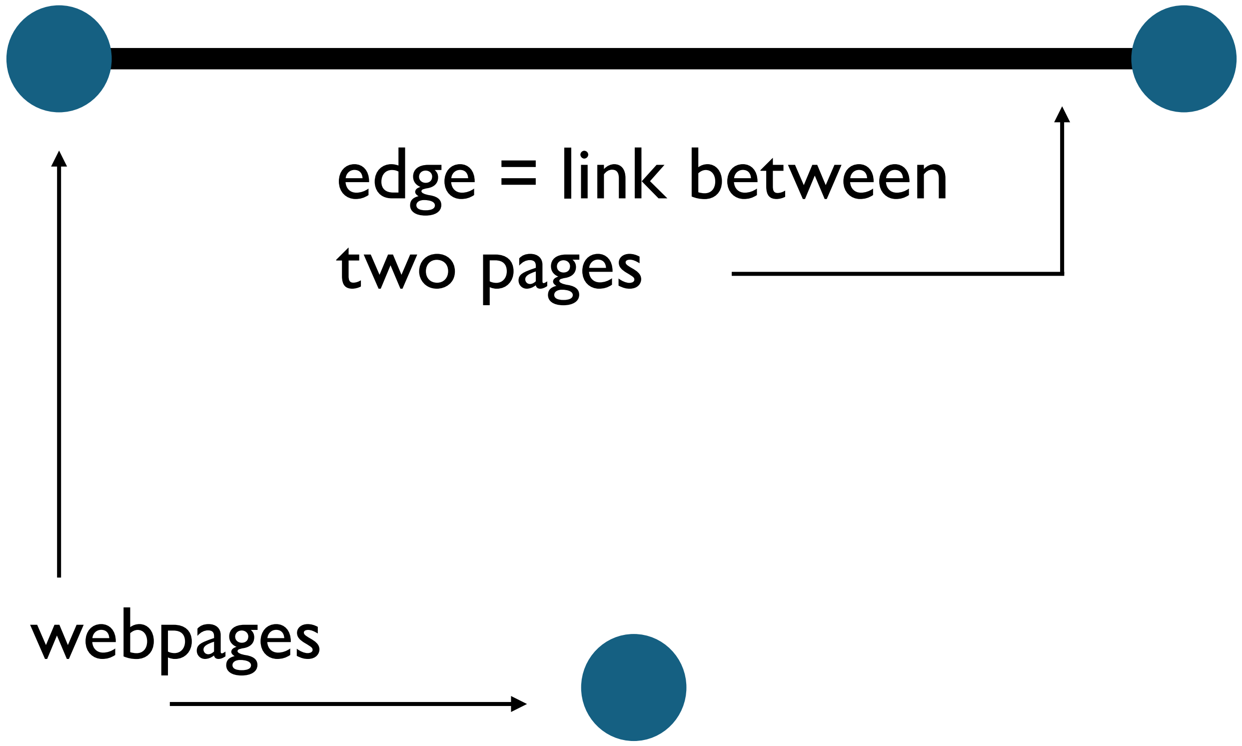
$G[S] = (S, E(S))$

This is the subgraph induced by S

okay, I know that was a lot…

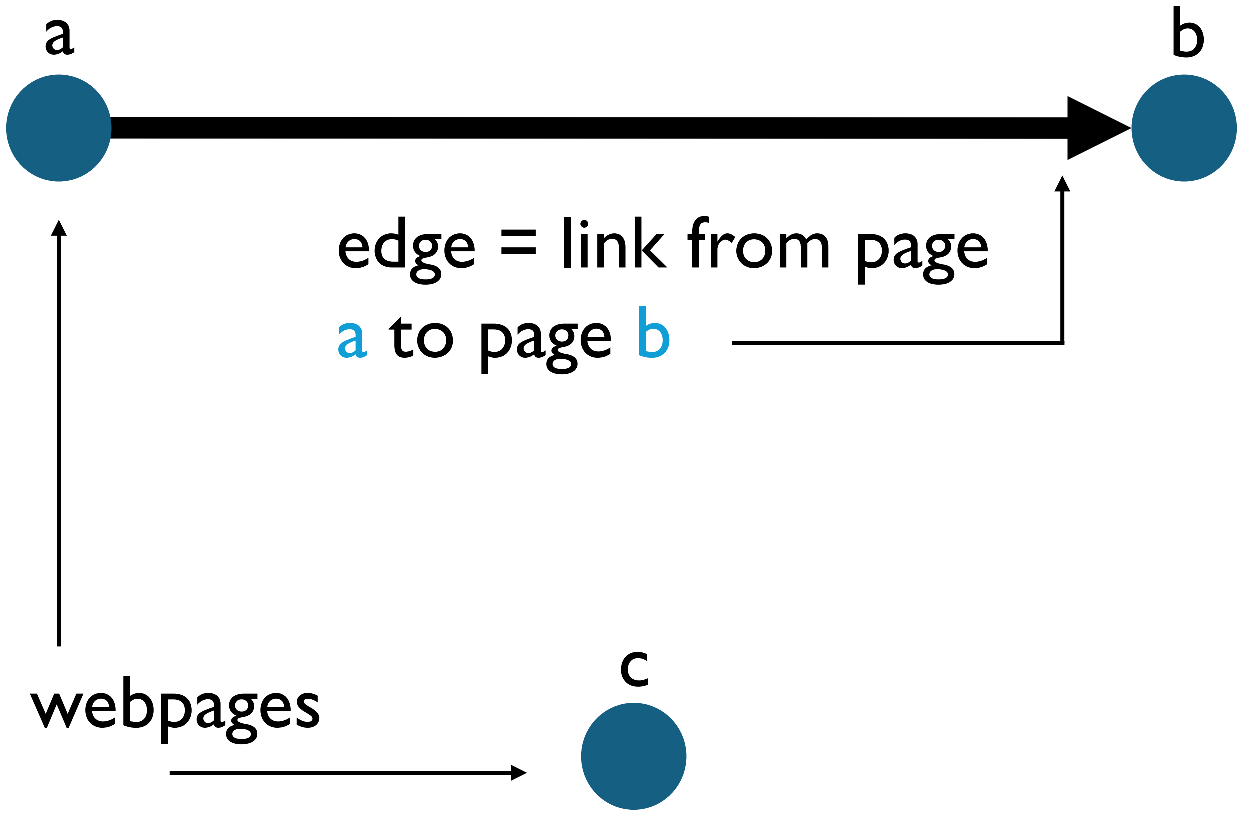
Let’s go back to the original problems.

real-world problems → graphs



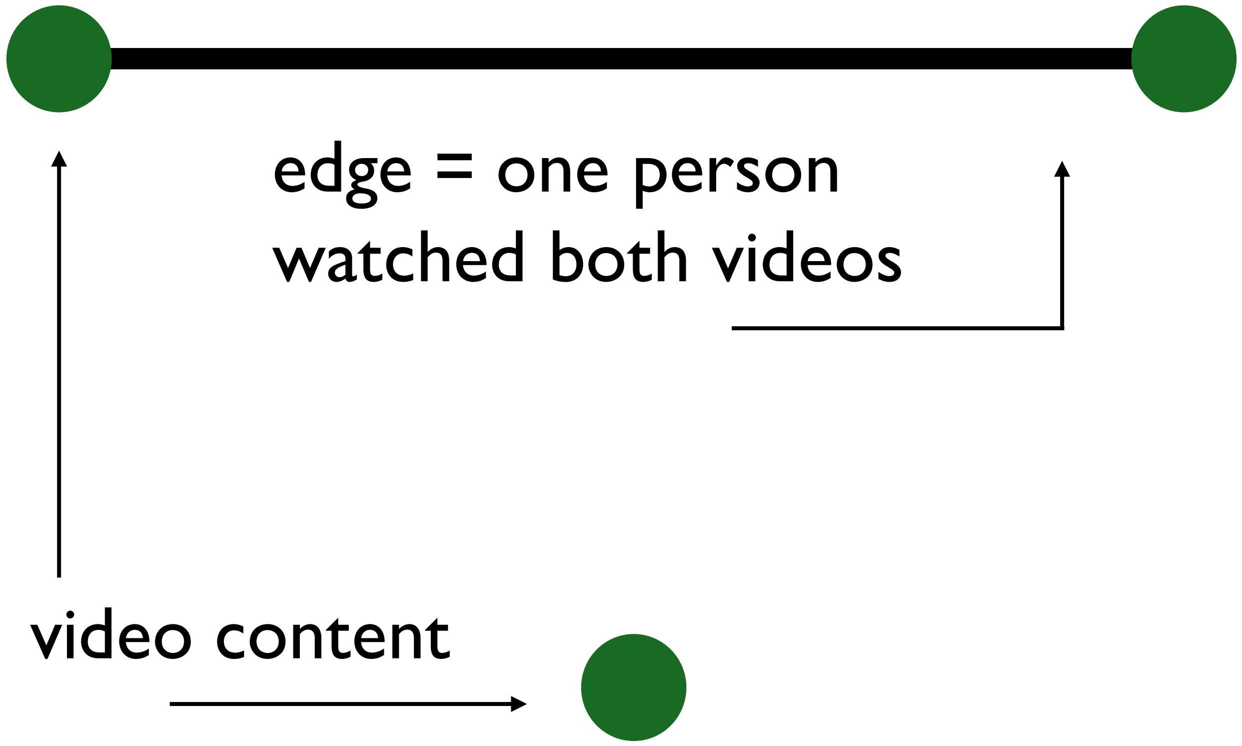
community detection

real-world problems → graphs



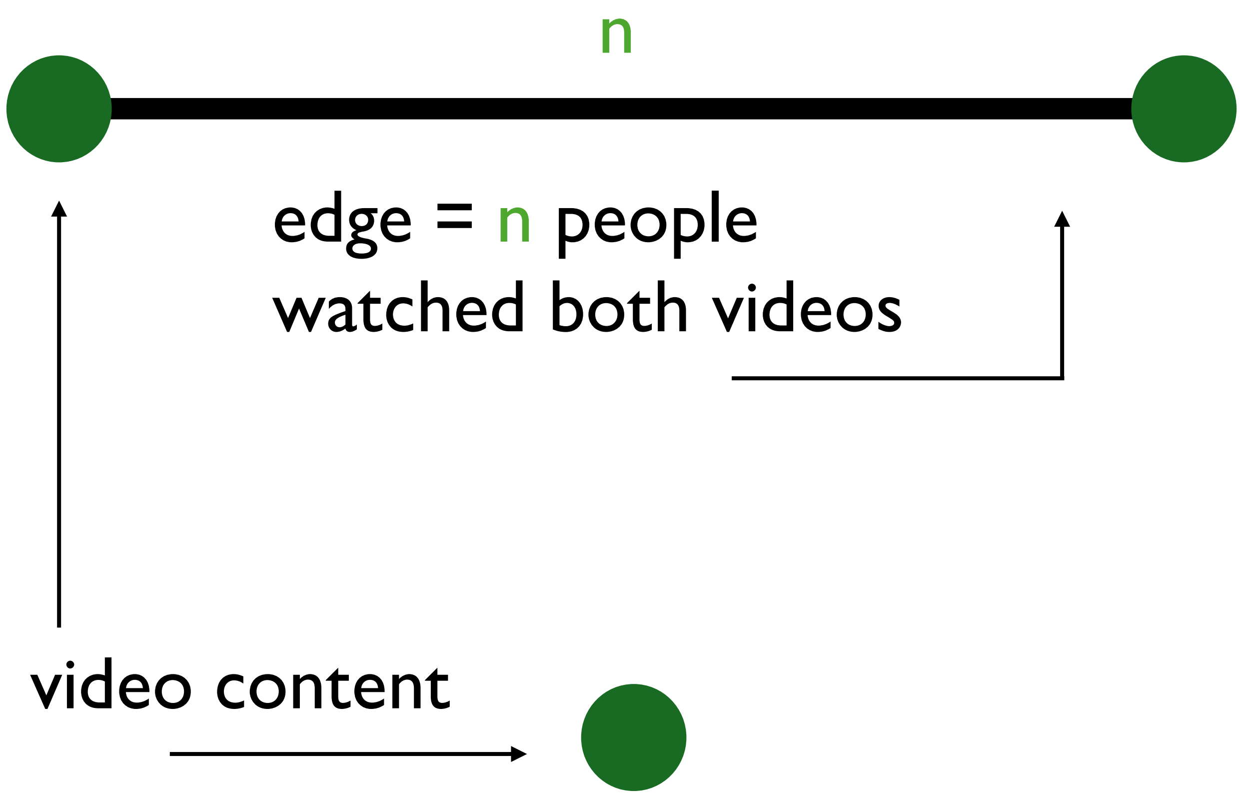
community detection

real-world problems → graphs



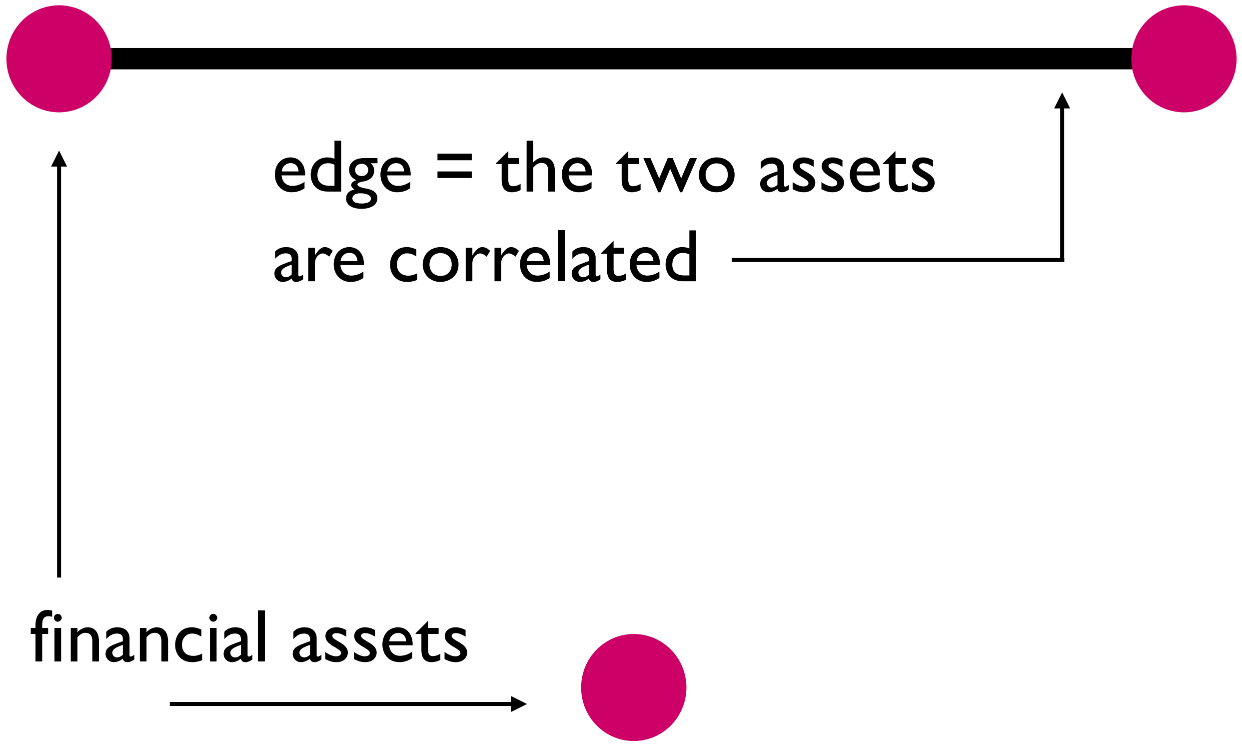
topic clustering

real-world problems → graphs



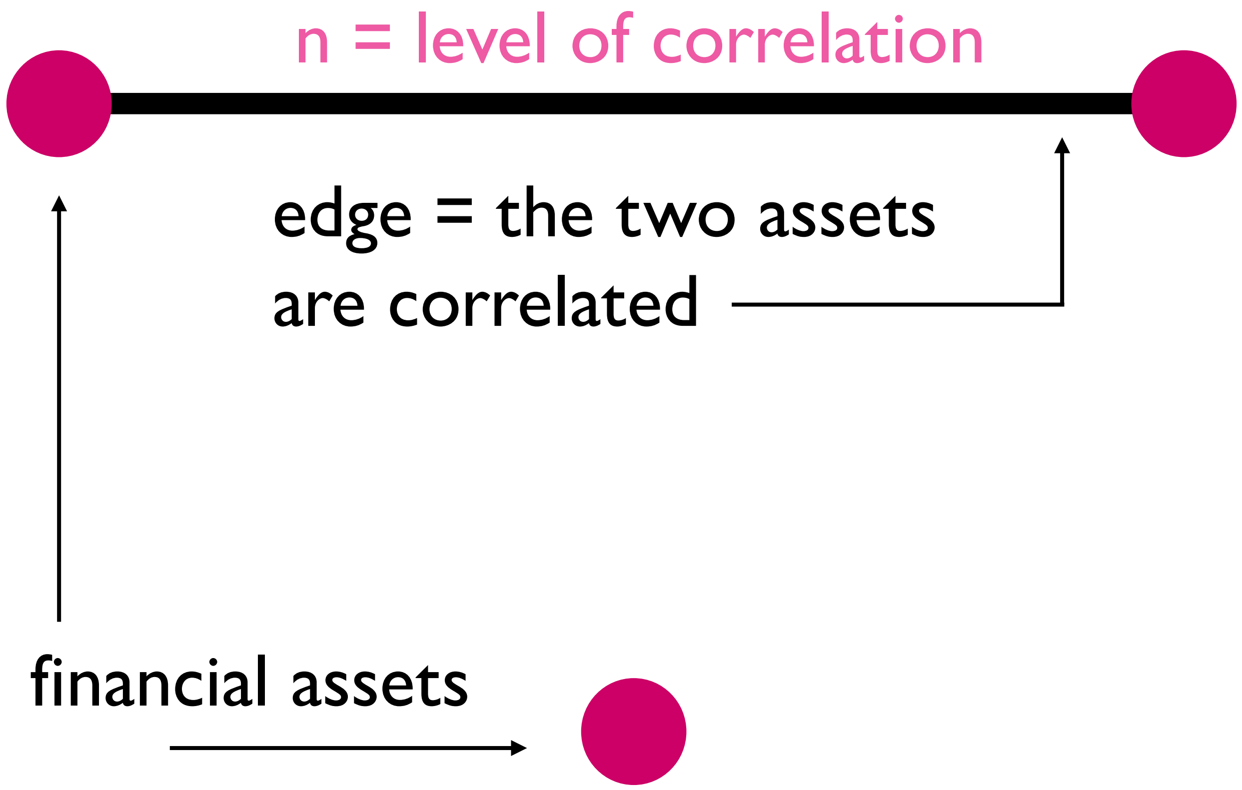
topic clustering

real-world problems → graphs



correlation mining

real-world problems → graphs

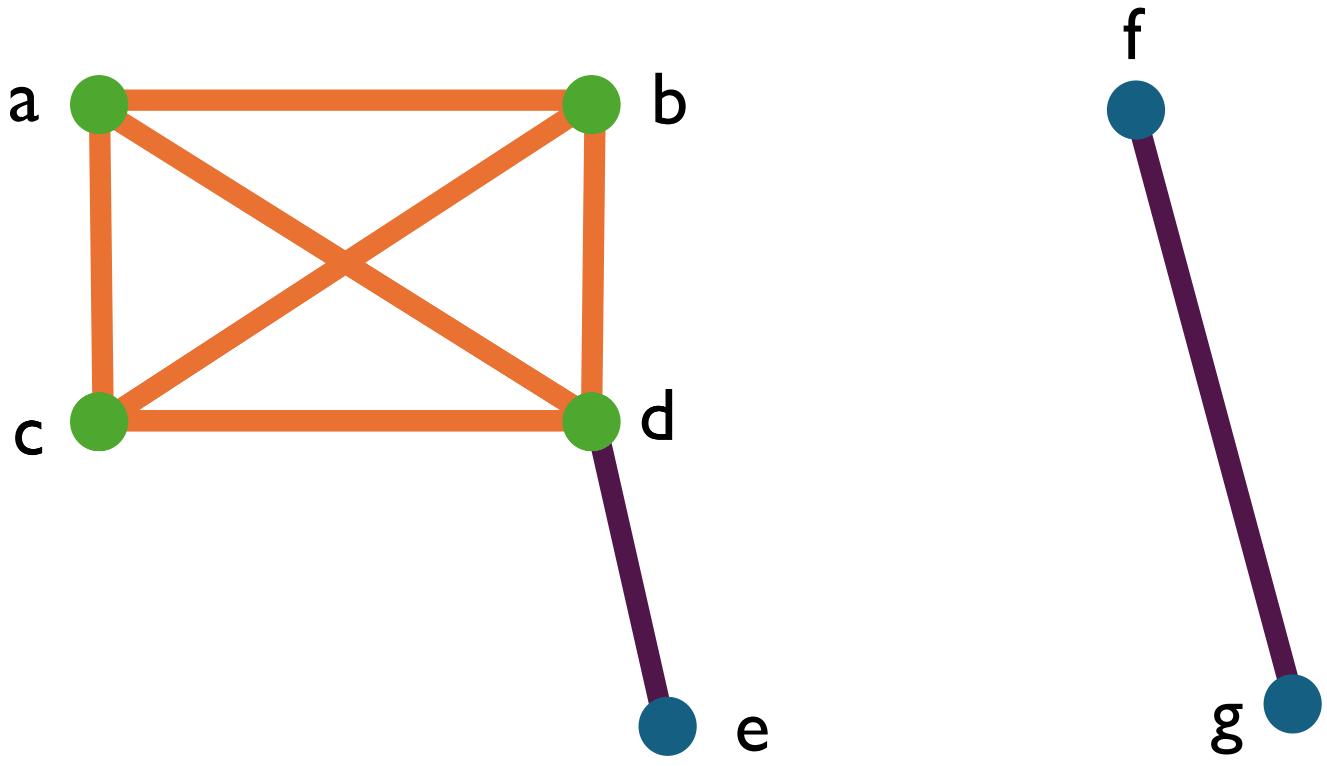


correlation mining

the densest subgraph problem

The density of a graph is the number of edges it has divided by the number of vertices.

We want to find the subgraph with the highest density.



Here, the densest subgraph is induced by the set S = {a, b, c, d}.

The DSP, formally.

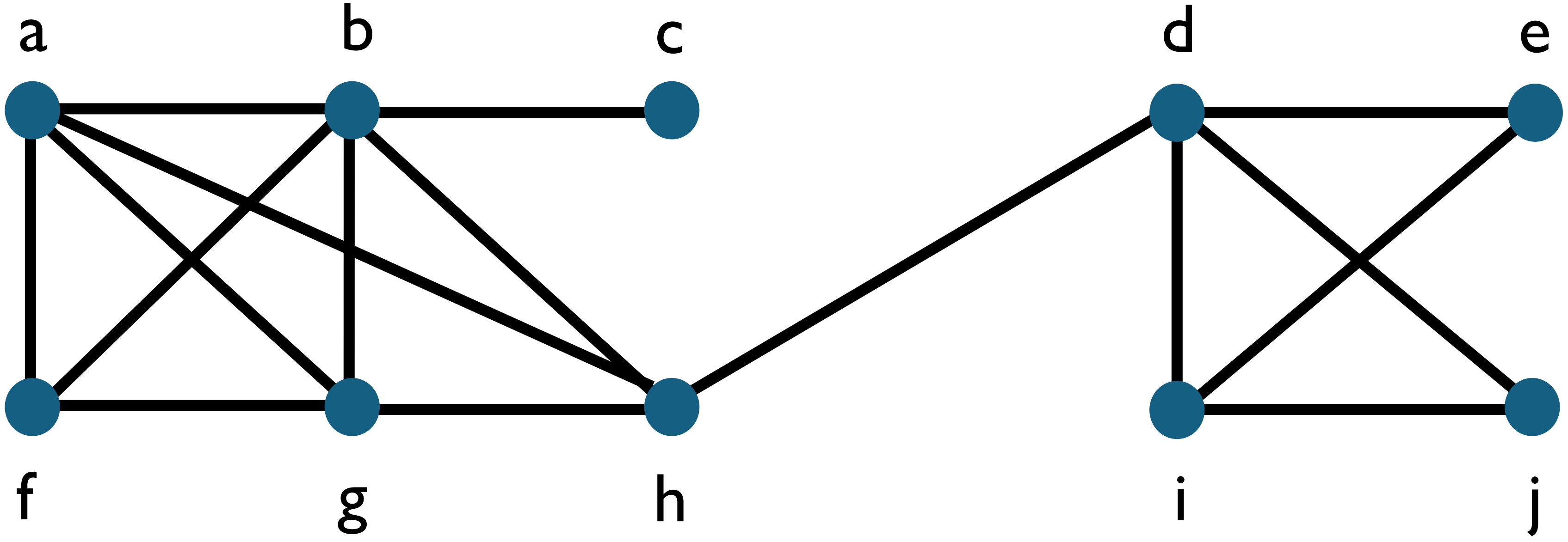
Given a graph $G = (V, E)$, let $S \subseteq V$. Then $G[S] = (S, E(S))$ and

$$density(S) = \frac{|E(S)|}{|S|}$$

We want to find the subgraph with the optimal density, λ^*:

$$\lambda^* = \max_{S \subseteq V} \frac{|E(S)|}{|S|}$$

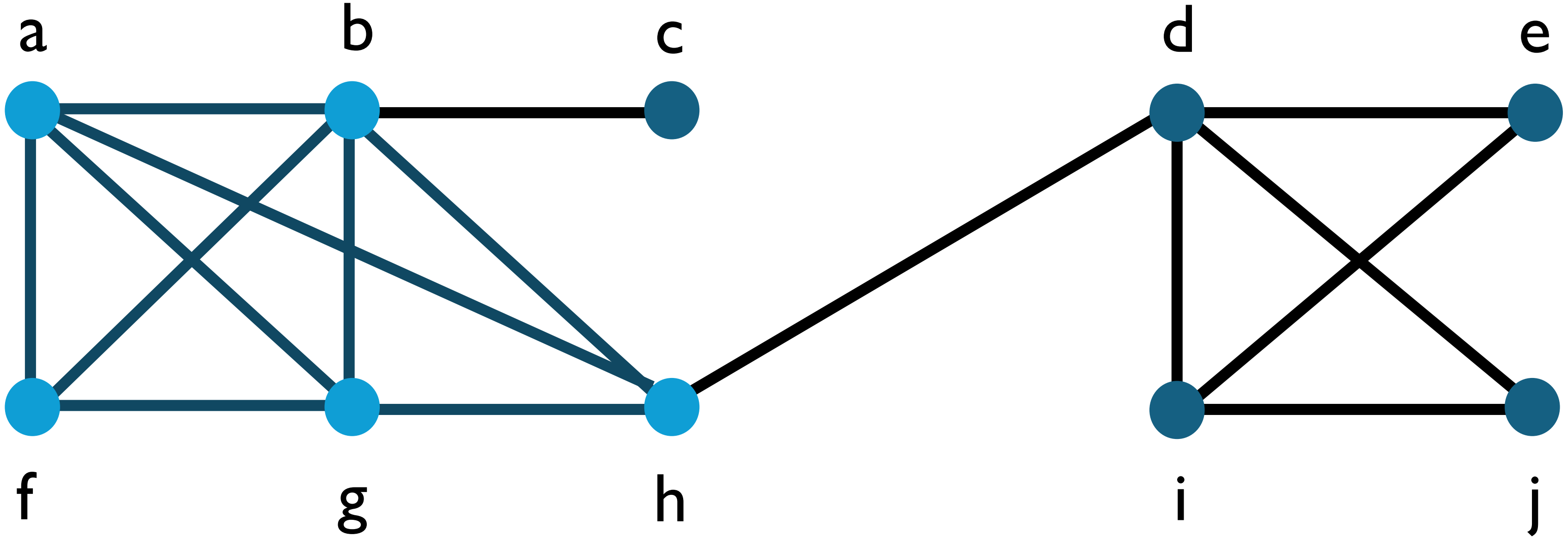
peeling: an algorithm



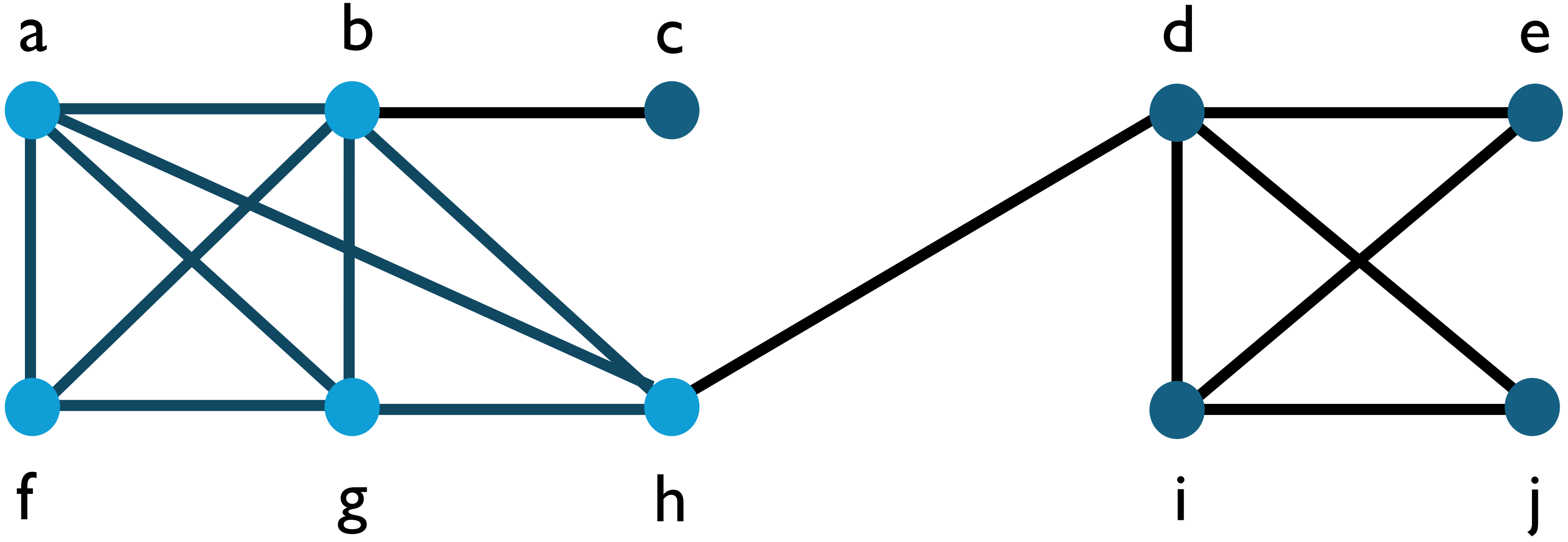
(Asahiro et. al, 1996; Charikar, 2000)

peeling: an algorithm

optimal density:
9/5 = 1.8



peeling: an algorithm



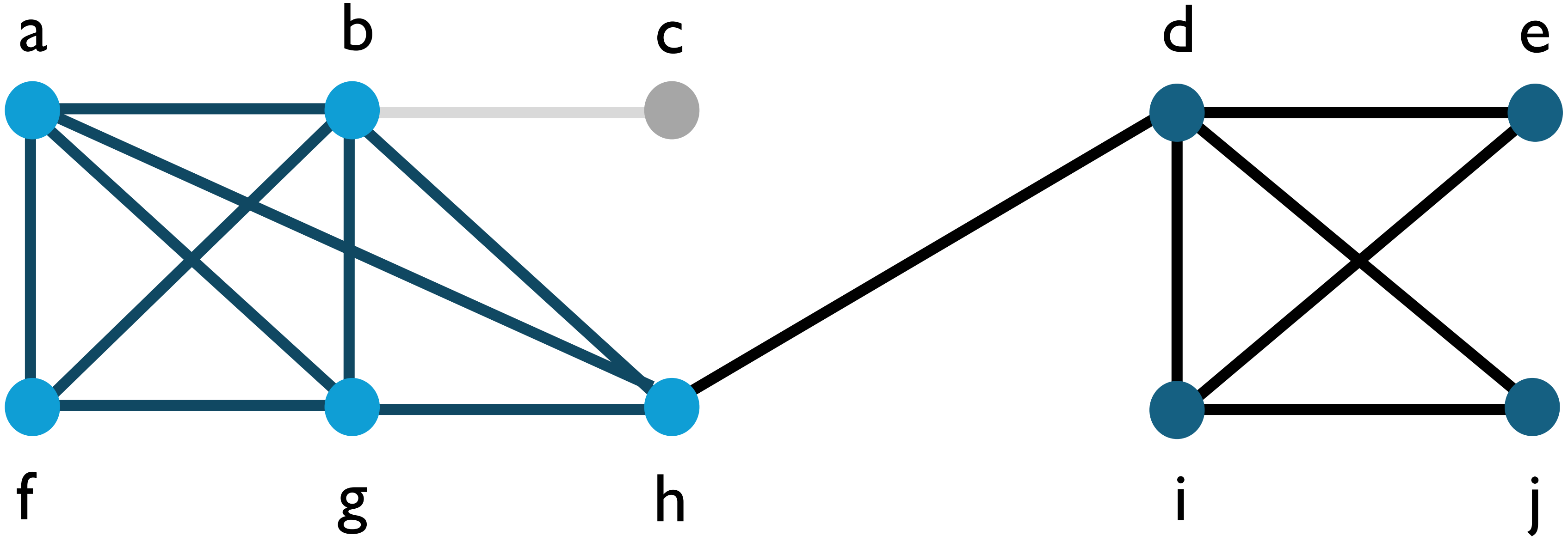
optimal density:
9/5 = 1.8

current density:
16/10 = 1.6

highest density:
16/10 = 1.6

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
4	5	1	4	2	3	4	3	3	2	curr. degree
										final degree

peeling: an algorithm



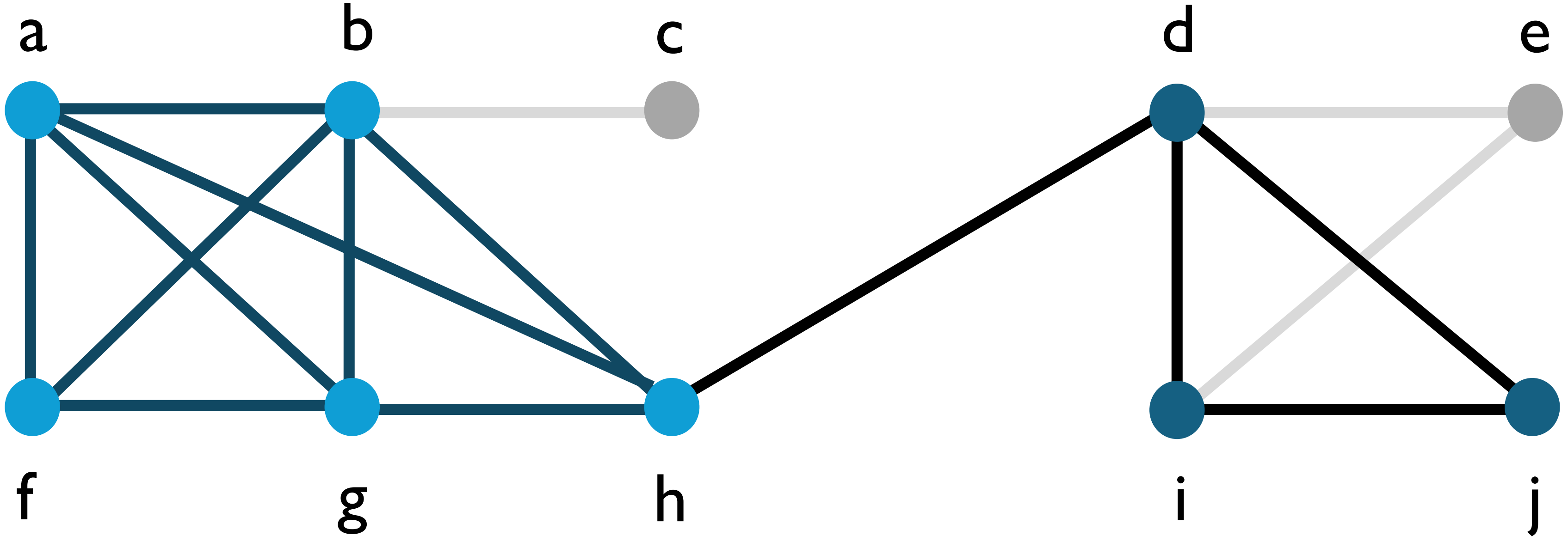
optimal density:
9/5 = 1.8

current density:
15/9 = 1.6666667

highest density:
15/9 = 1.6666667

	a	b	c	d	e	f	g	h	i	j
degree	4	5	1	4	2	3	4	4	3	2
curr. degree	4	4	X	4	2	3	4	3	3	2
final degree			1							

peeling: an algorithm



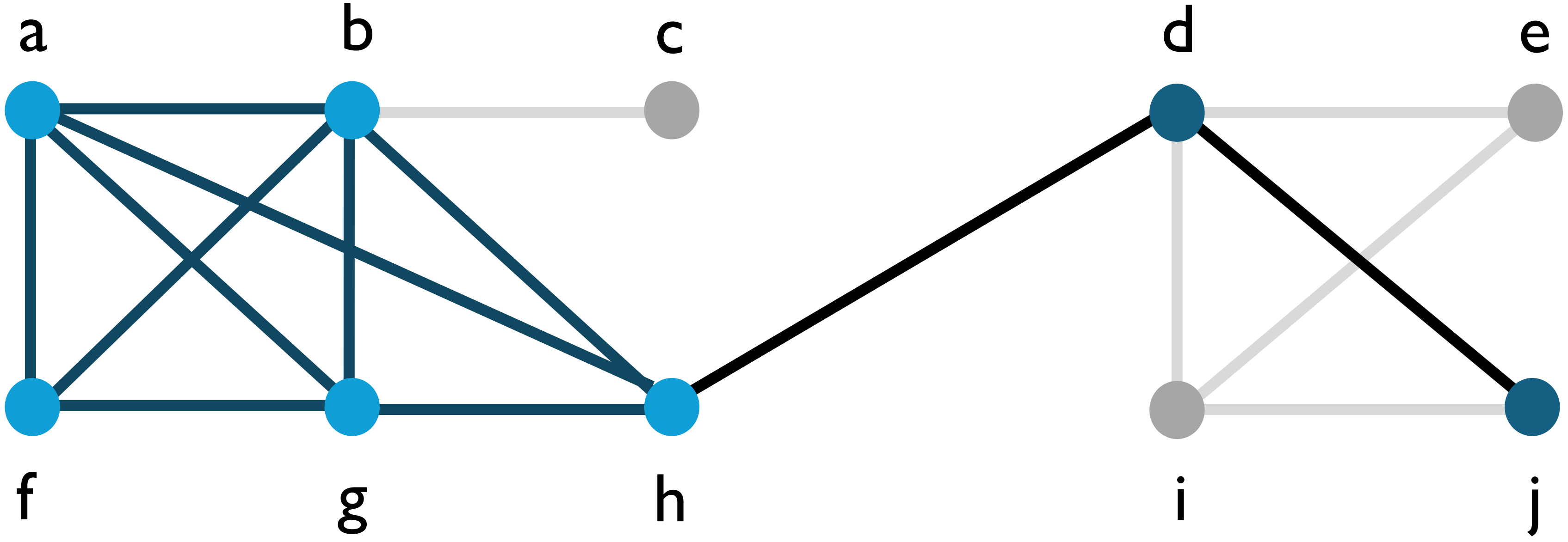
optimal density:
9/5 = 1.8

current density:
13/8 = 1.625

highest density:
15/9 = 1.6666667

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
4	4	X	3	X	3	4	3	2	2	curr. degree
		1		2						final degree

peeling: an algorithm



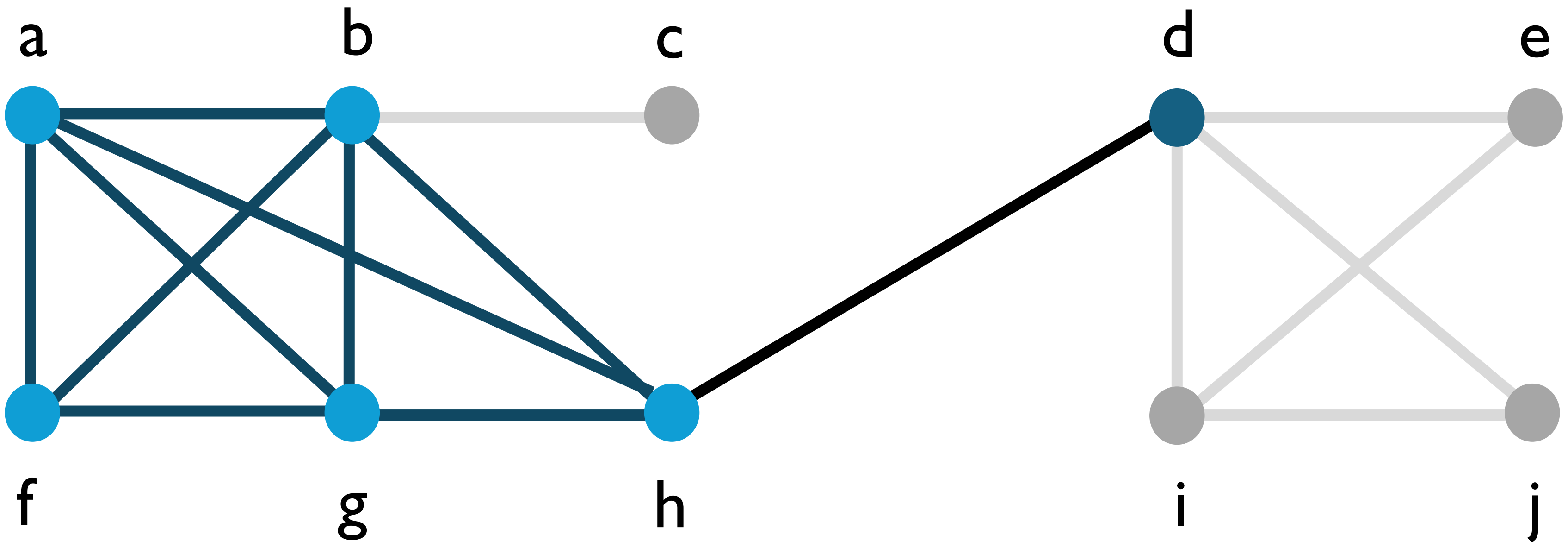
optimal density:
9/5 = 1.8

current density:
11/7 = 1.625

highest density:
15/9 = 1.6666667

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
4	4	X	2	X	3	4	3	X	1	curr. degree
		1		2				2		final degree

peeling: an algorithm



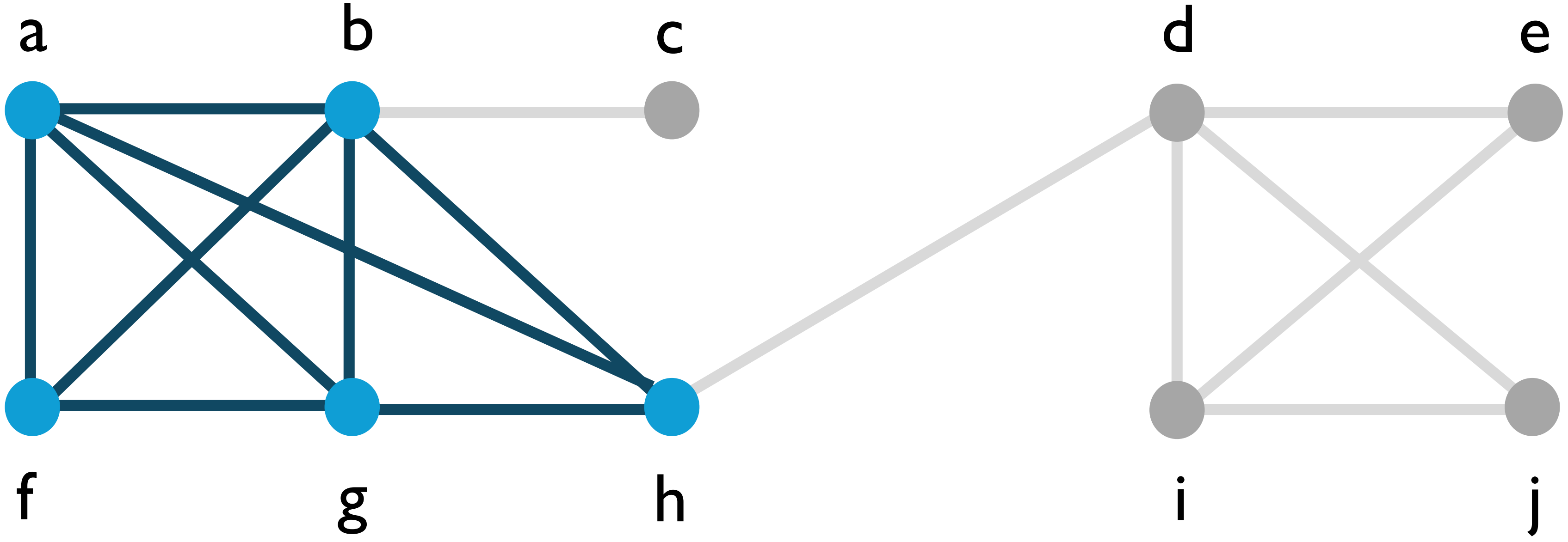
optimal density:
9/5 = 1.8

current density:
10/6 = 1.6666667

highest density:
15/9 = 1.6666667

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
4	4	X	1	X	3	4	3	X	X	curr. degree
		1		2				2	1	final degree

peeling: an algorithm



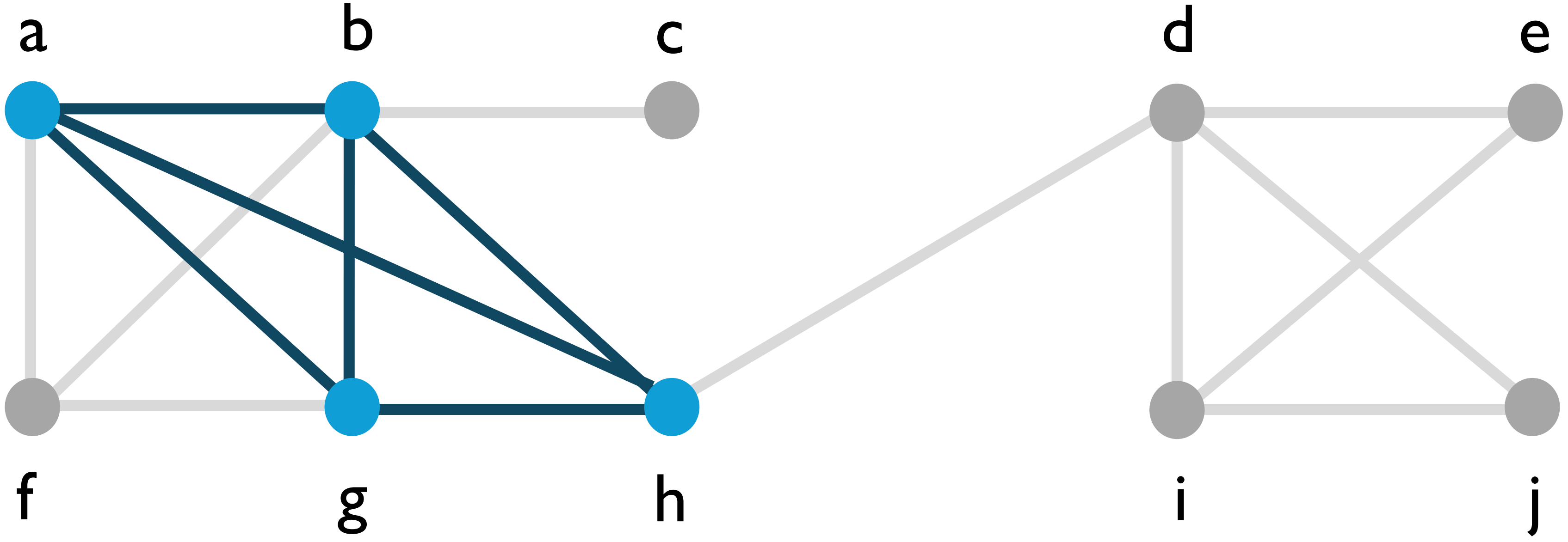
optimal density:
9/5 = 1.8

current density:
9/5 = 1.8

highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
4	4	X	X	X	3	4	3	X	X	curr. degree
		1	1	2				2	1	final degree

peeling: an algorithm



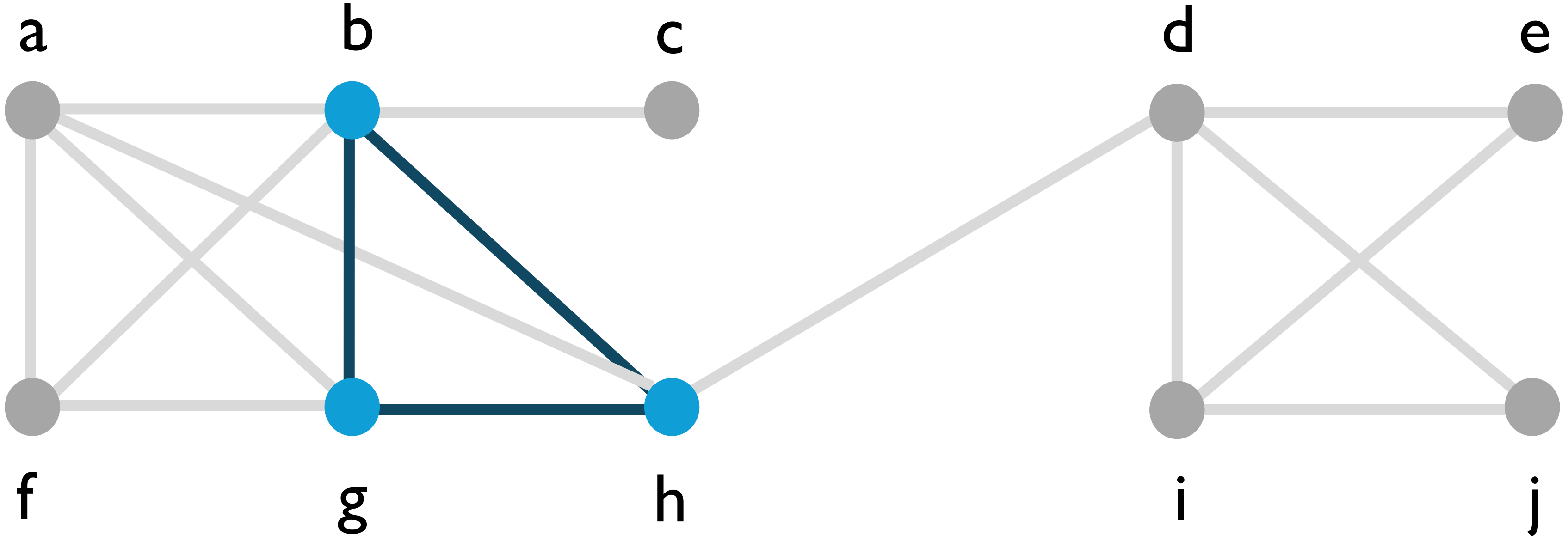
optimal density:
9/5 = 1.8

current density:
6/4 = 1.5

highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
3	3	X	X	X	X	3	3	X	X	curr. degree
		1	1	2	3			2	1	final degree

peeling: an algorithm



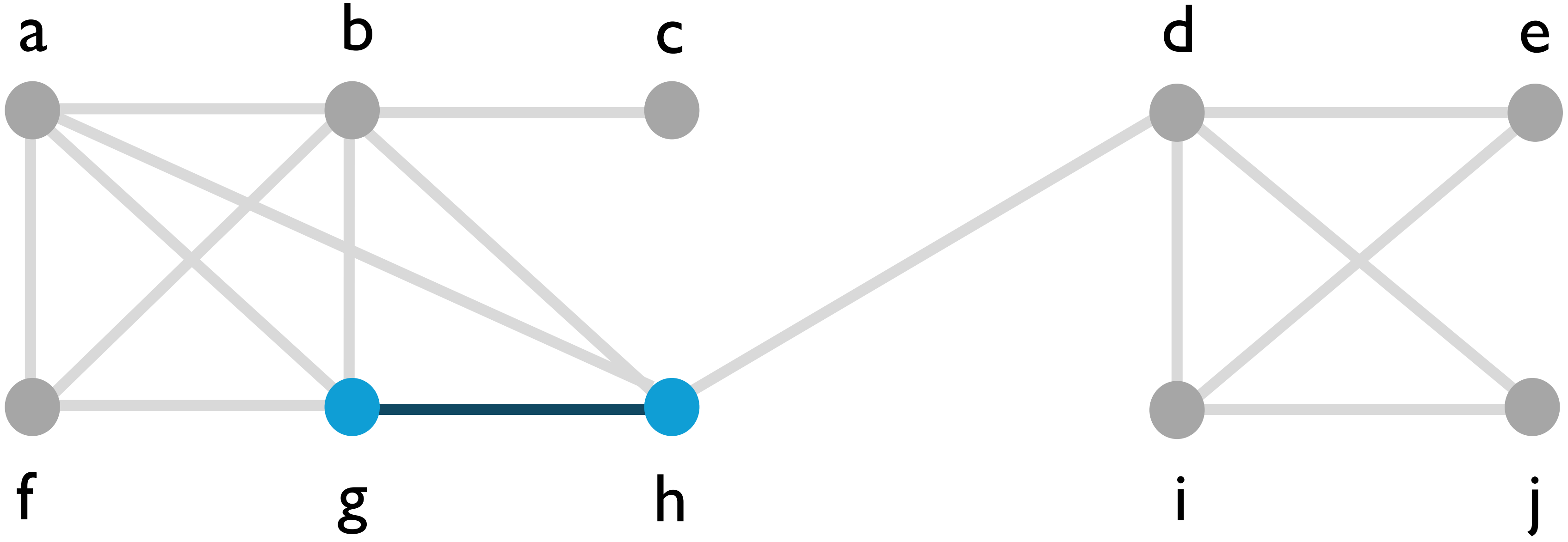
optimal density:
9/5 = 1.8

current density:
3/3 = 1

highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	2	X	X	X	X	2	2	X	X	curr. degree
3		1	1	2	3			2	1	final degree

peeling: an algorithm



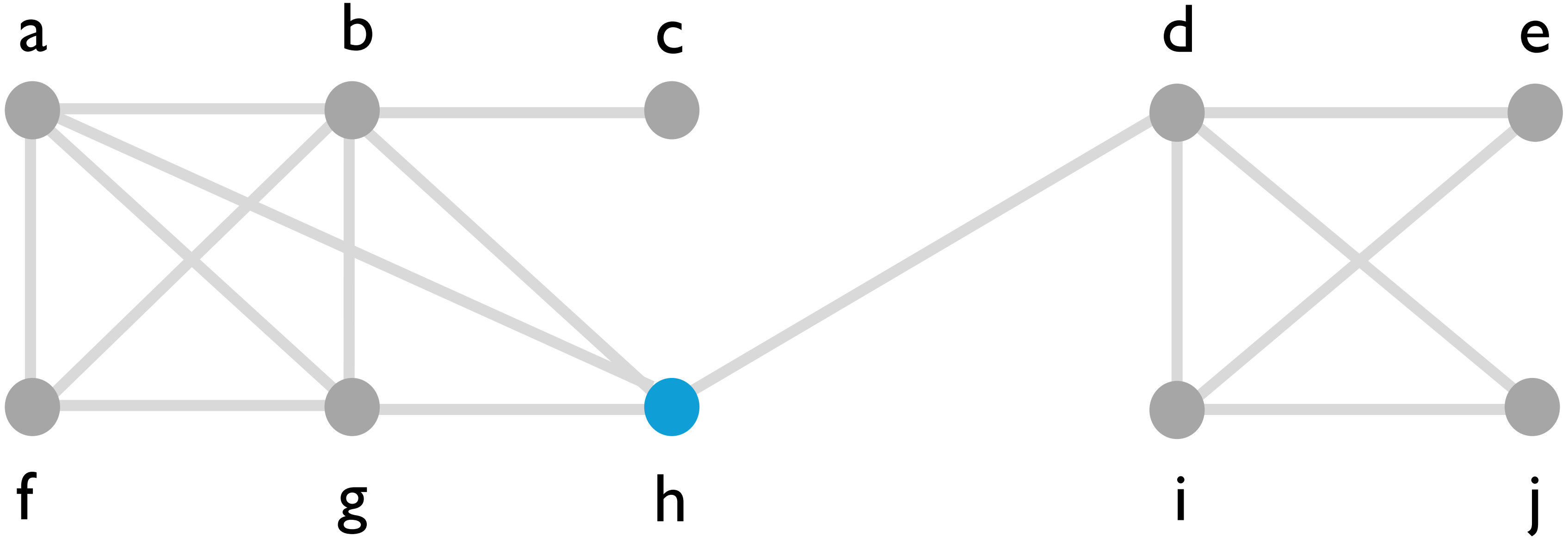
optimal density:
9/5 = 1.8

current density:
1/2 = 0.5

highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	X	X	X	X	X	1	1	X	X	curr. degree
3	2	1	1	2	3			2	1	final degree

peeling: an algorithm



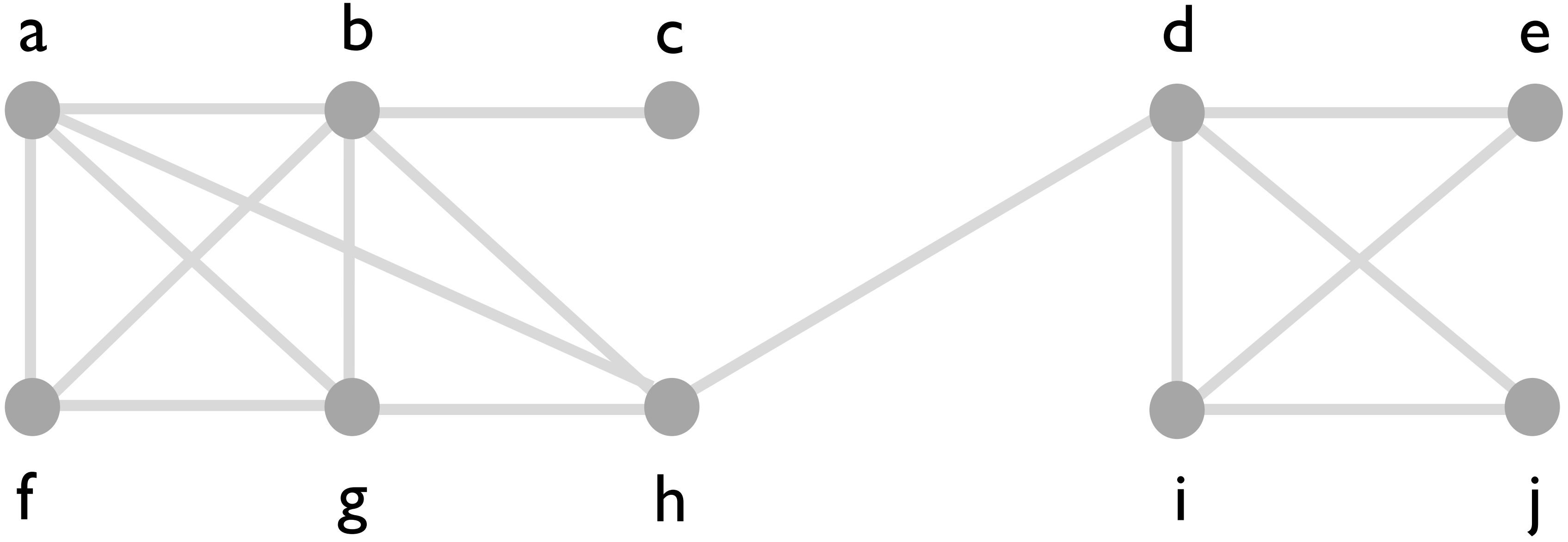
optimal density:
9/5 = 1.8

current density:
0/1 = 0

highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	X	X	X	X	X	X	0	X	X	curr. degree
3	2	1	1	2	3	1		2	1	final degree

peeling: an algorithm



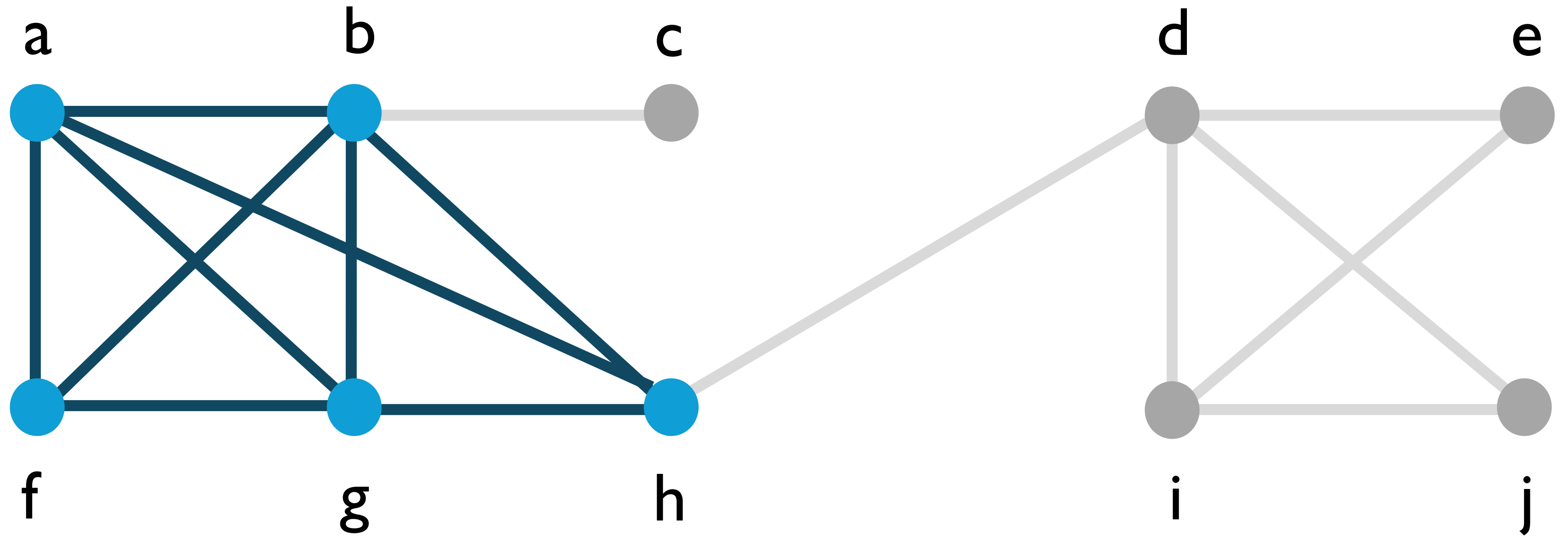
optimal density:
9/5 = 1.8

current density:
0/0 = undefined

highest density:
9/5 = 1.8

a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	X	X	X	X	X	X	1	X	X	curr. degree
3	2	1	1	2	3	1	0	2	1	final degree

peeling: an algorithm



In this case, the algorithm did produce the expected densest subgraph. However, this is not always true…

peeling: summary

- very fast! (runs in linear time)
- *usually* about 80% good on real world graphs
- used in practice (real applications)
- only guaranteed to be 50% good in worst case (and there are known bad cases)

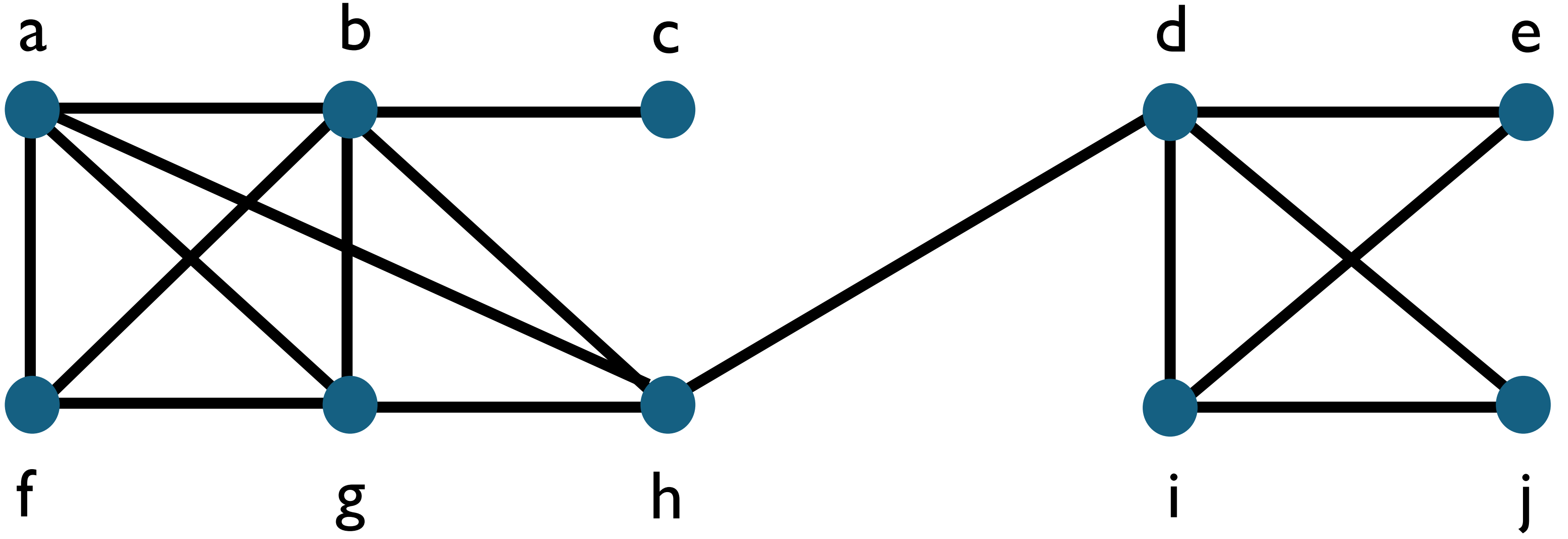
peeling: summary

- very fast! (runs in linear time)
- *usually* about 80% good on real world graphs
- used in practice (real applications)
- only guaranteed to be 50% good in worst case (and there are known bad cases)

what if we could do better?

iterative peeling

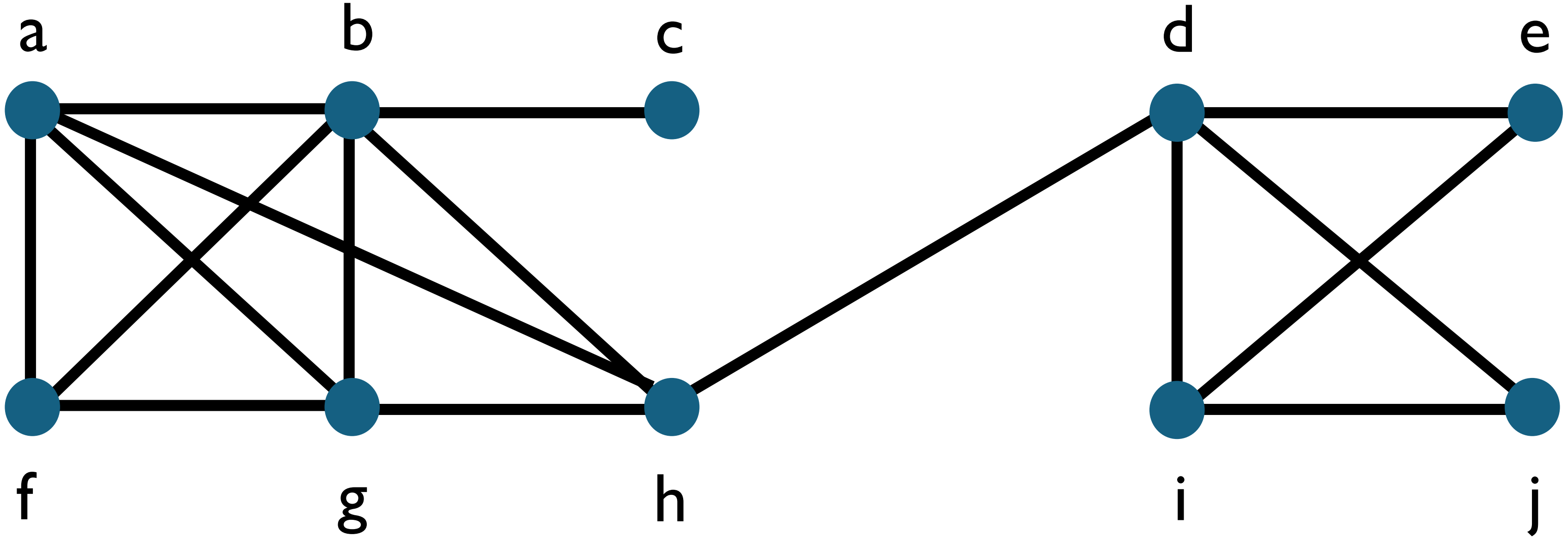
(Boob et. al, 2019)



a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	X	X	X	X	X	X	1	X	X	curr. degree
3	2	1	1	2	3	1	0	2	1	final degree

iterative peeling

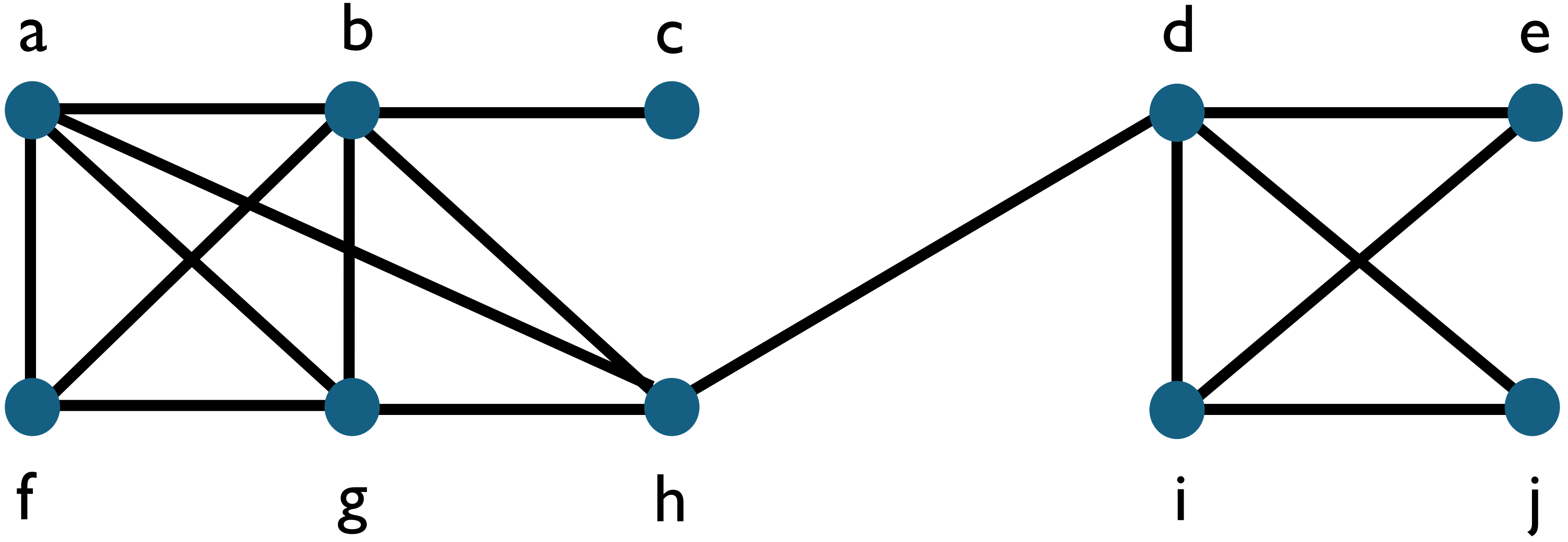
(Boob et. al, 2019)



a	b	c	d	e	f	g	h	i	j	
4	5	1	4	2	3	4	4	3	2	degree
X	X	X	X	X	X	X	1	X	X	curr. degree
3	2	1	1	2	3	1	0	2	1	loads

iterative peeling

(Boob et. al, 2019)

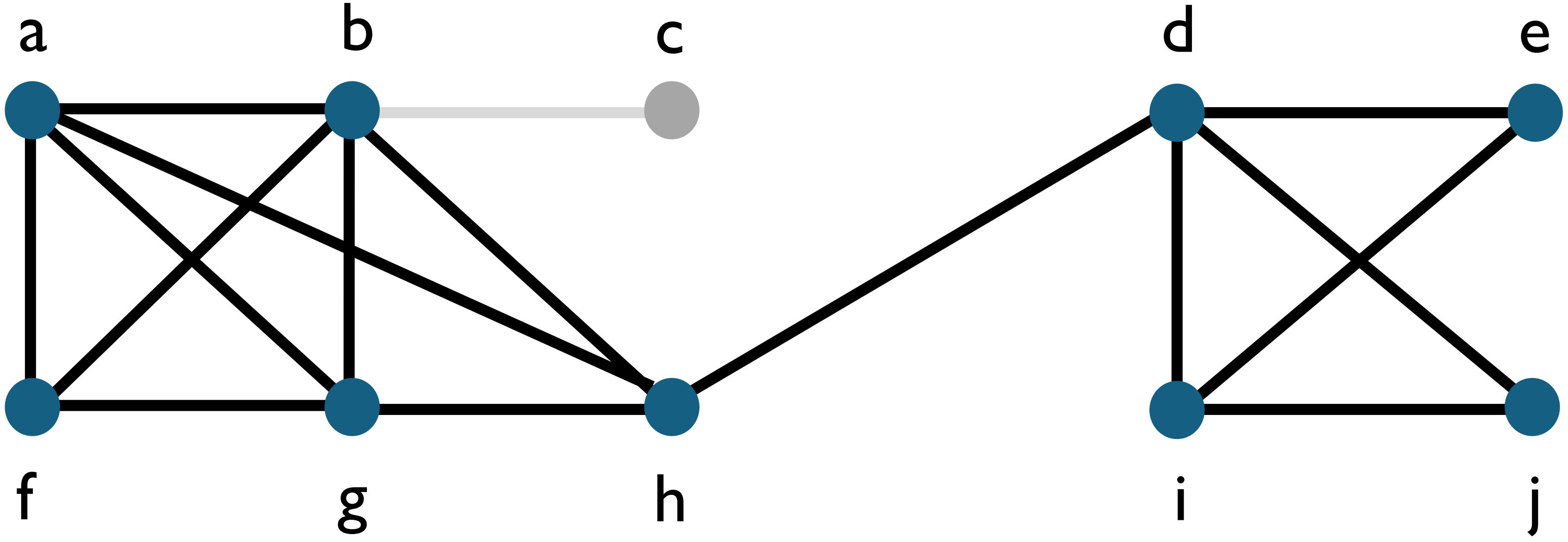


We obtain a new peeling order that changes with further iterations and eventually stabilizes.

a	b	c	d	e	f	g	h	i	j	
3	2	1	1	2	3	1	0	2	1	load
4	5	1	4	2	3	4	4	3	2	degree
7	7	2	5	4	6	5	4	5	3	curr load + deg
										load update

iterative peeling

(Boob et. al, 2019)



We obtain a new peeling order that changes with further iterations and eventually stabilizes.

a	b	c	d	e	f	g	h	i	j	
3	2	1	1	2	3	1	0	2	1	load
4	5	X	4	2	3	4	4	3	2	degree
7	7	1	5	4	6	5	4	5	3	curr load + deg
		1								load update

iterative peeling: summary

- Boob et. al experimentally showed that this seems to always eventually work!
- can we make our solution as good as we want?
- if so, at what cost?
- can we use this to solve other problems?

Part 4:

(sub, super)modularity and set functions

(a digression)

set functions

- the powerset of a set S is the set of all subsets of S
- typically denoted $\mathcal{P}(S)$; we will use 2^S
- If $S = \{a, b, c\}$, then $2^S = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$

set functions

A set function assigns values to subsets of a set. We call the overall set we are working with the ground set.

In other words, a set function is a function from the powerset of S to the real numbers.

$$f : 2^S \to \mathbb{R} \cup \{\pm\infty\}$$

set functions (subtypes)

Let V be a ground set, and let $f : 2^V \to \mathbb{R}$.

f is:

- normalized if $f(\emptyset) = 0$
- monotone increasing if
  $A \subset B \implies f(A) < f(B)$

set functions (subtypes)

Let V be a ground set, and let $f : 2^V \to \mathbb{R}$.

f is:

- normalized if $f(\emptyset) = 0$
- monotone decreasing if
  $A \subset B \implies f(A) > f(B)$

set functions (subtypes)

Let V be a ground set, and let $f : 2^V \to \mathbb{R}$.

f is:

- additive if $f(A \dot\cup B) = f(A) + f(B)$
- subadditive if
  $f(A \dot\cup B) \leq f(A) + f(B)$

set functions (subtypes)

Let V be a ground set, and let $f : 2^V \to \mathbb{R}$.

f is:

- additive if $f(A \dot{\cup} B) = f(A) + f(B)$
- superadditive if $f(A \dot{\cup} B) \geq f(A) + f(B)$

set functions (marginal values)

Let V be a ground set, and let $f : 2^V \to \mathbb{R}$.

The marginal value of adding a new element to a set is the gain or loss incurred by adding that element to the set. Formally,

$$f(v|S) = f(S \cup \{v\}) - f(S), \;\; S \subsetneq V, v \notin S$$

interlude:

some basic economics

submodularity

Submodularity is characterized by diminishing returns. Abstractly:

10 dollars is worth more to a poor person than to a millionaire.

modularity

Modularity is characterized by constant returns. Abstractly:

Adding 10 dollars to your bank account always increases your purchasing power by the same amount.

supermodularity

Supermodularity is characterized by increasing returns. Abstractly:

Adding one brick to a stack of two bricks is less useful than adding one brick to a stack of one million bricks.

submodularity, formally

A submodular function is a real-valued set function characterized by diminishing returns:

$$f(A+v) - f(A) \geq f(B+v) - f(B)$$

$$A \subsetneq B, \; v \notin B.$$

supermodularity, formally

A supermodular function is a real-valued set function characterized by increasing returns:

$$f(A+v) - f(A) \leq f(B+v) - f(B)$$

$$A \subsetneq B,\ v \notin B.$$

modularity, formally

A modular function is both submodular and supermodular.

$$f(A+v) - f(A) = f(B+v) - f(B)$$

$$A \subsetneq B, \;\; v \notin B.$$

Notice that modular functions are additive!

rewritten using marginal values…

submodular:

$$f(v|A) \geq f(v|B),\ A \subsetneq B,\ v \notin B.$$

supermodular:

$$f(v|A) \leq f(v|B),\ A \subsetneq B,\ v \notin B.$$

modular:

$$f(v|A) = f(v|B),\ A \subsetneq B,\ v \notin B.$$

Part 5:

The Densest Supermodular Set Problem (DSSP)

(a generalization)

The DSSP, formally.

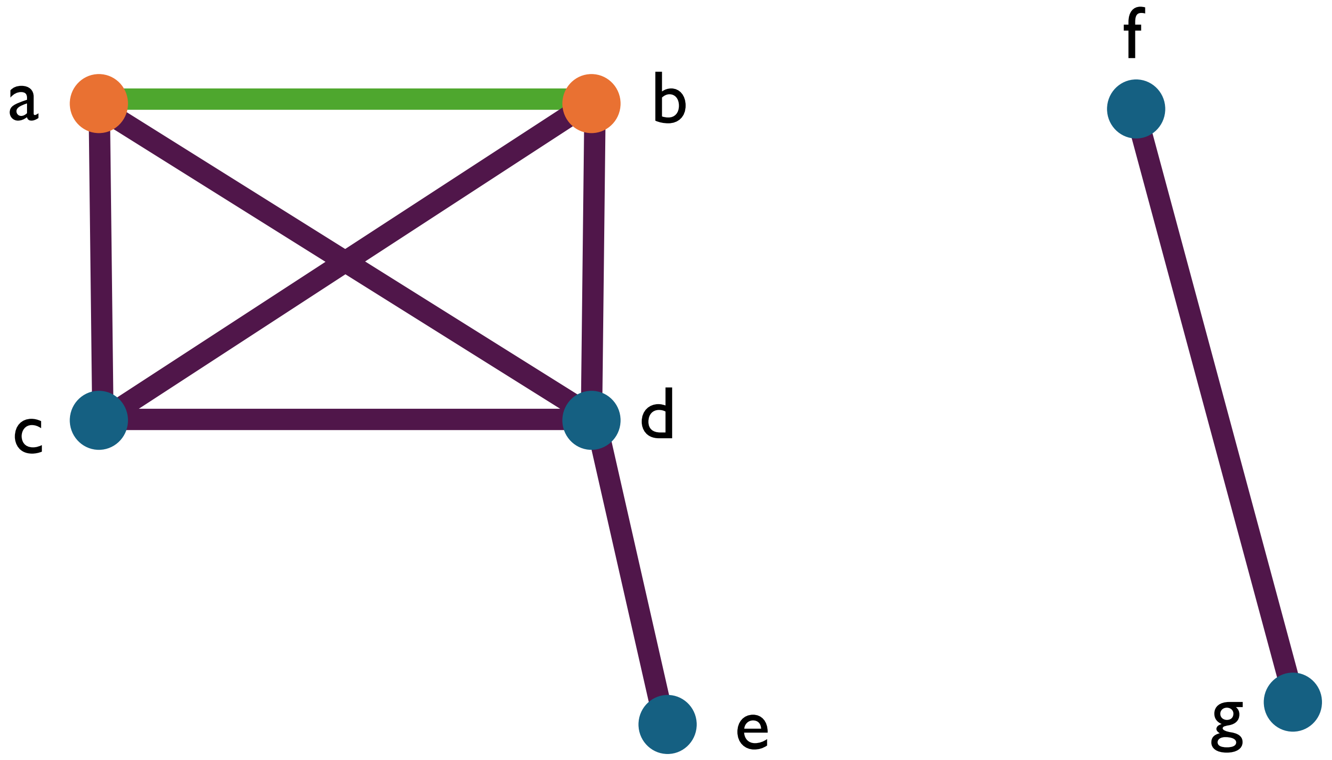
Given a non-negative supermodular function $f : 2^V \to \mathbb{R}^{\geq 0}$, let $S \subseteq V$. Then,

$$density(S) = \frac{f(S)}{|S|}$$

and we want to find the subset with the optimal density, λ^*:

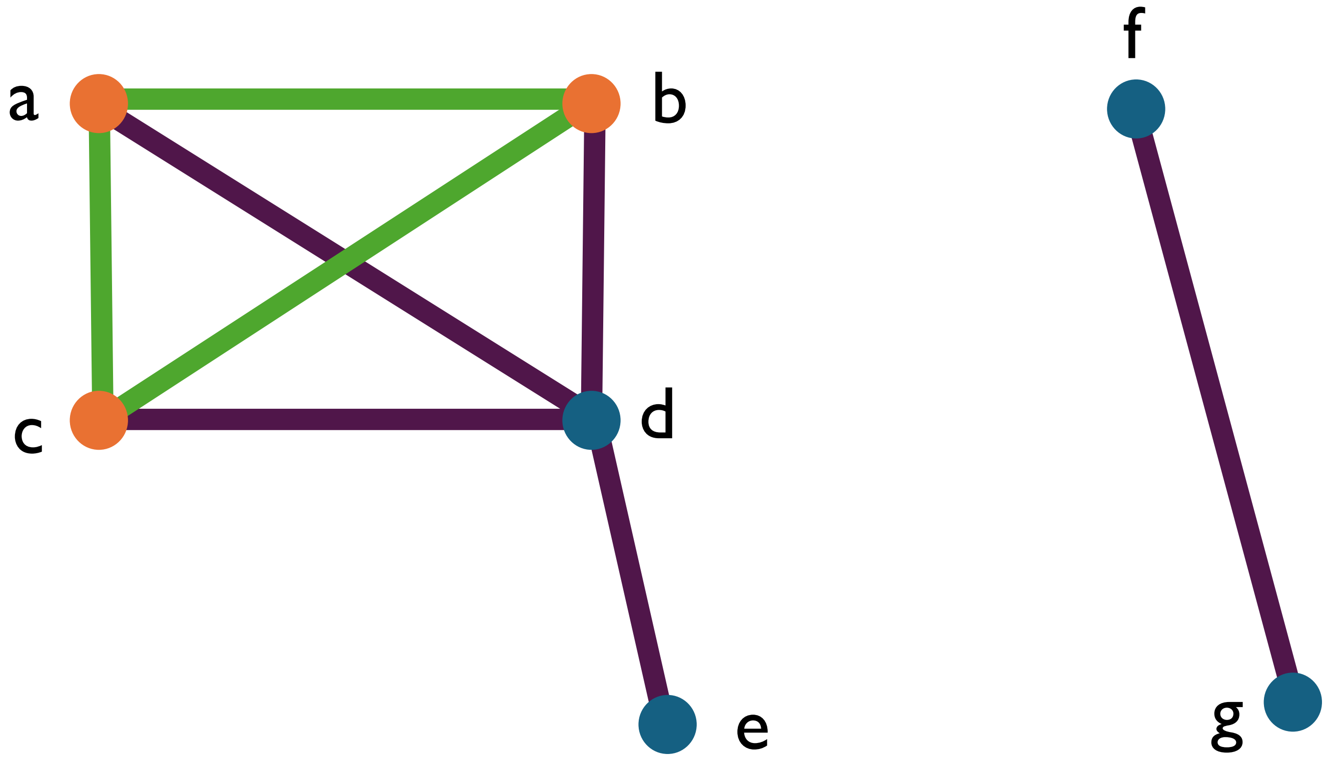
$$\lambda^* = \max_{S \subseteq V} \frac{f(S)}{|S|}$$

|E(S)| is supermodular!



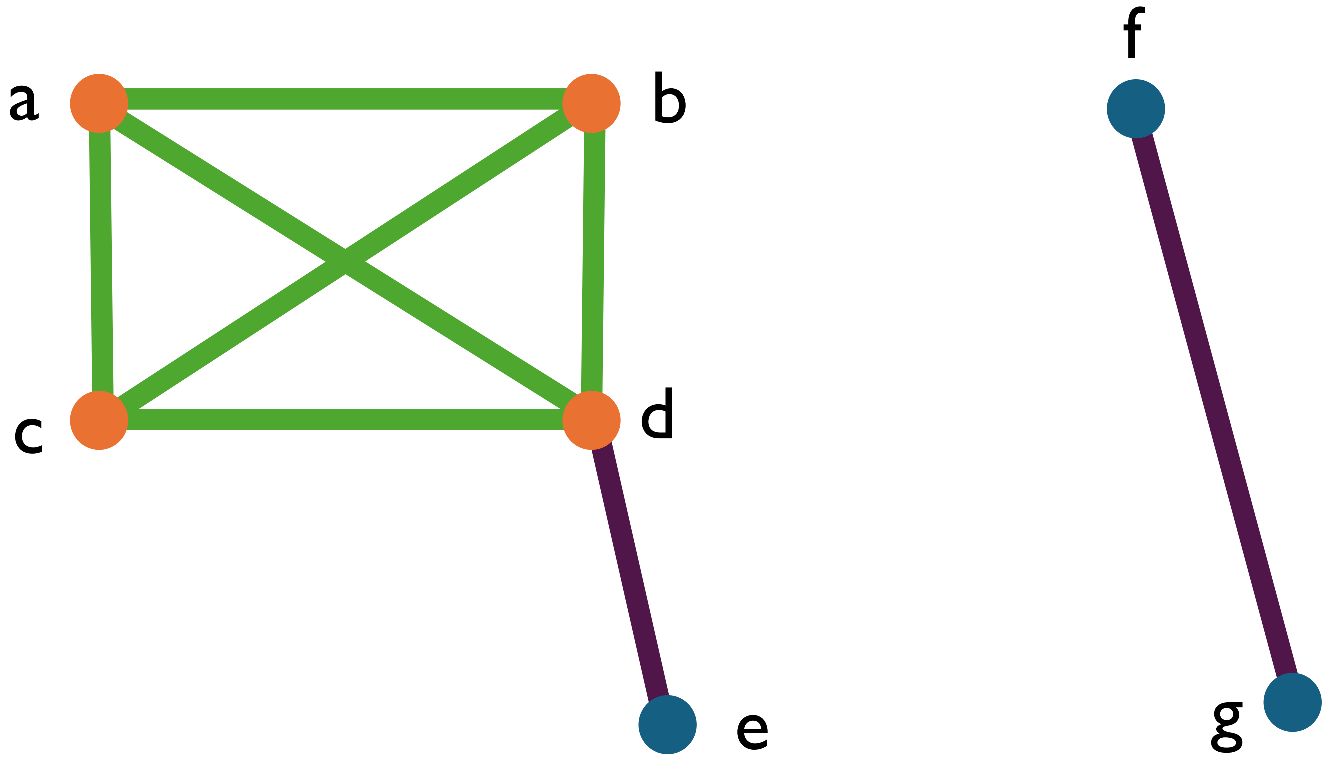
If we take S = {a, b}, then |E(S)| = 1.

|E(S)| is supermodular!



If we take T = {a, b, c}, then |E(T)| = 3. So f(c|S) = 2.

|E(S)| is supermodular!



If we take U = {a, b, c, d}, then |E(U)| = 6. So f(d|U) = 3!

The densest subgraph problem (DSP) is a special case of the densest supermodular set problem (DSSP).

(Charikar, Quanrud, Torres, 2022)

If iterative peeling works for all supermodular set functions, then we can use it to solve more problems.

(Charikar, Quanrud, Torres, 2022)

If iterative peeling works for all supermodular set functions, then perhaps we can use it to solve more problems.

Amazingly, this works!

(Charikar, Quanrud, Torres, 2022)

Part 6:

In Conclusion…

Theory helps us solve problems.

- we can describe problems more precisely (language)
- we can see how problems are related (abstraction)
- we can see how a solution might be reused (generalization)