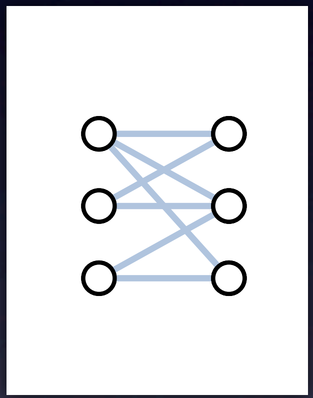
Exponential Time Algorithms

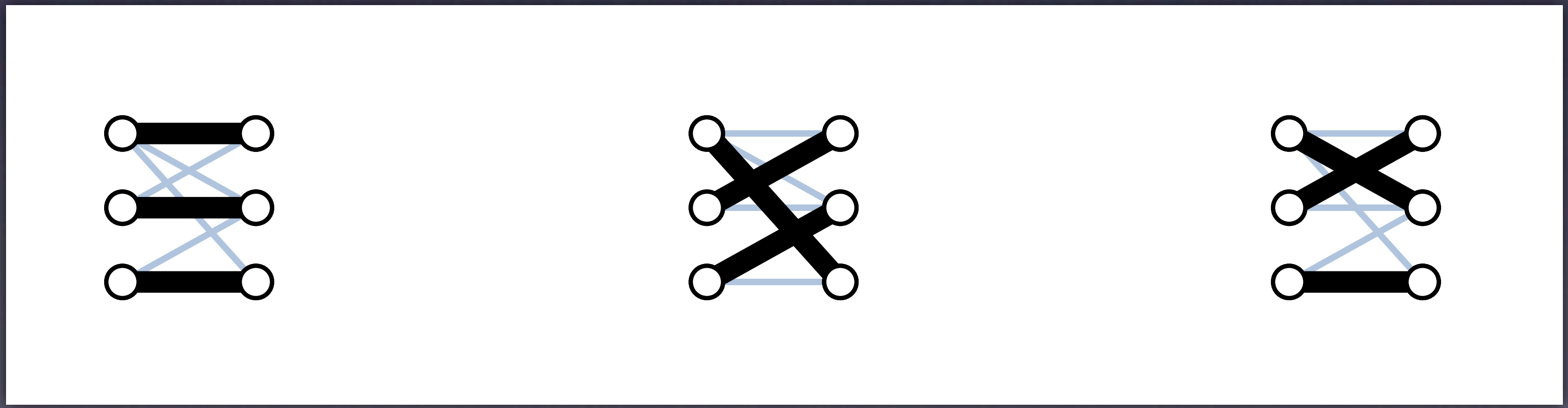
Thore Husfeldt

IT University of Copenhagen
Lund University

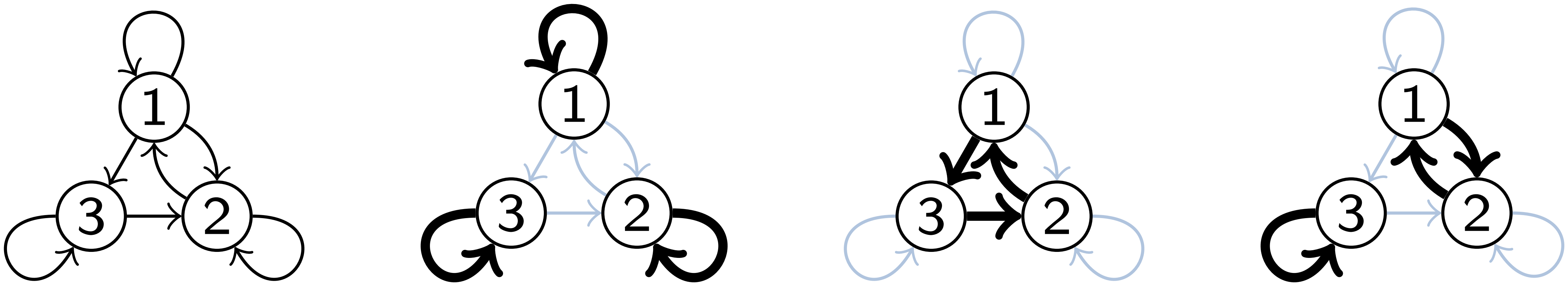
Perfect Matchings

Perfect Matchings in Bipartite Graphs

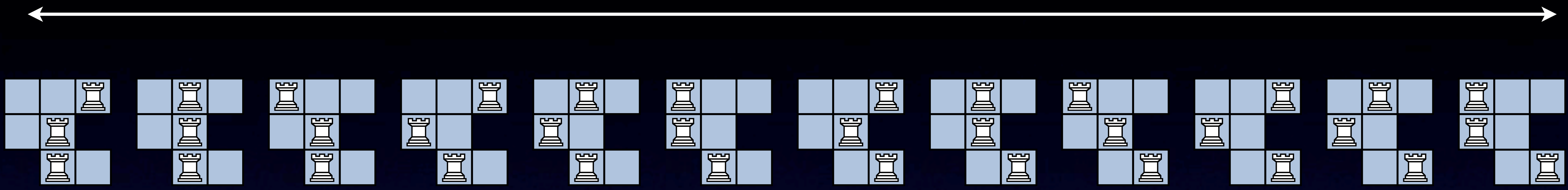




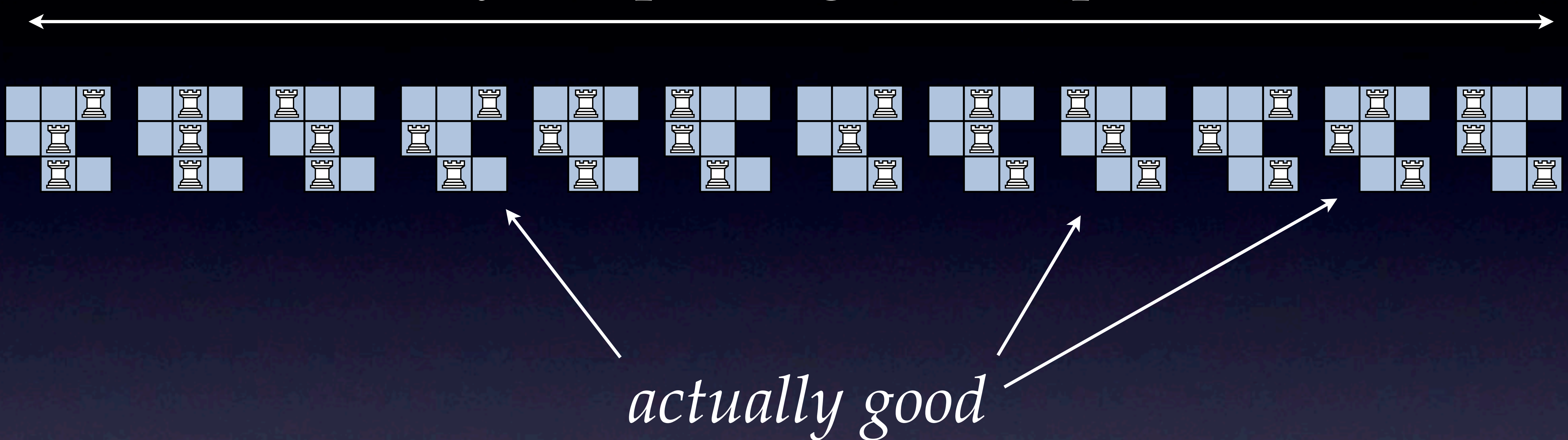
$$\begin{matrix}1&1&1\\1&1&0\\0&1&1\end{matrix}\qquad\begin{matrix}1&1&1\\1&1&0\\0&1&1\end{matrix}\qquad\begin{matrix}1&1&1\\1&1&0\\0&1&1\end{matrix}\qquad\begin{matrix}1&1&1\\1&1&0\\0&1&1\end{matrix}$$



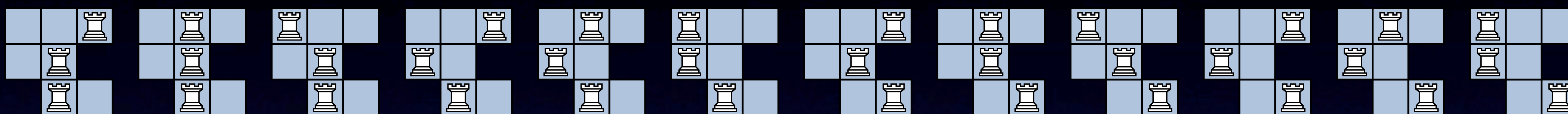
All ways of placing 1 rook per row



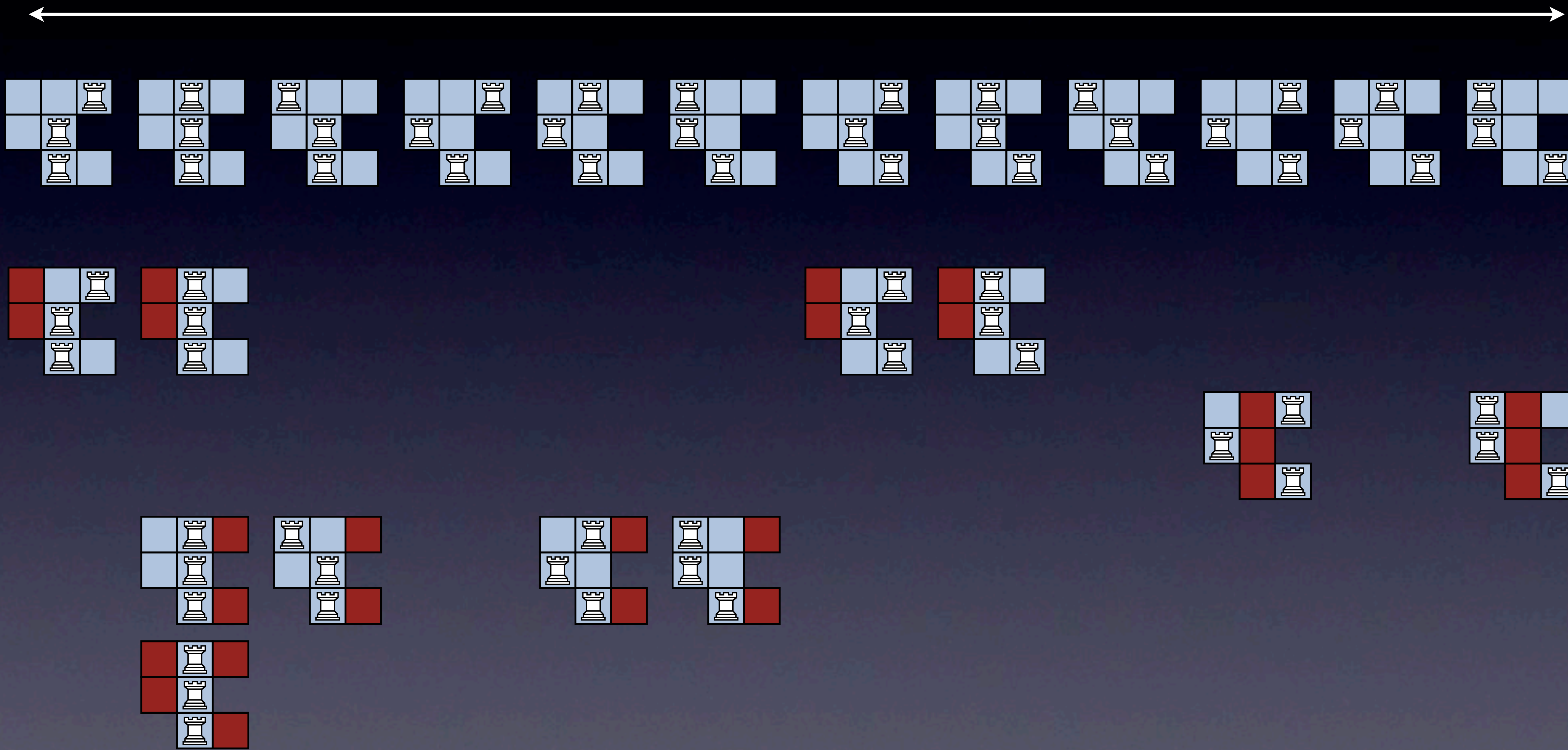
All ways of placing 1 rook per row



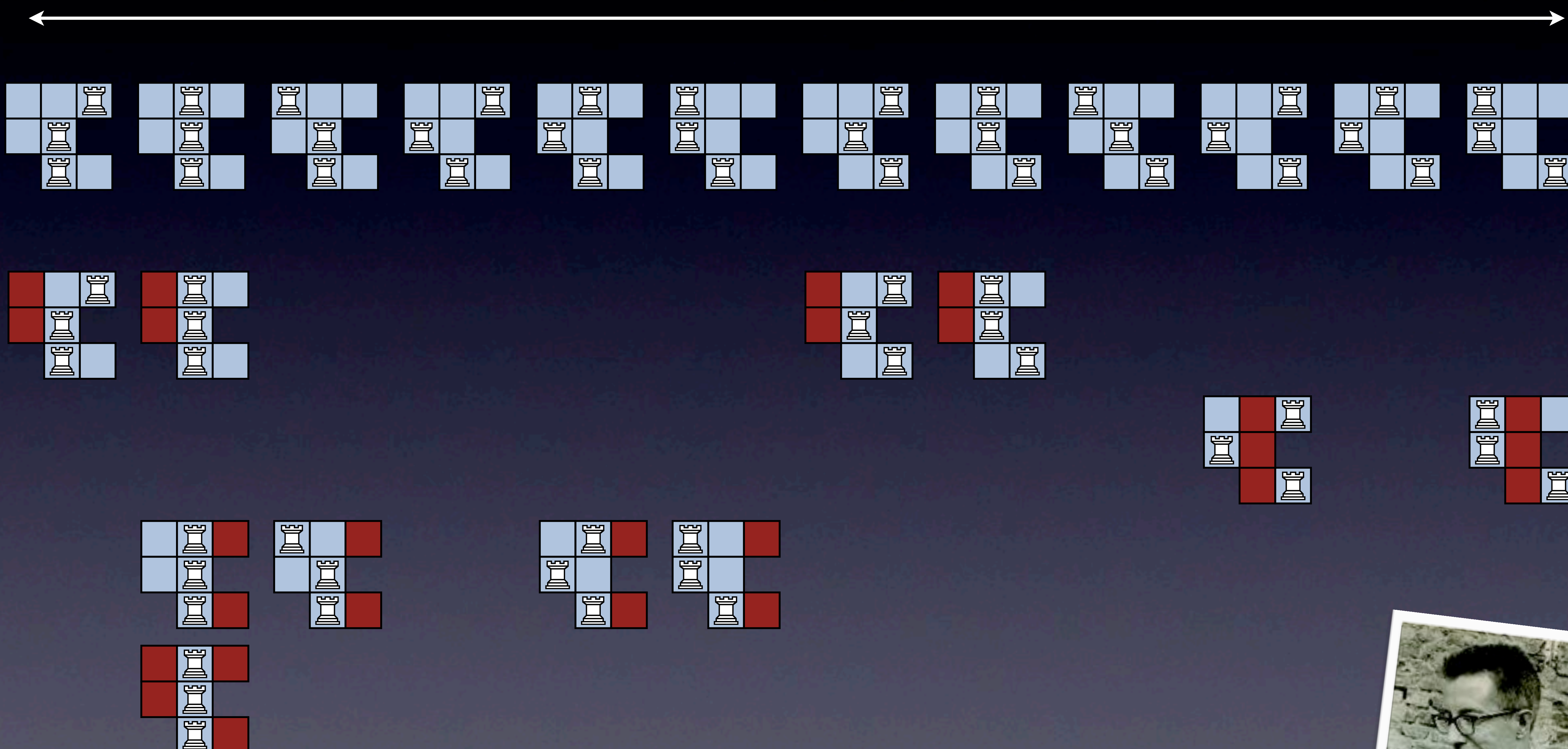
All ways of placing 1 rook per row



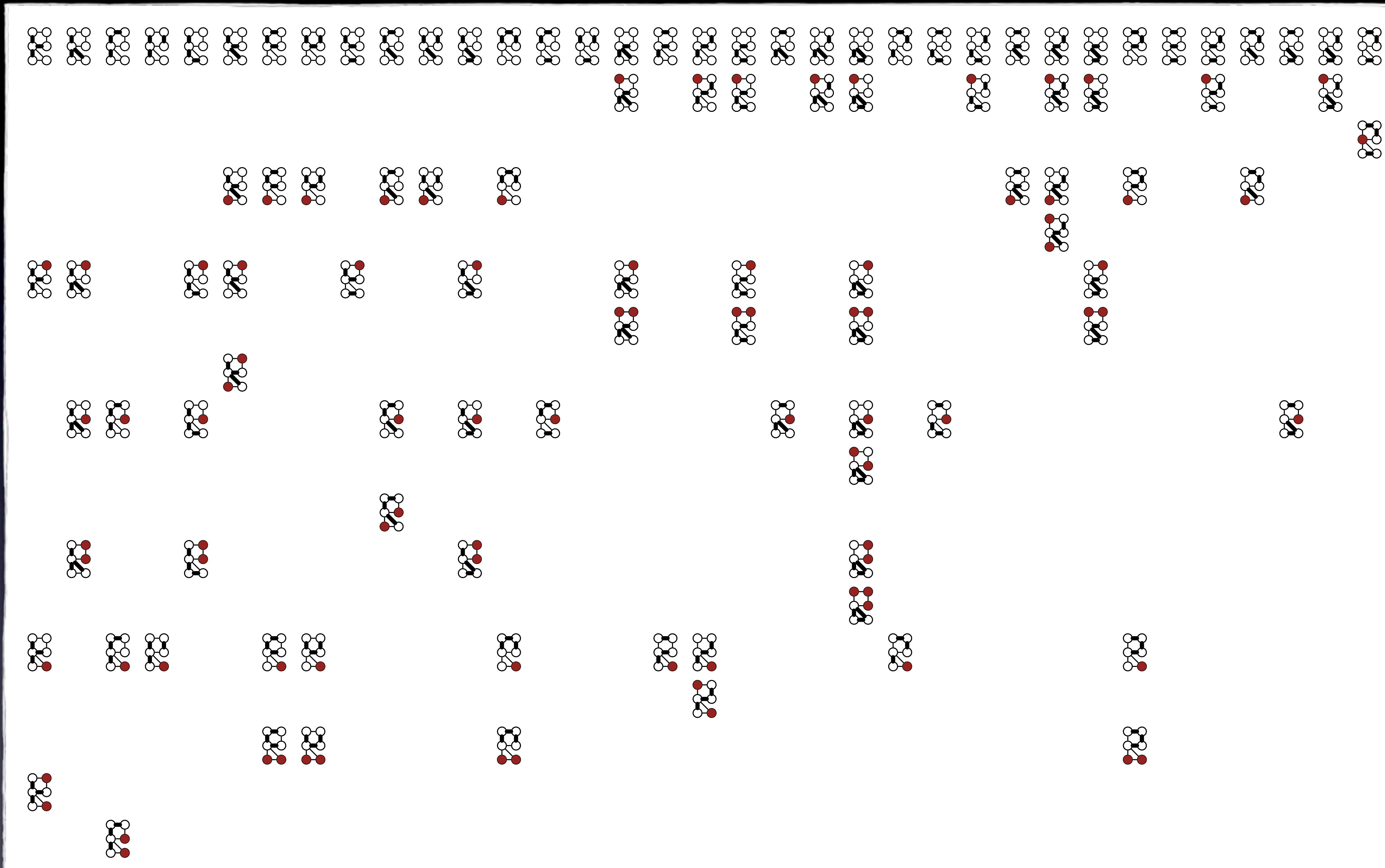
All ways of placing 1 rook per row

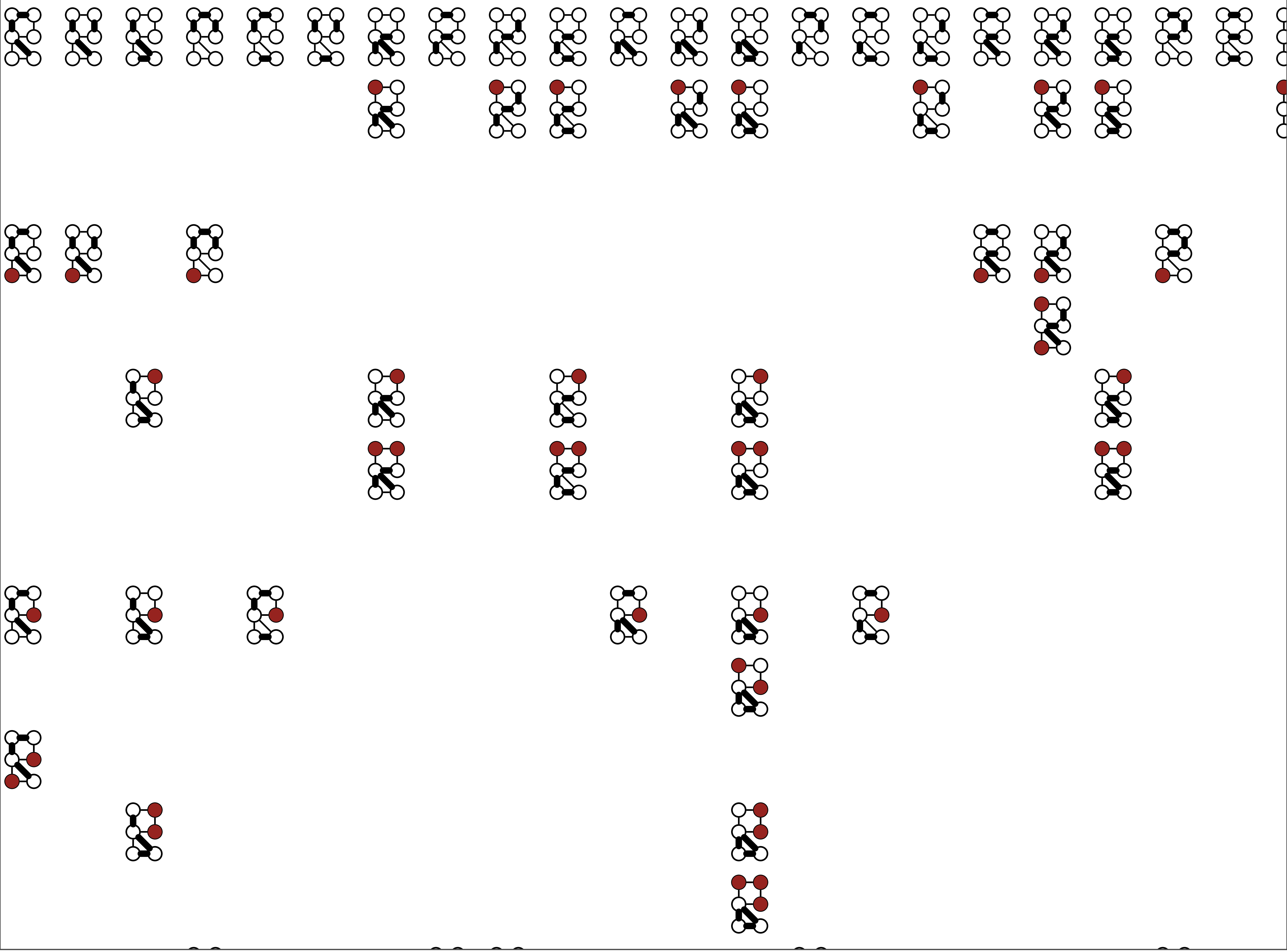


All ways of placing 1 rook per row

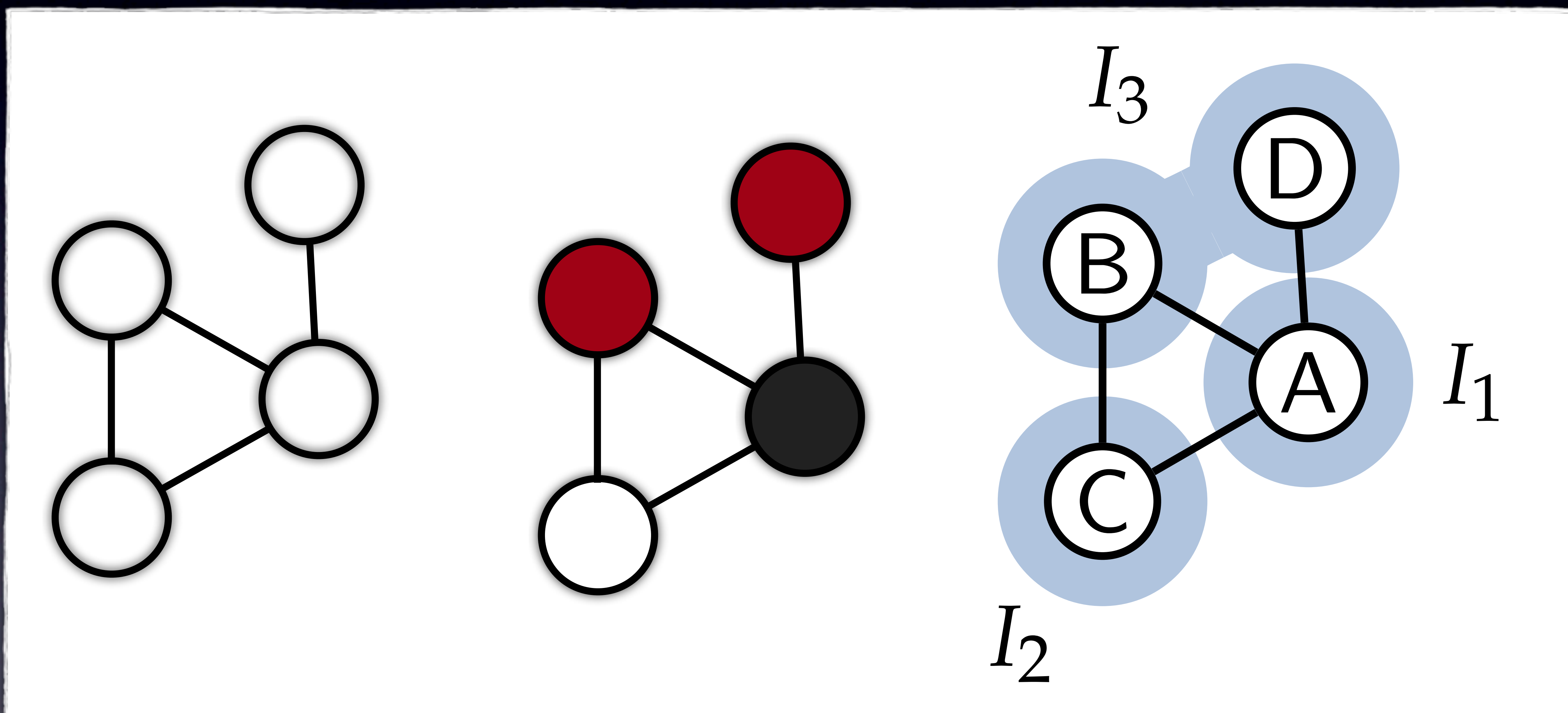


Ryser



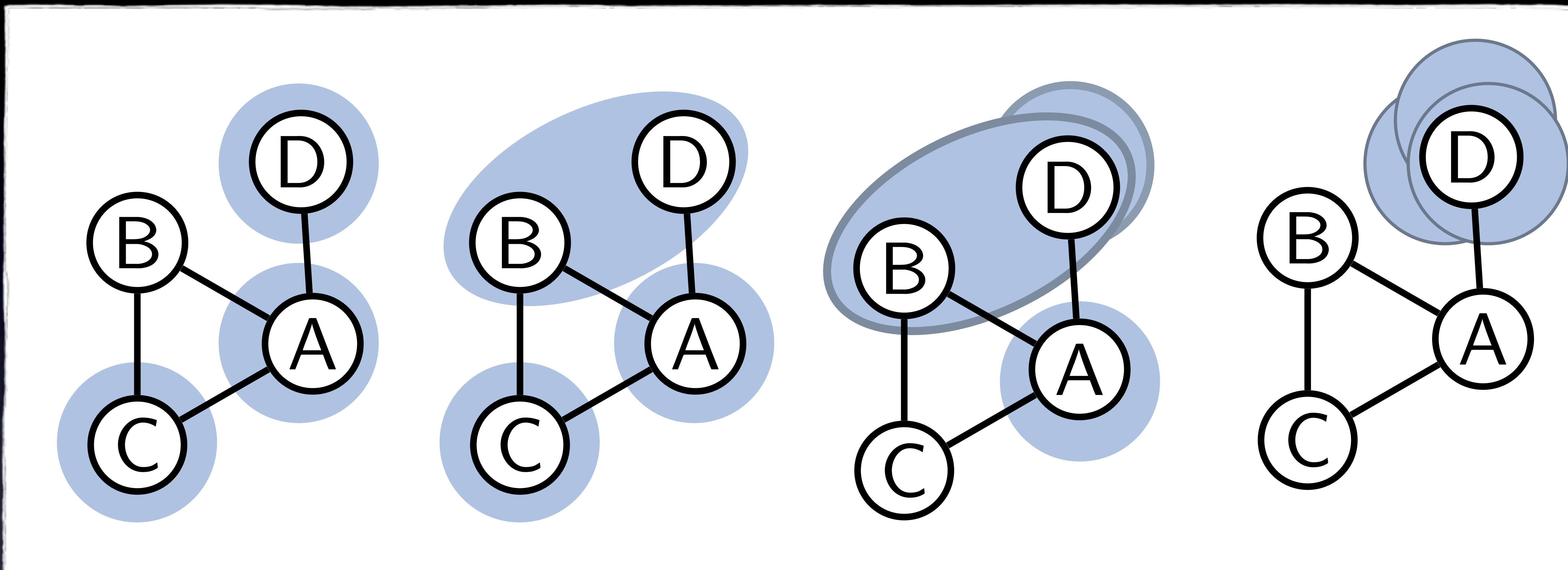


Vertex colouring



I_3
D
B
A
I_1
C
I_2

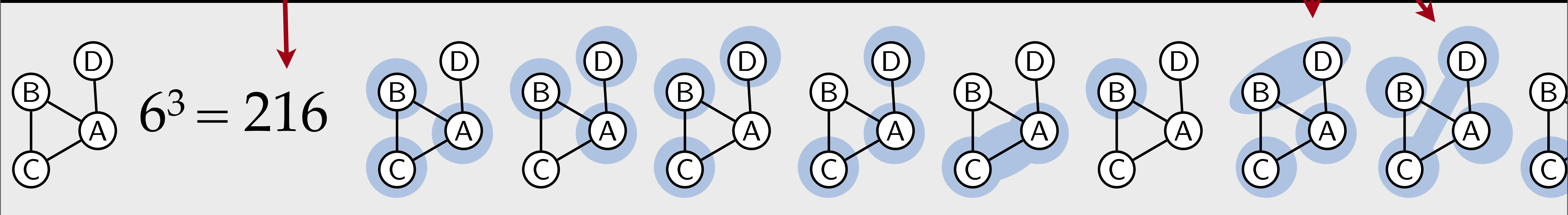
Picking 3 independent sets



A	B	C	D	AB	AC	AD	BC	BD	CD	ABC	ABD	ACD	BCD	ABCD
1	1	1	1	0	0	0	0	1	1	0	0	0	0	0

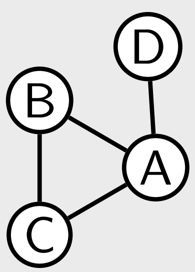
ways pick 3 indep. sets

actually useful ones

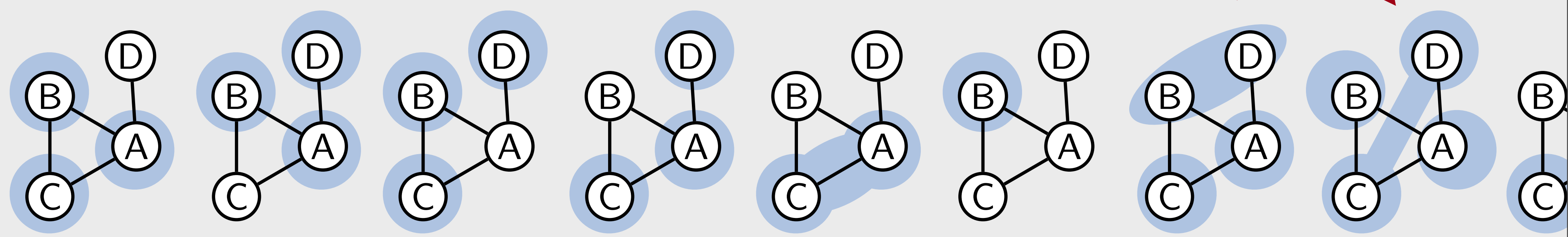


$\#$ ways pick 3 indep. sets

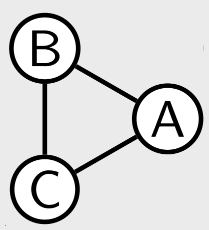
actually useful ones



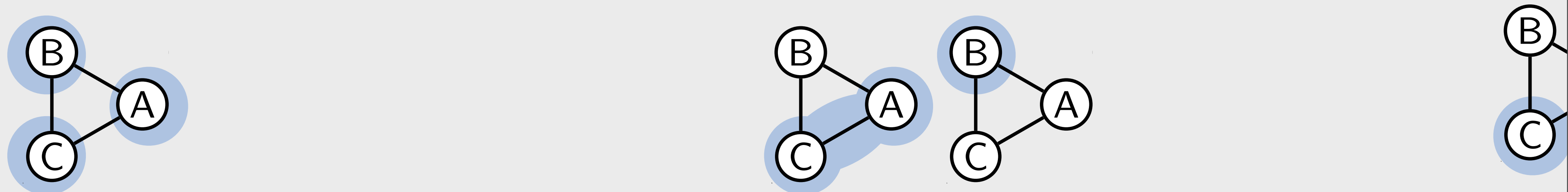
$6^3 = 216$



vertex subsets

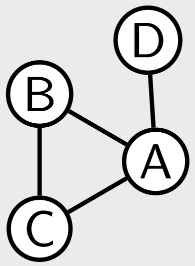


$3^3 = 27$

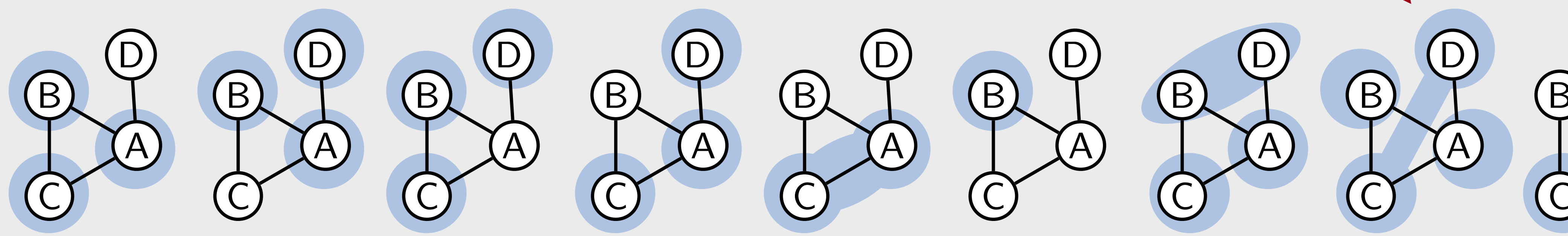


ways pick 3 indep. sets

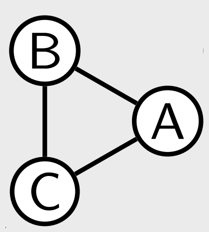
actually useful ones



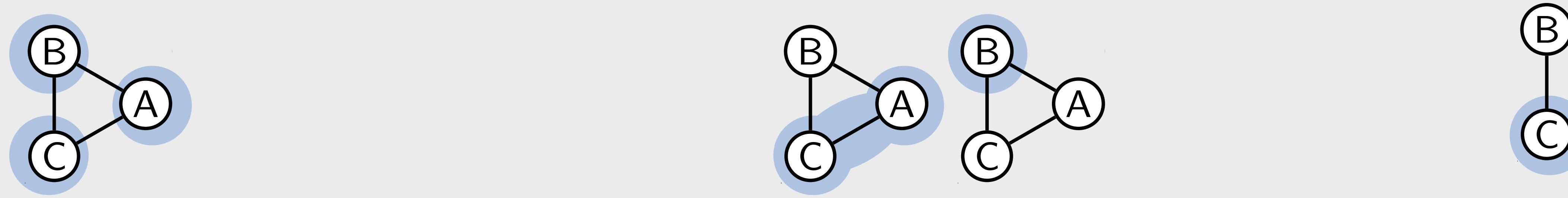
$6^3 = 216$

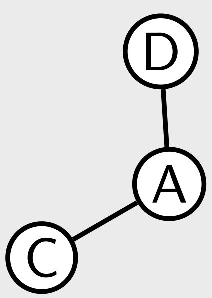


vertex subsets



$3^3 = 27$

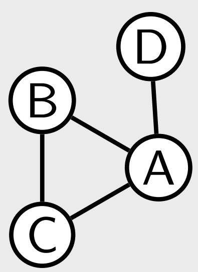




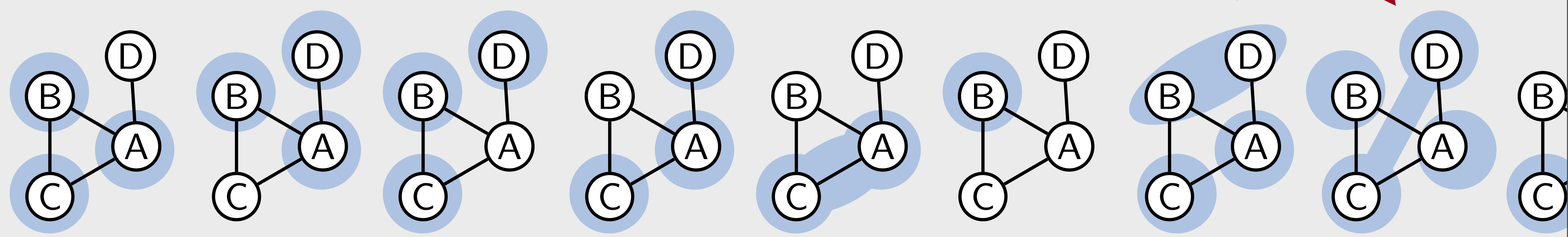
$4^3 = 64$

 # ways pick 3 indep. sets

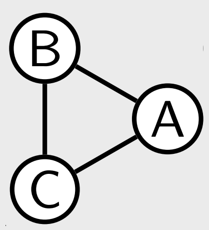
actually useful ones



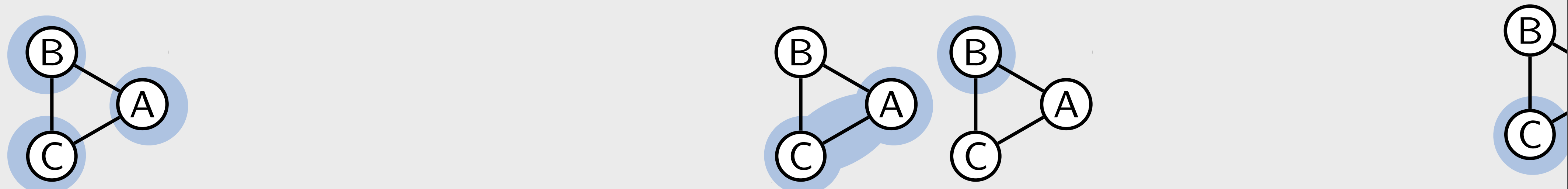
$6^3 = 216$

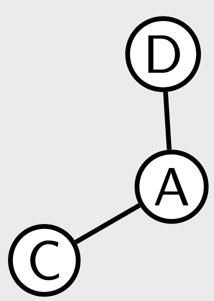


vertex subsets

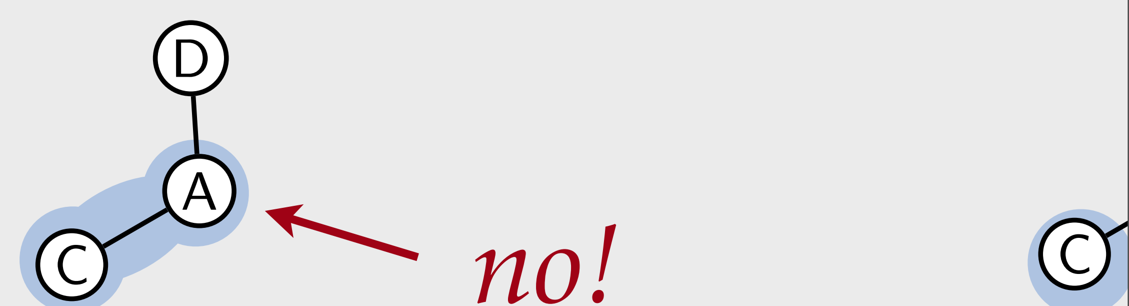


$3^3 = 27$

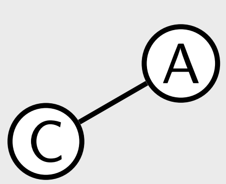




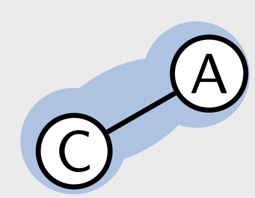
$4^3 = 64$



no!



$2^3 = 8$



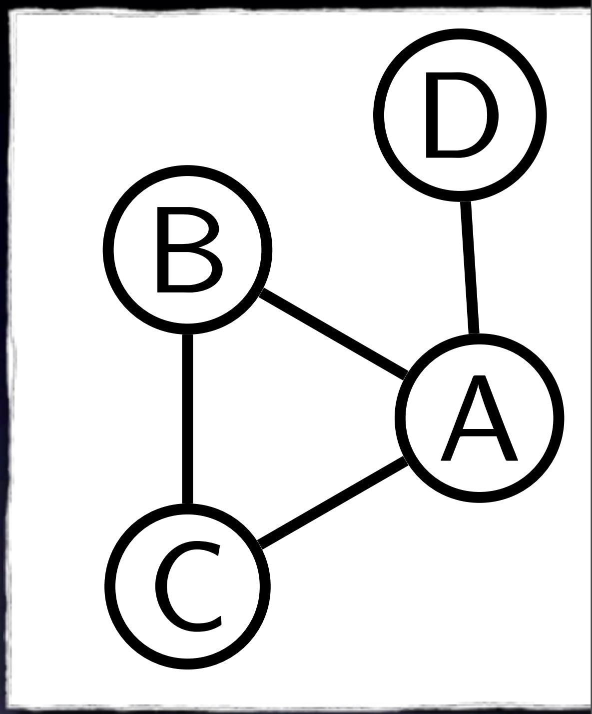
yes!

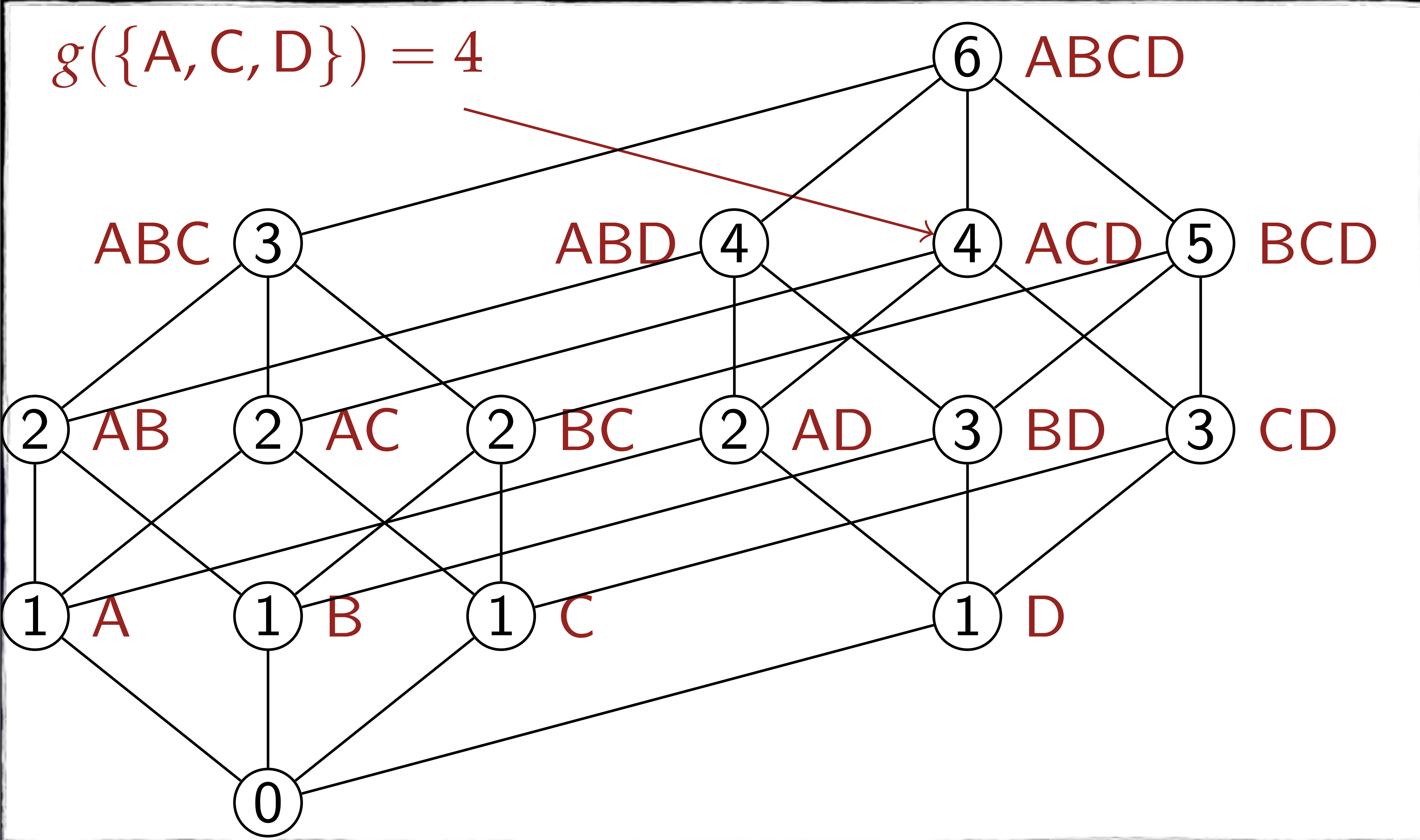
Vertex subset S	# indep. subsets, $g(S)$	$(g(S))^3$
A	1	1
B	1	1
C	1	1
D	1	1
AB	2	8
AC	2	8
AD	2	8
BC	2	8
BD	3	27
CD	3	27
ABC	3	27
ABD	4	64
ACD	4	64
BCD	5	125
ABCD	6	216

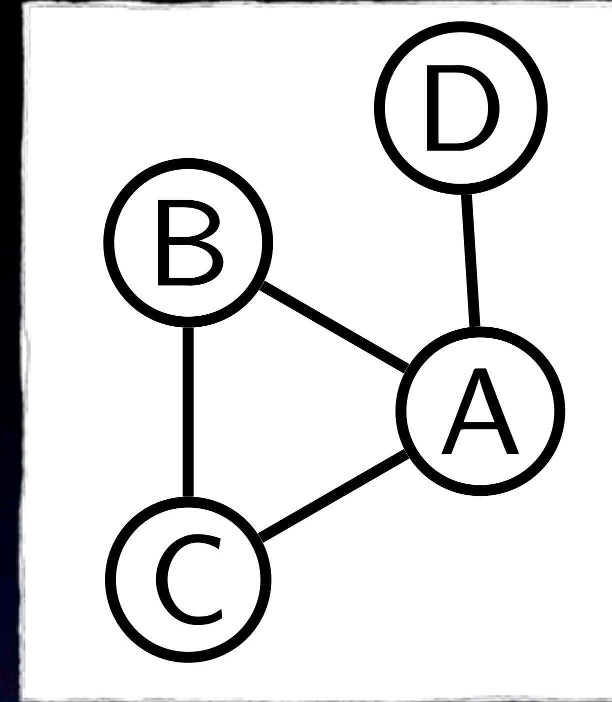
−

+

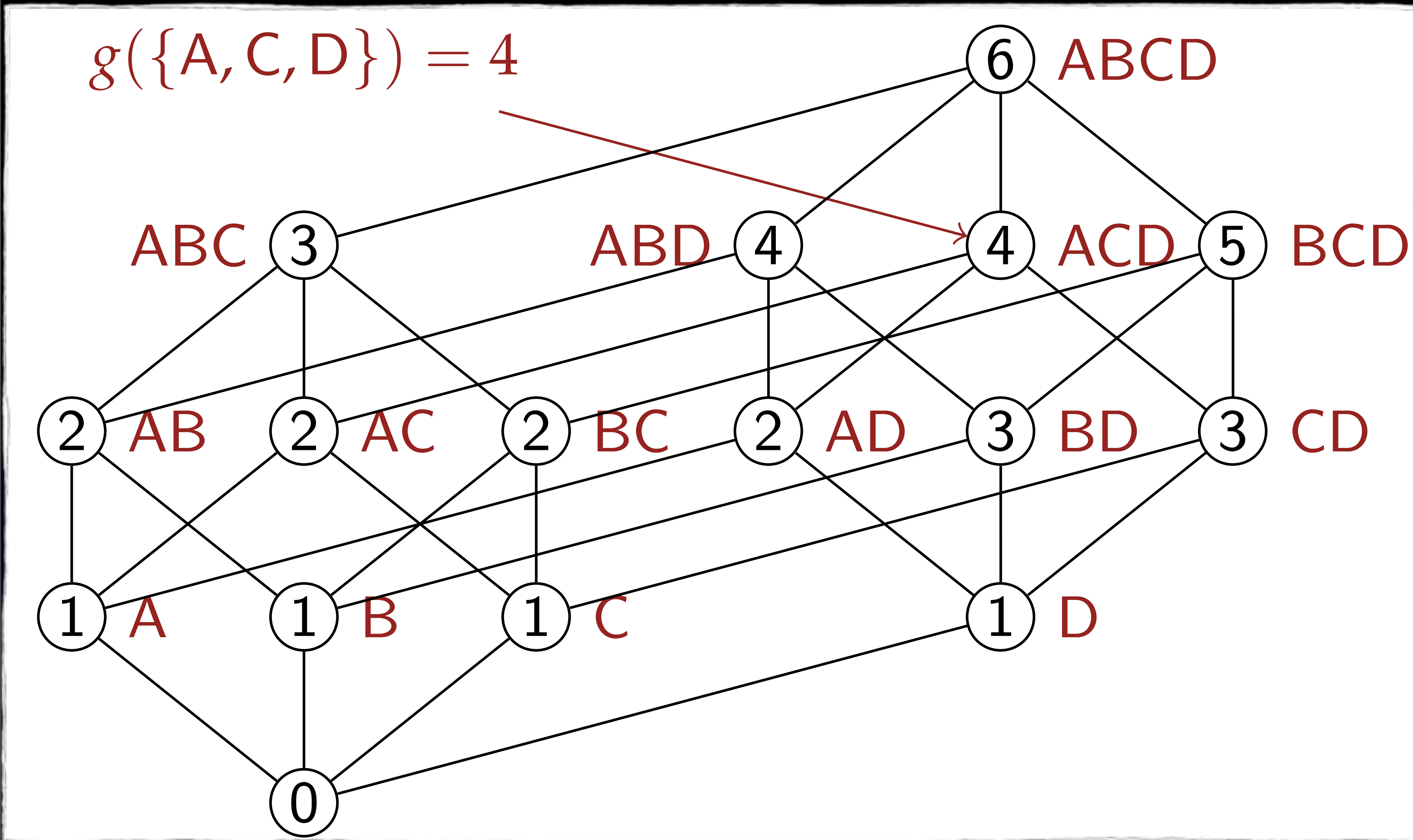
−

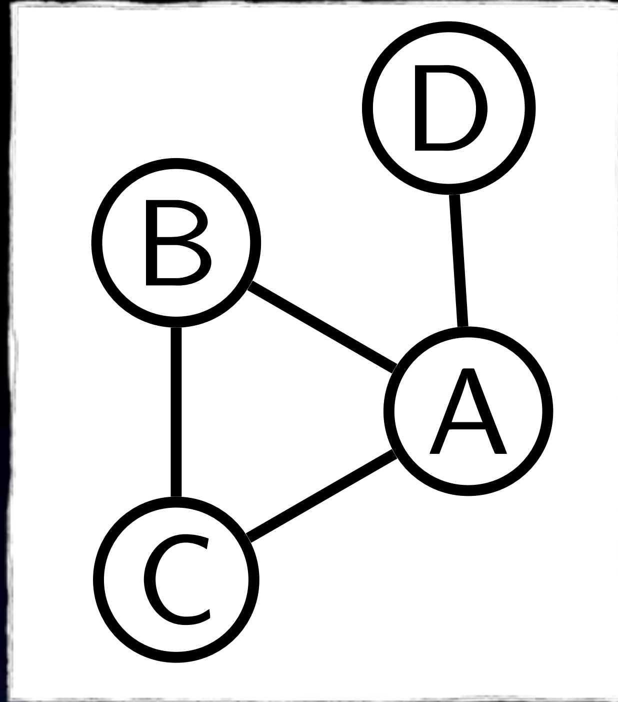


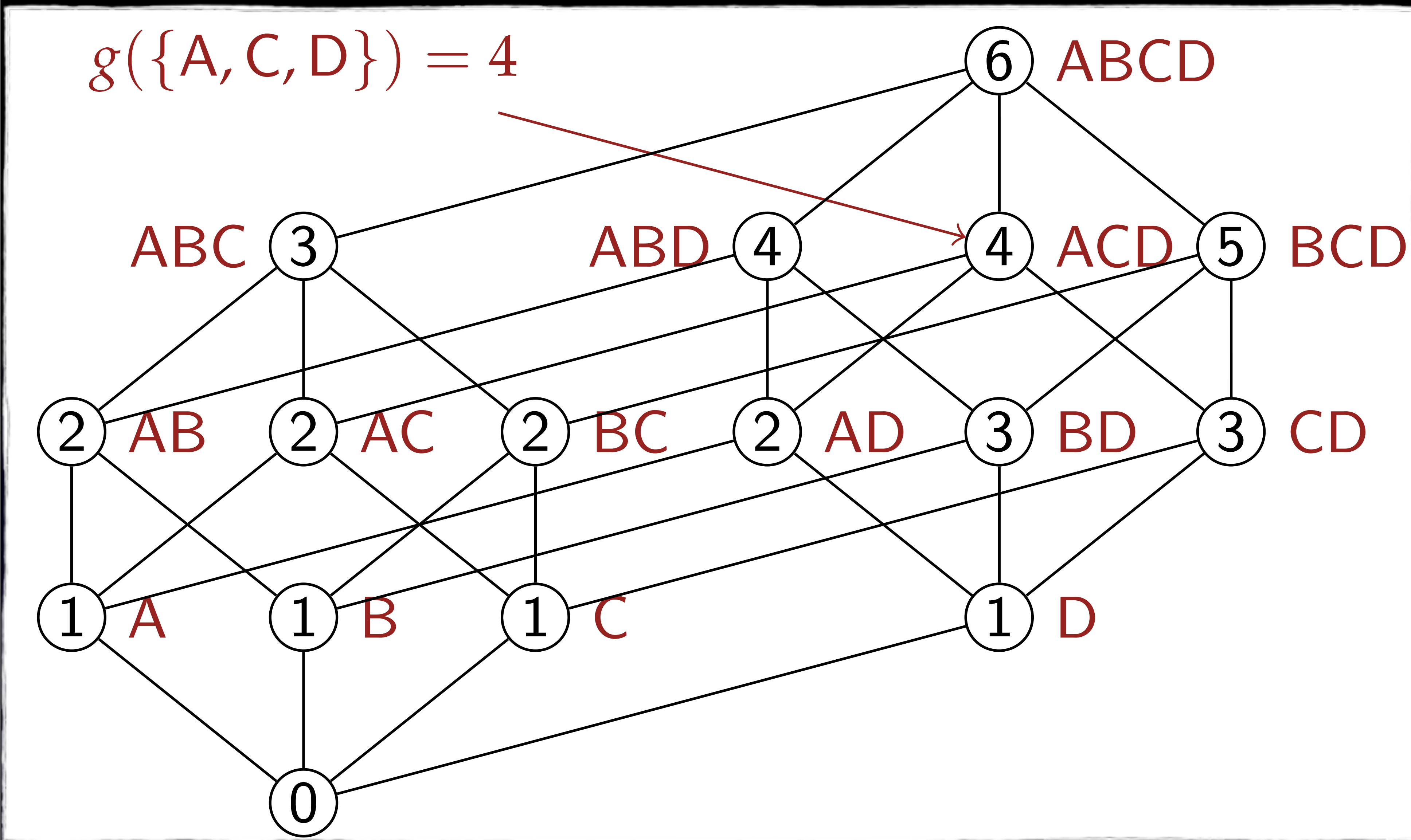


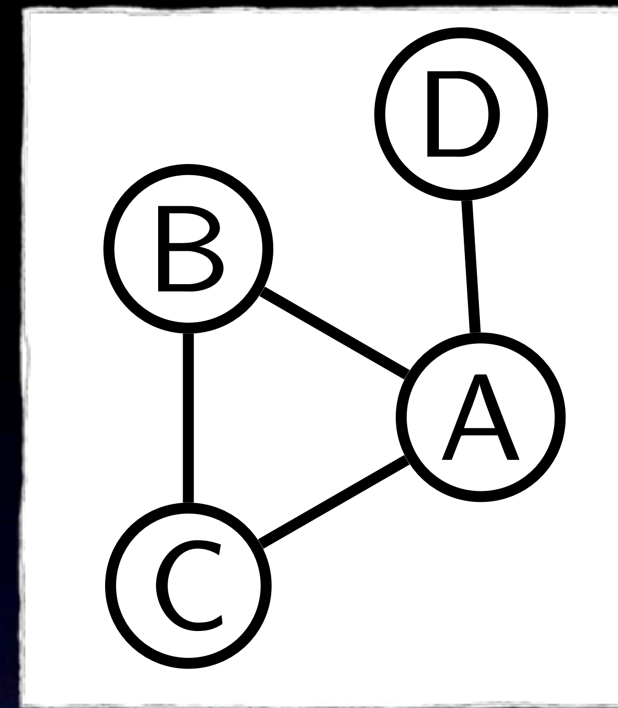


Vertex subset S	# indep.subsets, $g(S)$	$(g(S))^3$
A	1	1
B	1	1
C	1	1
D	1	1
AB	2	8
AC	2	8
AD	2	8
BC	2	8
BD	3	27
CD	3	27
ABC	3	27
ABD	4	64
ACD	4	64
BCD	5	125
ABCD	6	216


$g(\{A, C, D\}) = 4$
6 ABCD
ABC 3
ABD 4
4 ACD
5 BCD
2 AB
2 AC
2 BC
2 AD
3 BD
3 CD
1 A
1 B
1 C
1 D
0

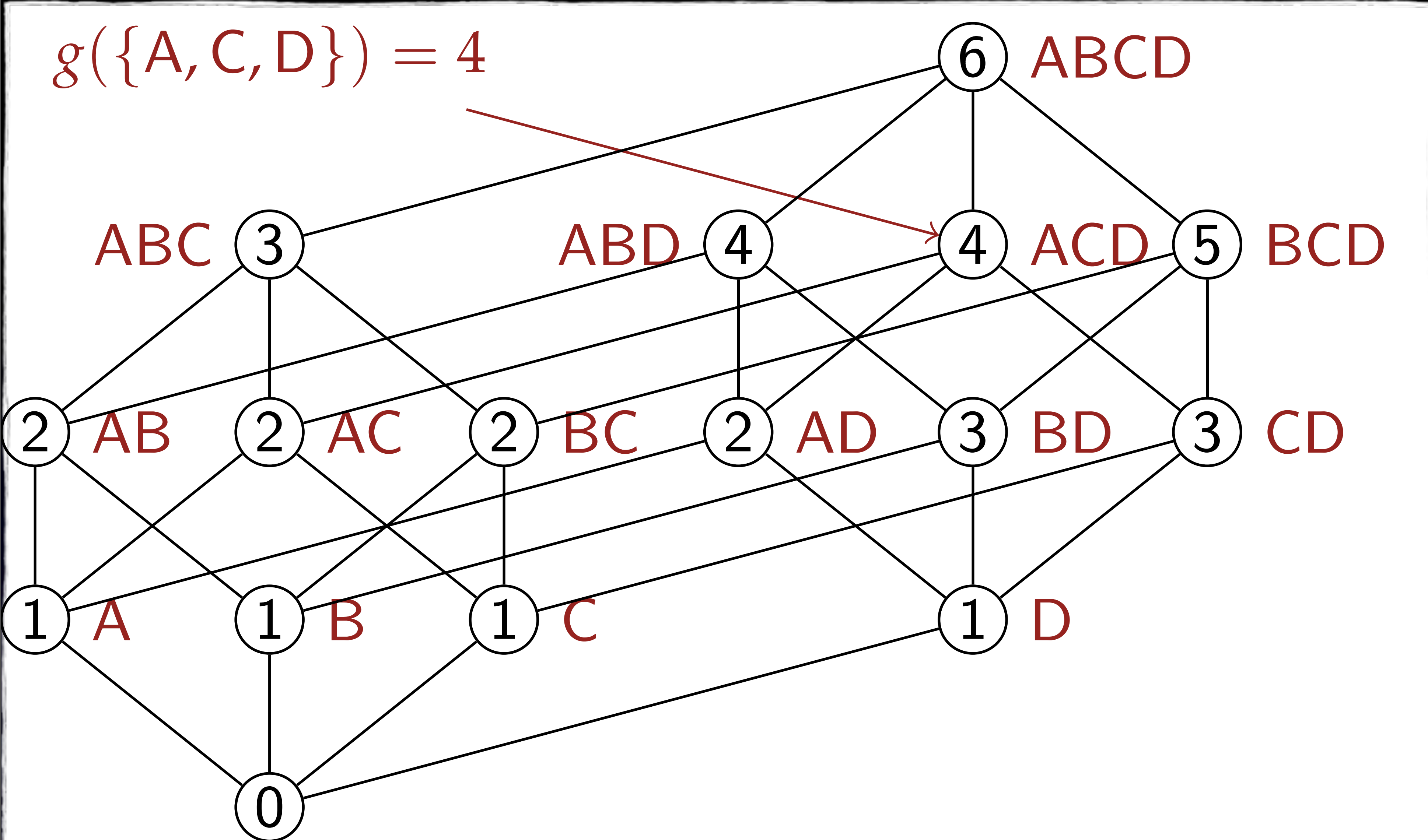

D
B
A
C

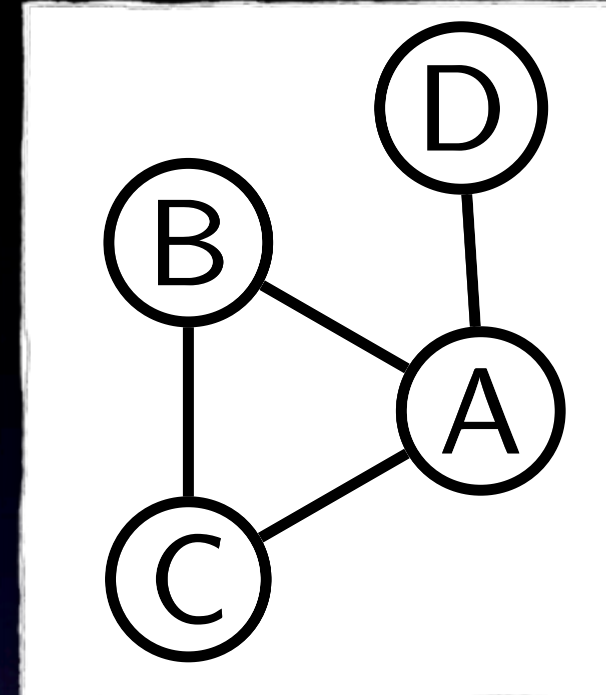




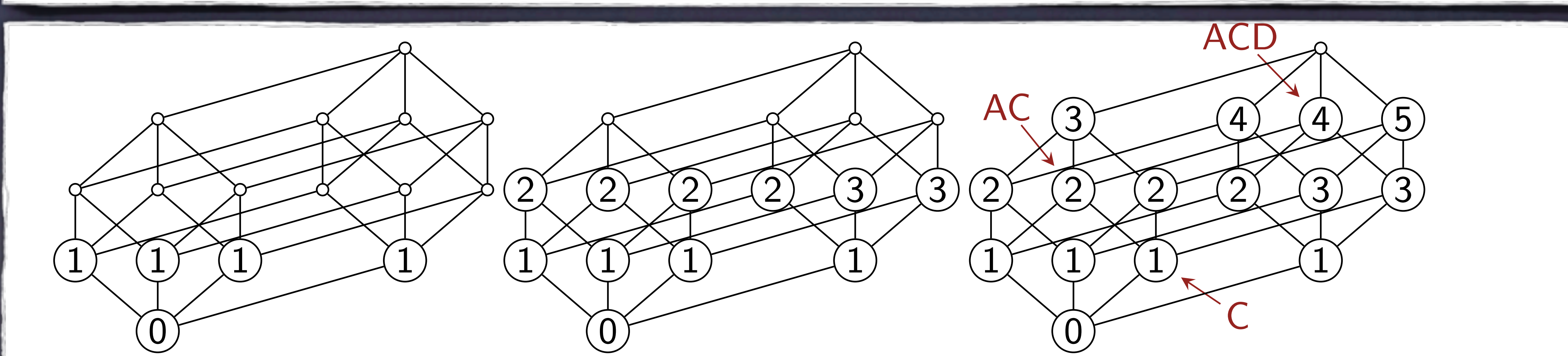
$$\sum_{S \subseteq N} 2^{|S|} = \sum_{i=1}^{n} \binom{n}{i} 2^i = 3^n$$

polynomial space





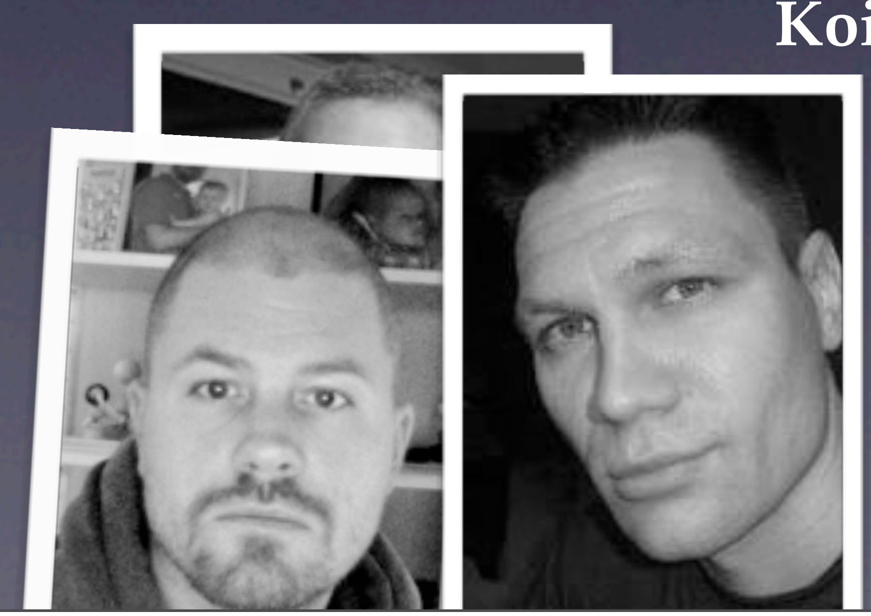
$$g(S) = g(S \setminus \{v\}) + g(S \setminus N[v]) + 1 \qquad (v \in S)$$



Graph colouring

Compute $\sum_{S \subseteq N} (-1)^{n-|S|} (g(S))^k$

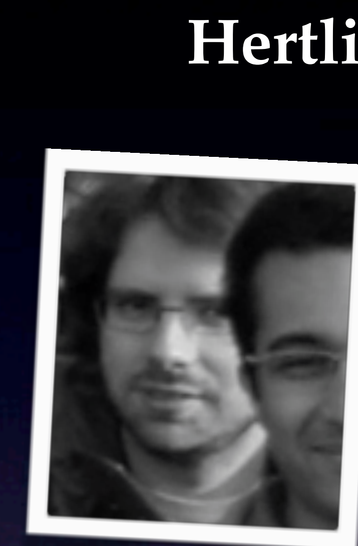
$O^*(3^n)$ time polynomial space	$O^*(2^n)$ time $O^*(2^n)$ space



Exponential Time Hypothesis

Exponential Time Hypothesis

Can do 3-Sat *in*
time 1.308^n



Exponential Time Hypothesis

Hertli

Can do 3-Sat *in time* 1.308^n

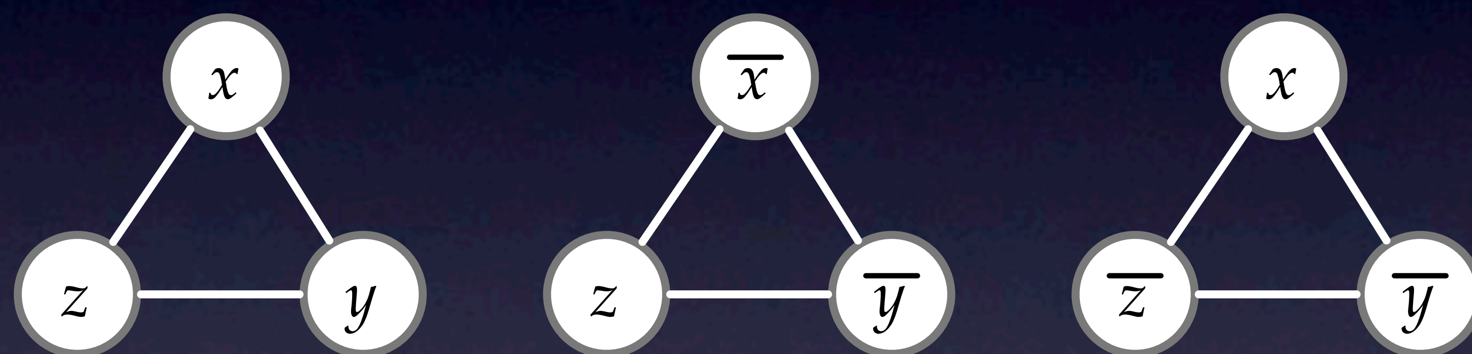
Zane

Impagliazzo

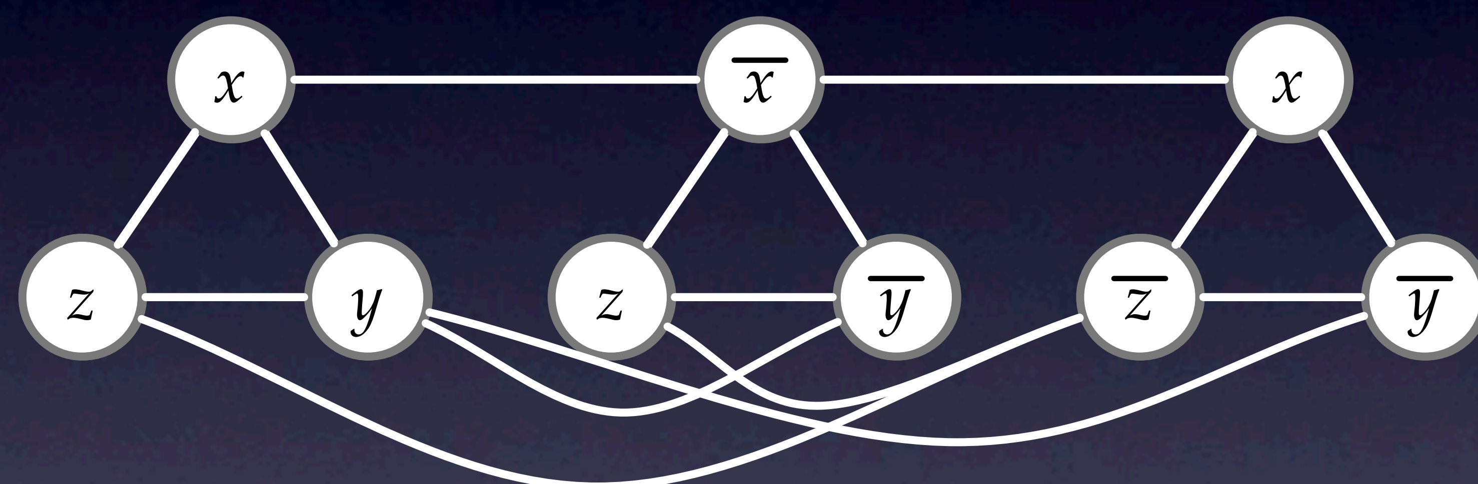
Paturi

Can't do 3-Sat *in time* $\exp(o(n))$

$$(x \vee y \vee z) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z})$$



$$(x \vee y \vee z) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z})$$



n vars

m clauses

$3m=O(n^3)$ verts

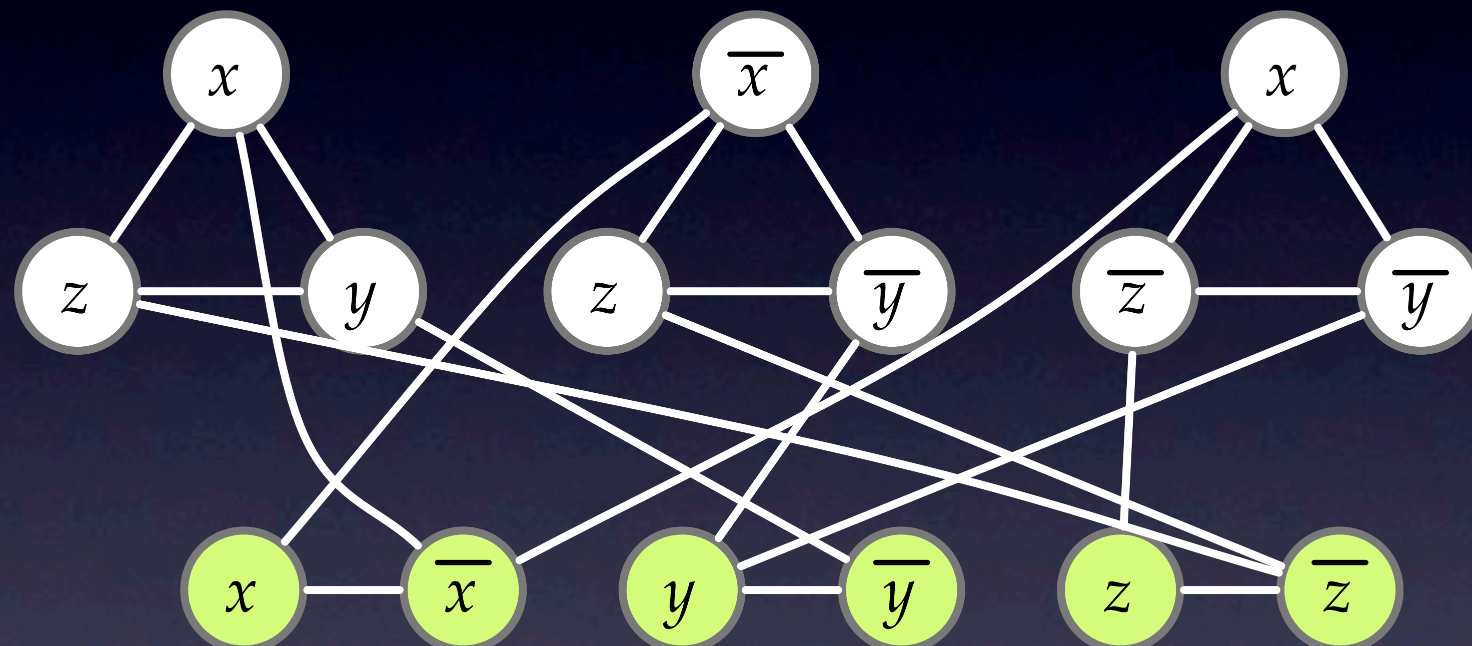
$O(m^2)$ edges

$\exp(o(n))$ alg for 3-SAT

$\exp(o(n^{1/3}))$ alg for I.S.

$\exp(o(m^{1/2}))$ alg for I.S.

$$(x \vee y \vee z) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z})$$



n vars

m clauses

exp(o(n)) alg for 3-SAT

$3m = O(n^3)$ verts

O(m) edges

exp(o($n^{1/3}$)) alg for I.S.

exp(o(m)) alg for I.S.

Independent set
n vertices m edges

1.1888^n

c^m

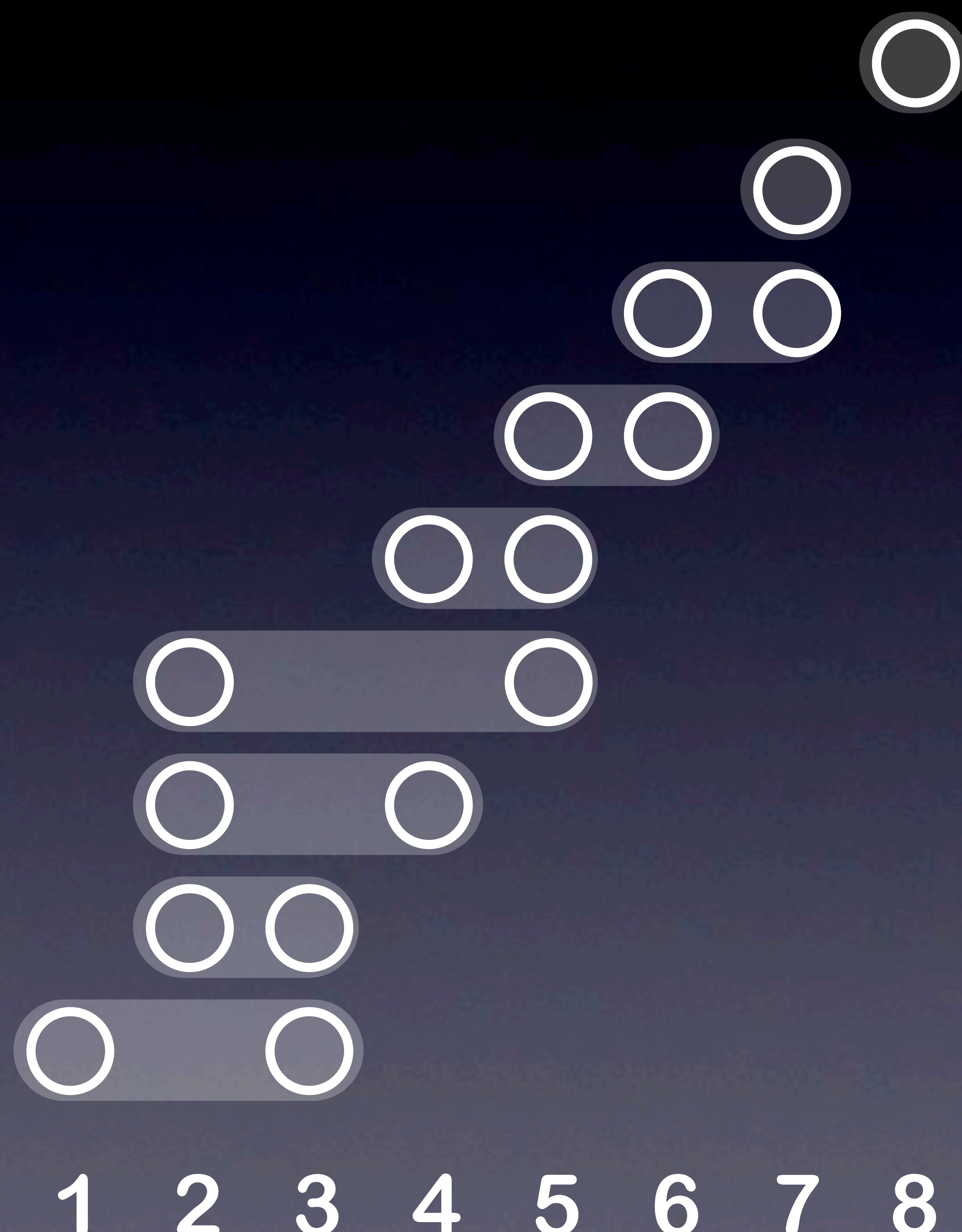
Clique
n vertices m edges

1.1888^n

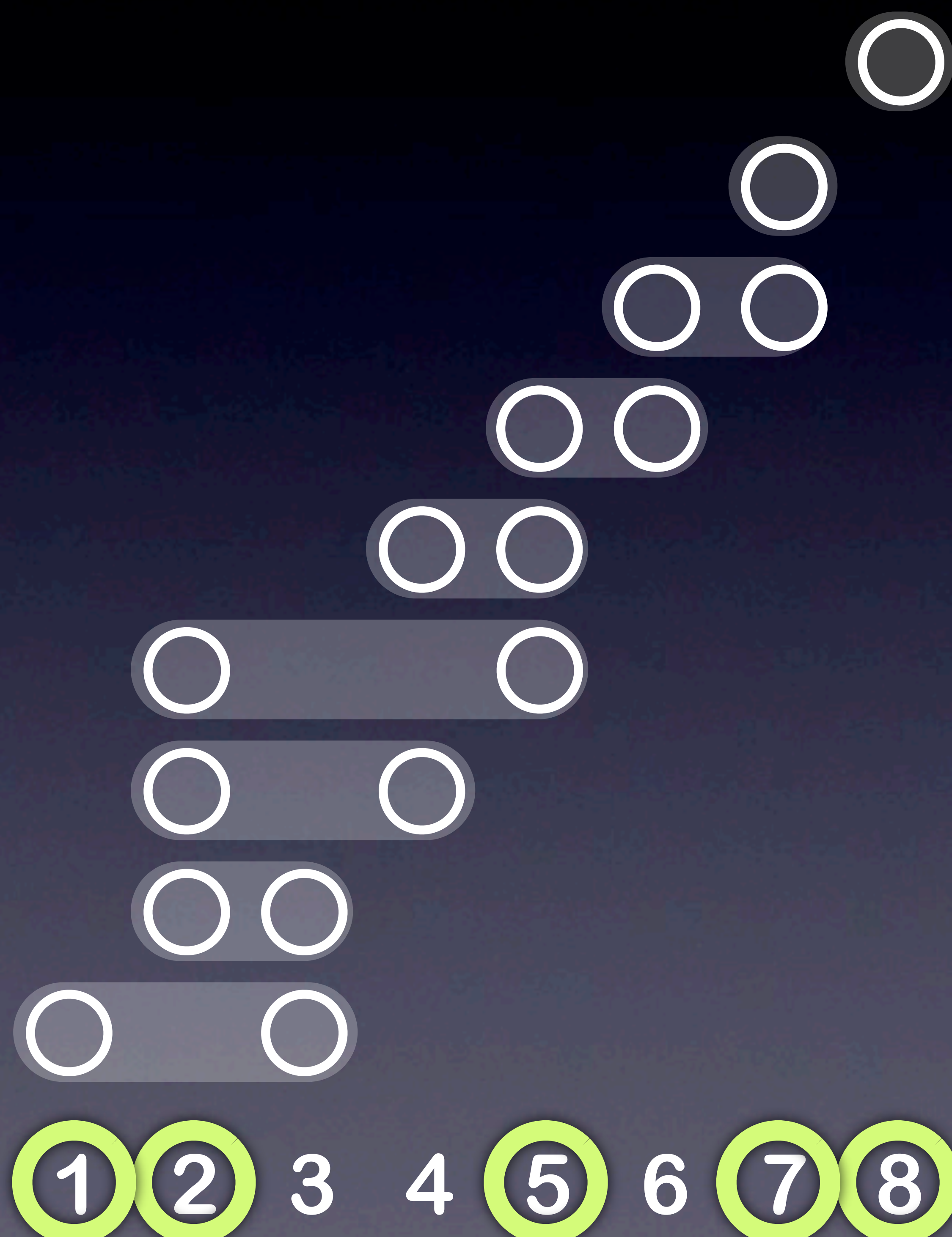
$2^{\sqrt{m}\log n}$

Sparsification

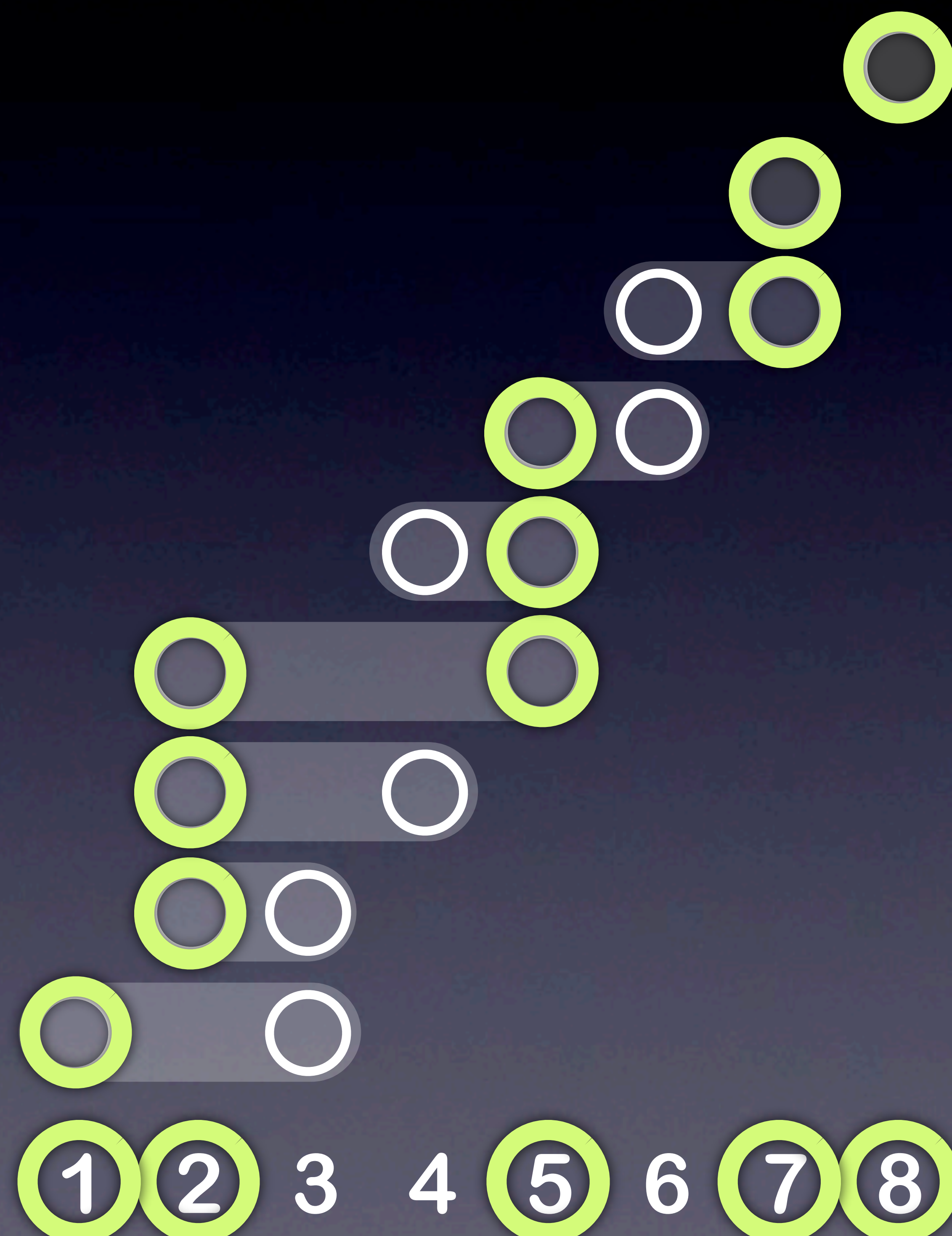
Hitting Set



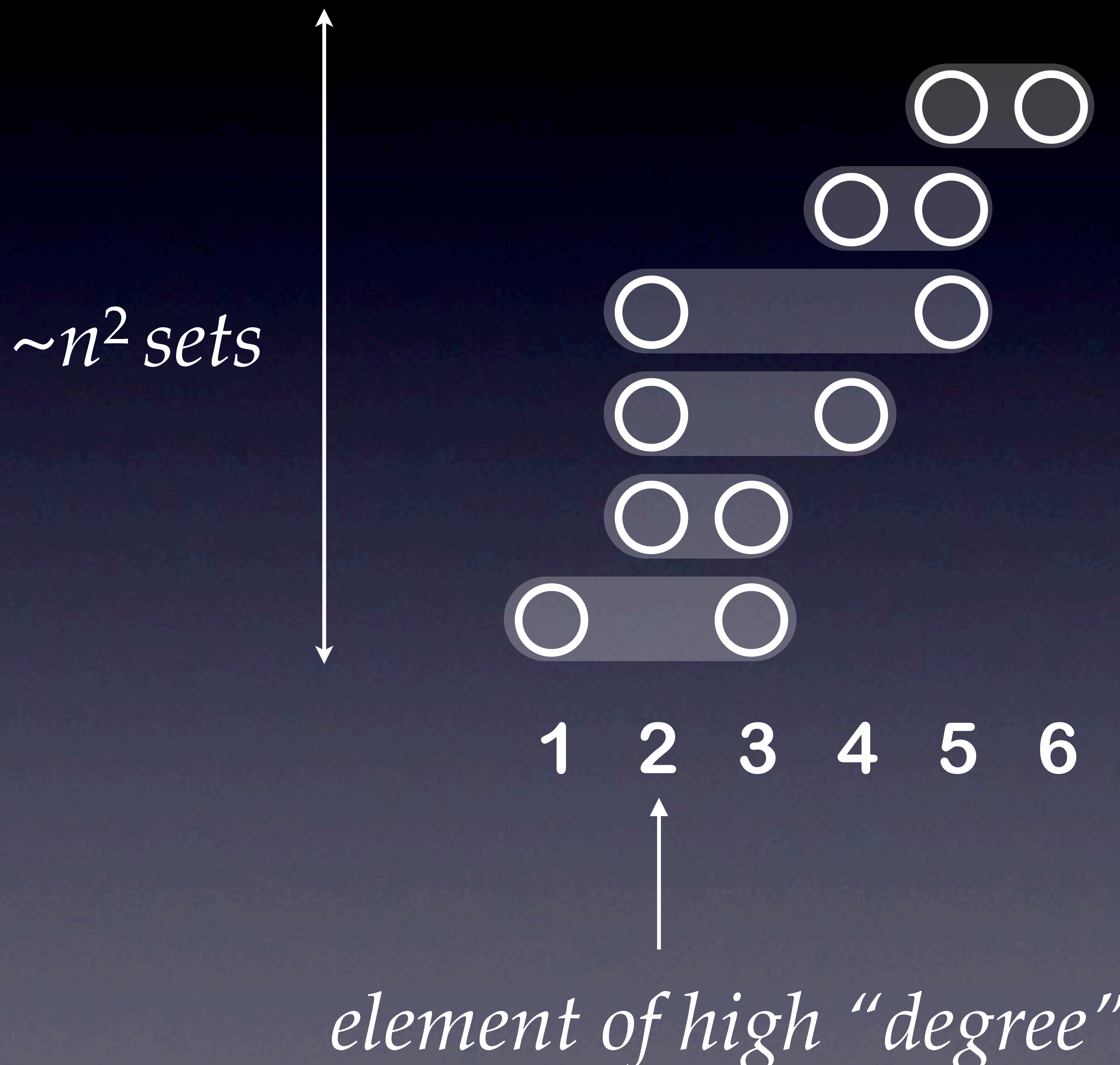
Hitting Set



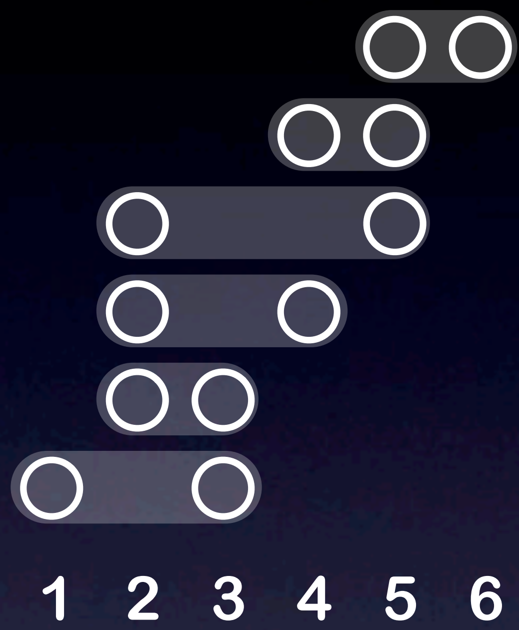
Hitting Set



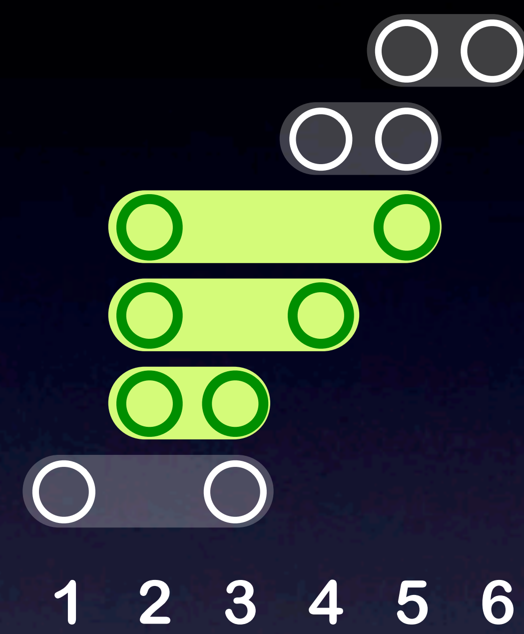
Sparsifying a Hitting Set Instance



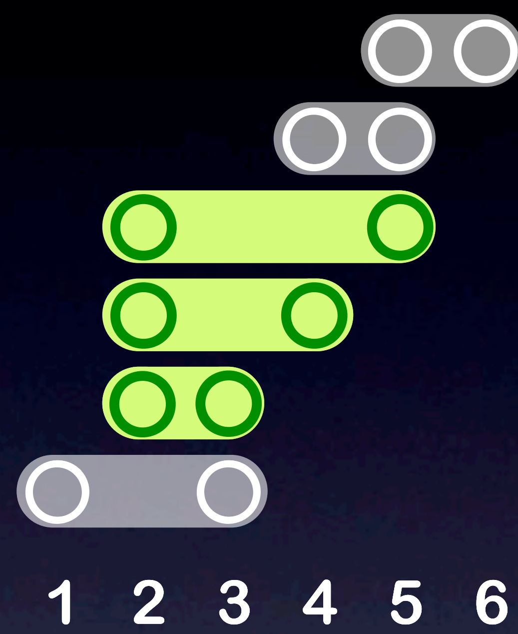
Sparsifying a Hitting Set Instance



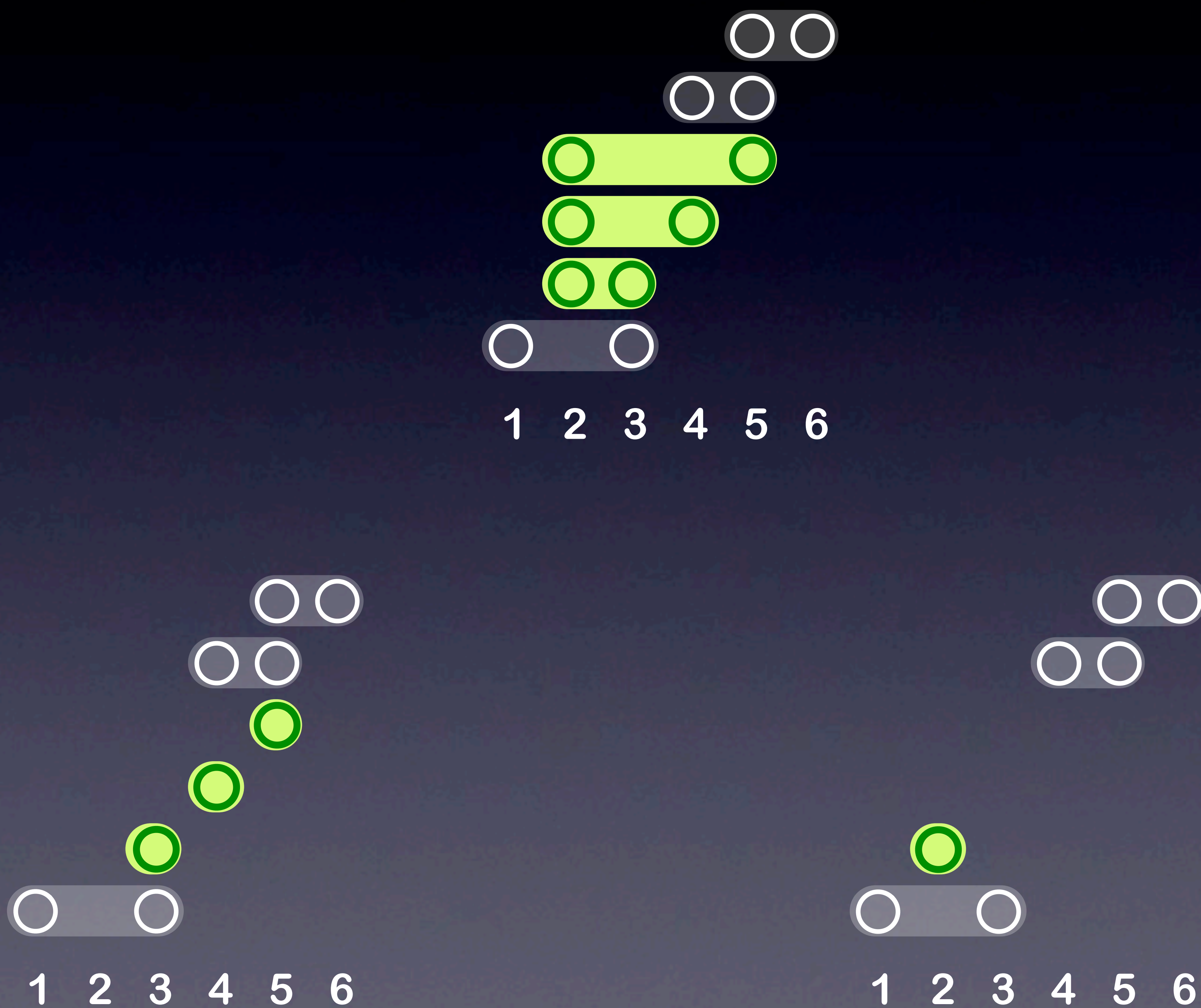
Sparsifying a Hitting Set Instance



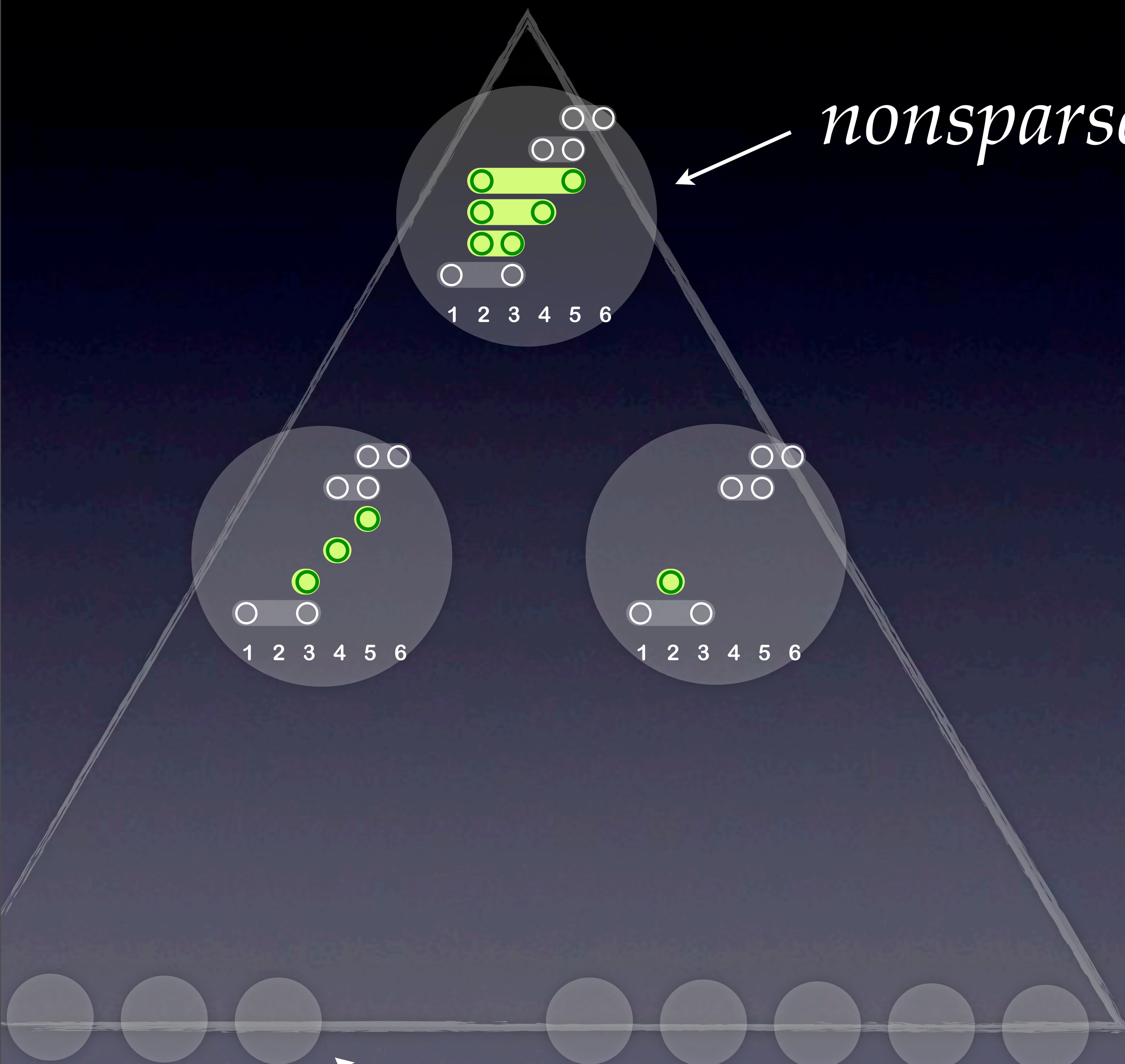
Sparsifying a Hitting Set Instance



Sparsifying a Hitting Set Instance



Sparsifying a Hitting Set Instance



nonsparse instance

high degree

max degree r·n

sparse instances

$\exp(o(n)) \cdot 2^n = \exp(n)$

2^n leaves

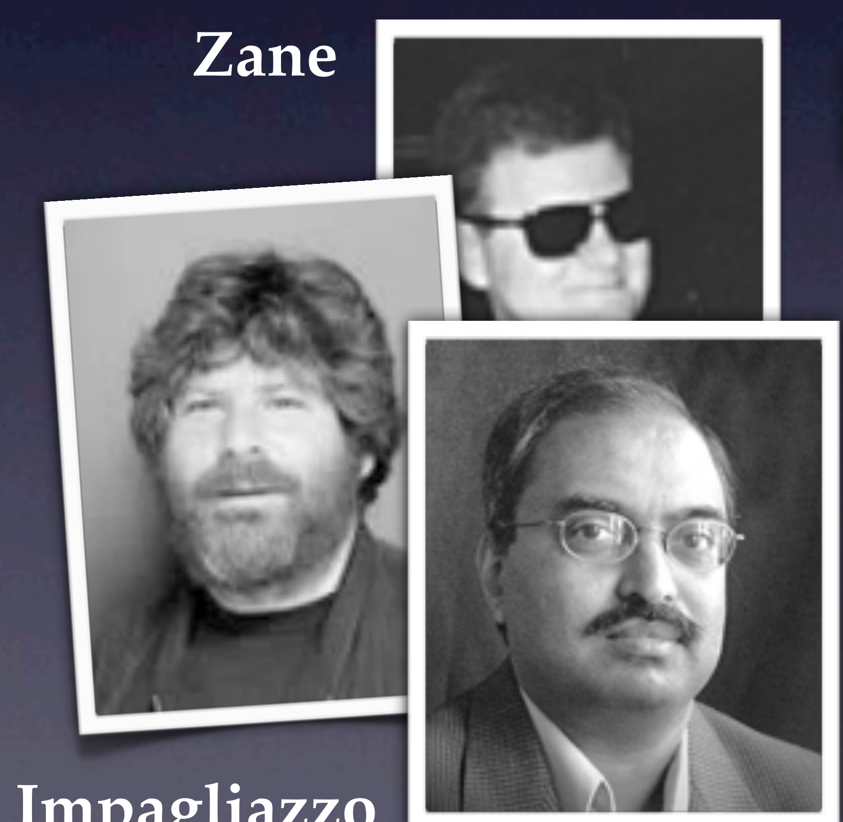
$\exp(o(n)) \cdot \exp(H(1/r)n) =$
$\exp(o(n))$

$C(n,1) + \ldots + C(n,n/r)$
leaves

Exponential Time Hypothesis

Can't do 3-Sat *in time* exp(o(n))

Zane



Can't do 18-Sat *in time* exp(o(m))

Can't do Independent Set *in time* exp(o(n))

Impagliazzo

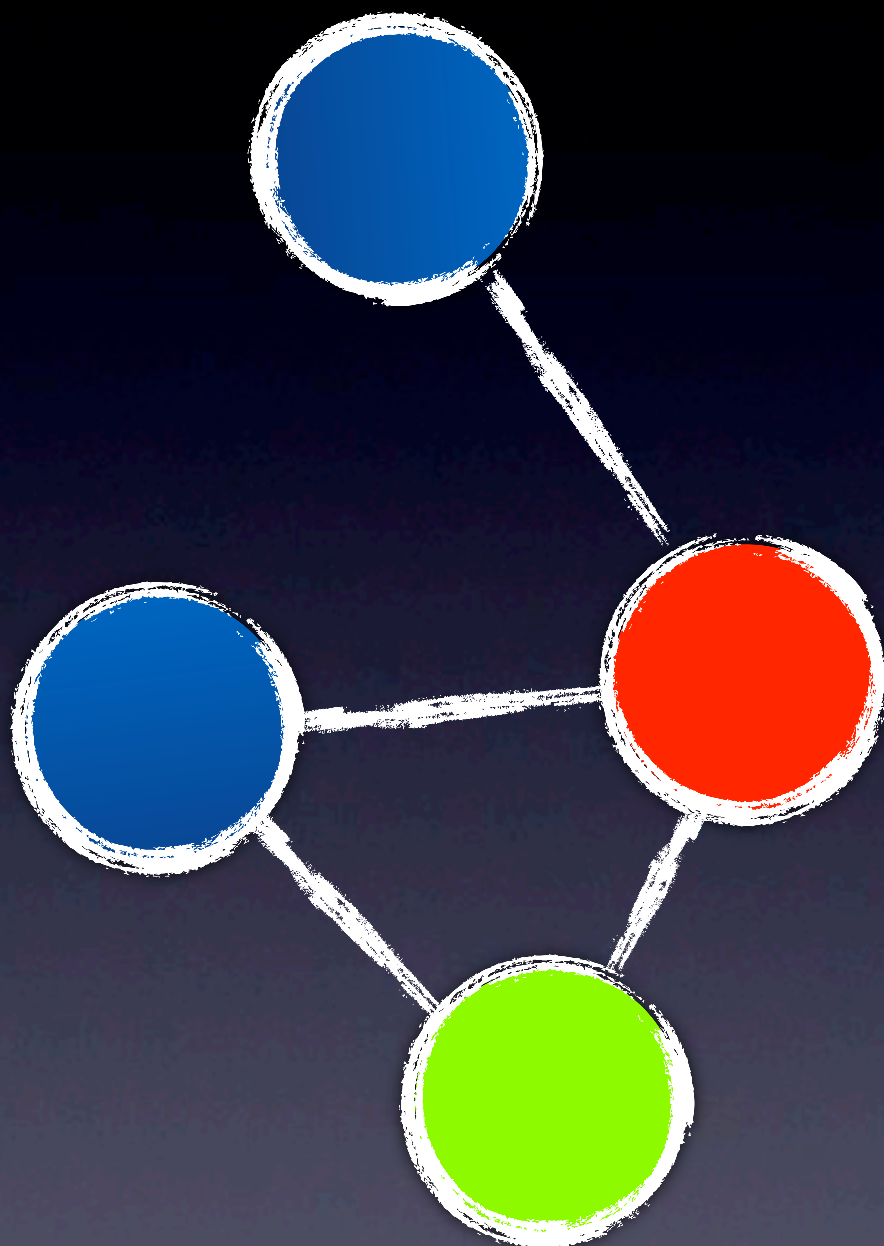
Paturi

Why No Dependency on # Colours is Surprising

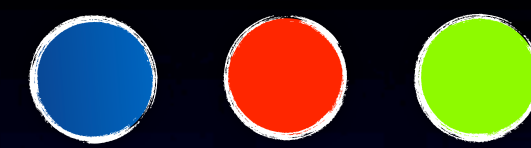
CSP(q,2)

q states, pairwise constraints

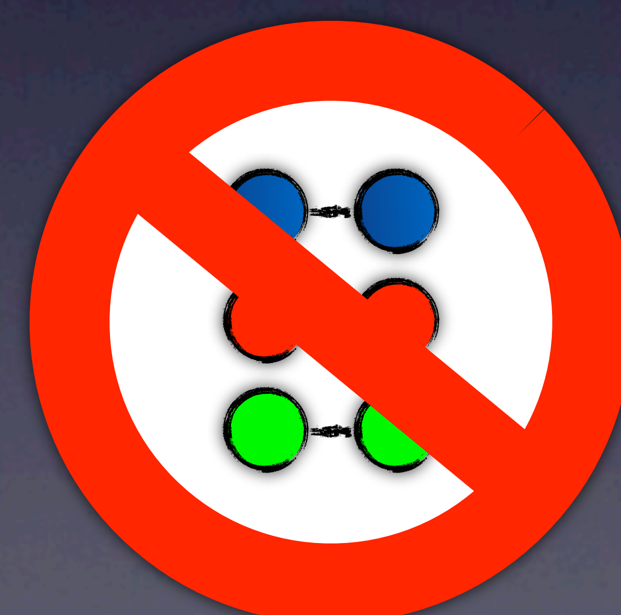
Traxler

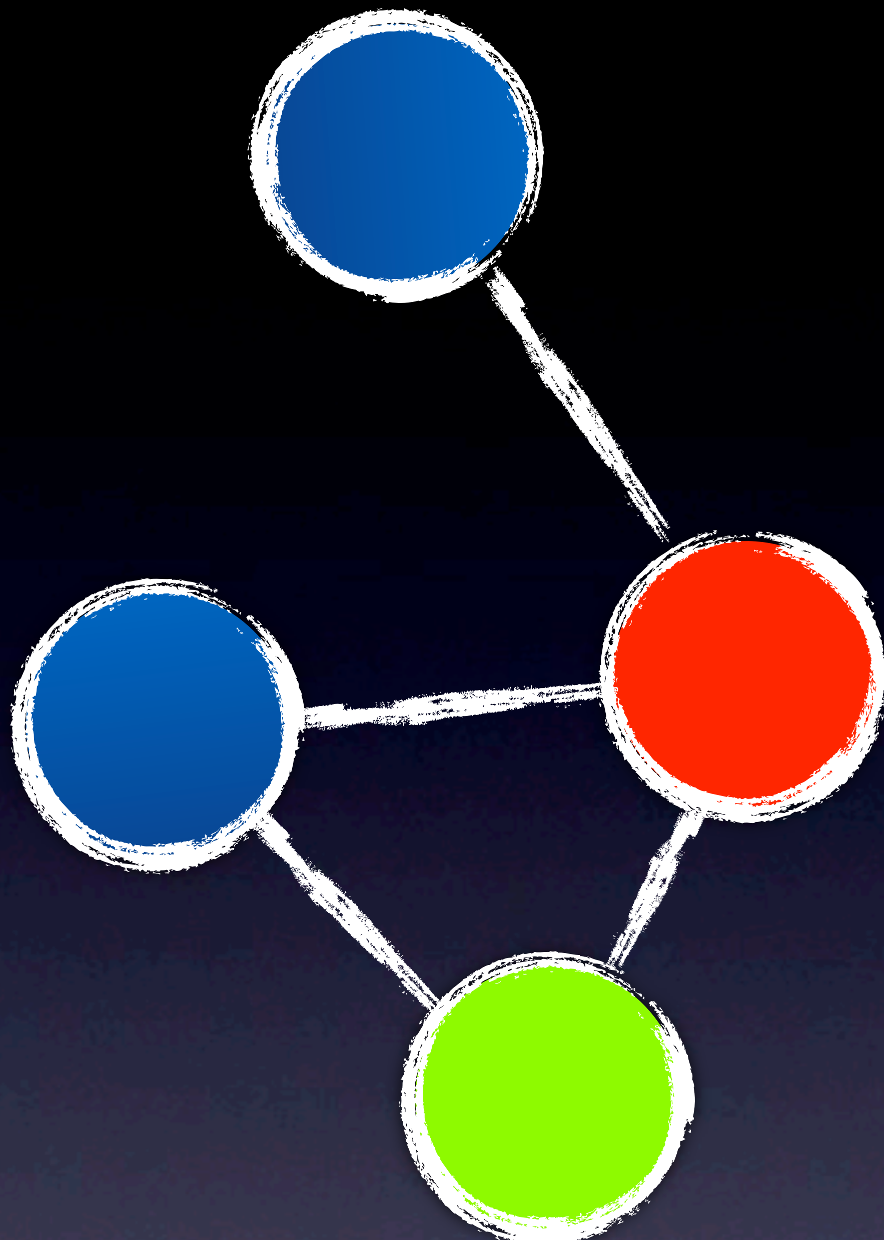


states:

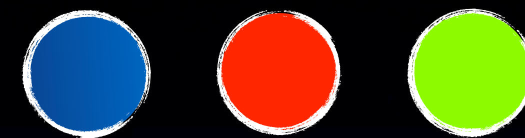


constraints

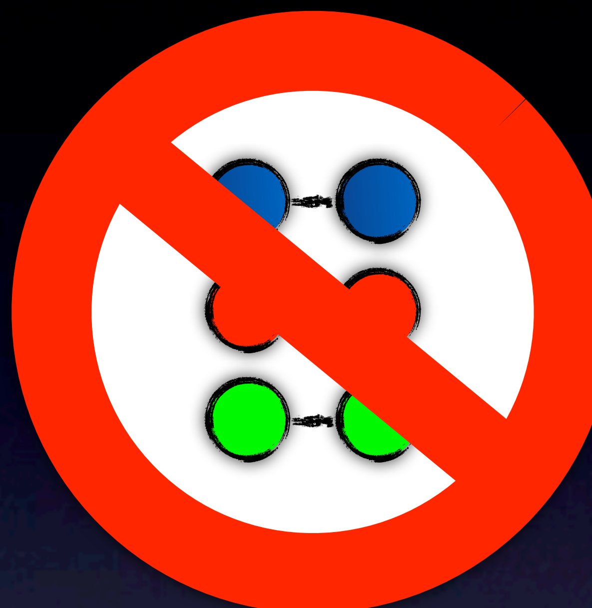


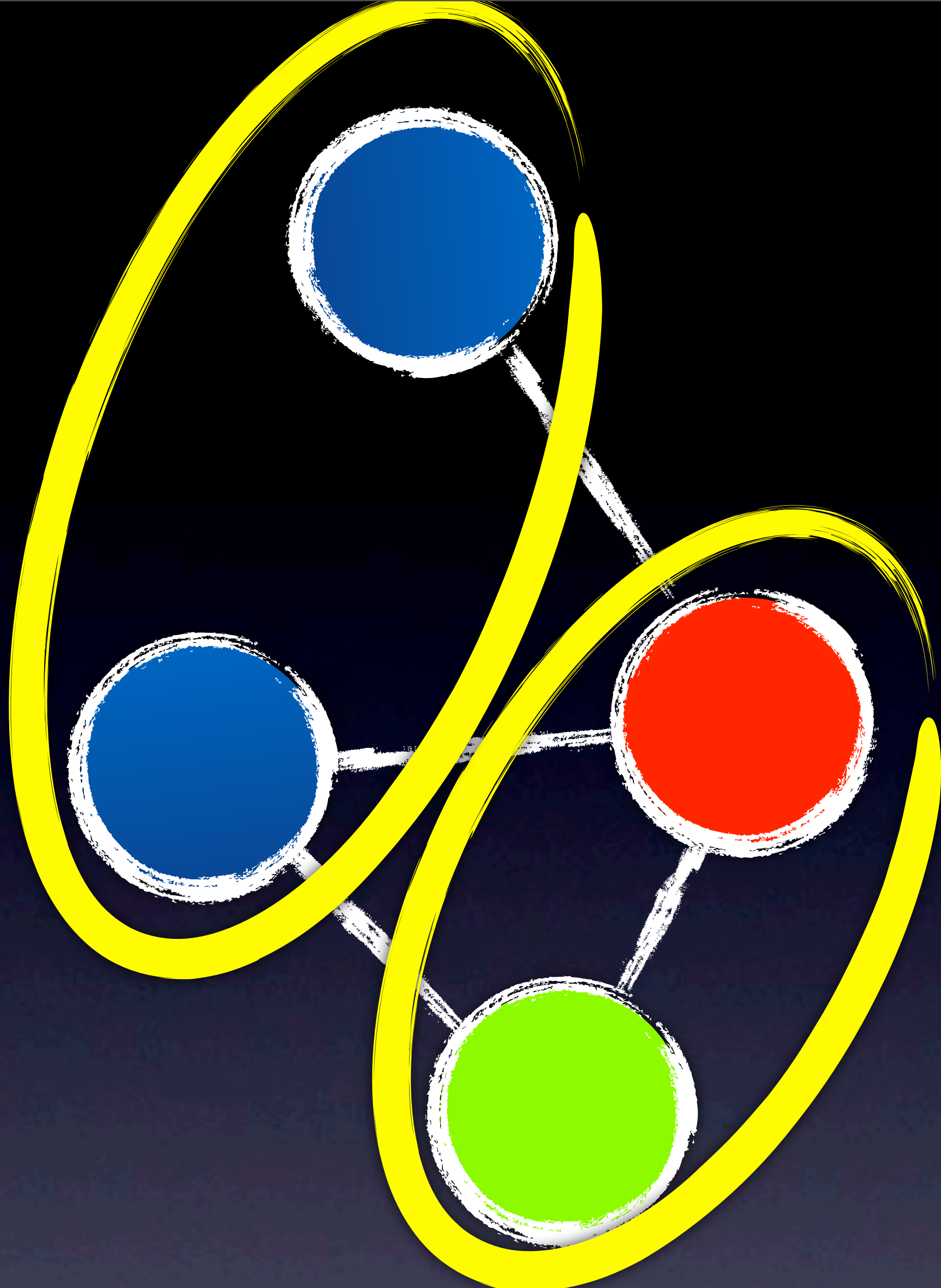


states:

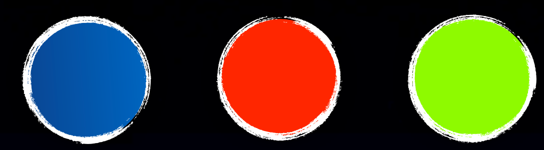


constraints

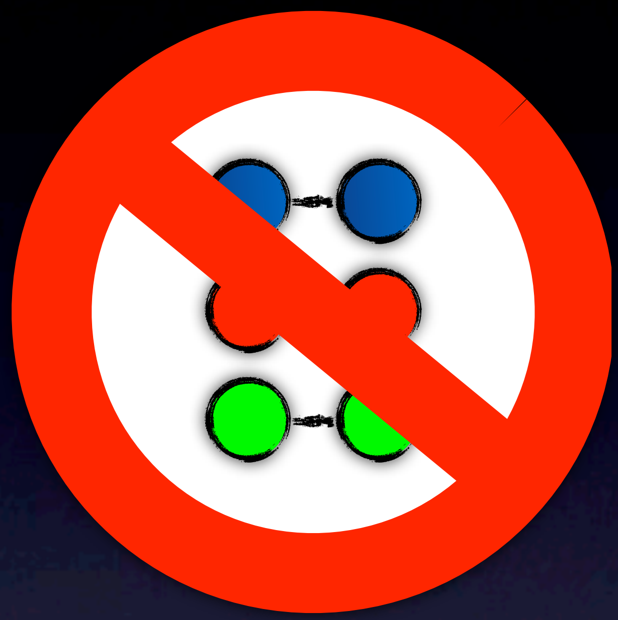


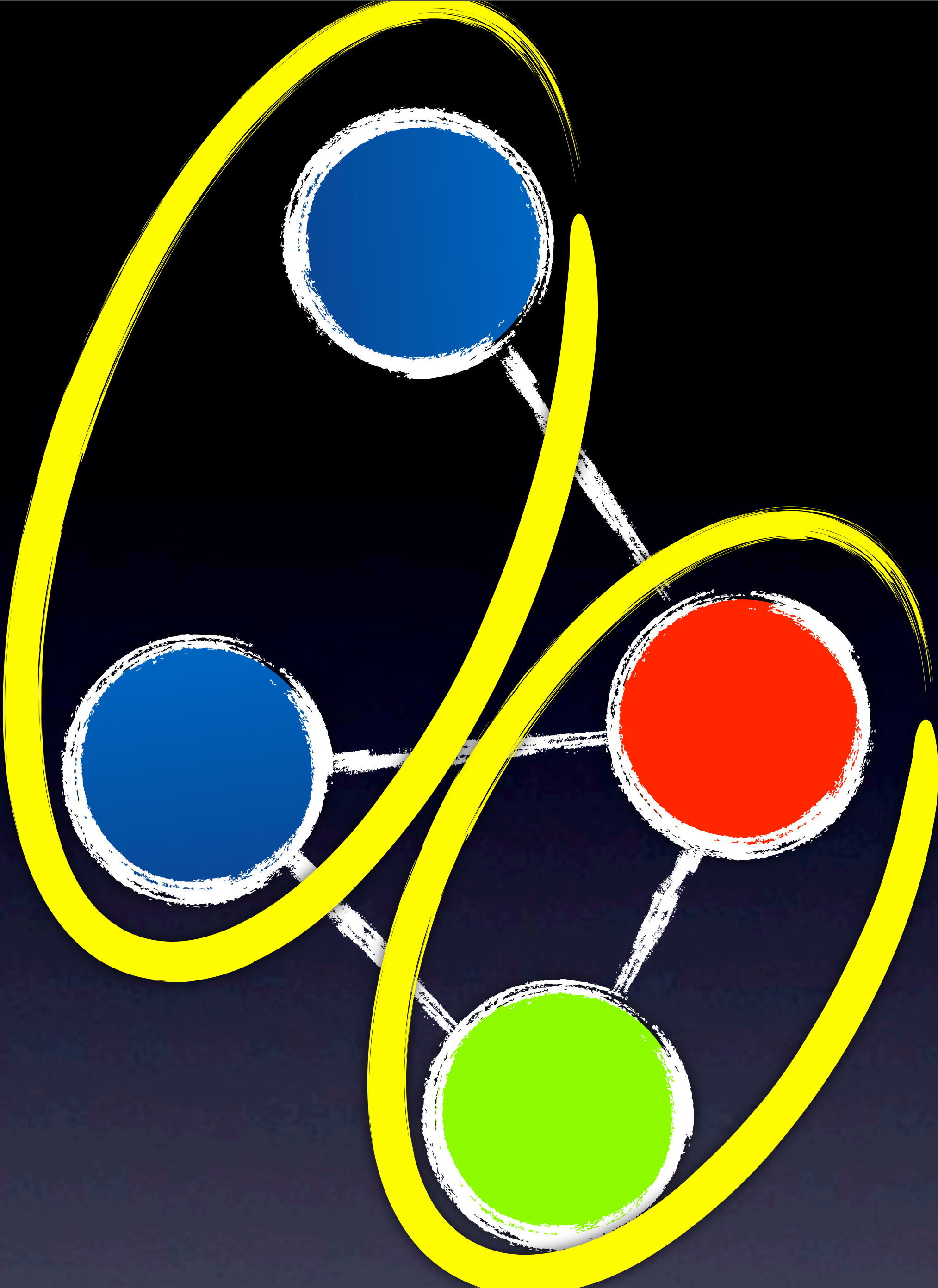


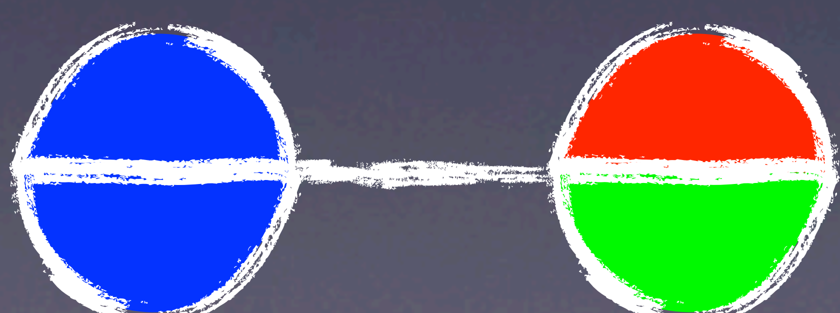
states:



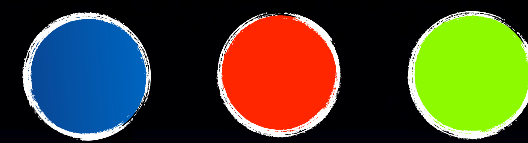
constraints



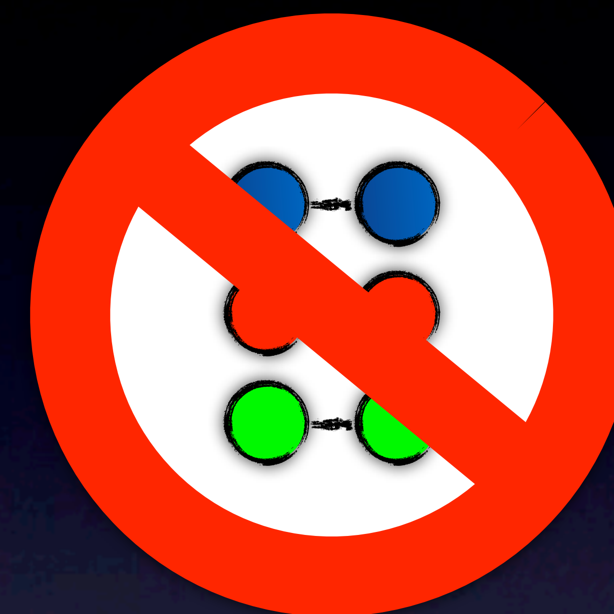


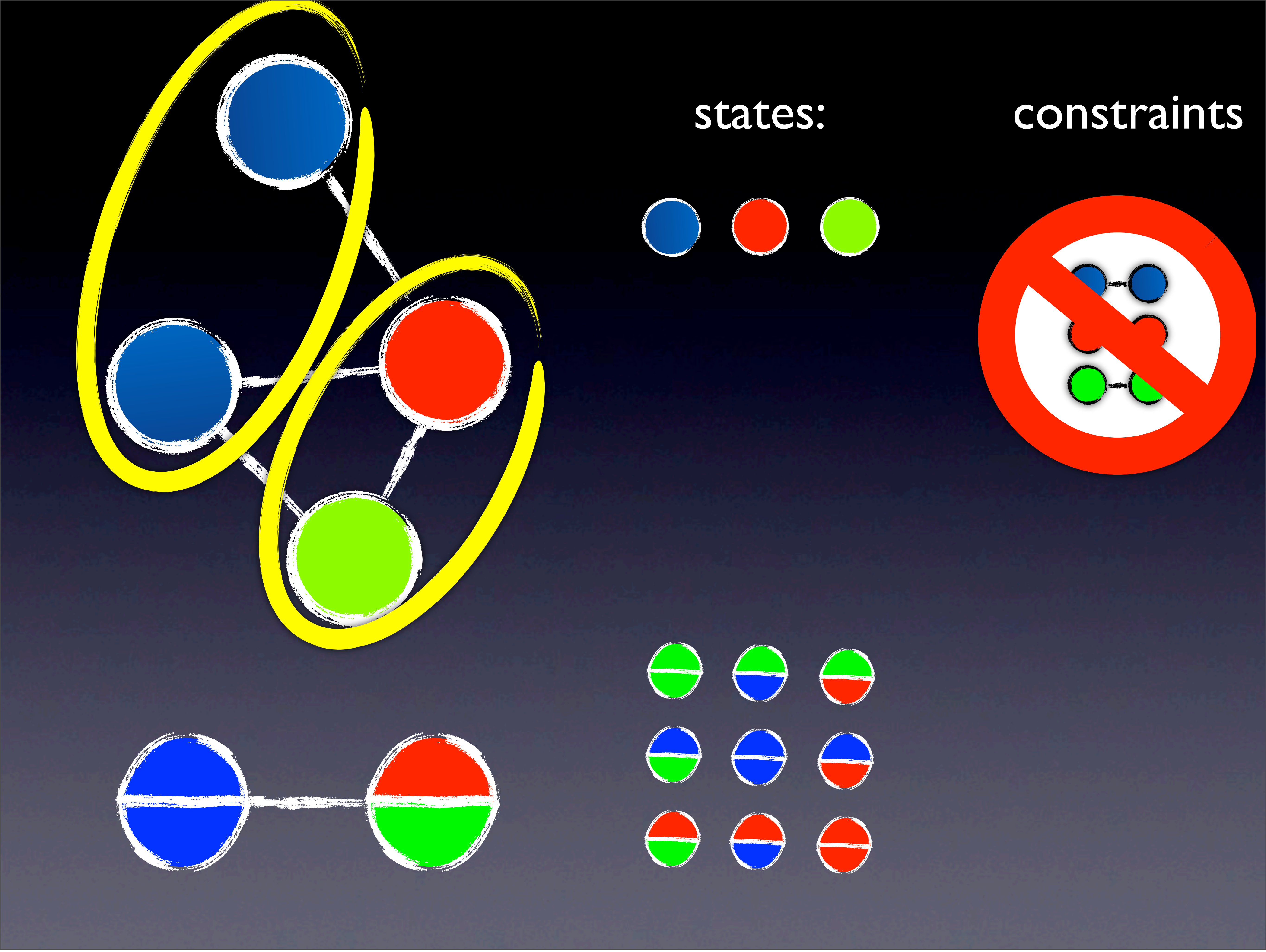


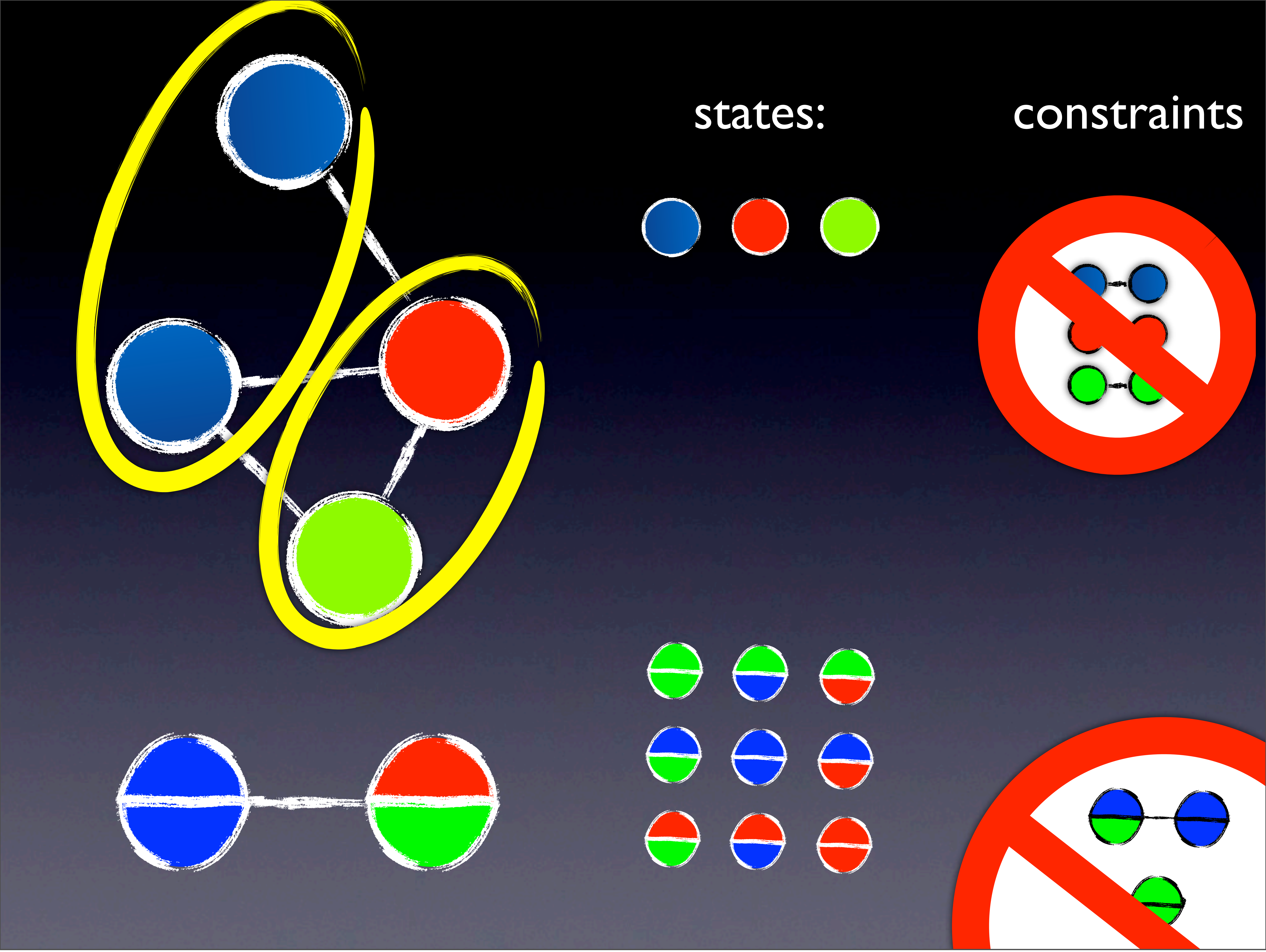
states:



constraints




states:
constraints


states:
constraints

n verts

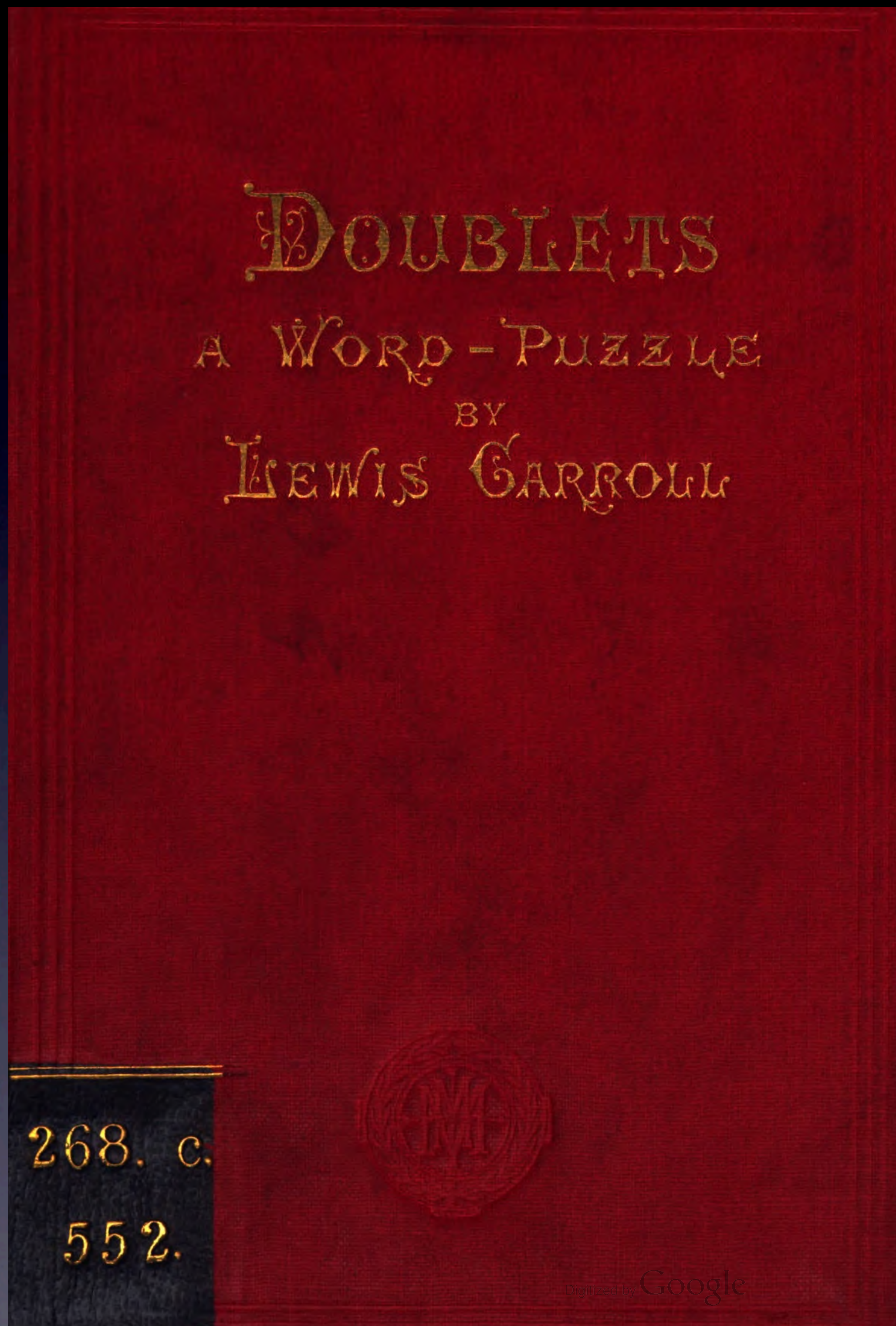
d states

n/2 verts

d^2 states

Must have $d^n = (d^2)^{n/2}$

Path through specified vertices

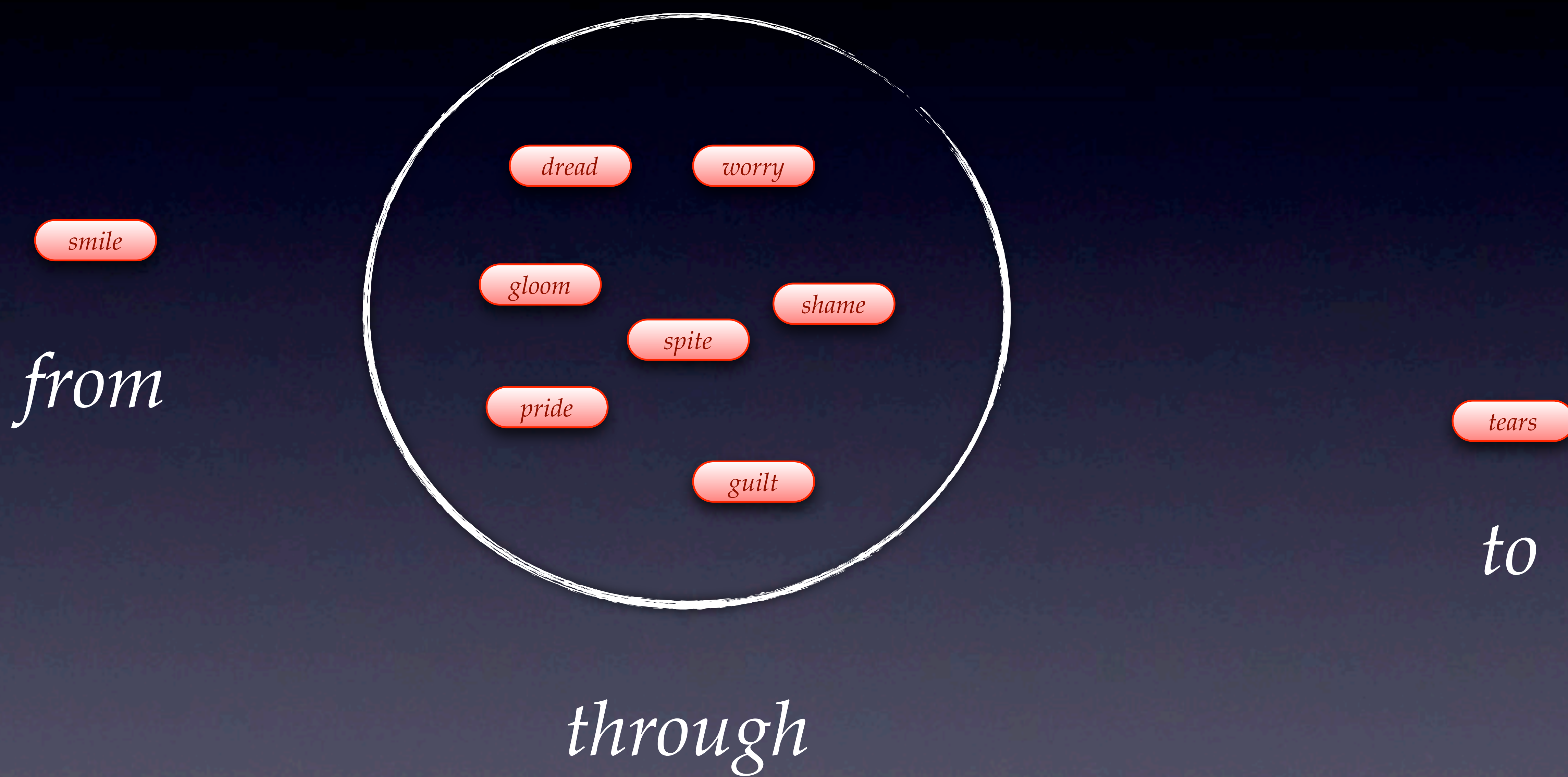


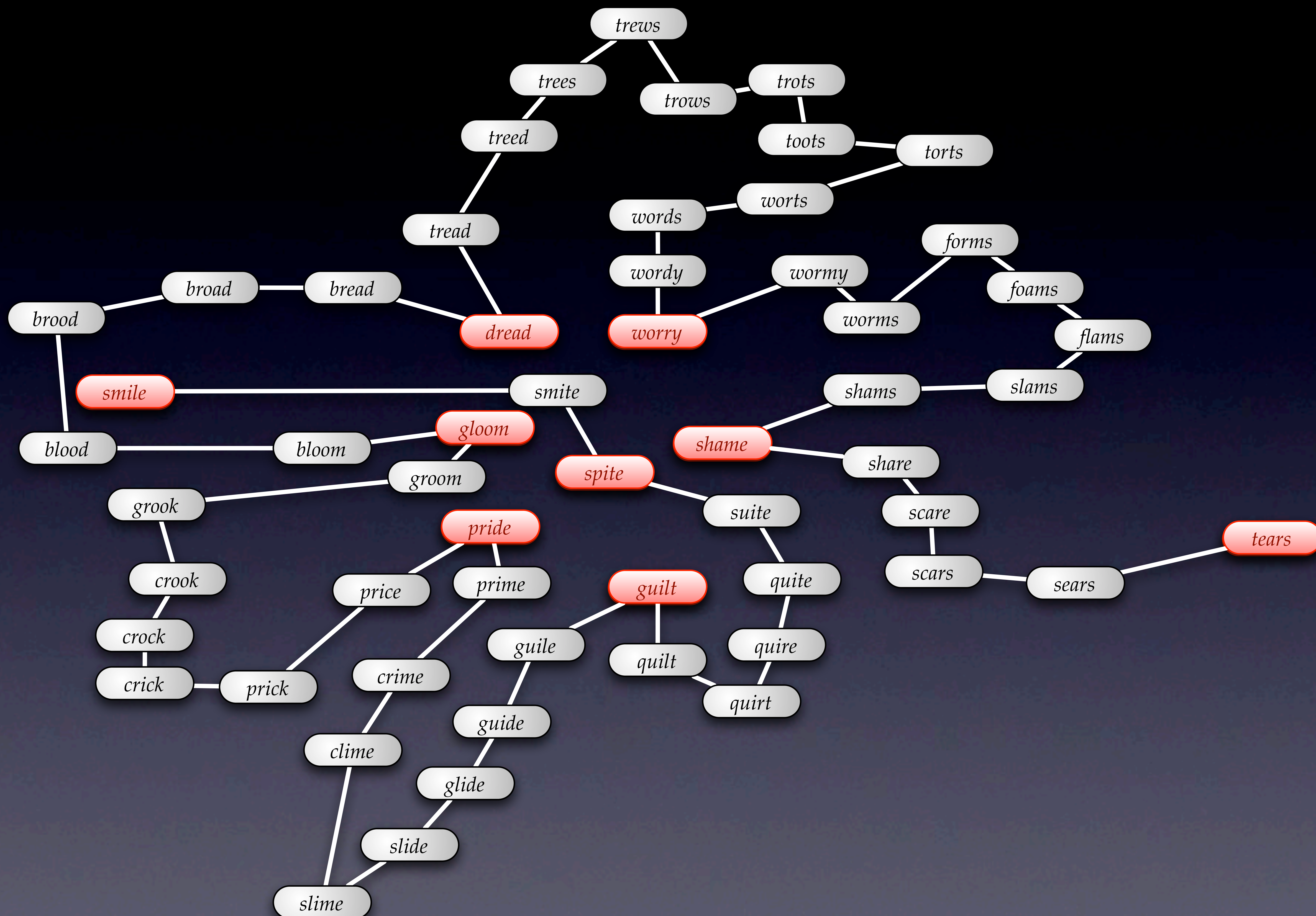
DEAR VANITY,—Just a year ago last Christmas, two young ladies —smarting under that sorest scourge of feminine humanity, the having "nothing to do"—besought me to send them "some riddles." But riddles I had none at hand, and therefore set myself to devise some

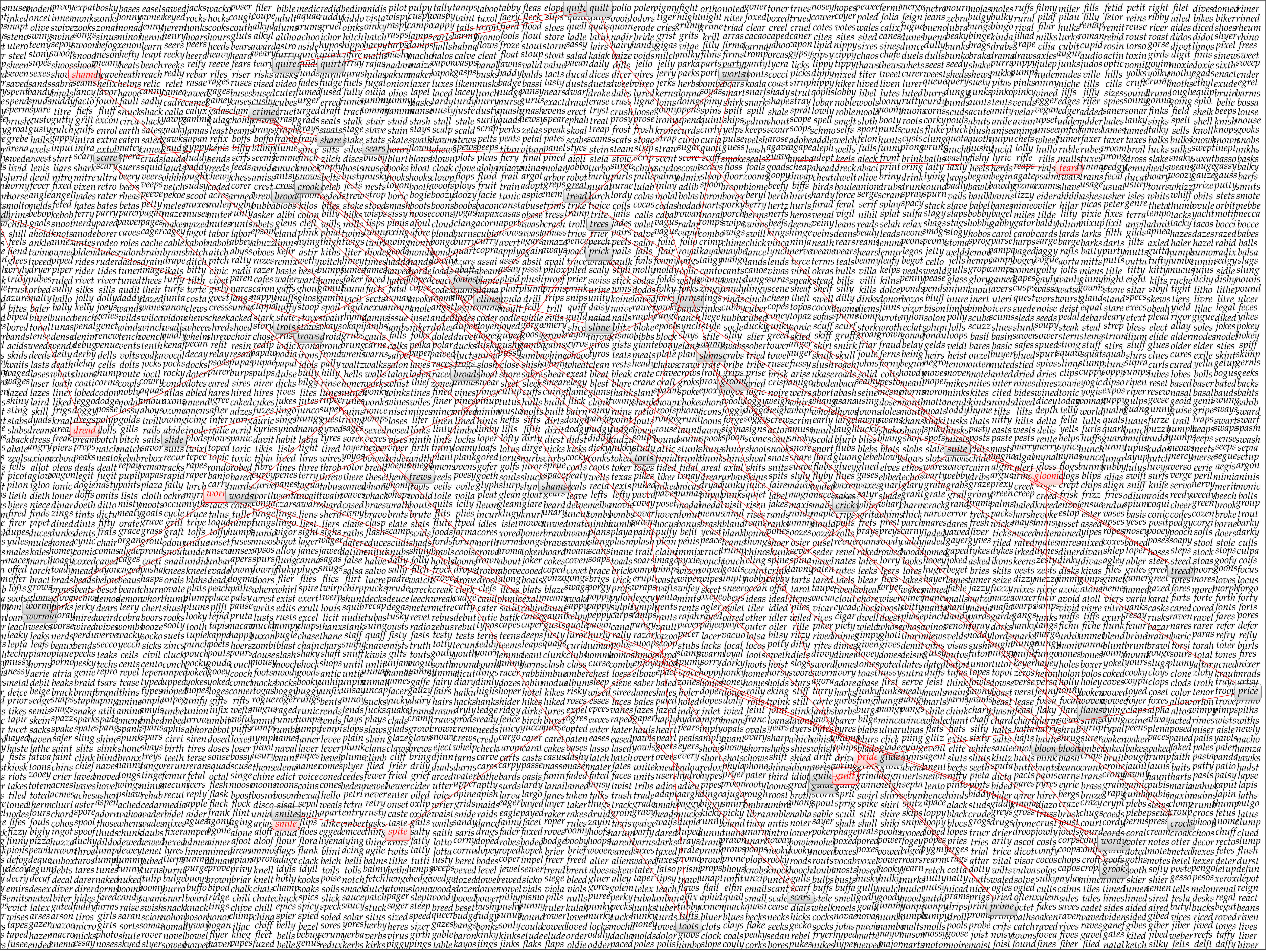
HEAD
h e a l
t e a l
t e l l
t a l l
TAIL

April 5.—Dip PEN into INK.
Touch CHIN with NOSE.
Change TEARS into SMILE.

Path through specified vertices

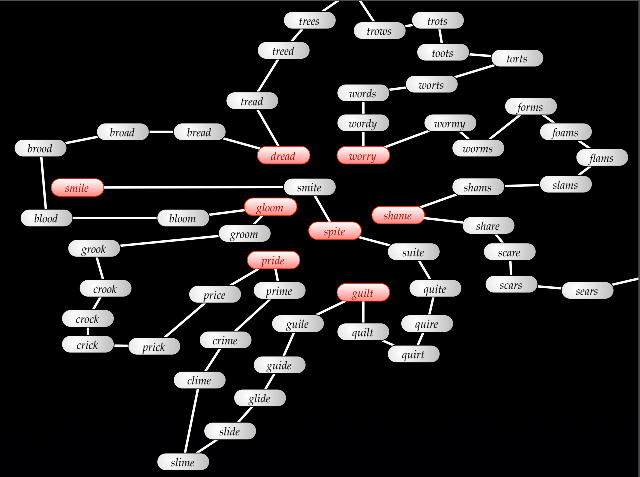



trews
trees
trows
trots
treed
toots
torts
worts
words
tread
forms
wordy
wormy
broad
bread
foams
brood
dread
worry
worms
flams
smile
smite
shams
slams
gloom
blood
bloom
shame
groom
share
spite
grook
suite
scare
pride
tears
crook
price
prime
guilt
quite
scars
sears
crock
guile
quilt
quire
crick
prick
crime
quirt
guide
clime
glide
slide
slime

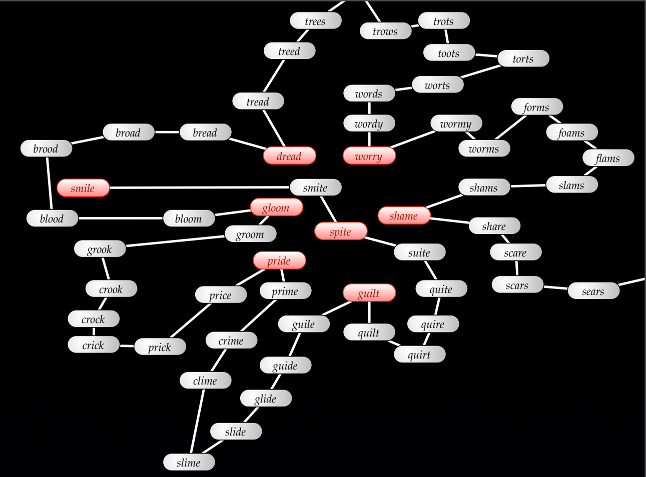


onsdag 12 oktober 11

k specified vertices, n vertices

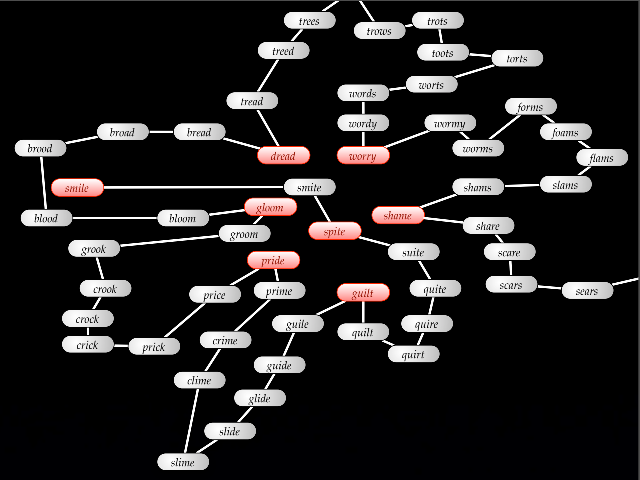


k specified vertices, n vertices



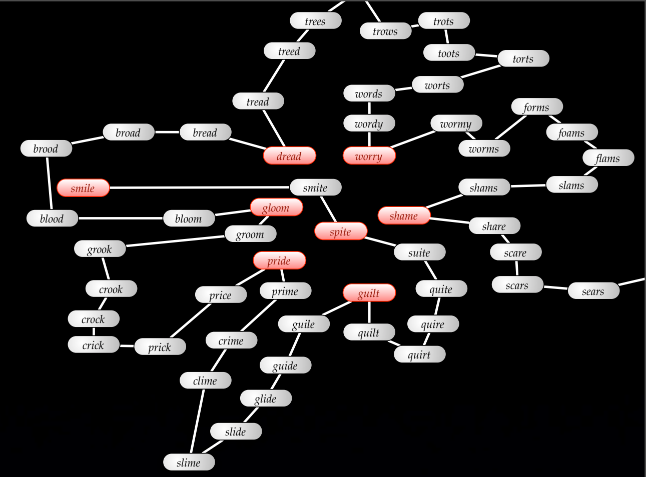
- $k=n$: Hamilton path

k specified vertices, n vertices

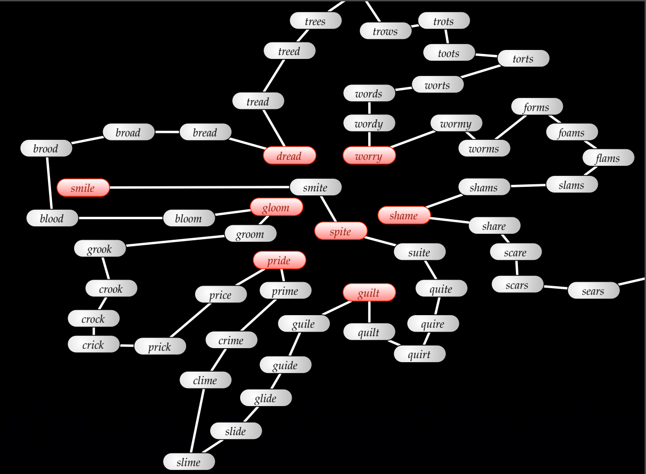


- $k=n$: Hamilton path
- no poly(k)-algorithm under P vs NP

k specified vertices, n vertices

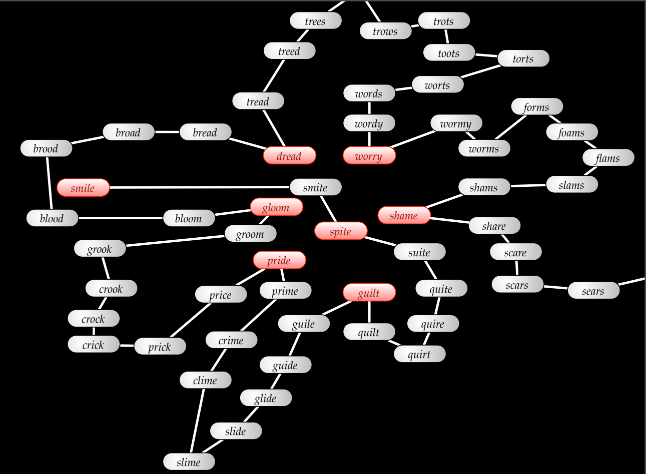


- $k=n$: Hamilton path
- no poly(k)-algorithm under P vs NP
- no exp($o(k)$)-algorithm under ETH



k specified vertices, n vertices

- $k=n$: Hamilton path
- no poly(k)-algorithm under P vs NP
- no exp($o(k)$)-algorithm under ETH
- Brute force: $O(n!)$ (note: not n^k)

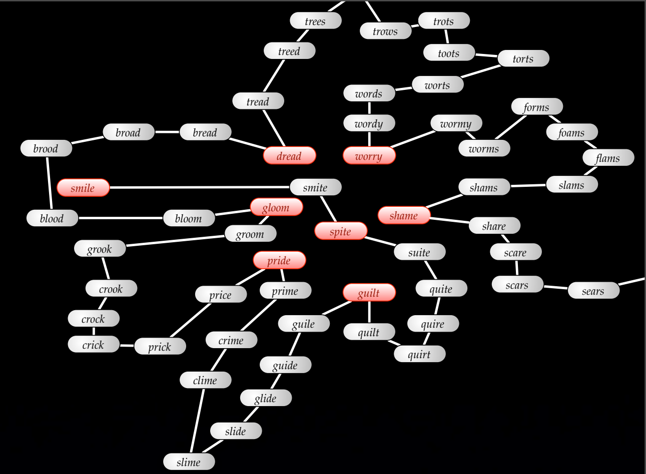


k specified vertices, n vertices

- $k{=}n$: Hamilton path
- no poly(k)-algorithm under P vs NP
- no exp($o(k)$)-algorithm under ETH
- Brute force: $O(n!)$ (note: not n^k)
- Disjoint paths: $f(k)\cdot\text{poly}(n)$

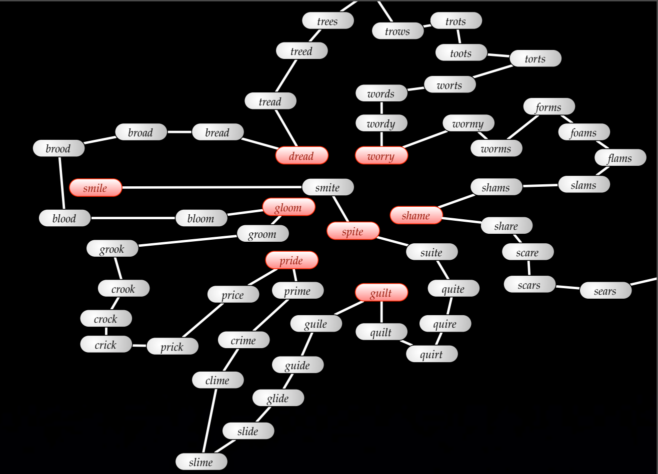
Robertson–Seymour

k specified vertices, n vertices



- $k=n$: Hamilton path
- no poly(k)-algorithm under P vs NP
- no exp($o(k)$)-algorithm under ETH
- Brute force: $O(n!)$ (note: not n^k)
- Disjoint paths: $f(k)\cdot\text{poly}(n)$
- Algorithms for k=1,2,3.

Robertson–Seymour



k specified vertices, n vertices

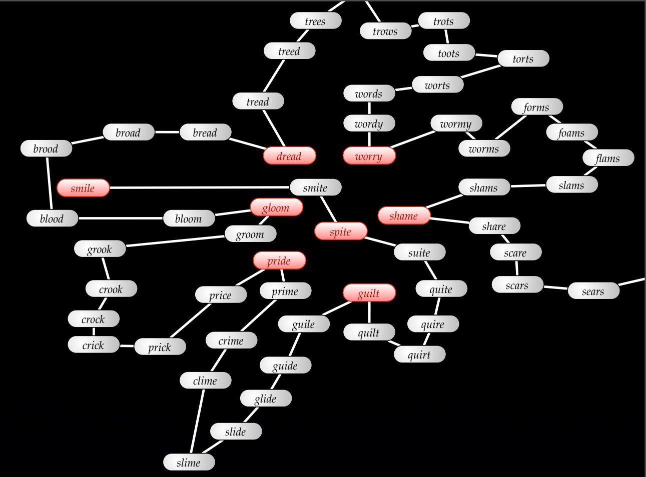
- $k=n$: Hamilton path
- no poly(k)-algorithm under P vs NP
- no exp($o(k)$)-algorithm under ETH

- Brute force: $O(n!)$ (note: not n^k)
- Disjoint paths: $f(k)\cdot\text{poly}(n)$
- Algorithms for k=1,2,3.
- Algorithm in $\exp(\exp(k^{10}))$

Robertson–Seymour

Kawarabayashi

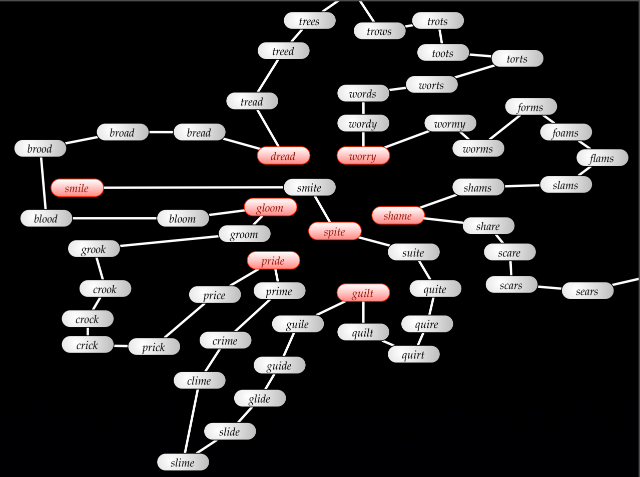
k specified vertices, n vertices



New result [SODA12]:
randomised algorithm in time
$\exp(k)\mathrm{poly}(n)$

Taslaman **Björklund**

k specified vertices, n vertices

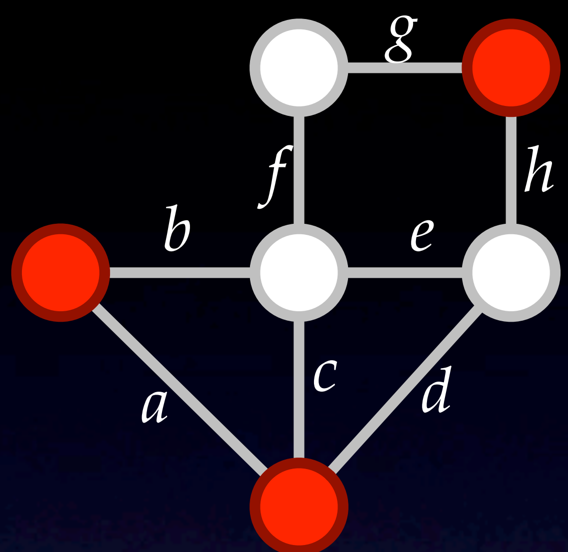


New result [SODA12]:
randomised algorithm in time
$\exp(k)\text{poly}(n)$

Thm: Shortest(!) cycle through k given vertices or edges in time $2^k\text{poly}(n)$ with exponentially small one-sided error.

Taslaman **Björklund**

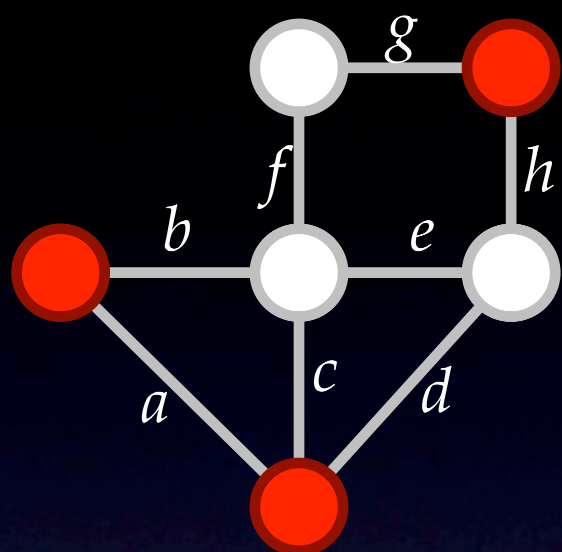
Trick: Look at Polynomials Instead



Koutis

Williams

Trick: Look at Polynomials Instead

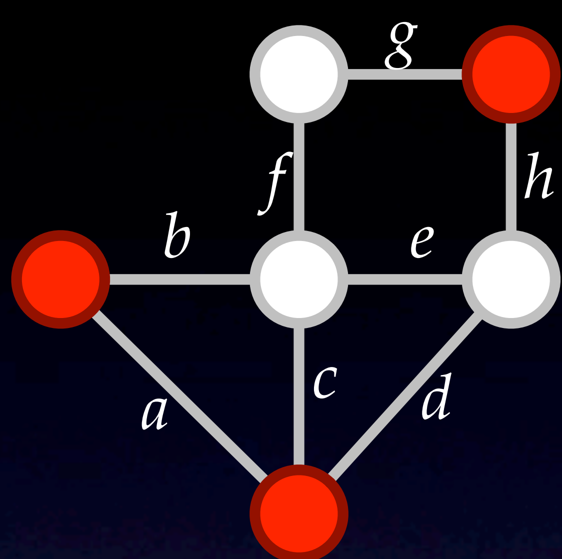


Koutis

Williams

Björklund *et al.*

Trick: Look at Polynomials Instead



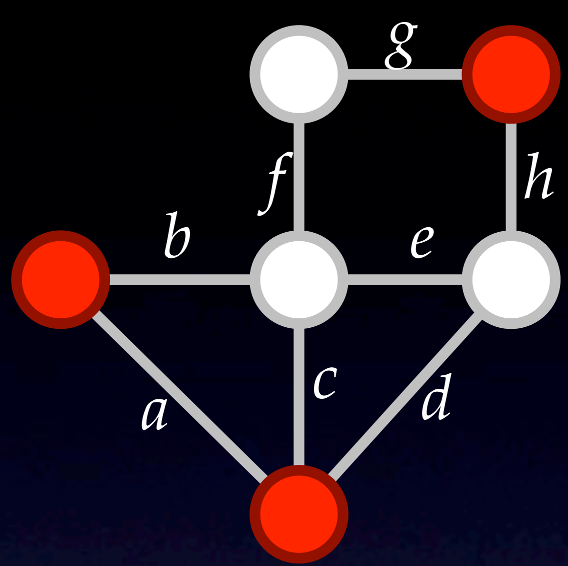
Koutis

Williams

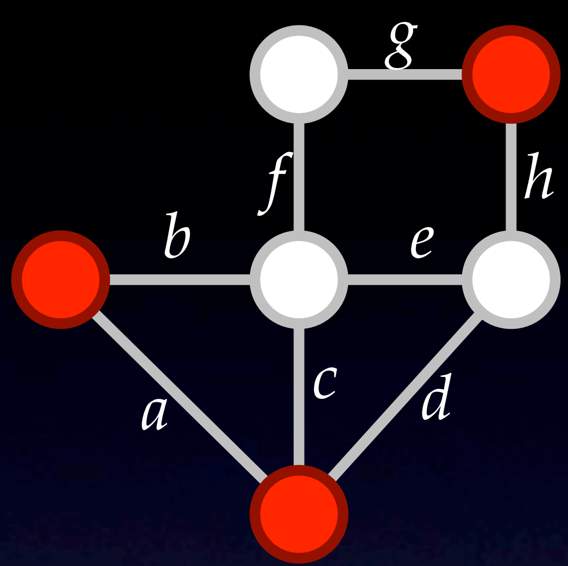
Björklund *et al.*

Tutte

Trick: Look at Polynomials Instead

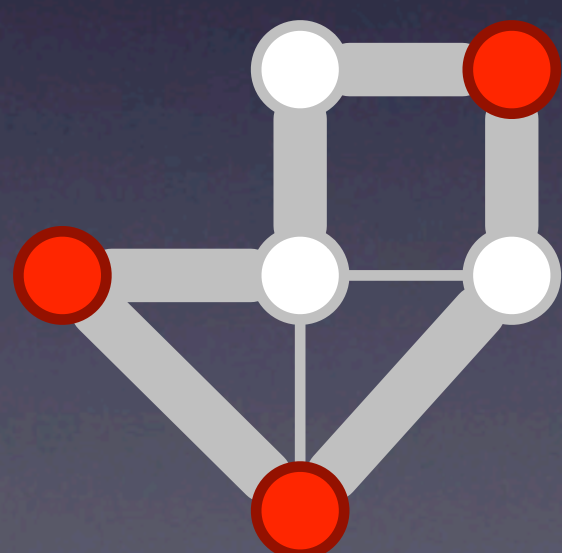


Trick: Look at Polynomials Instead

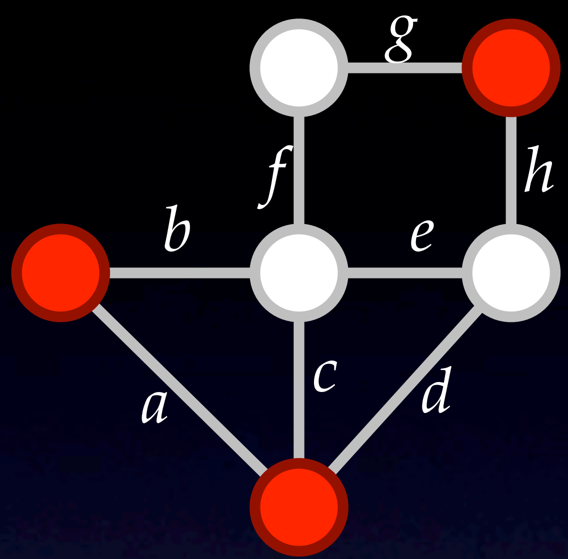


monomial for every walk

$a \cdot b \cdot f \cdot g \cdot h \cdot d$



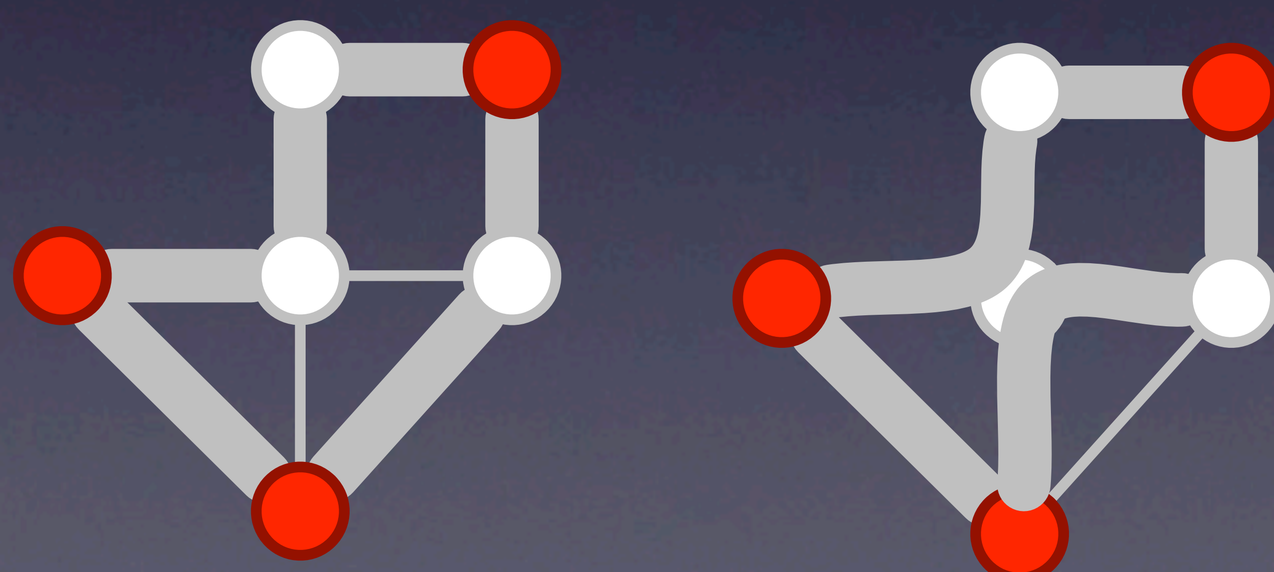
Trick: Look at Polynomials Instead



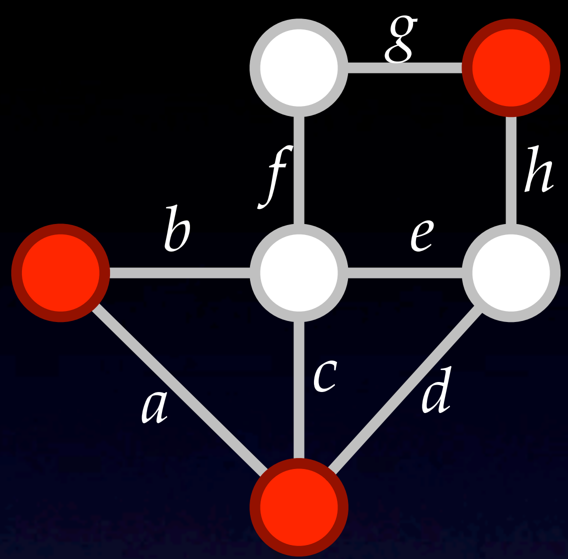
monomial for every walk

sum over all walks

$a \cdot b \cdot f \cdot g \cdot h \cdot d$ $\qquad$ $a \cdot b \cdot c \cdot e \cdot f \cdot g \cdot h$

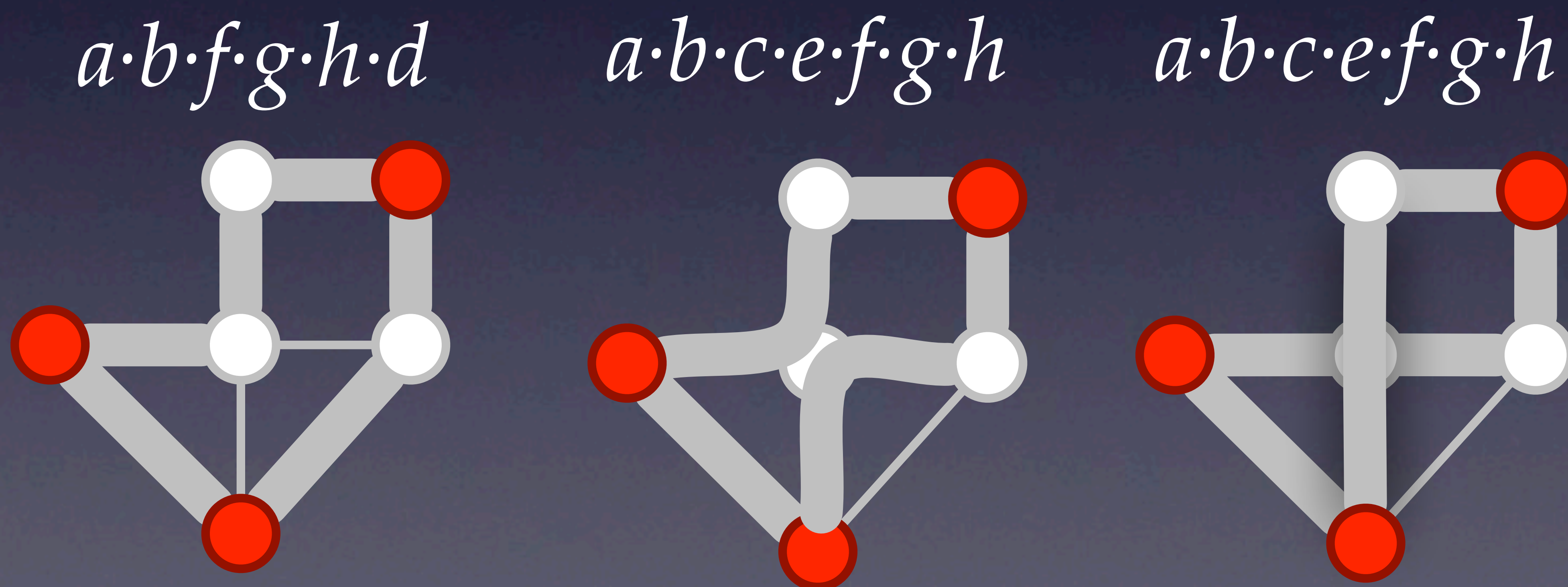


Trick: Look at Polynomials Instead

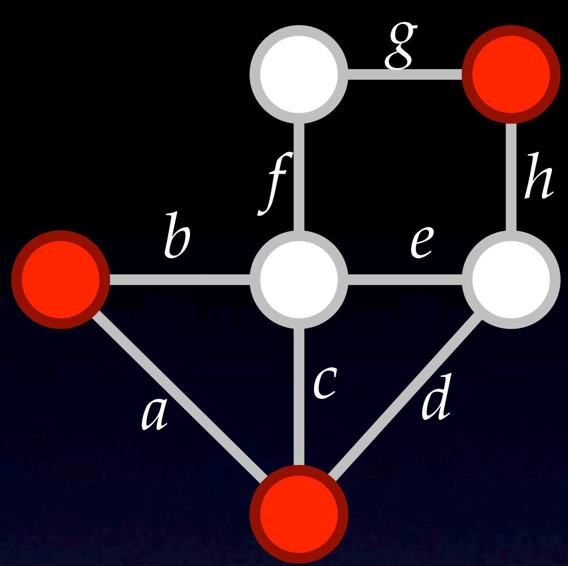


monomial for every walk

sum over all walks



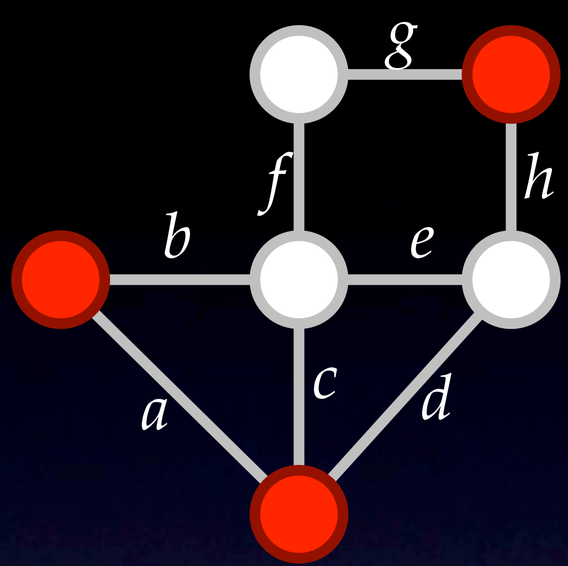
Trick: Look at Polynomials Instead



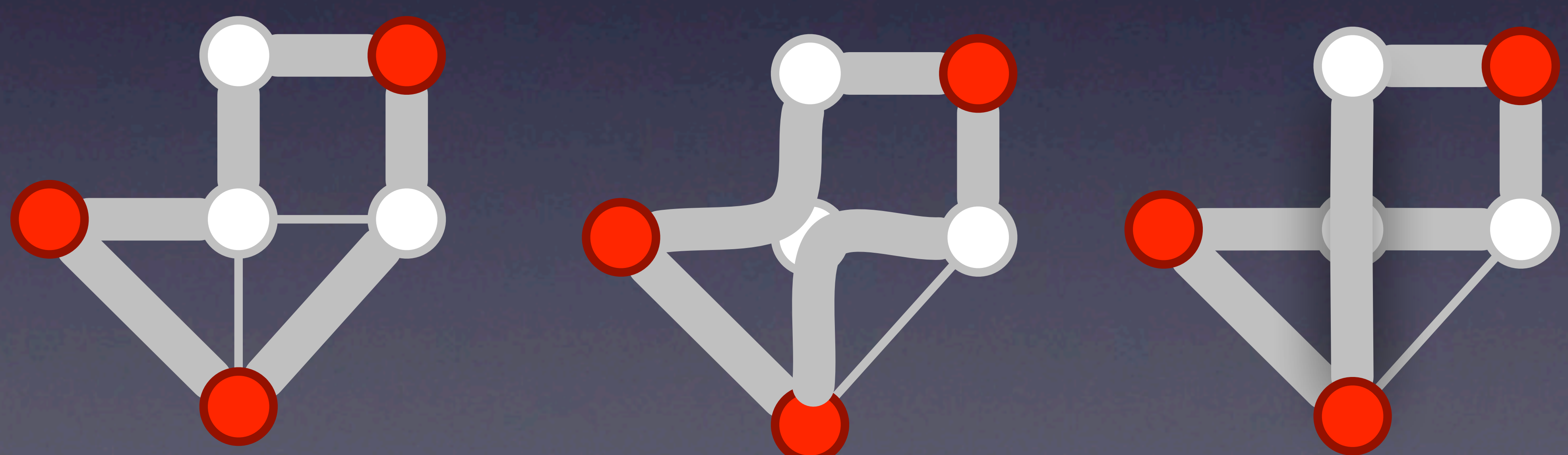
monomial for every walk

sum over all walks mod 2

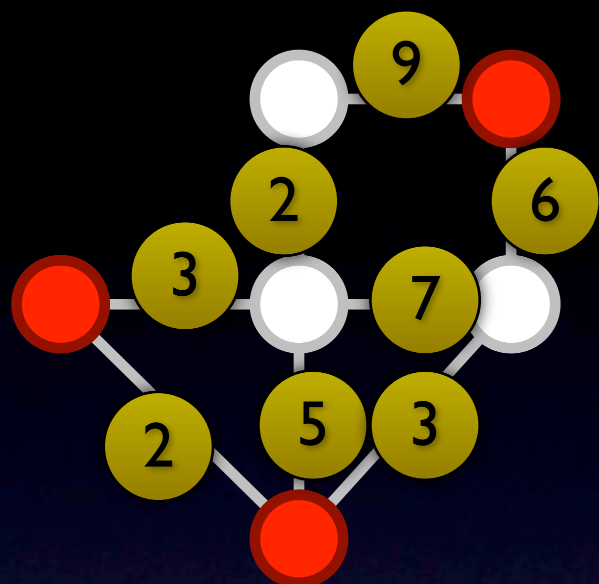
Trick: Look at Polynomials Instead



(Not really. Look at random numbers and *interpret* them as polynomial evaluations.)

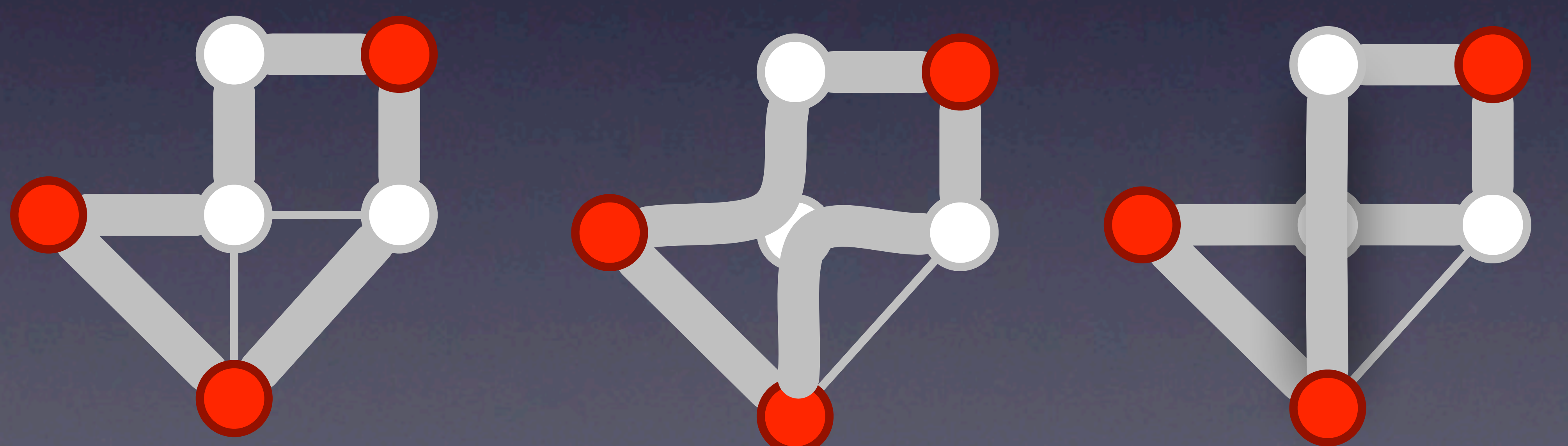


Trick: Look at Polynomials Instead

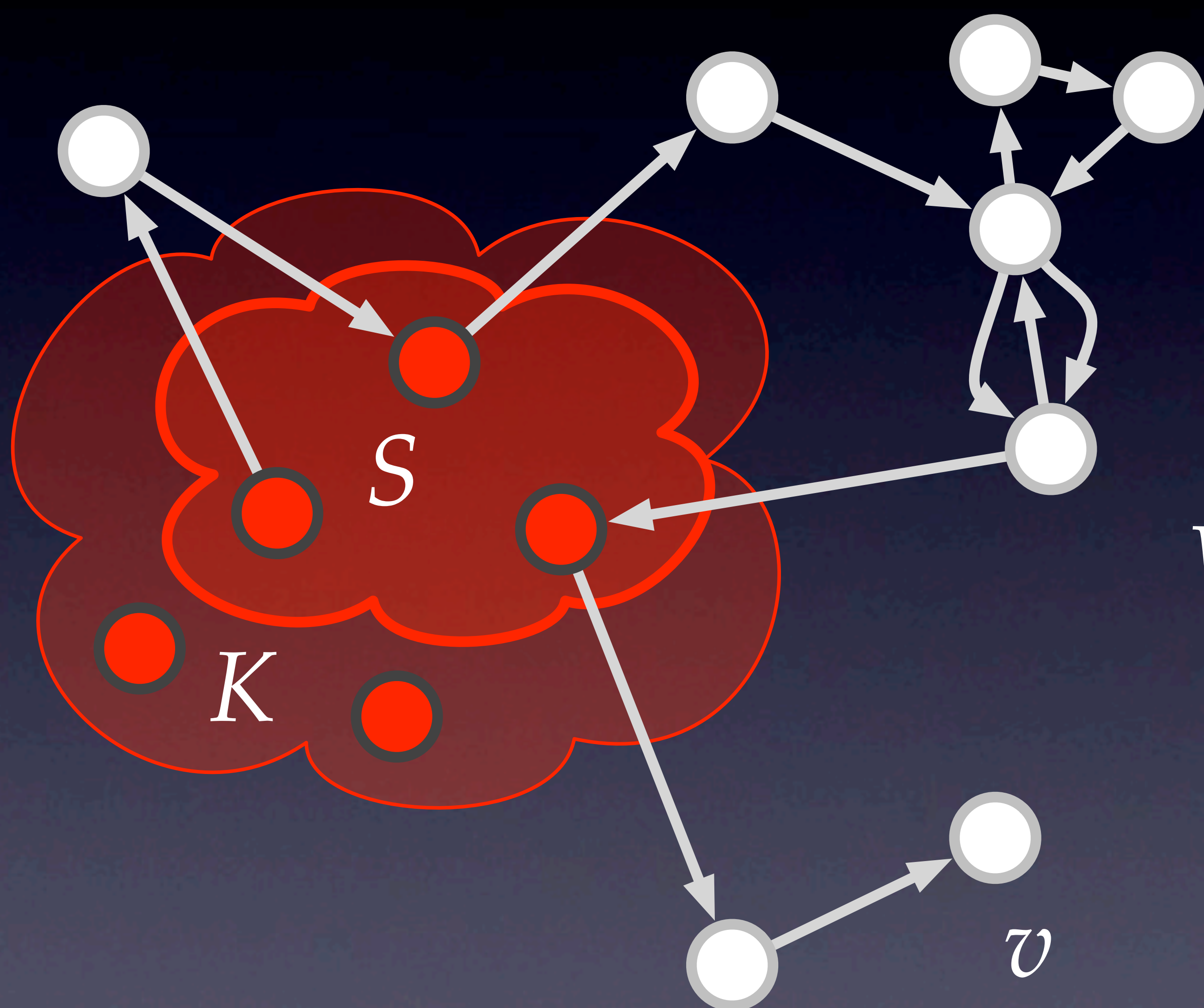


(Not really. Look at random numbers and *interpret* them as polynomial evaluations.)

$3{\cdot}2{\cdot}9{\cdot}6{\cdot}3{\cdot}2$ $\quad$ $3{\cdot}2{\cdot}9{\cdot}6{\cdot}7{\cdot}5{\cdot}2$ $\quad$ $3{\cdot}7{\cdot}6{\cdot}9{\cdot}2{\cdot}5{\cdot}2$



Constructing all Walks: Dynamic Programming for Sequencing Problems



K = specifed vertices

$S \subseteq K$

W$(r,S,v) =$ *walks s.t.*

length: r

end in v

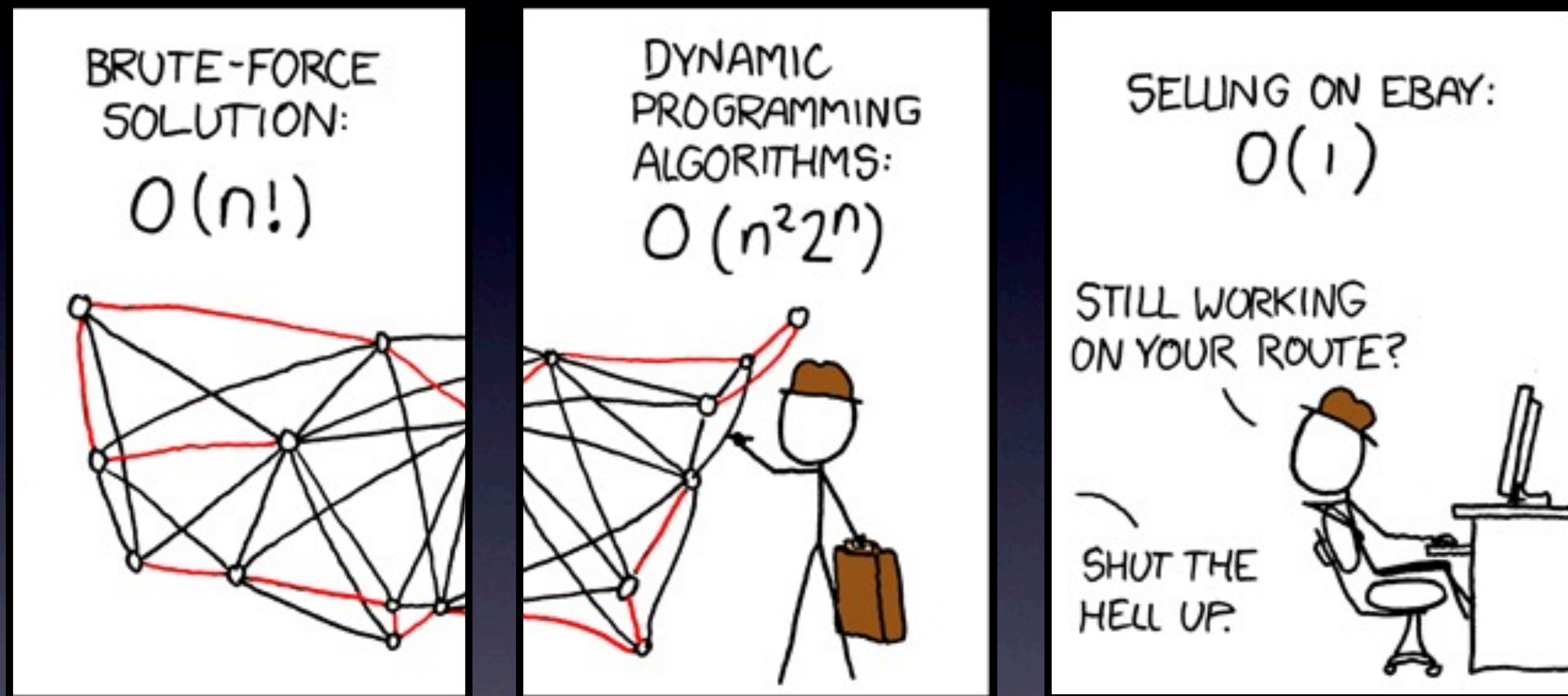
visit all in S exactly once

no other in K

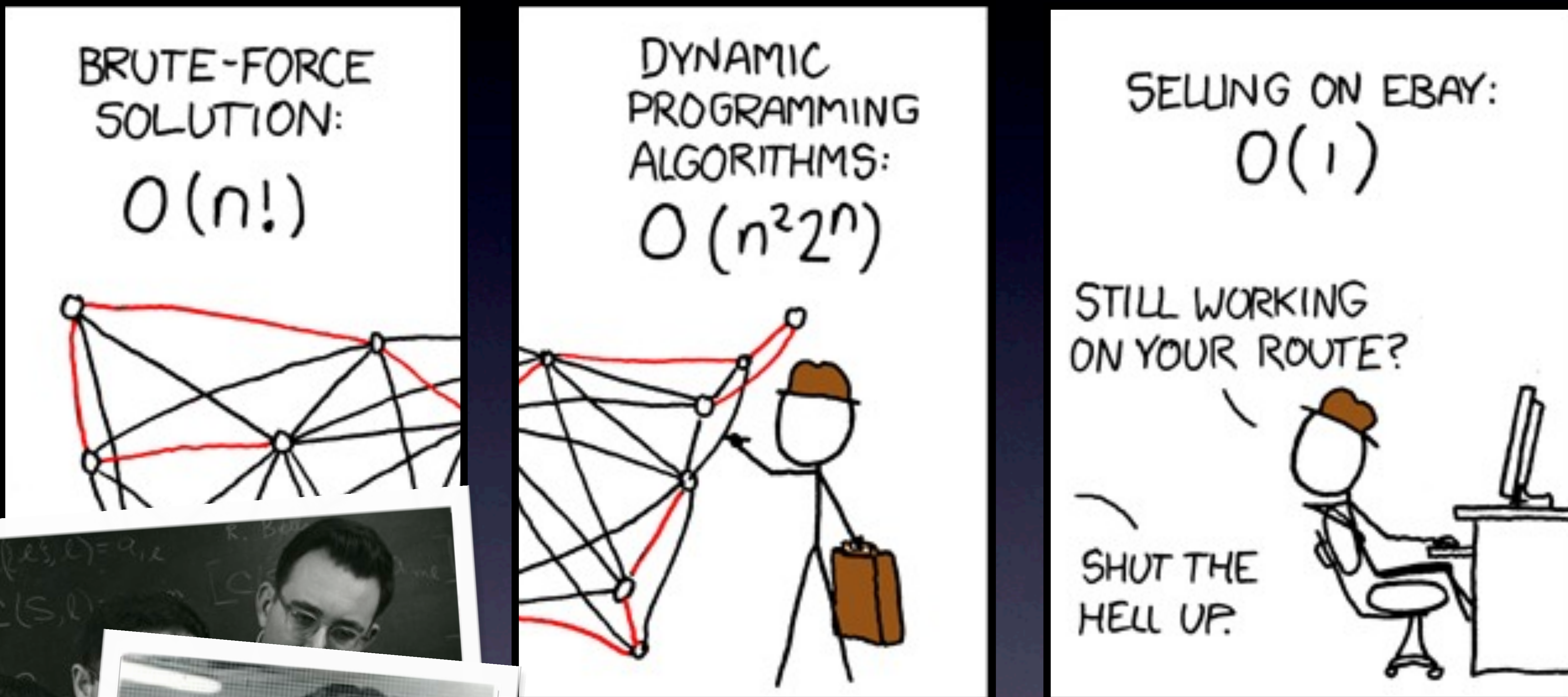
$$W(r,S,v) = \begin{cases} \bigcup_{uv \in E} W(r-1, S, u) & v \notin S \\ \bigcup_{uv \in E} W(r-1, S-v, u) & v \in S \end{cases}$$

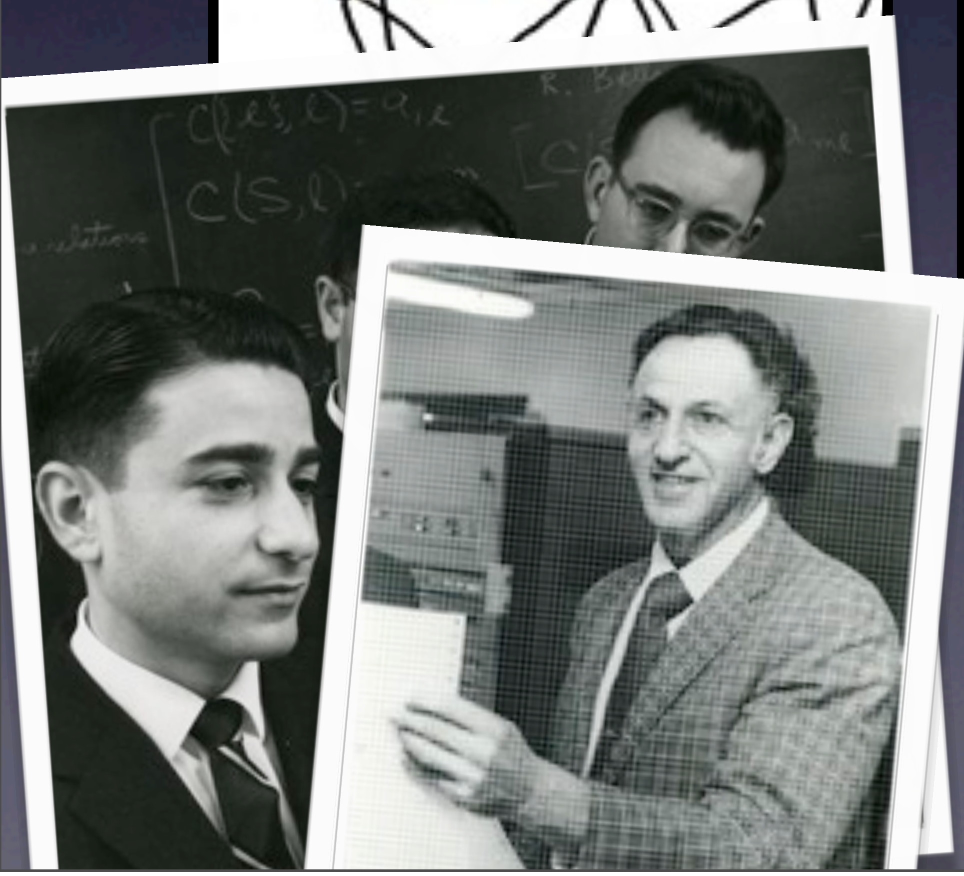
Time: 2^Kpoly(n)

Constructing all Walks:
Dynamic Programming for Sequencing Problems



Constructing all Walks: Dynamic Programming for Sequencing Problems

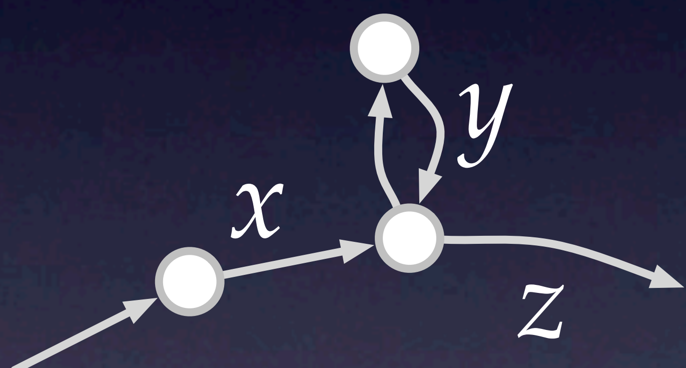




Held–Karp

Bellman

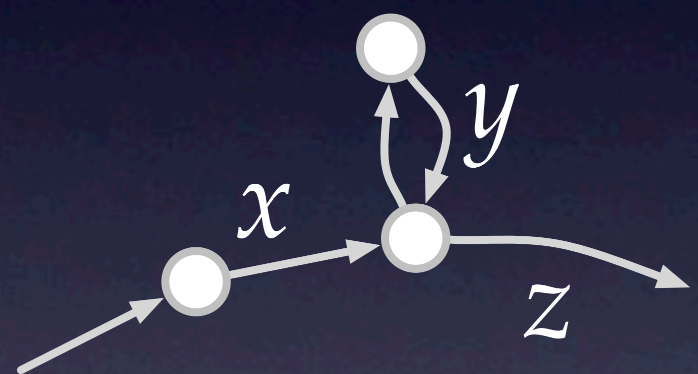
Some pitfalls…



does not cancel

xy^2z

Some pitfalls…

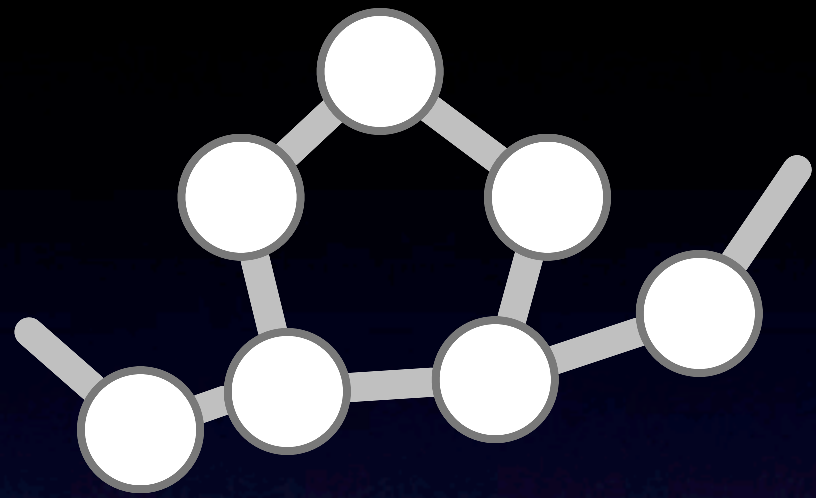


does not cancel

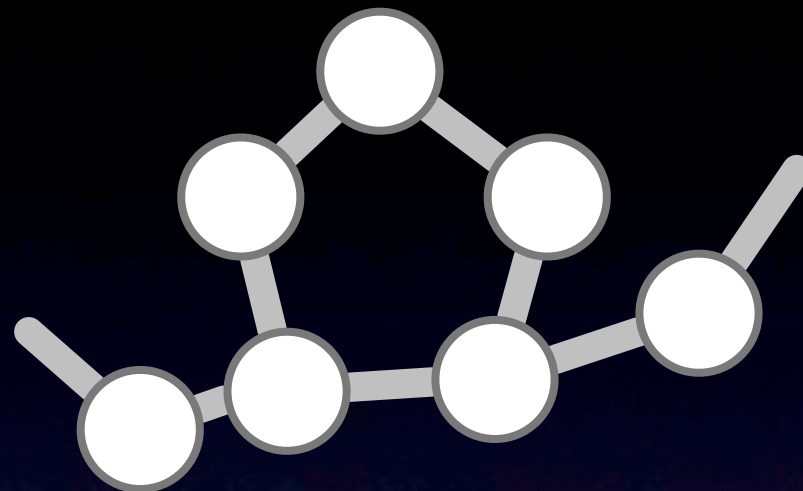
xy^2z

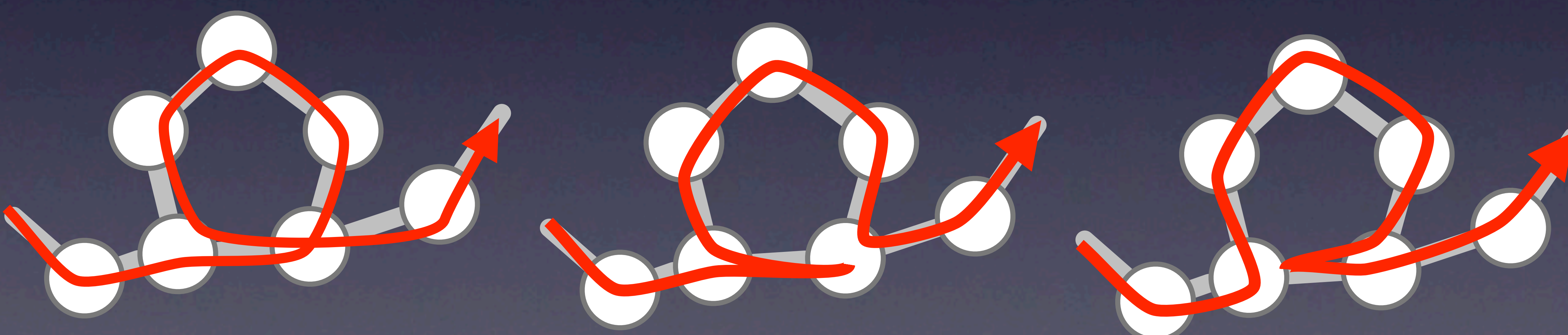
(solved in the dynamic program: just avoid “digons”)

Some pitfalls…

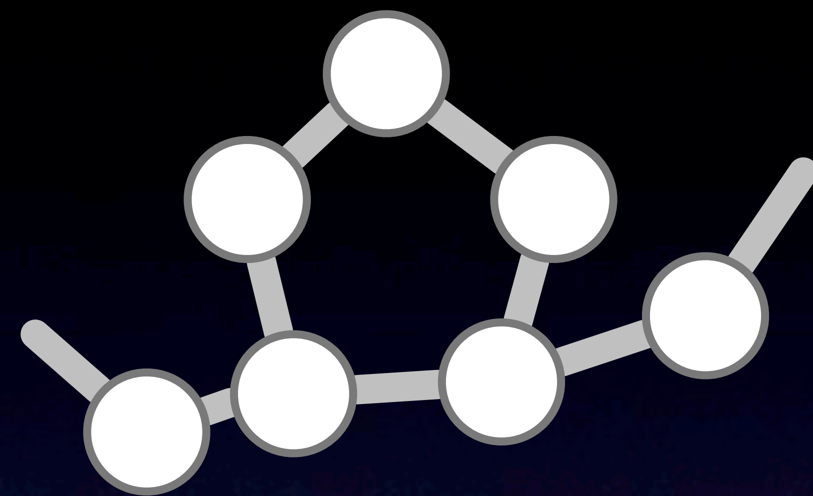


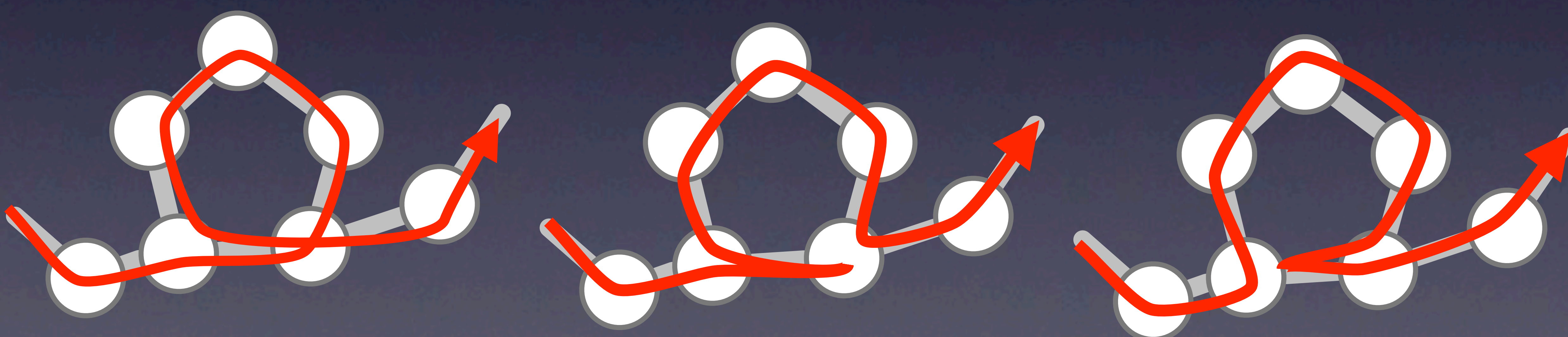
Some pitfalls…





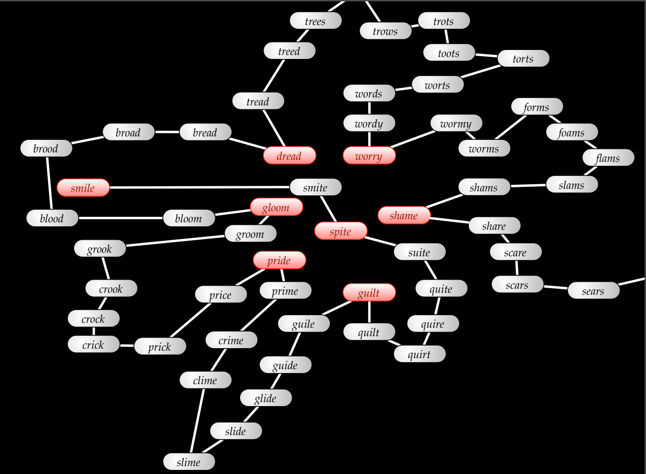
Some pitfalls…





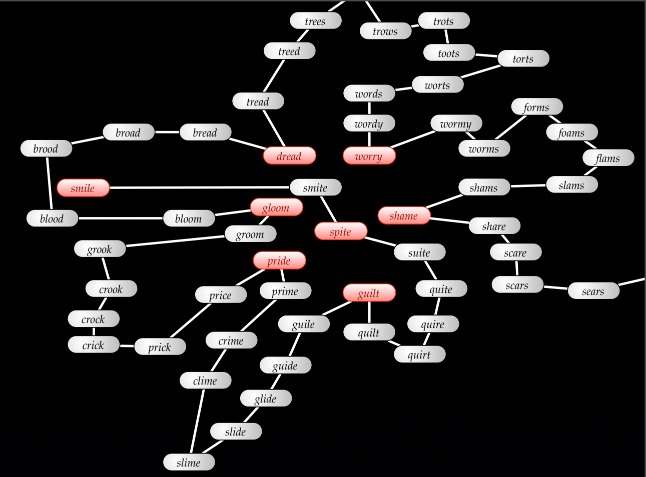
(solve shorter lengths first)

k specified vertices, n vertices



1. Associate random value from GF(2^q) to each edge
2. Use dynamic programming to count the contribution of all sufficiently well-behaved walks
3. Return "Found one!" if the result is nonzero

k specified vertices, n vertices

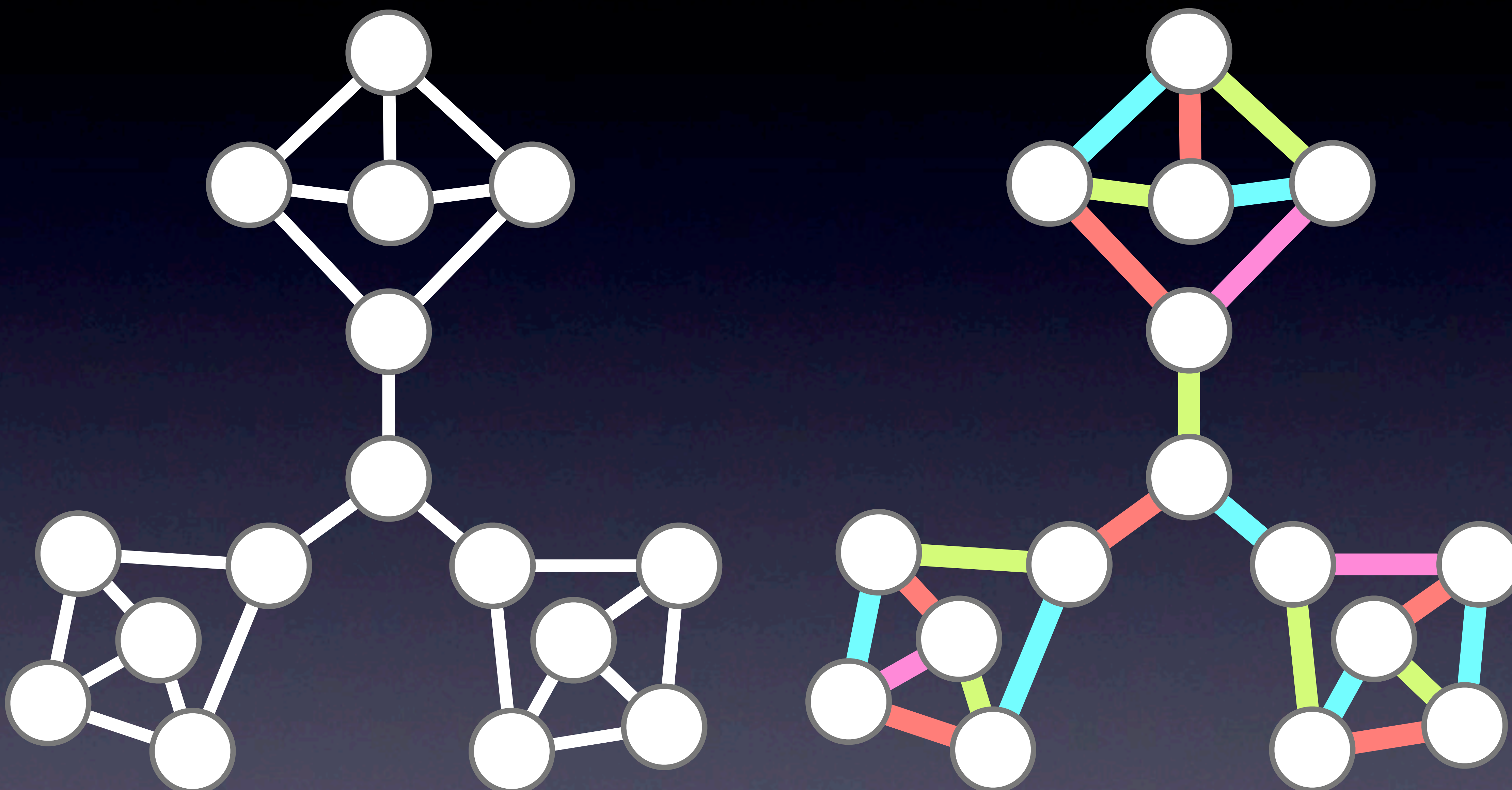


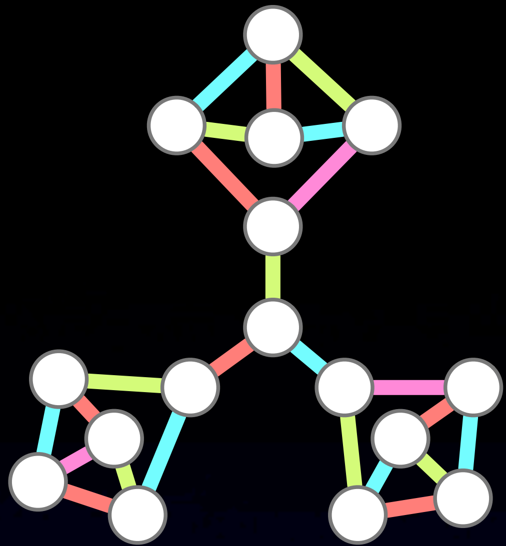
Theorem: *Shortest cycle through k given vertices or edges in time* $2^k\text{poly}(n)$ *with exponentially small one-sided error.*

1. Associate random value from GF(2^q) to each edge
2. Use dynamic programming to count the contribution of all sufficiently well-behaved walks
3. Return “Found one!” if the result is nonzero

Edge Colouring

Edge Colouring





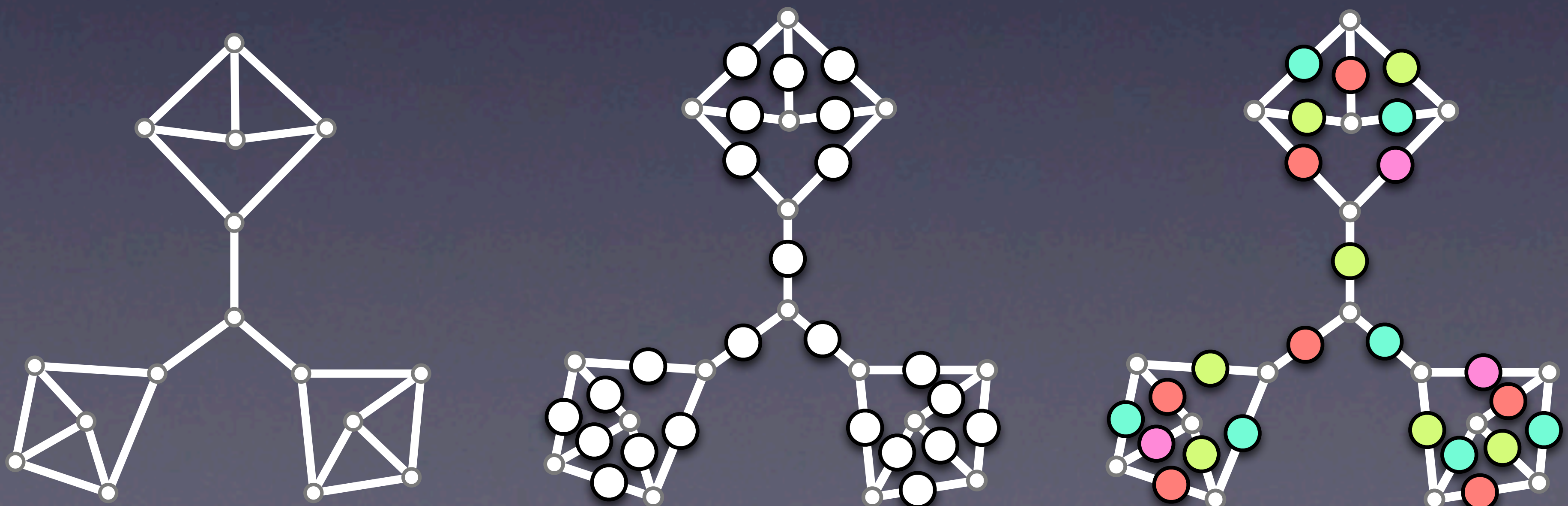
k: # colours

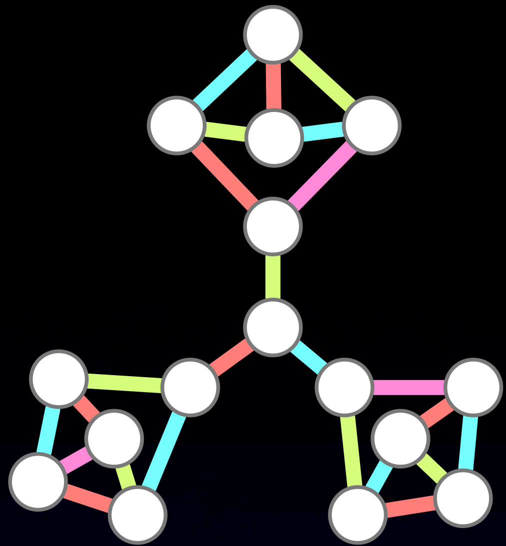
d: degree

Vizing: $k = d$ or $k = d$+1

Brute force: check all d^m possibilities

Vertex colour the line graph: time 2^m=$2^{nd/2}$





k: # colours

d: degree

Brute force	d^m
Vertex colour the line graph	$2^m = 2^{nd/2}$
"Narrow sieves" [*BHKK*]	$2^{n(d-1)/2}$

Under ETH: not in $\exp(o(n))$

Edge Colouring takes

- ☐ $\exp(n)$
- ☐ $d^n = \exp(n \log d)$
- ☐ $\exp(m) = \exp(nd)$

Tak fordi I kom