Introduction to Exact Algorithmics

Thore Husfeldt
IT University of Copenhagen
Lund University

Algorithm E
for problem T

Algorithm L
for problem T

Algorithm E'
for problem T

Algorithm C
for problem R

Algorithm A
for problem P

Algorithm H
for problem W

Algorithm D
for problem S

Algorithm I
for problem X

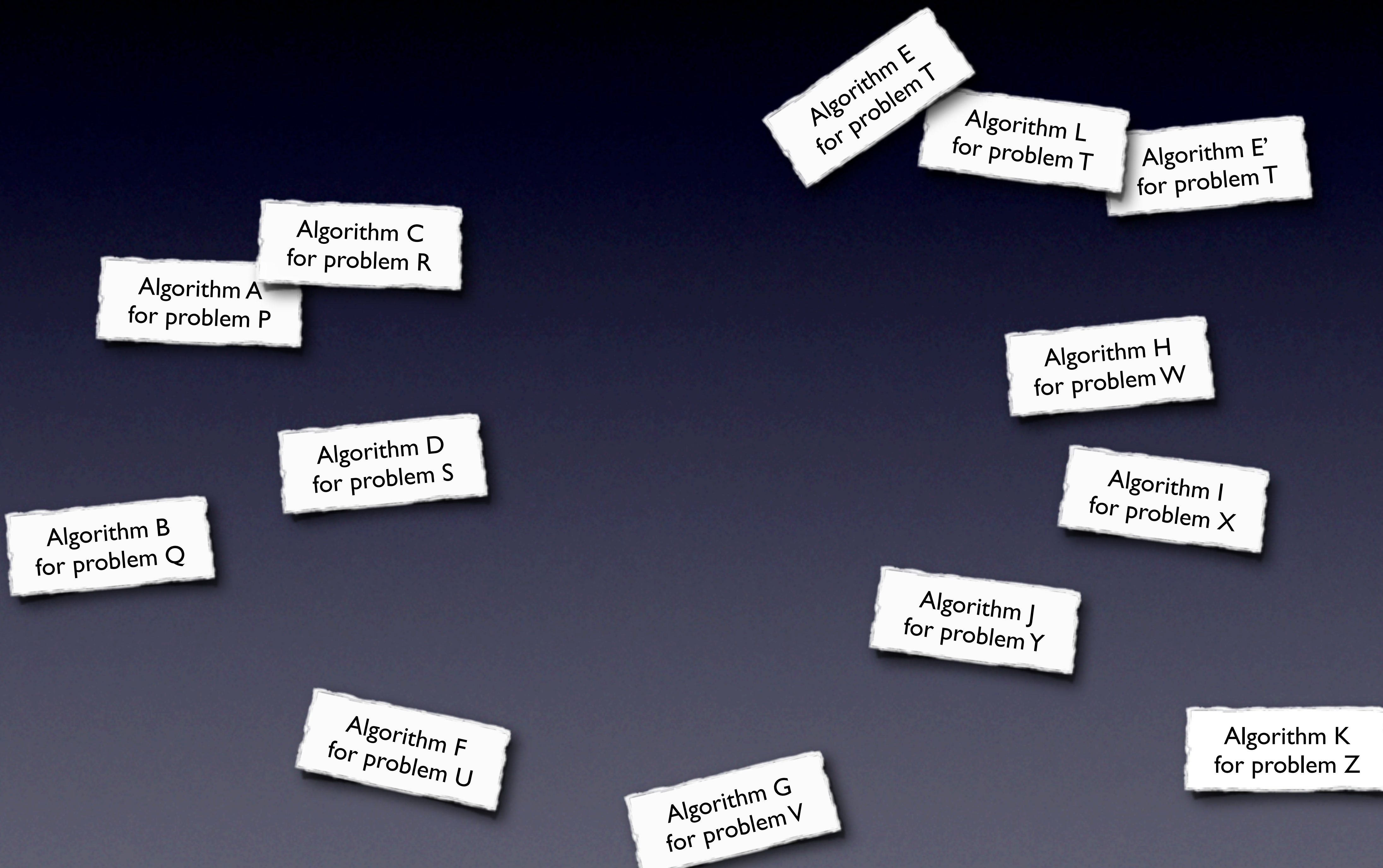
Algorithm B
for problem Q

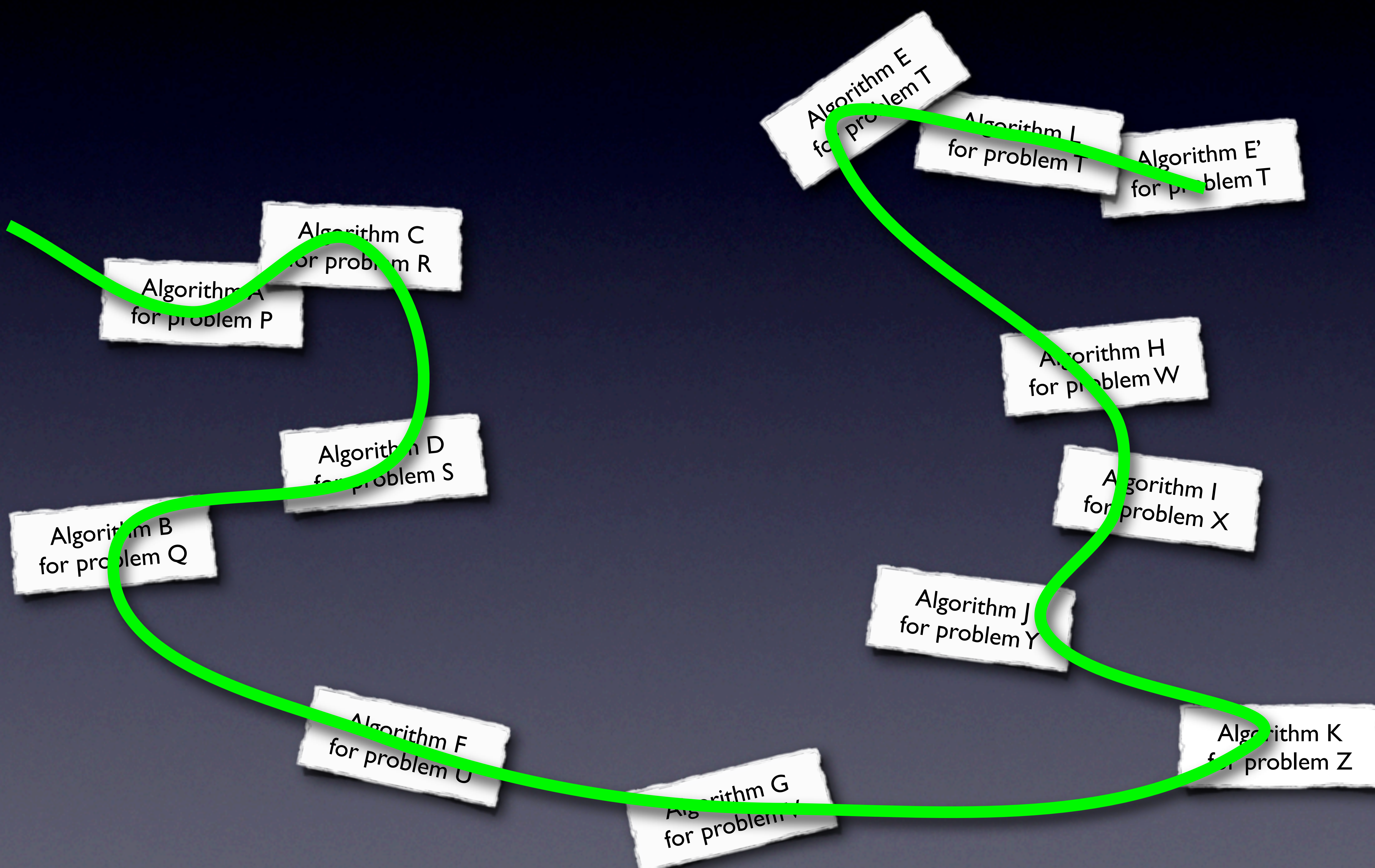
Algorithm J
for problem Y

Algorithm F
for problem U

Algorithm K
for problem Z

Algorithm G
for problem V


Algorithm E for problem T
Algorithm L for problem T
Algorithm E’ for problem T
Algorithm C for problem R
Algorithm A for problem P
Algorithm H for problem W
Algorithm D for problem S
Algorithm I for problem X
Algorithm B for problem Q
Algorithm J for problem Y
Algorithm F for problem U
Algorithm K for problem Z
Algorithm G for problem V


Algorithm E
Algorithm E’
for problem T
Algorithm J
for problem Y
for problem X

Algorithm E
for problem T

Algorithm L
for problem T

Algorithm E'
for problem T

Algorithm C
for problem R

Algorithm A
for problem P

Algorithm H
for problem W

Algorithm D
for problem S

Algorithm I
for problem X

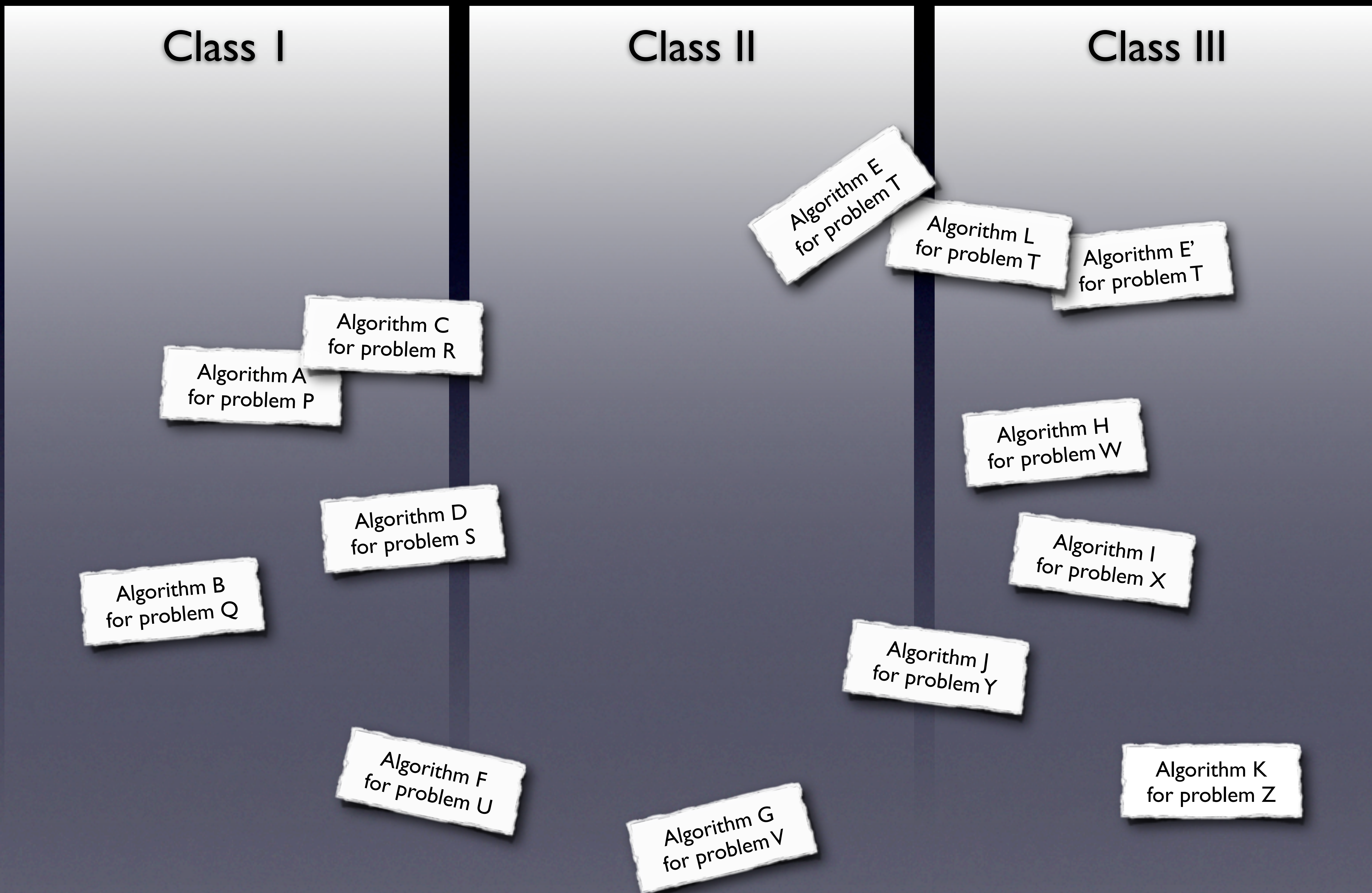
Algorithm B
for problem Q

Algorithm J
for problem Y

Algorithm F
for problem U

Algorithm K
for problem Z

Algorithm G
for problem V


Class I
Class II
Class III
Algorithm E for problem T
Algorithm L for problem T
Algorithm E' for problem T
Algorithm C for problem R
Algorithm A for problem P
Algorithm H for problem W
Algorithm D for problem S
Algorithm I for problem X
Algorithm B for problem Q
Algorithm J for problem Y
Algorithm F for problem U
Algorithm K for problem Z
Algorithm G for problem V

Class I

Algorithm B
for problem Q

Algorithm E
for problem T

Class II

Algorithm A
for problem P

Algorithm H
for problem W

Class III

Algorithm I
for problem X

Algorithm J
for problem Y

Perfect talk

Brute force
Algorithm A for problem P

Divide and conquer
Algorithm B for problem P

Greedy
Algorithm C for problem P

Dynamic programming
Algorithm D for problem P

Transforms
Algorithm E for problem P

Iteration
Algorithm F for problem P

Whatever
Algorithm G for problem P

…

Bad news

- Not comprehensive
- Not optimal
- Not standard

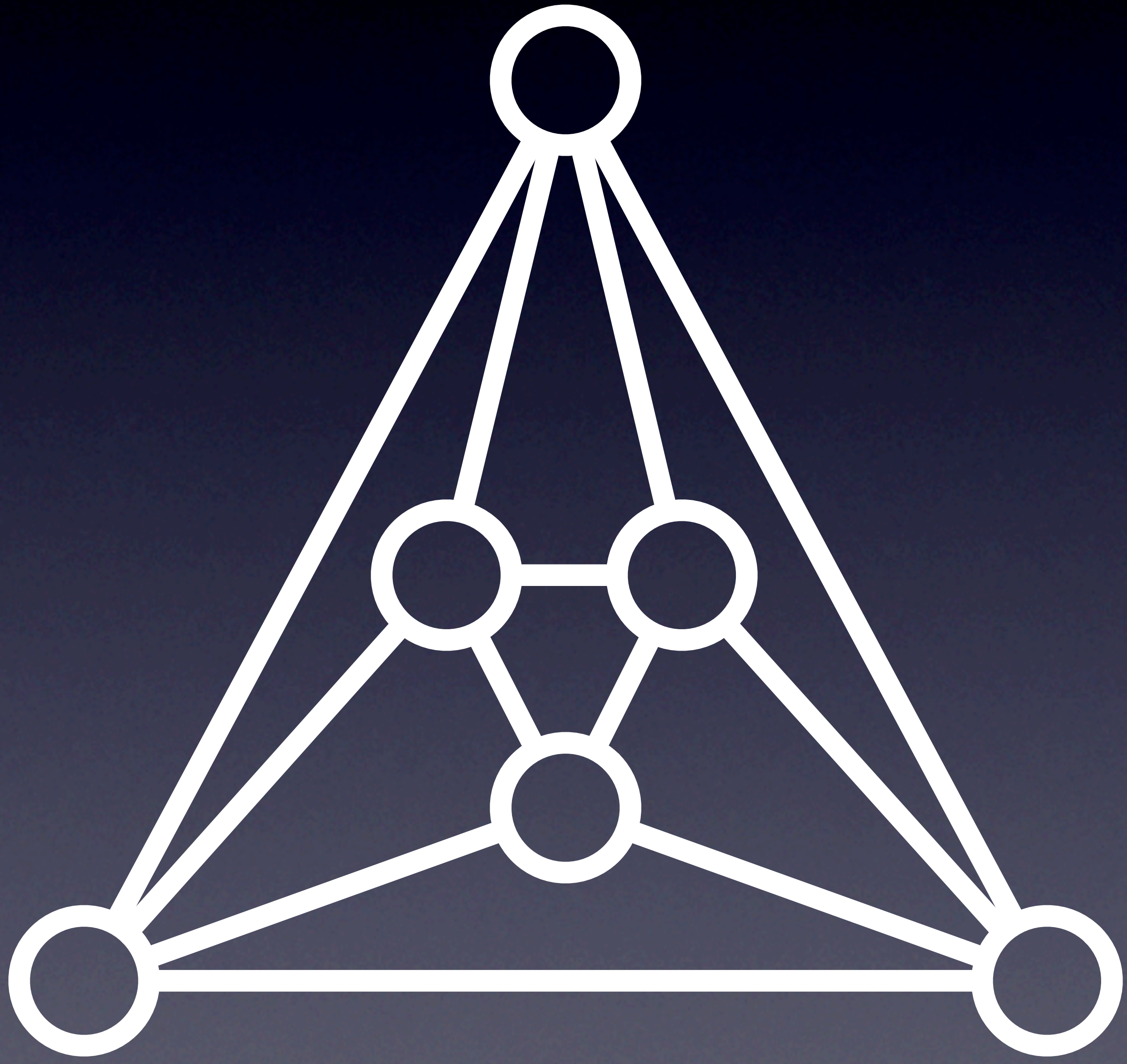
Even worse news

- Exercises during the talk

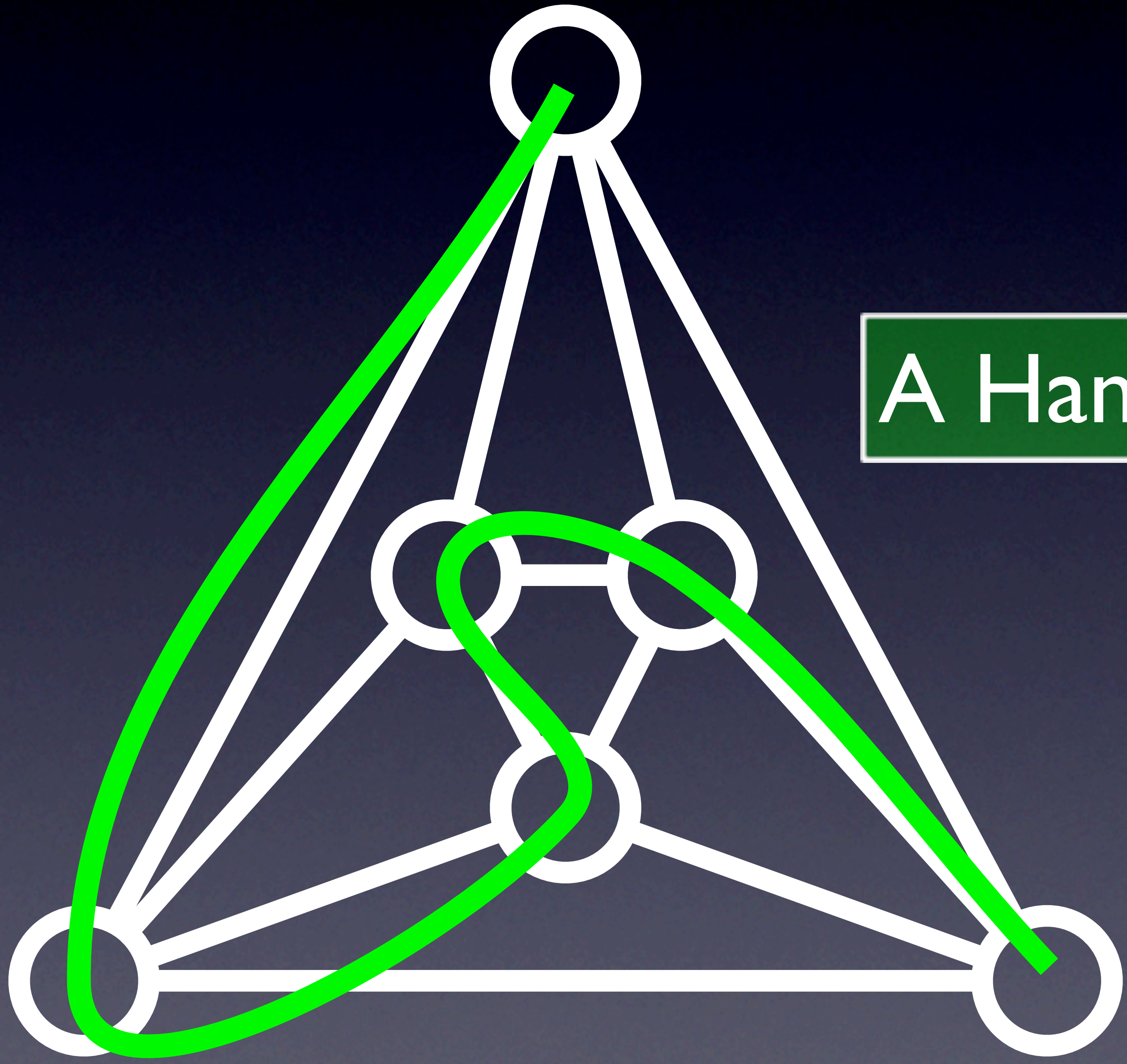
Brute Force

JUST DO IT.

Travelling Salesman

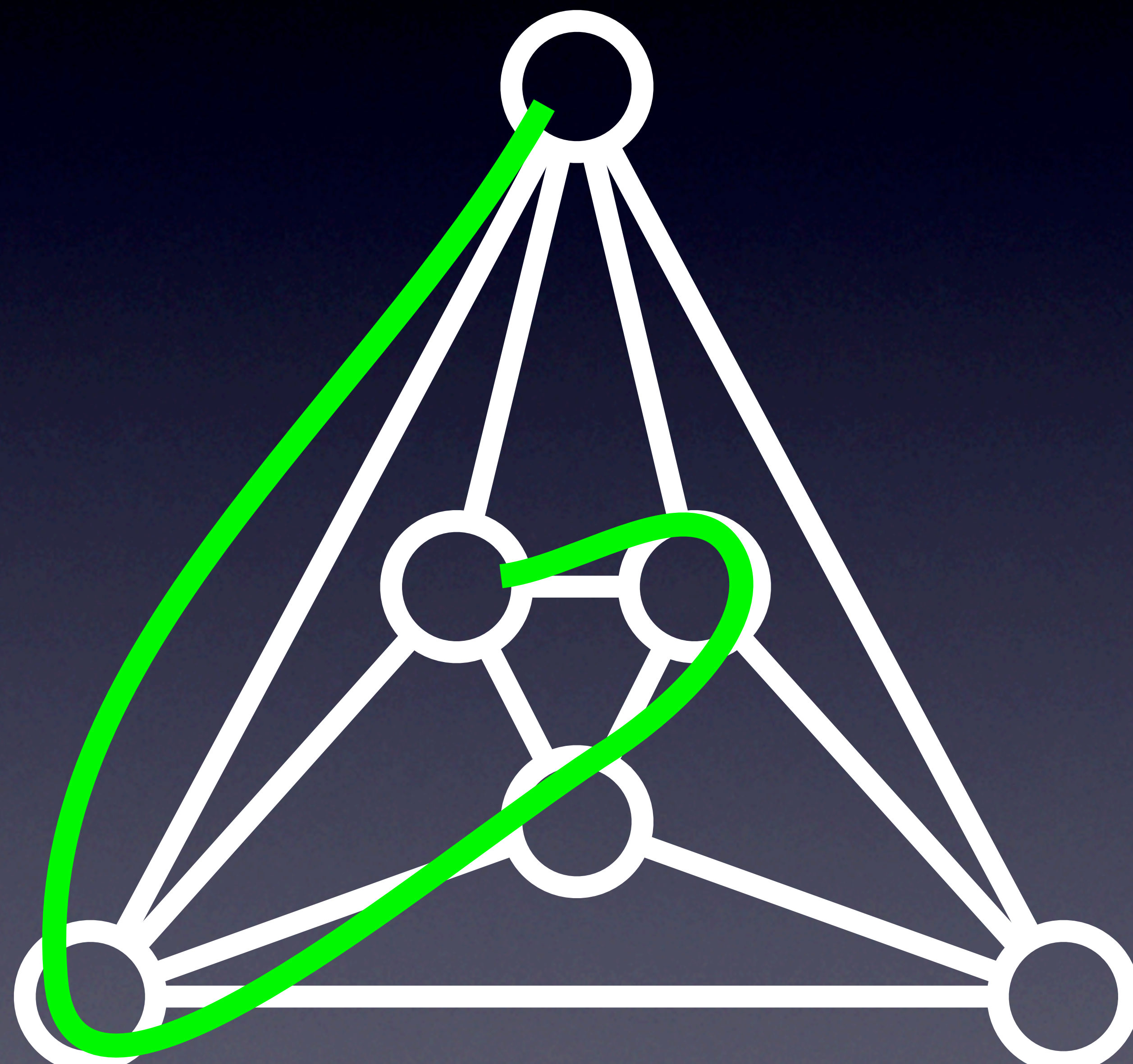


Travelling Salesman

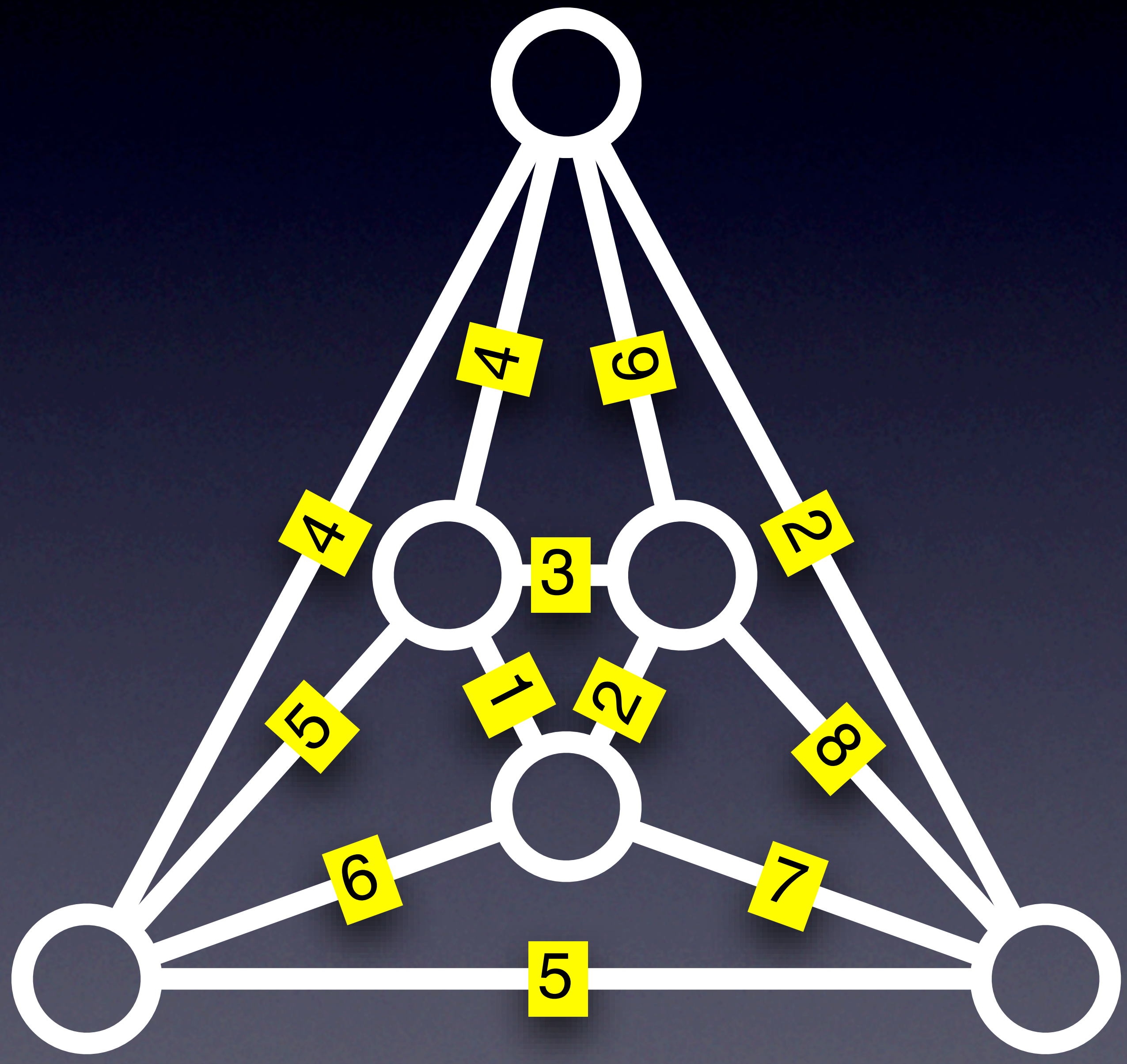


A Hamiltonian path

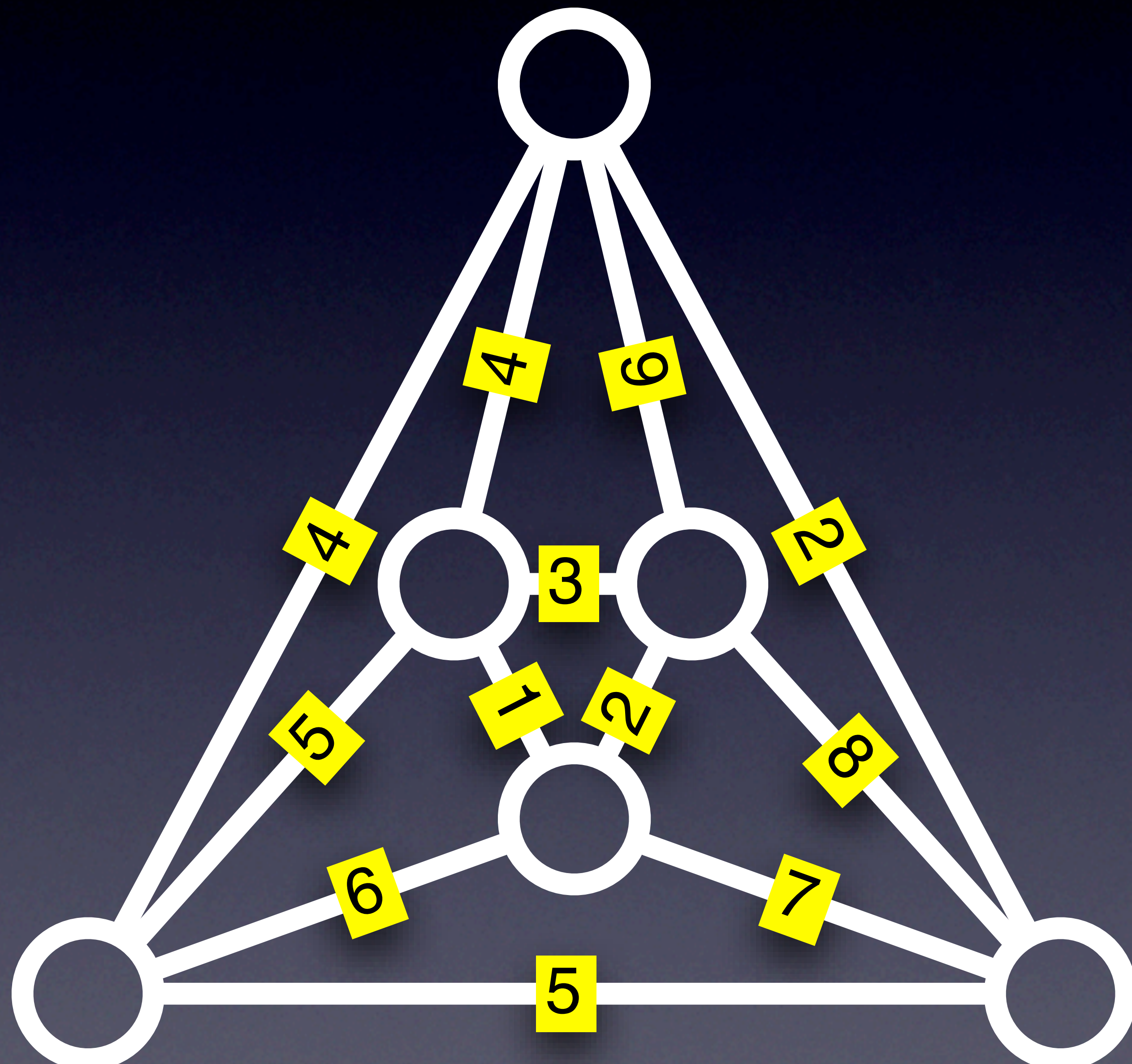
Travelling Salesman



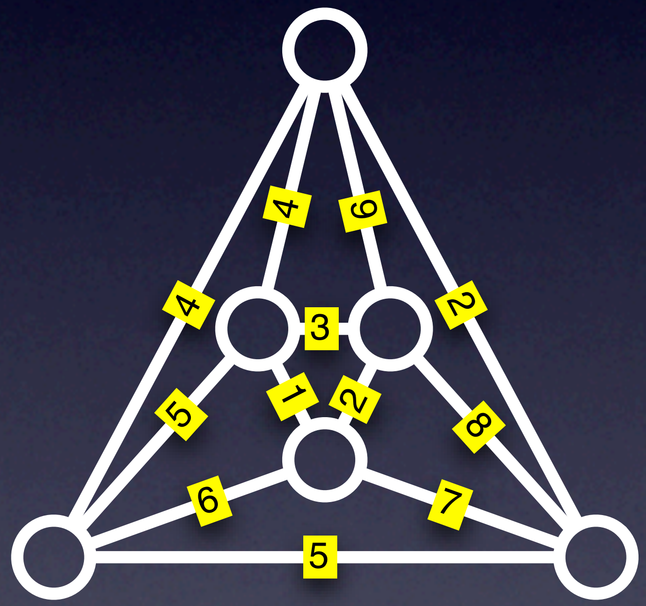
Travelling Salesman



Travelling Salesman



Travelling Salesman

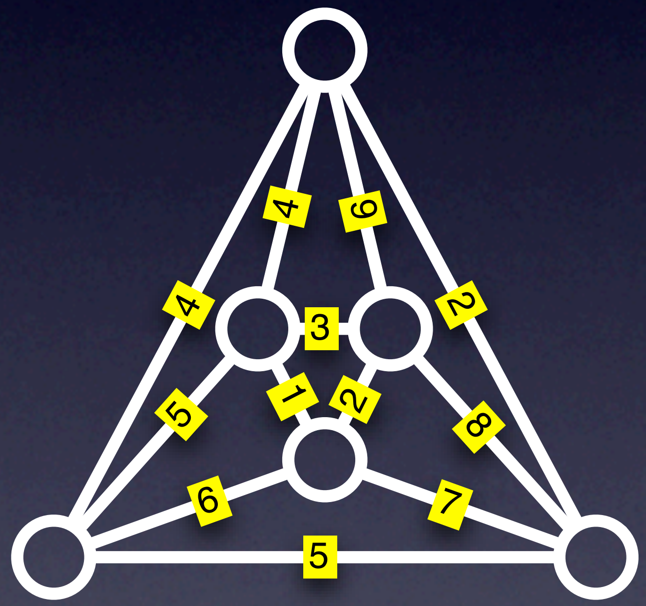


$$G = (V, E)$$

$$w \colon E \to \mathbf{N}$$

$$\min_{\pi} \sum_{i=1}^{n-1} w(\pi(i), \pi(i+1))$$

Travelling Salesman



$$G = (V, E)$$

$$w\colon E \to \mathbf{N}$$

$$\min_{\pi} \sum_{i=1}^{n-1} w(\pi(i), \pi(i+1))$$

Time $O(n!)$. Polynomial space.

$n!$ permutations of $1,2,\ldots,n$

Time $O(n!)$. Polynomial space.

Time $O(n!)$. Polynomial space.

$n!$ permutations of $1,2,\ldots,n$

Time $O((n\text{-}1)!)$

Time $O(n!)$. Polynomial space.

$n!$ permutations of $1,2,\ldots,n$

Don't need them all

Time $O((n-1)!)$

Time $O(n!n)$

Time $O(n!)$. Polynomial space.

$n!$ permutations of $1,2,\ldots,n$

Don't need them all

Polynomial computation
for each permutation

Time $O^*(n!)$

Polynomial space.

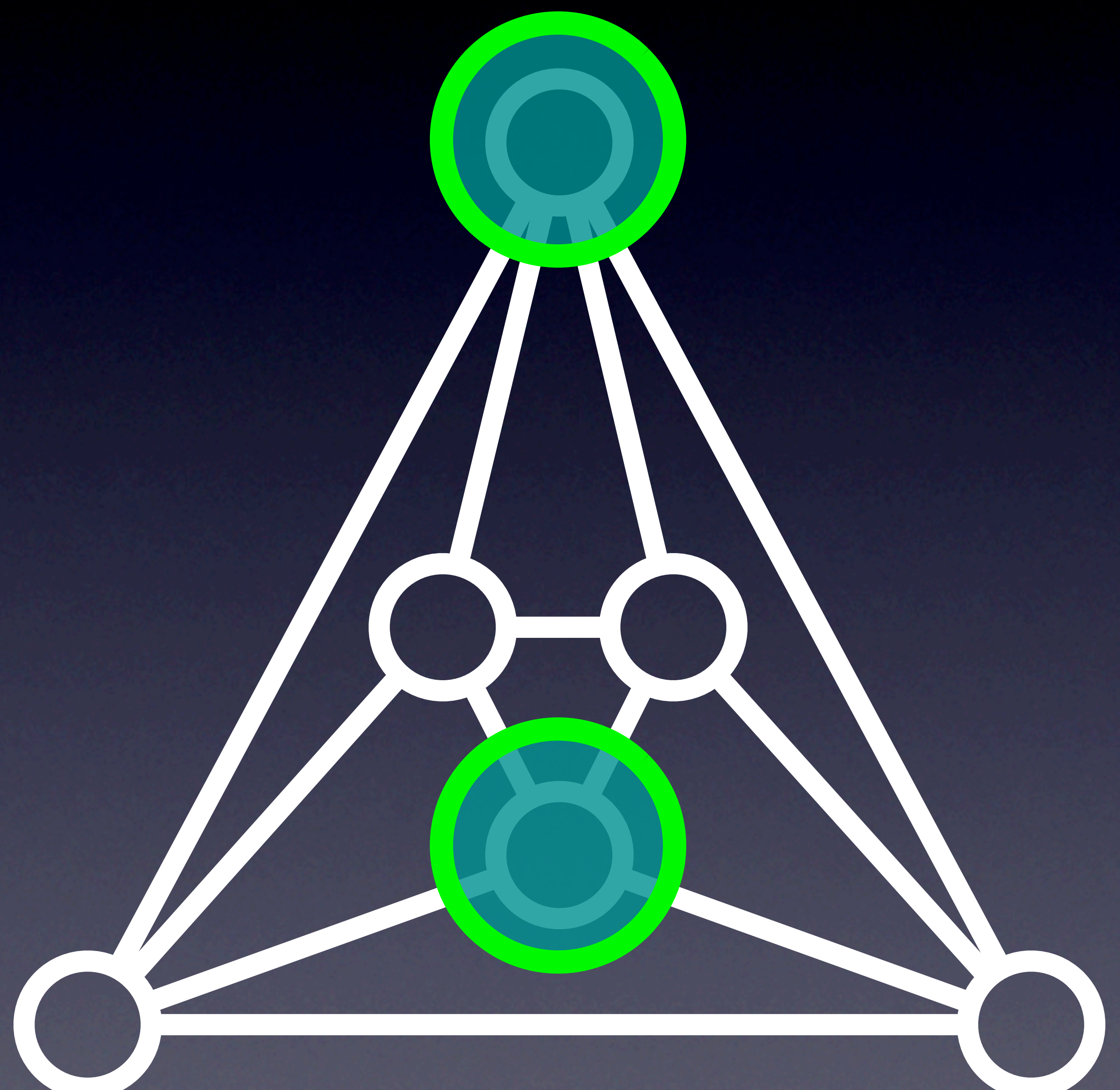
$n!$ permutations of $1,2,\ldots,n$

Don't need them all

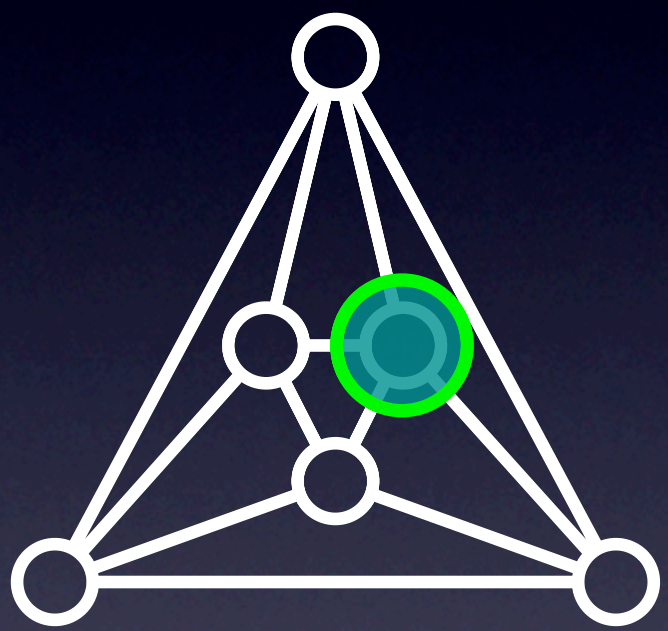
Polynomial computation for each permutation

Construct all permutations with constant delay?

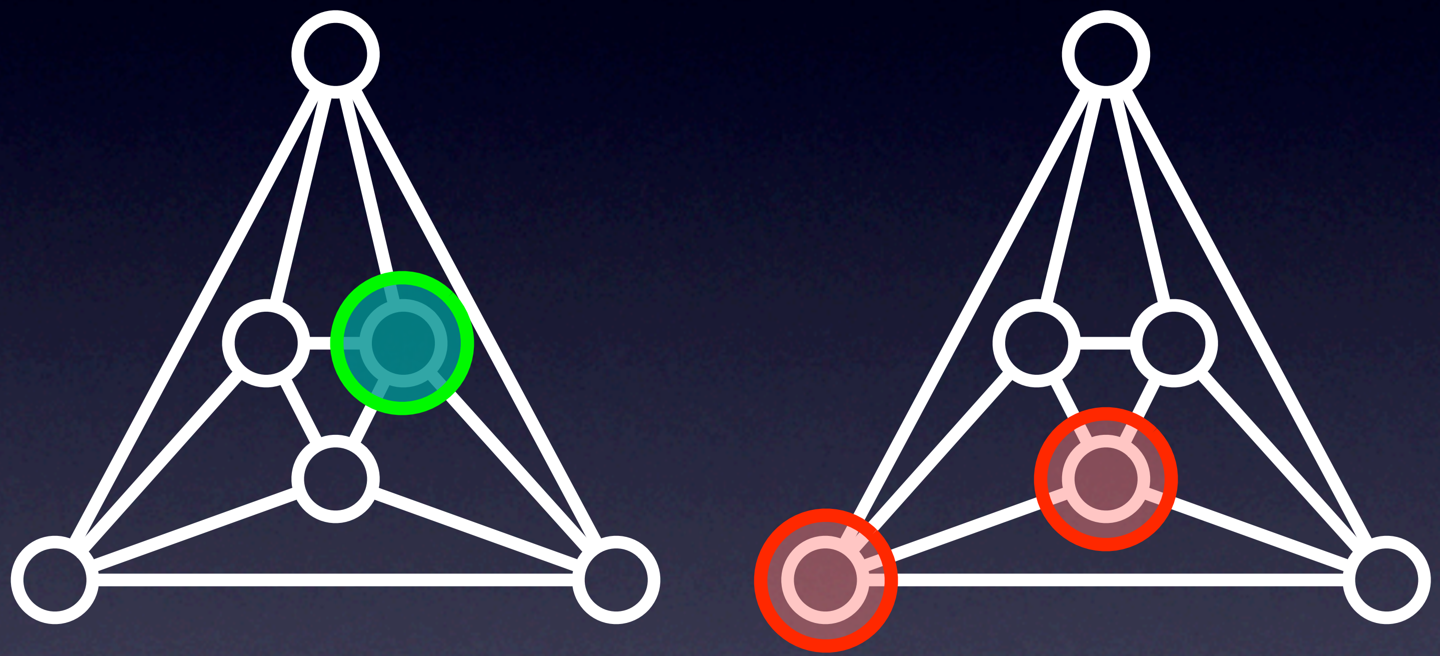
Independent set



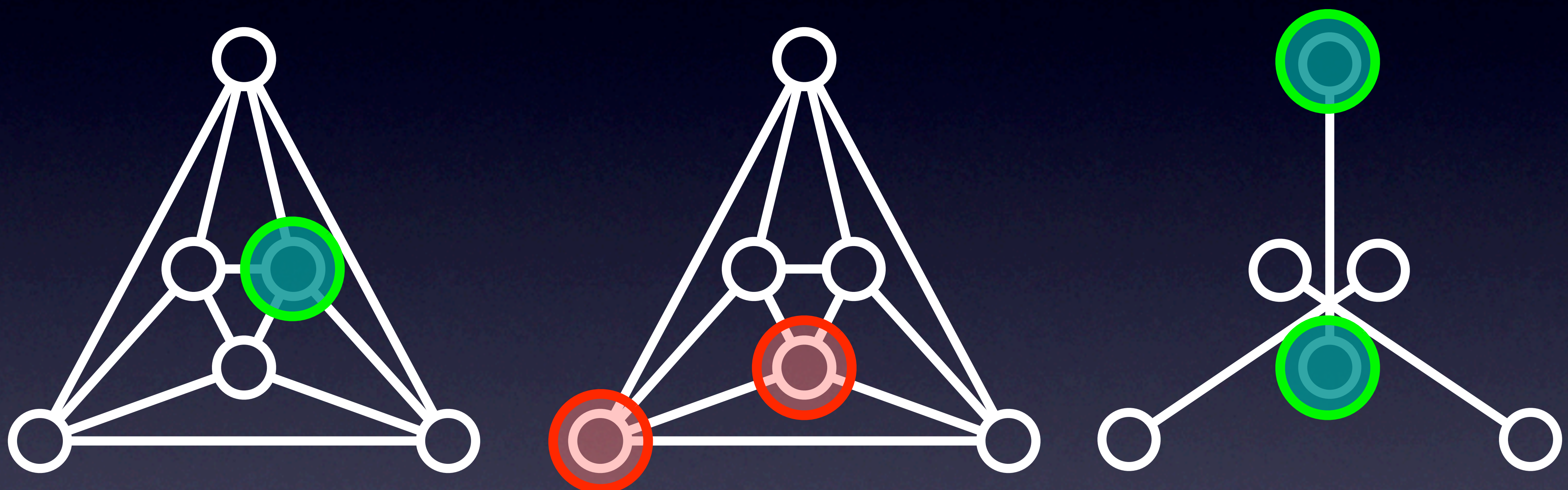
Independent set



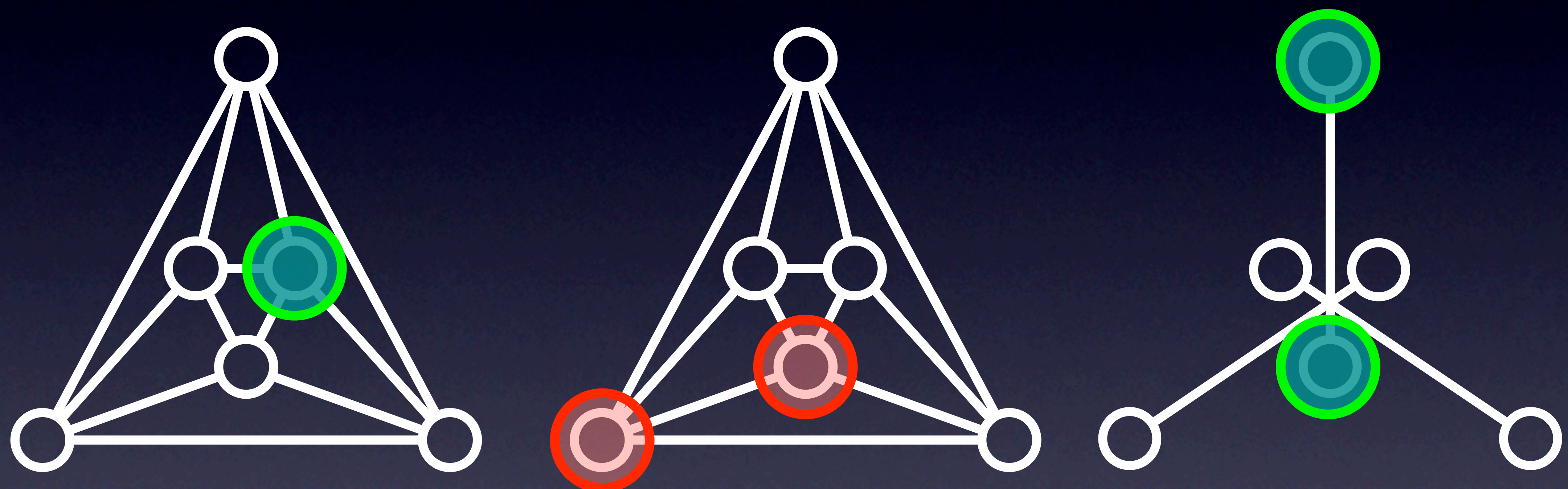
Independent set



Independent set



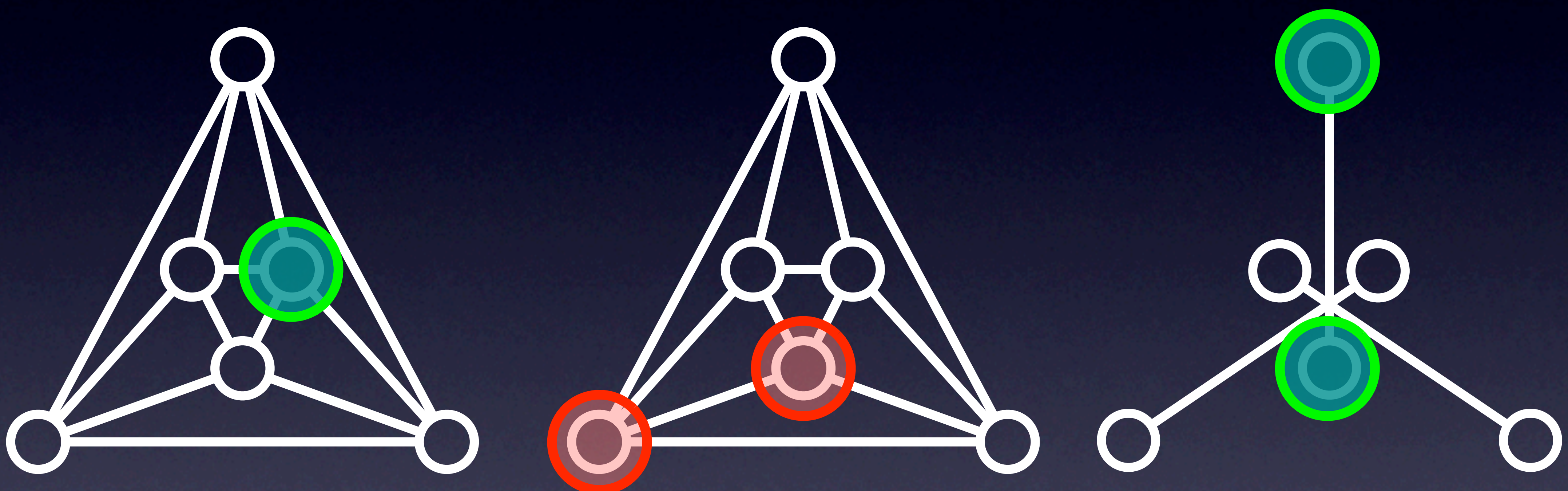
Independent set



$G = (V, E)$

$$\max_{I \subseteq V}\{\,|I| : u, v \in I \rightarrow uv \notin E\,\}$$

Independent set



$G = (V, E)$

$$\max_{I \subseteq V}\{\, |I| : u, v \in I \rightarrow uv \notin E \,\}$$

Time $O^*(2^n)$. Polyspace.

3-Satisfiability

$$(\neg x \vee y \vee z) \wedge (x \vee \neg y \vee z) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

n variables

m clauses

3-Satisfiability

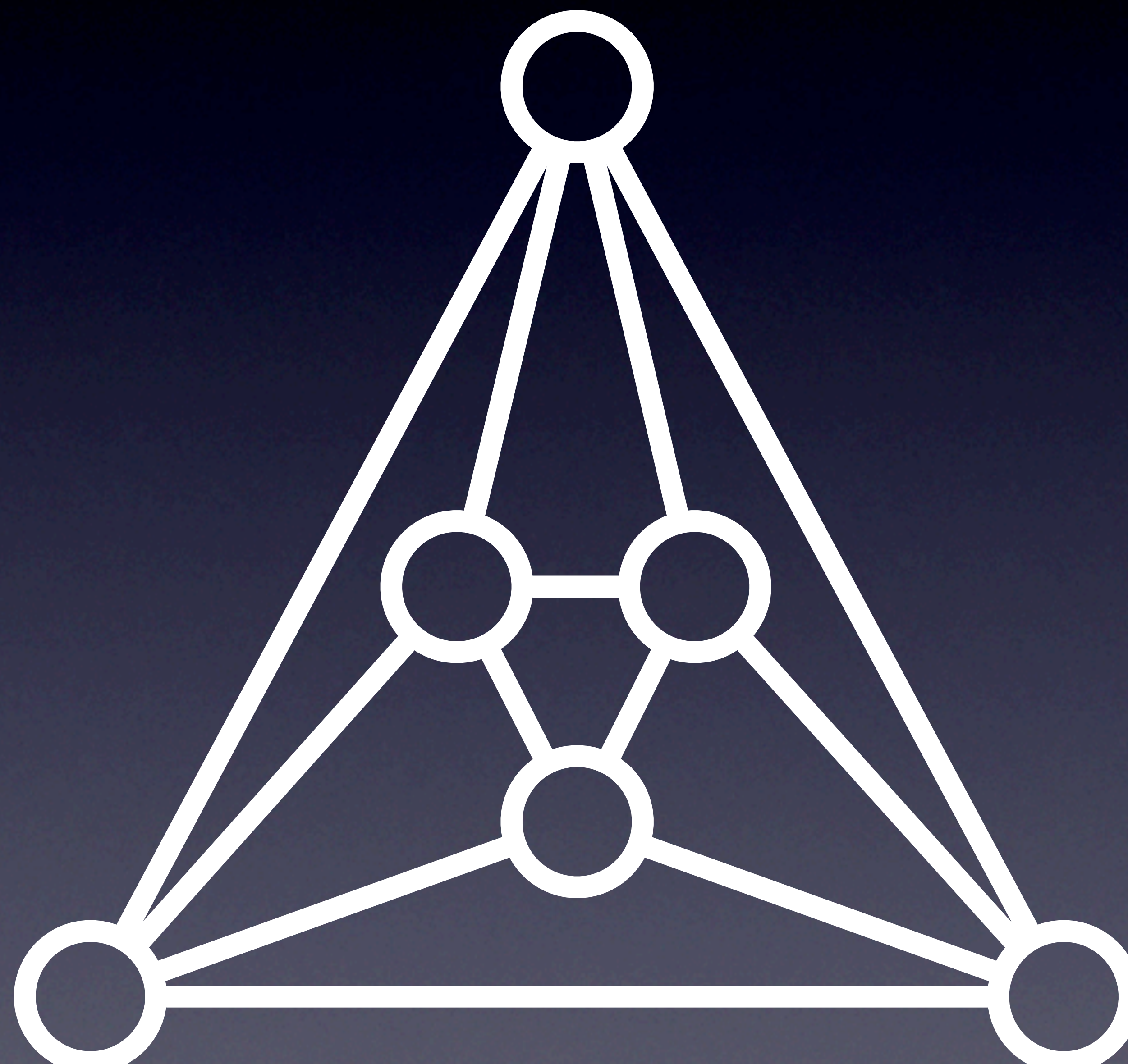
$$(\neg x \vee y \vee z) \wedge (x \vee \neg y \vee z) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

n variables

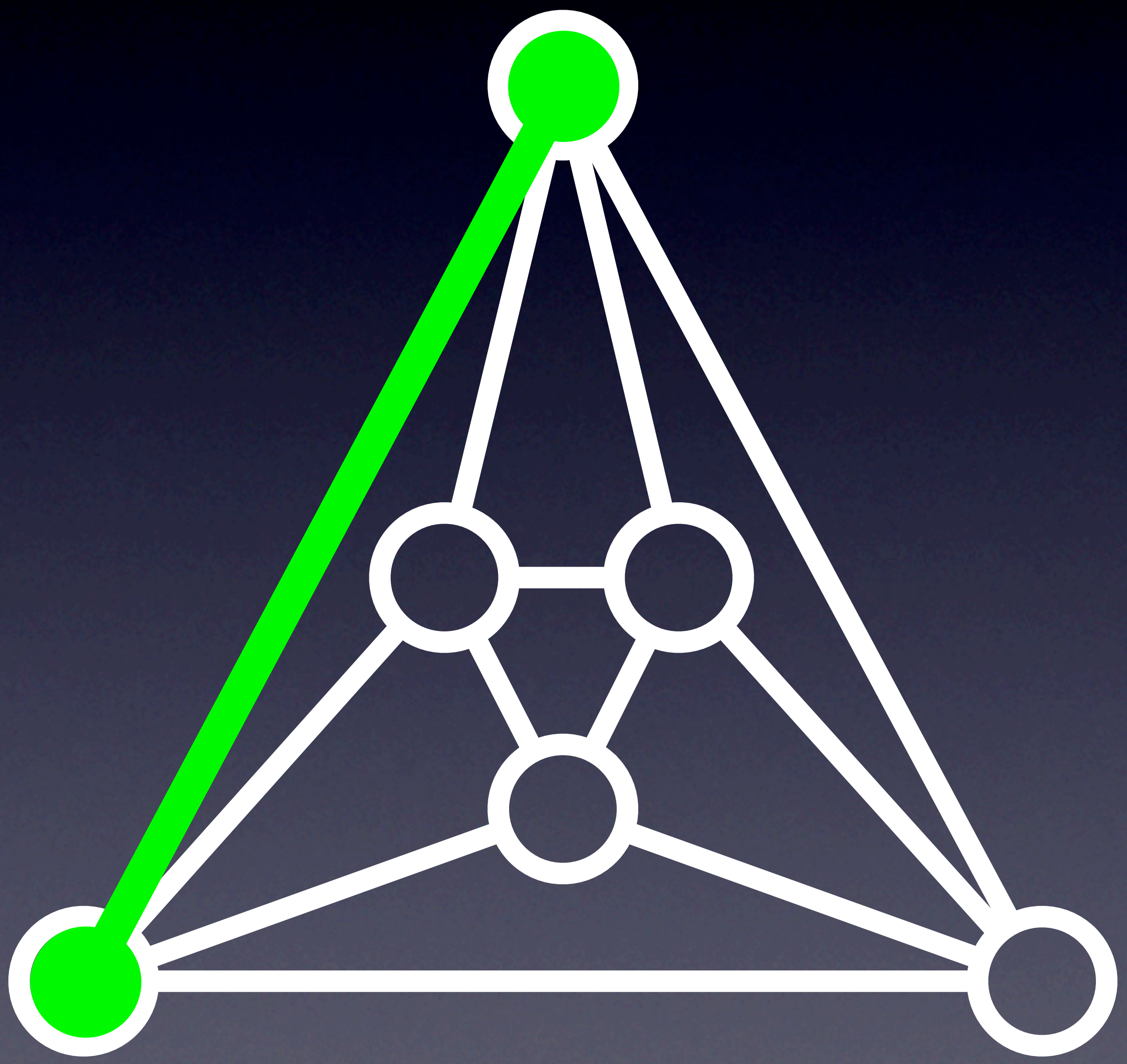
m clauses

Time $O^*(2^n)$. Polyspace.

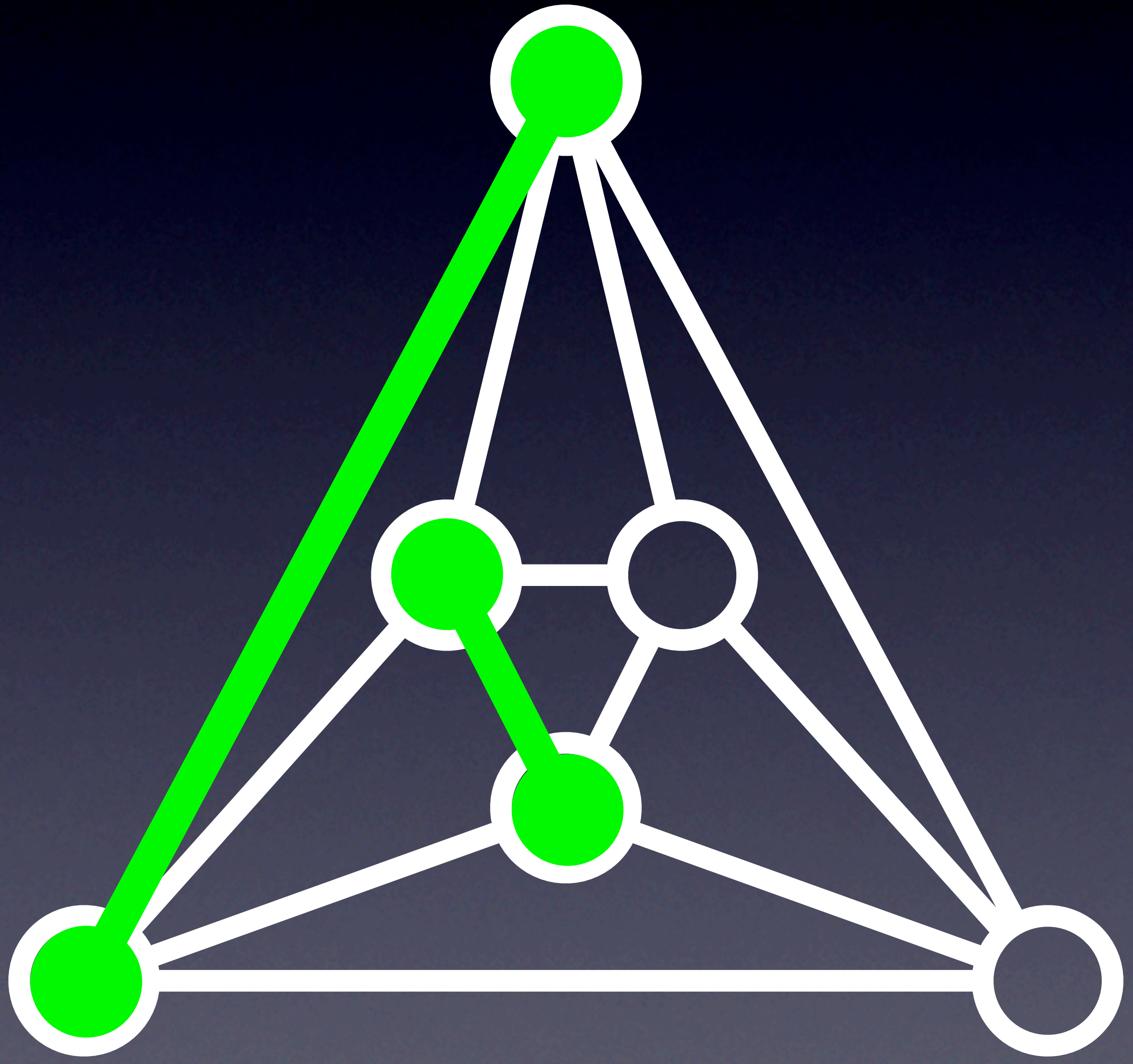
Perfect matchings



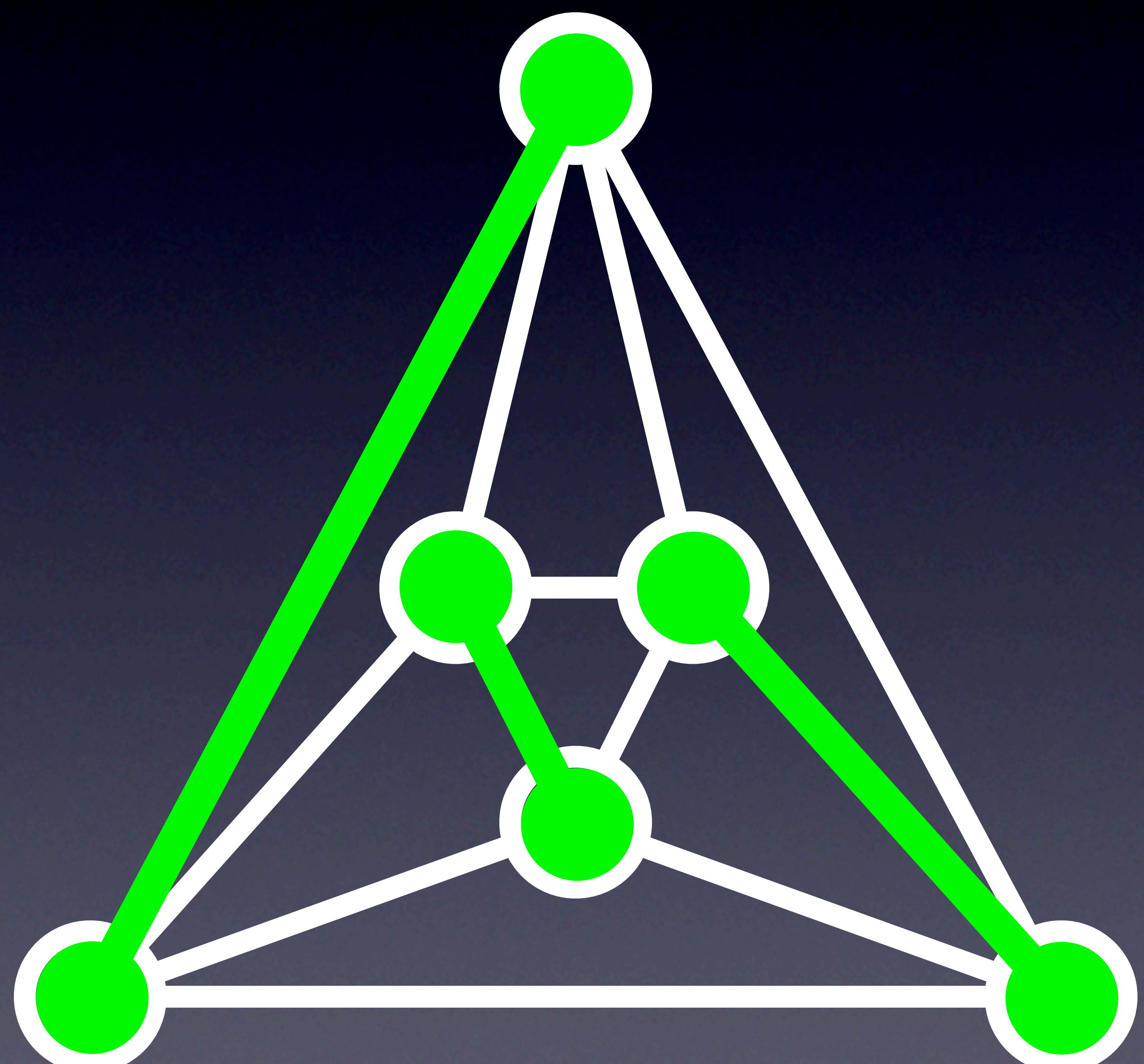
Perfect matchings



Perfect matchings

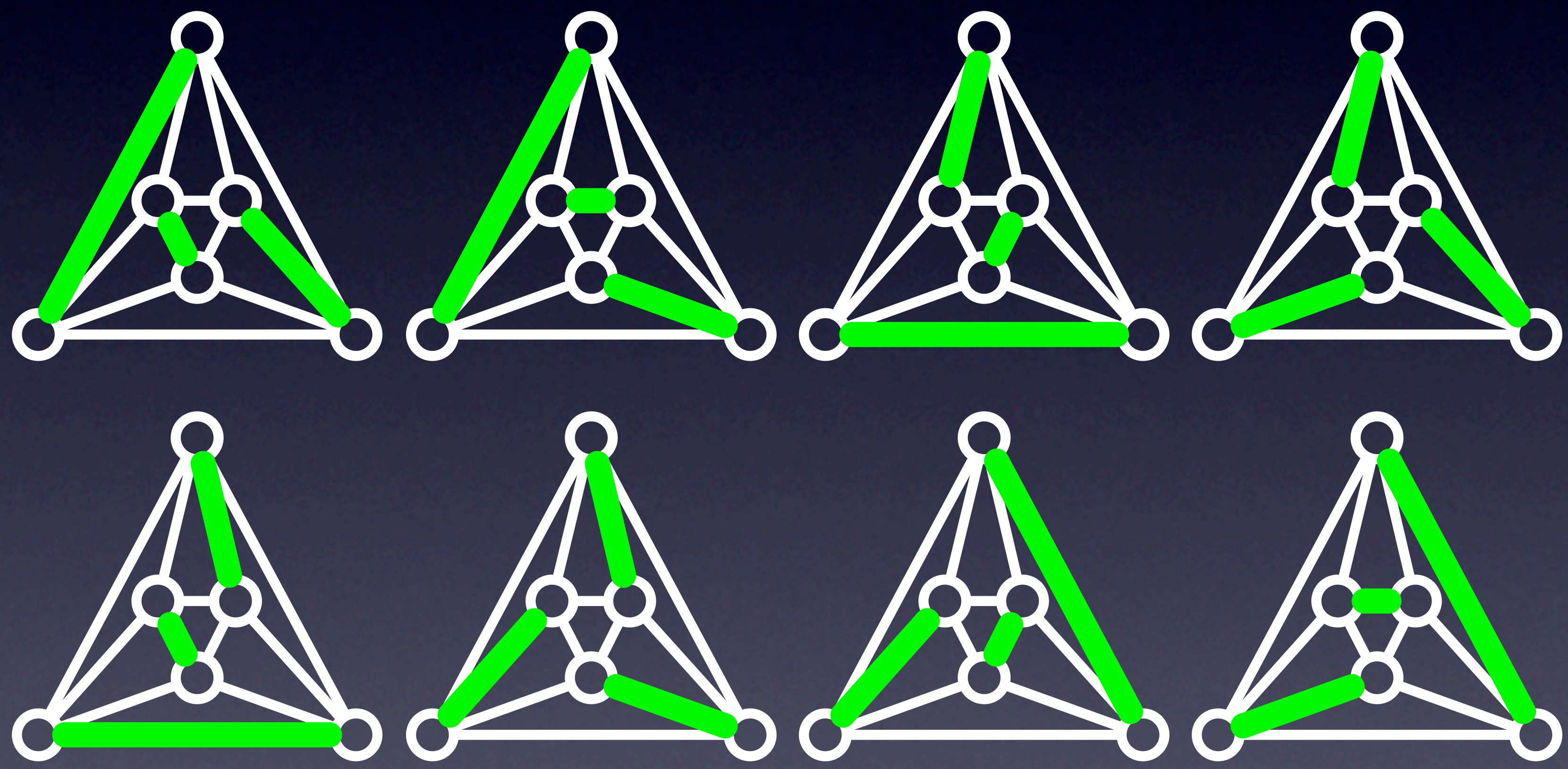


Perfect matchings



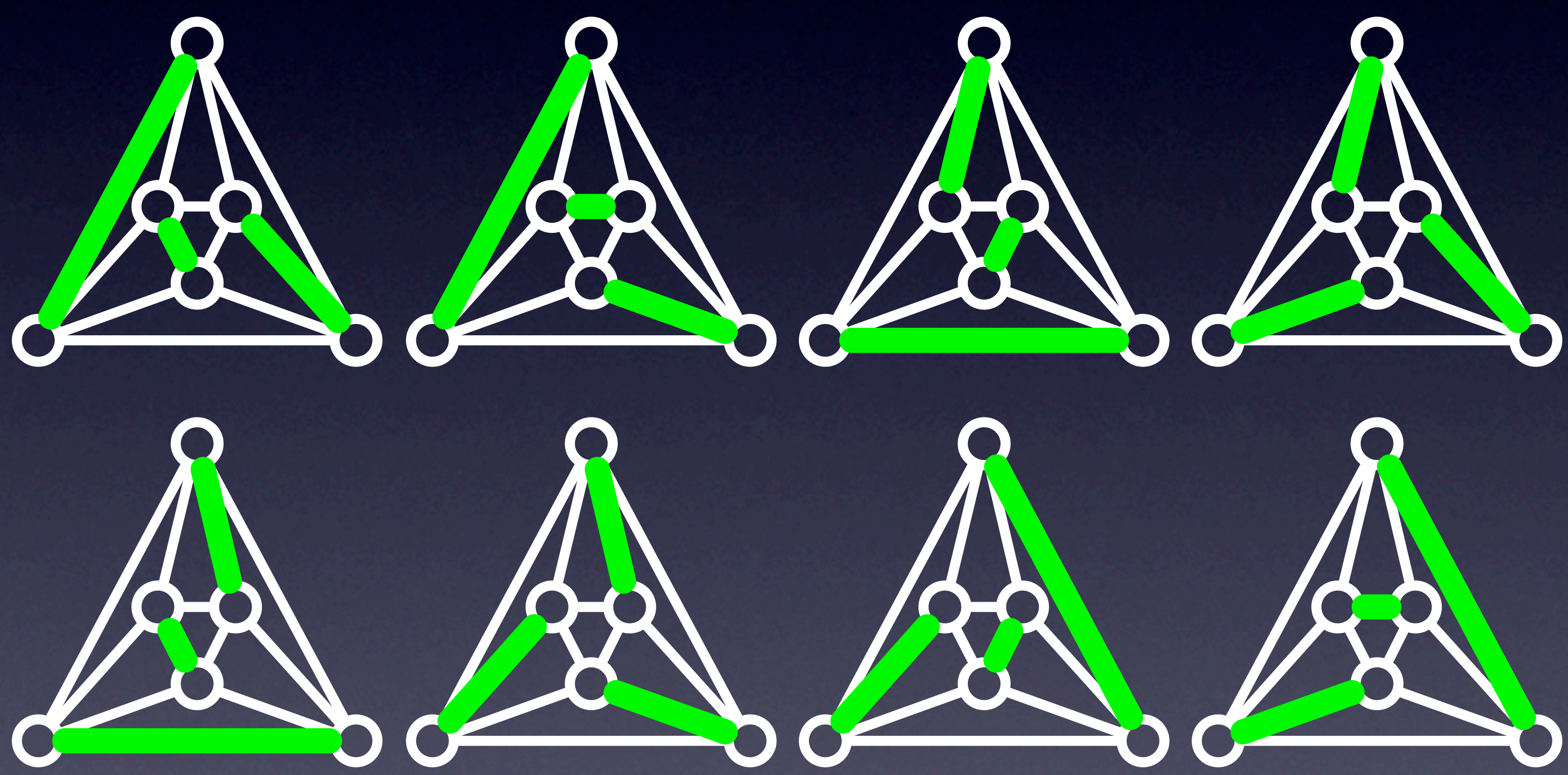
Perfect matchings

Count them (finding one is “easy”)



Perfect matchings

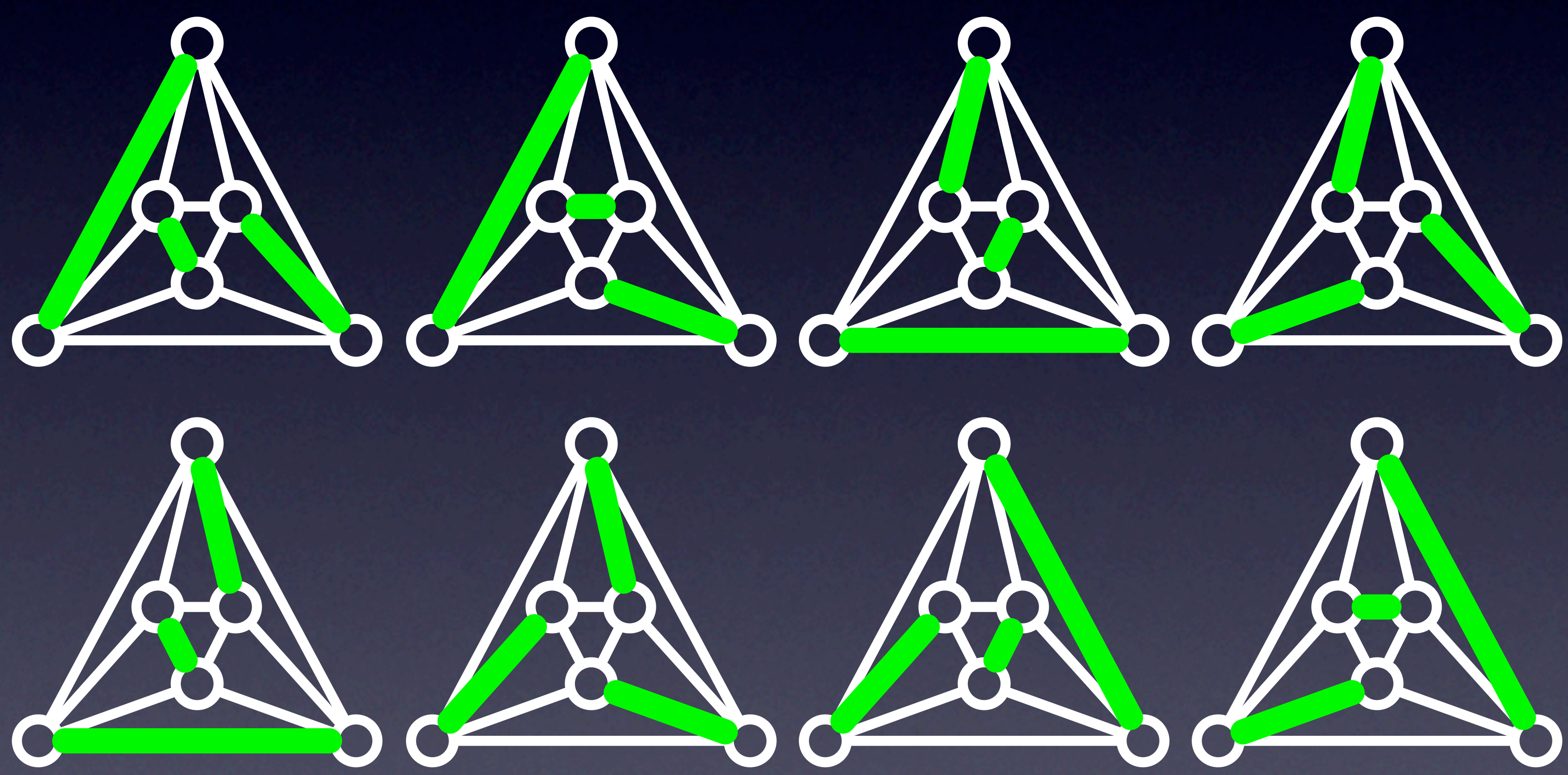
Count them (finding one is "easy")



Time $O^*(2^m)$. Polyspace.

Perfect matchings

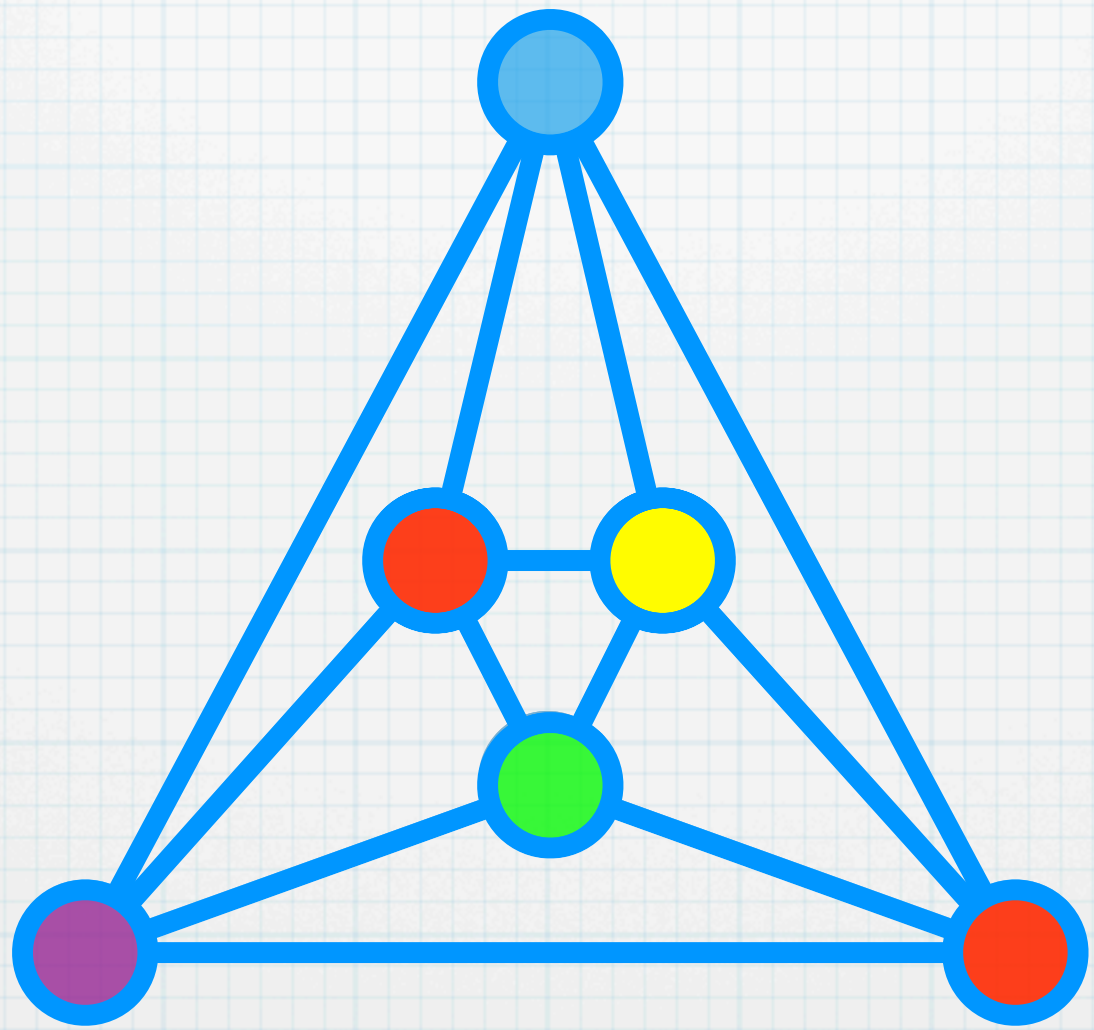
Count them (finding one is "easy")



Time $O^*(2^m)$. Polyspace.

Time $O^*(n!)$. Polyspace.

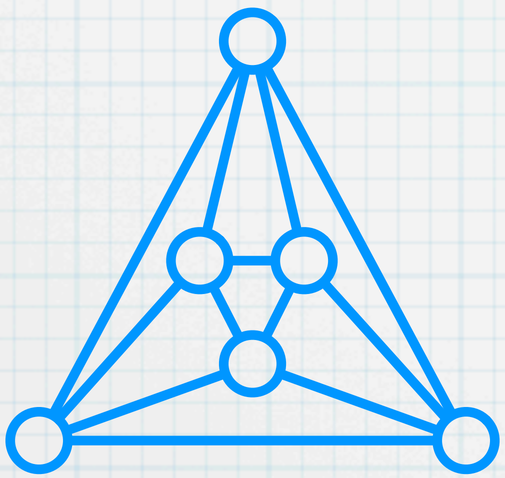
Exercise: Graph colouring



A five-colouring
No edge connects vertices of the same colour

Exercise: Graph colouring

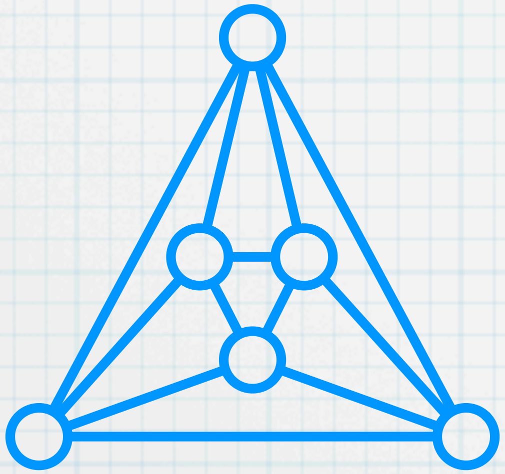
Input: Graph G=(V,E), integer k
Ouput: Can G be coloured with k colours?



Solve using brute force

Exercise: Graph colouring

Input: Graph G=(V,E), integer k
Ouput: Can G be coloured with k colours?

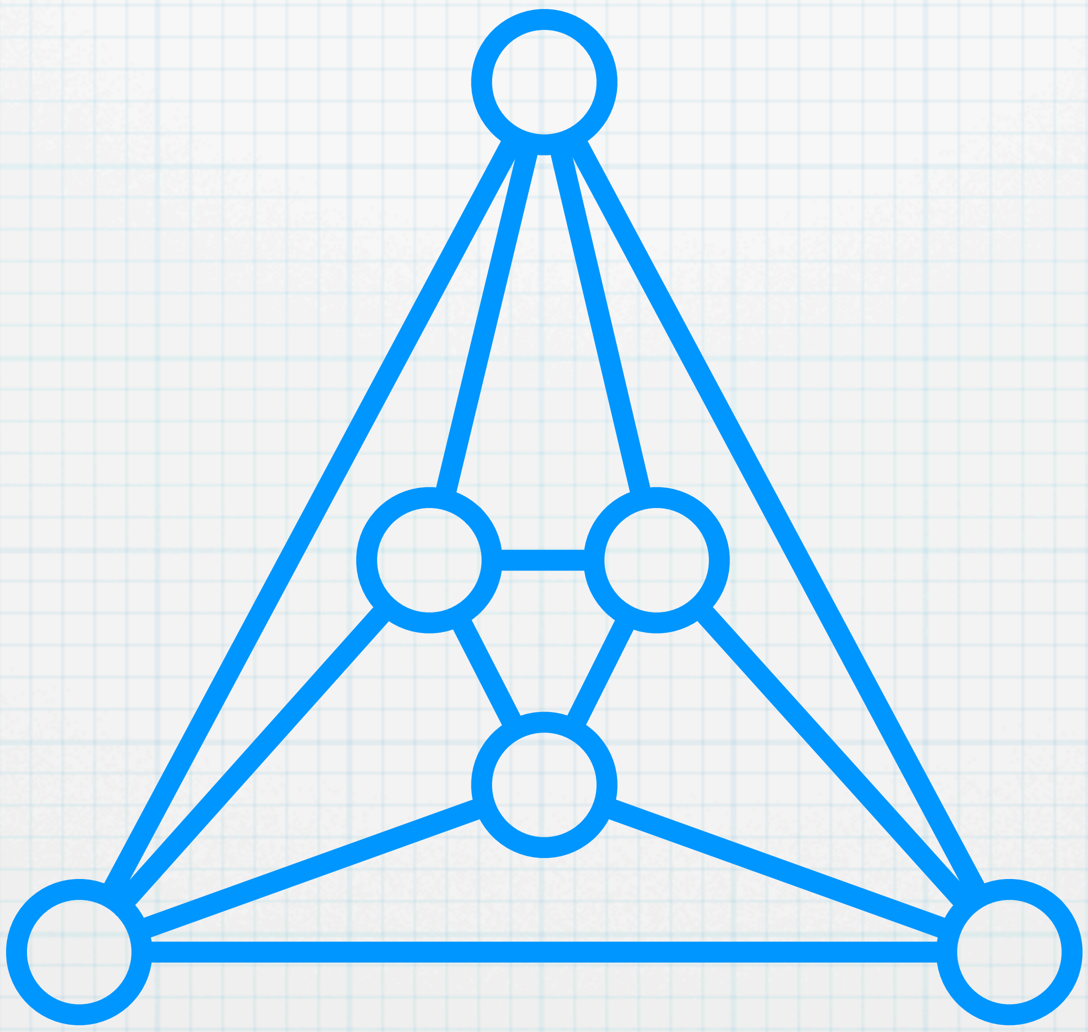


Solve using brute force

Time $O^*(n^k)$

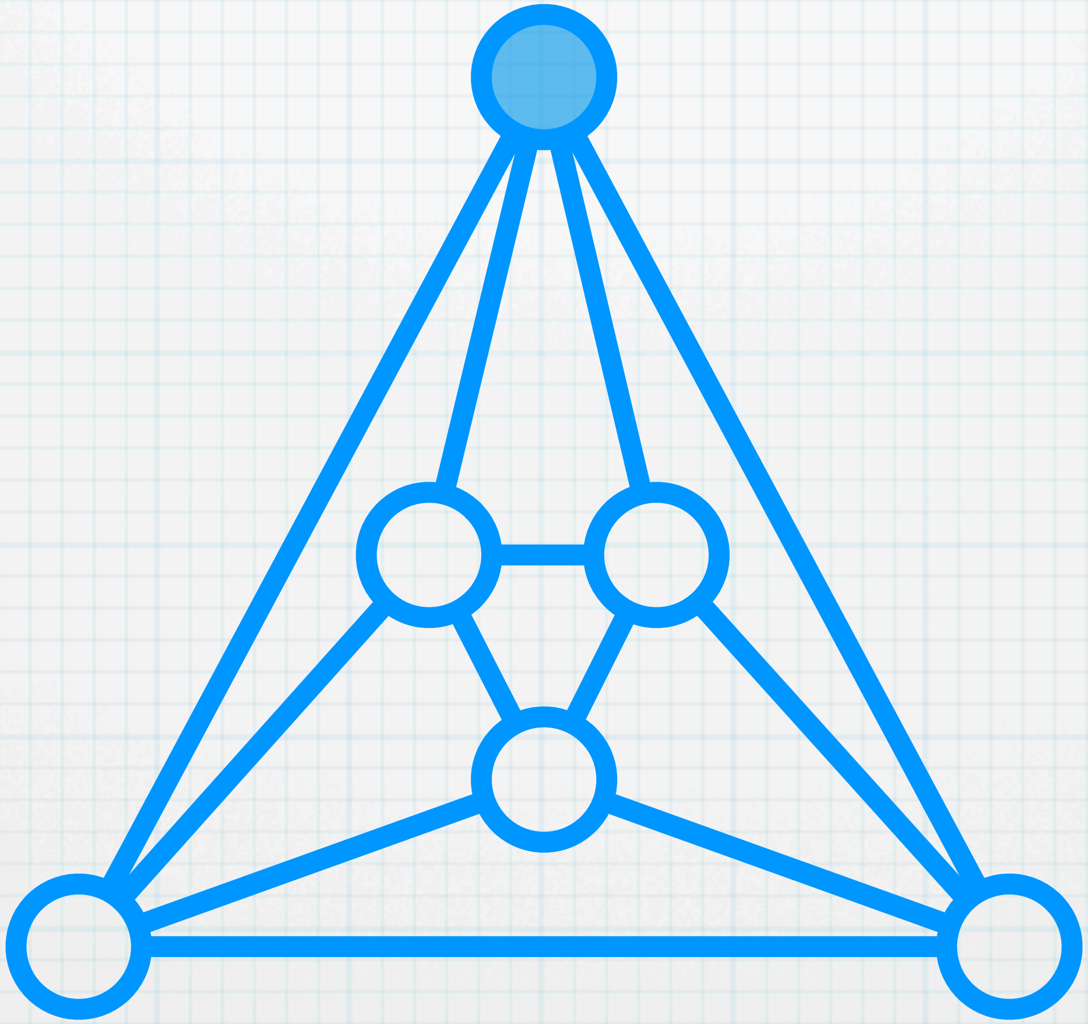
Greedy

Exercise: Graph colouring



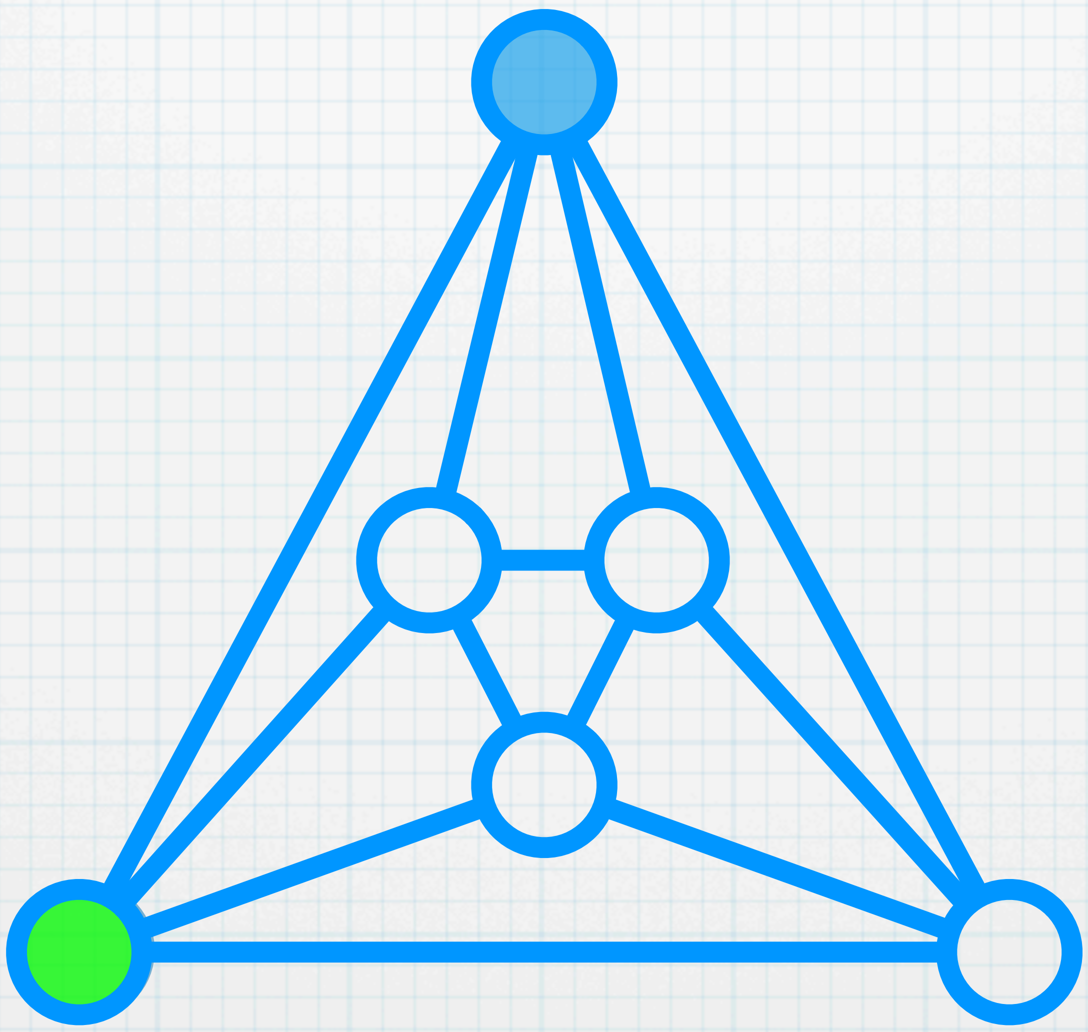
Does this always work?

Exercise: Graph colouring



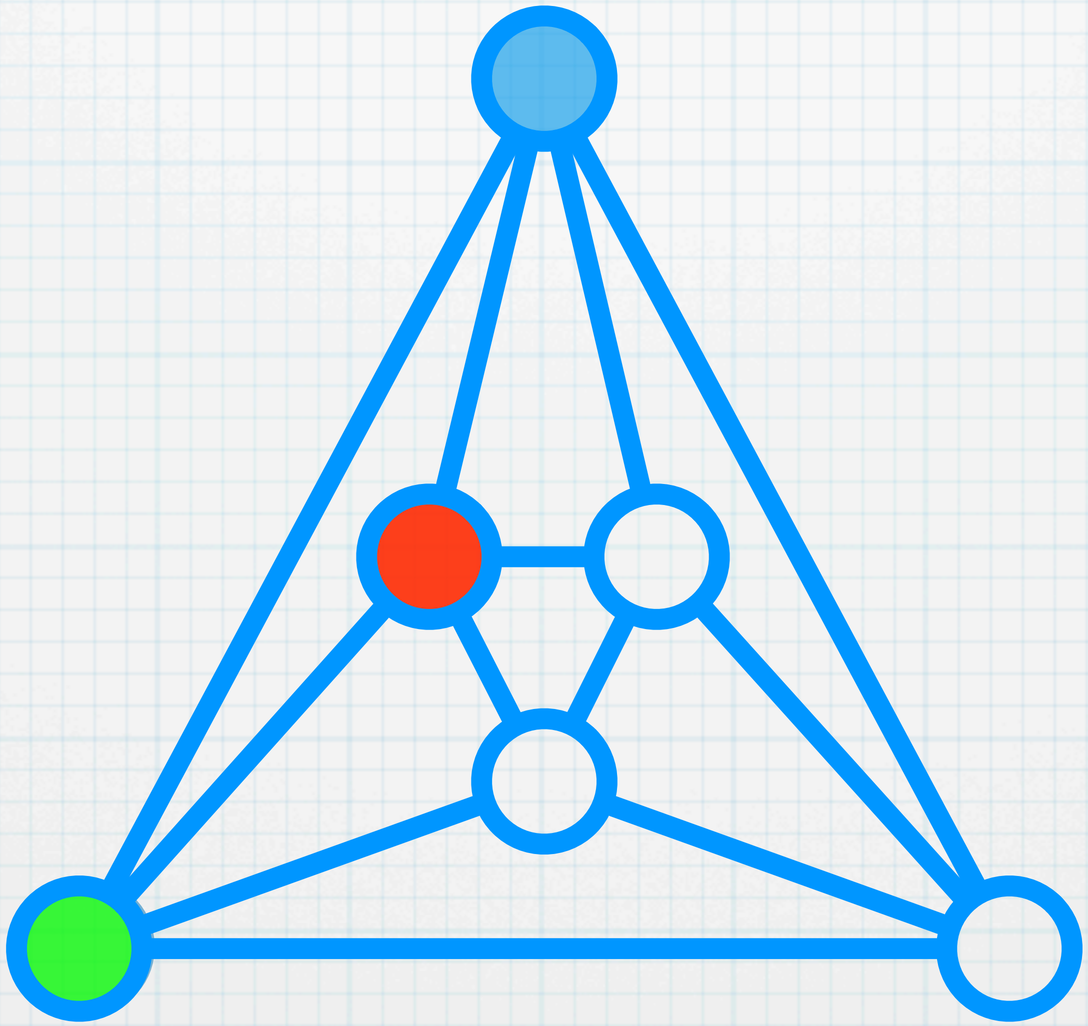
Does this always work?

Exercise: Graph colouring



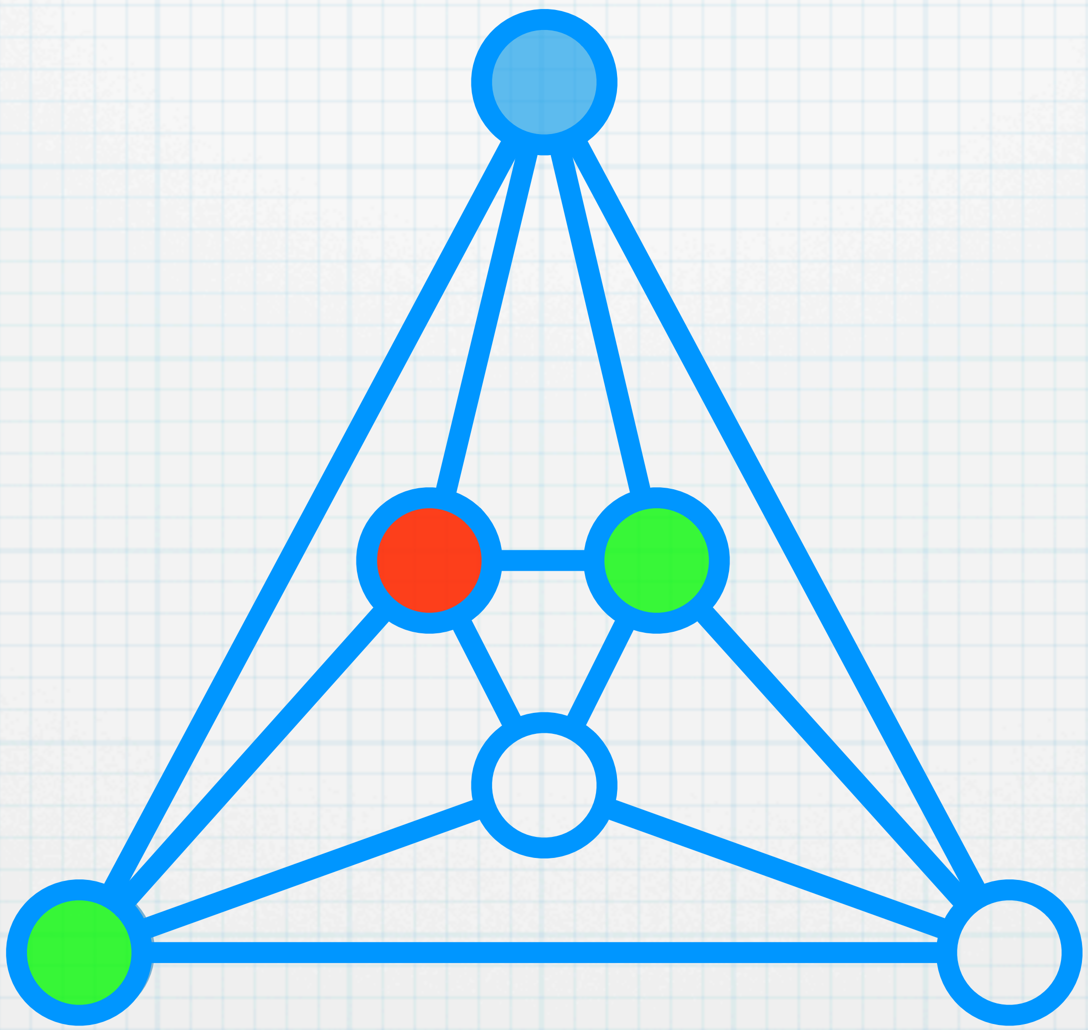
Does this always work?

Exercise: Graph colouring



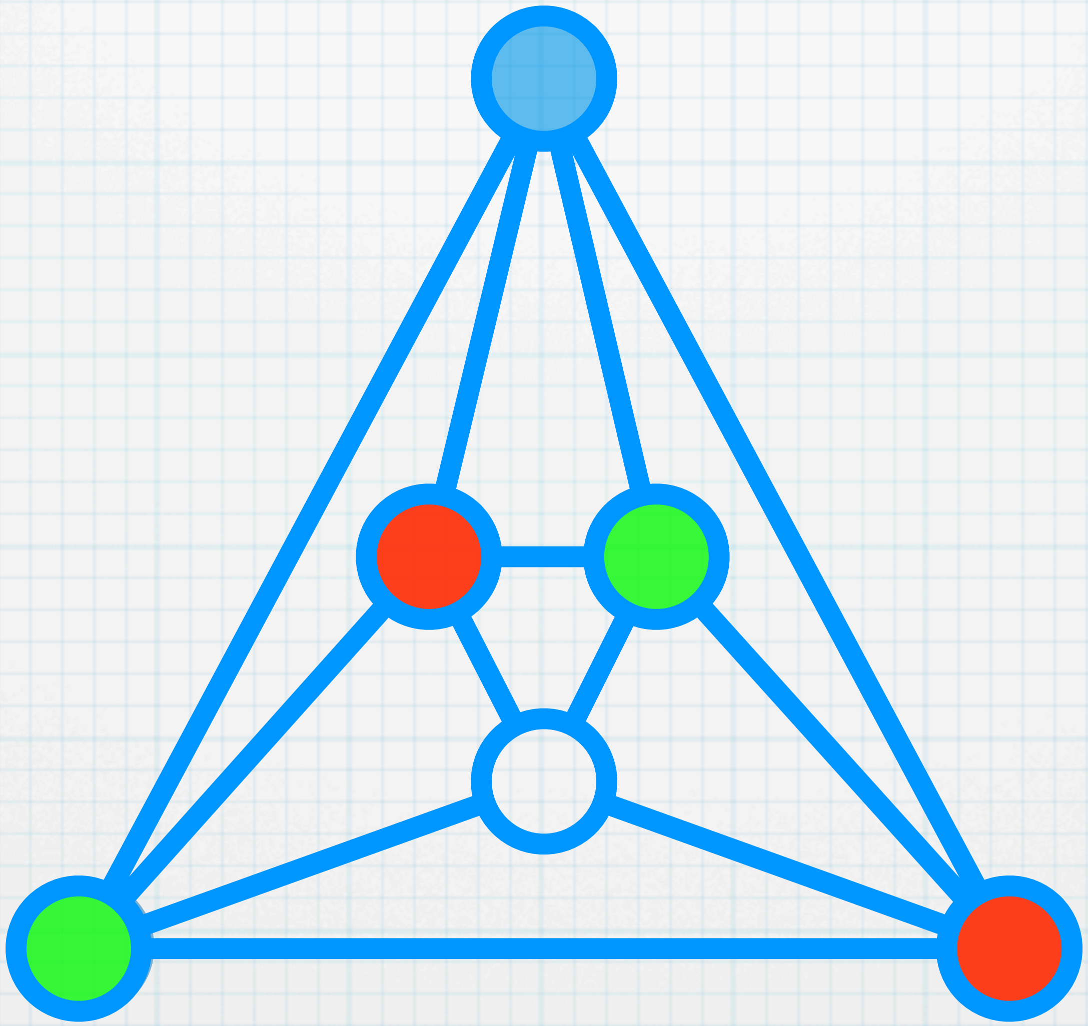
Does this always work?

Exercise: Graph colouring



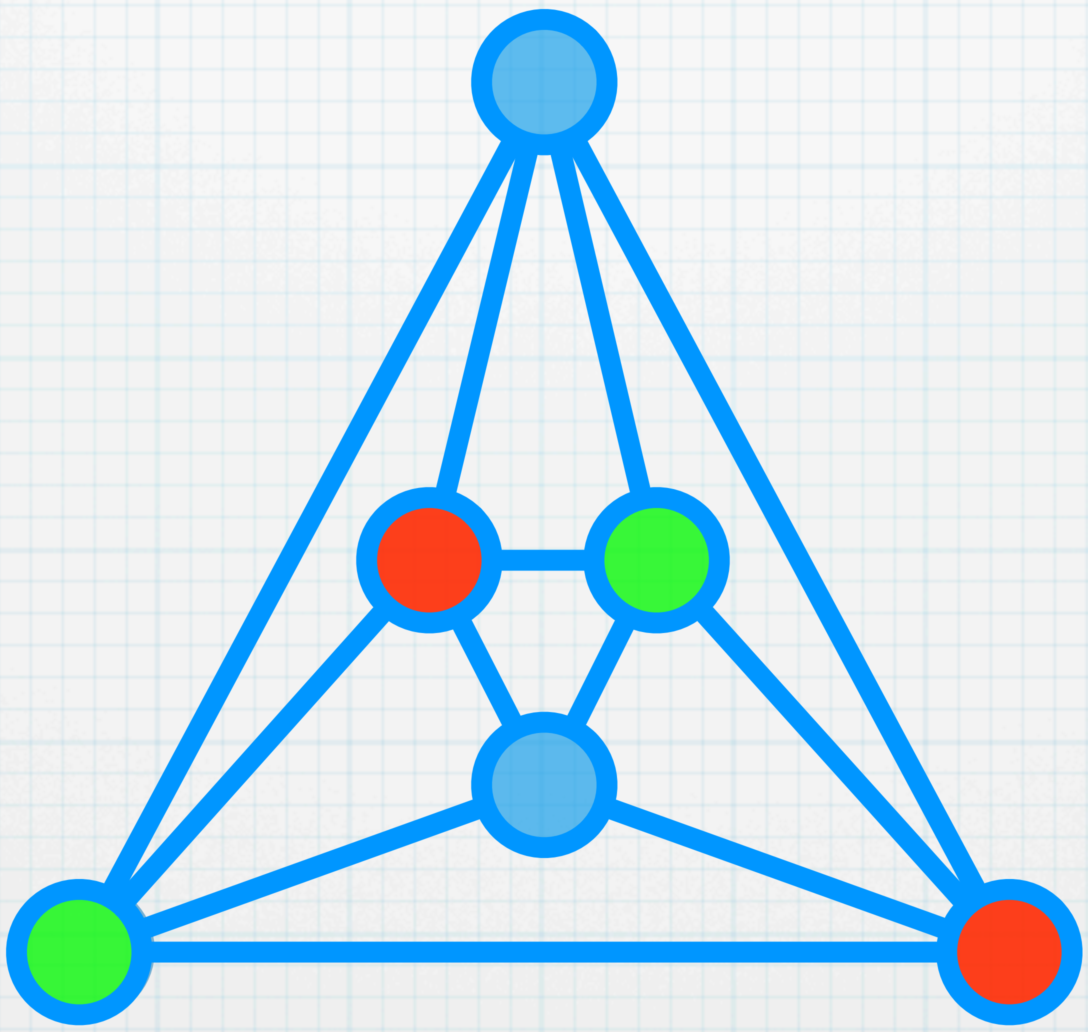
Does this always work?

Exercise: Graph colouring



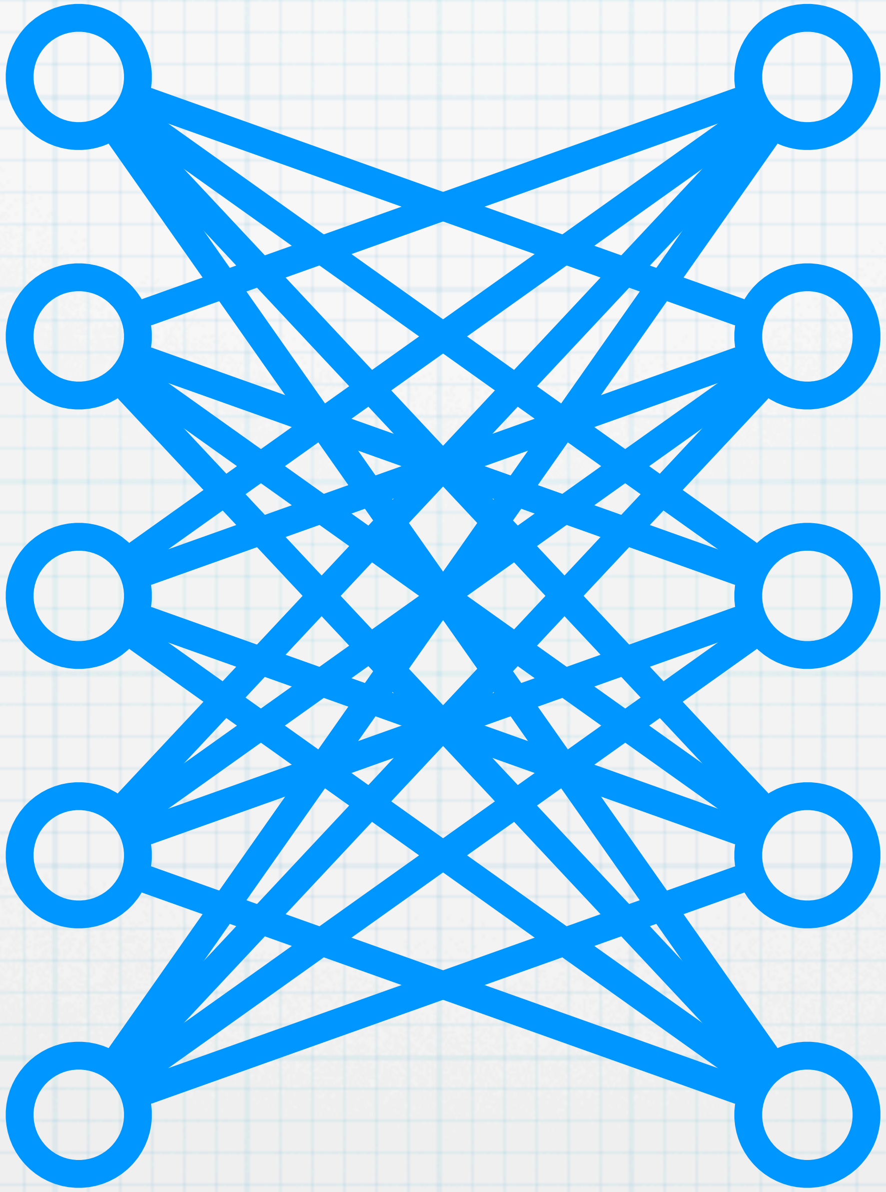
Does this always work?

Exercise: Graph colouring

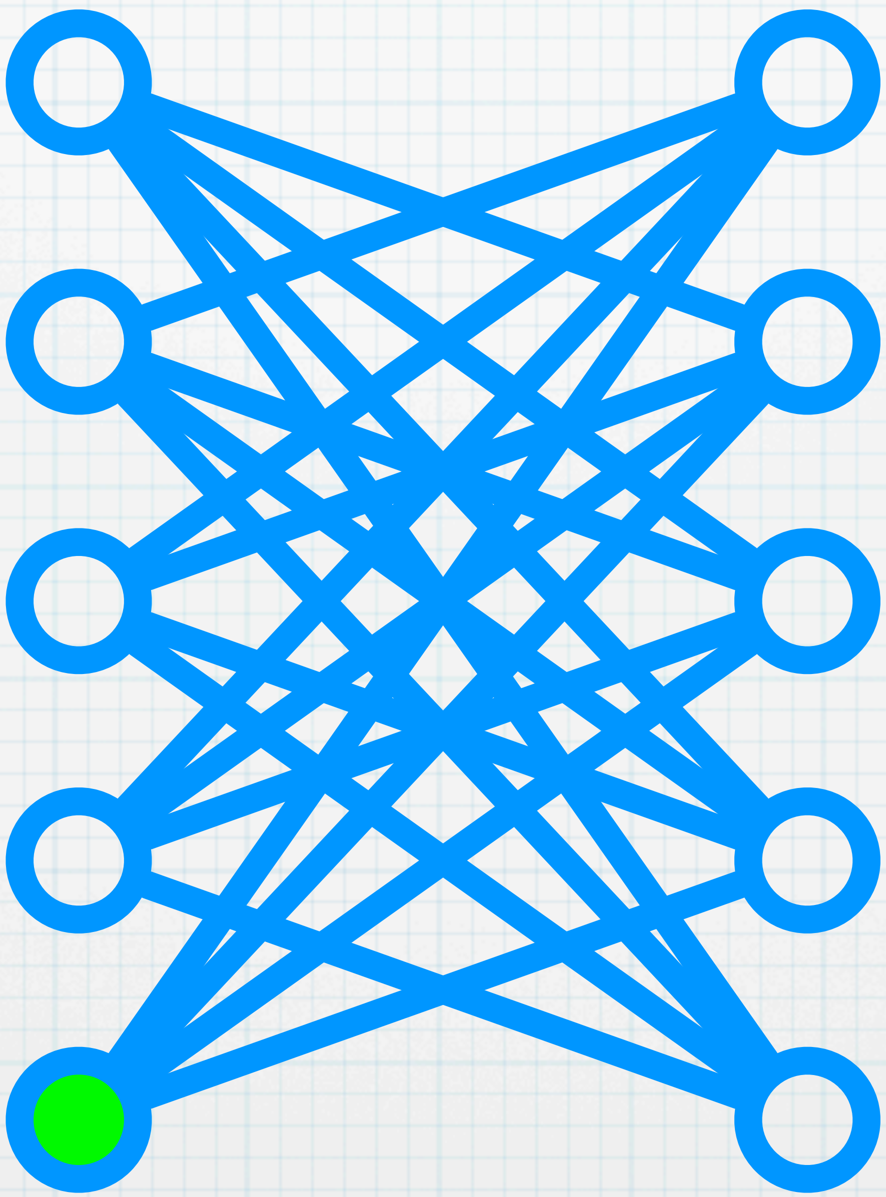


Does this always work?

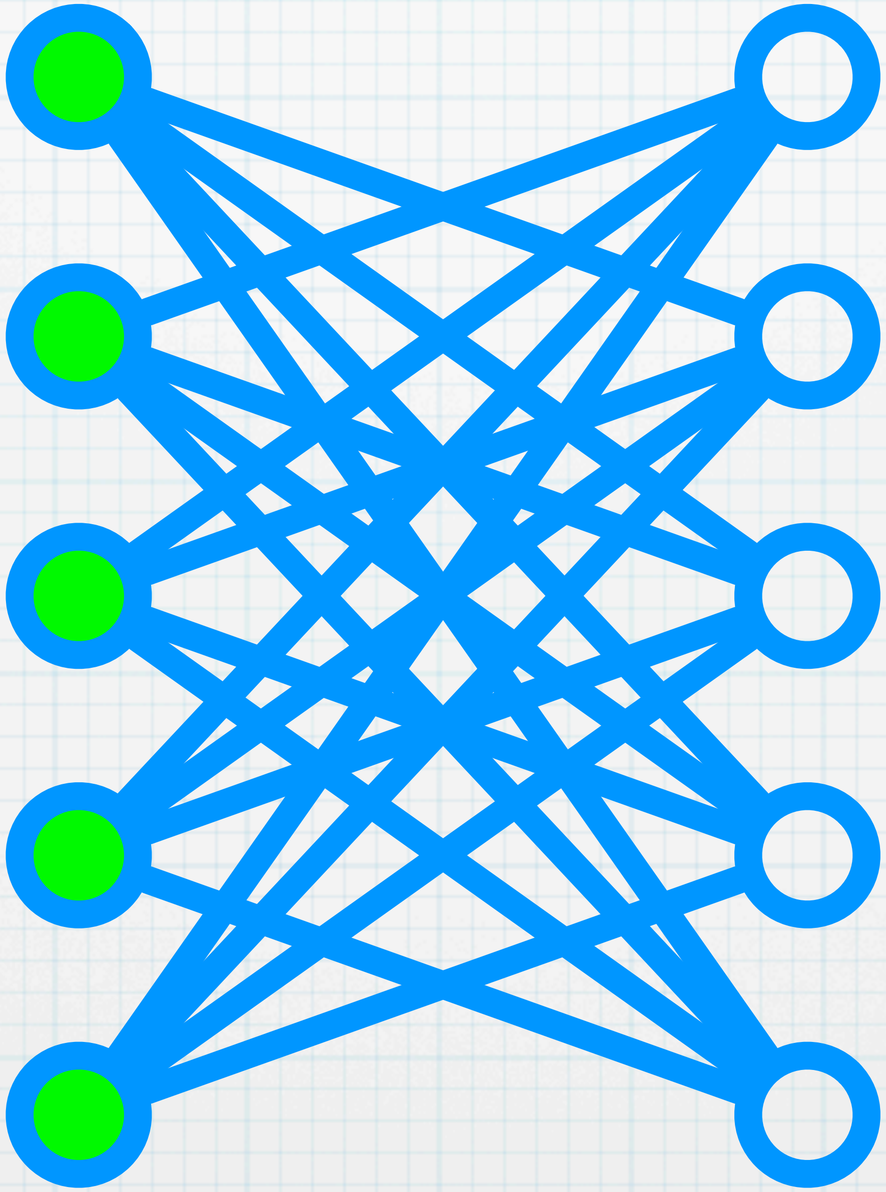
Exercise: Graph colouring



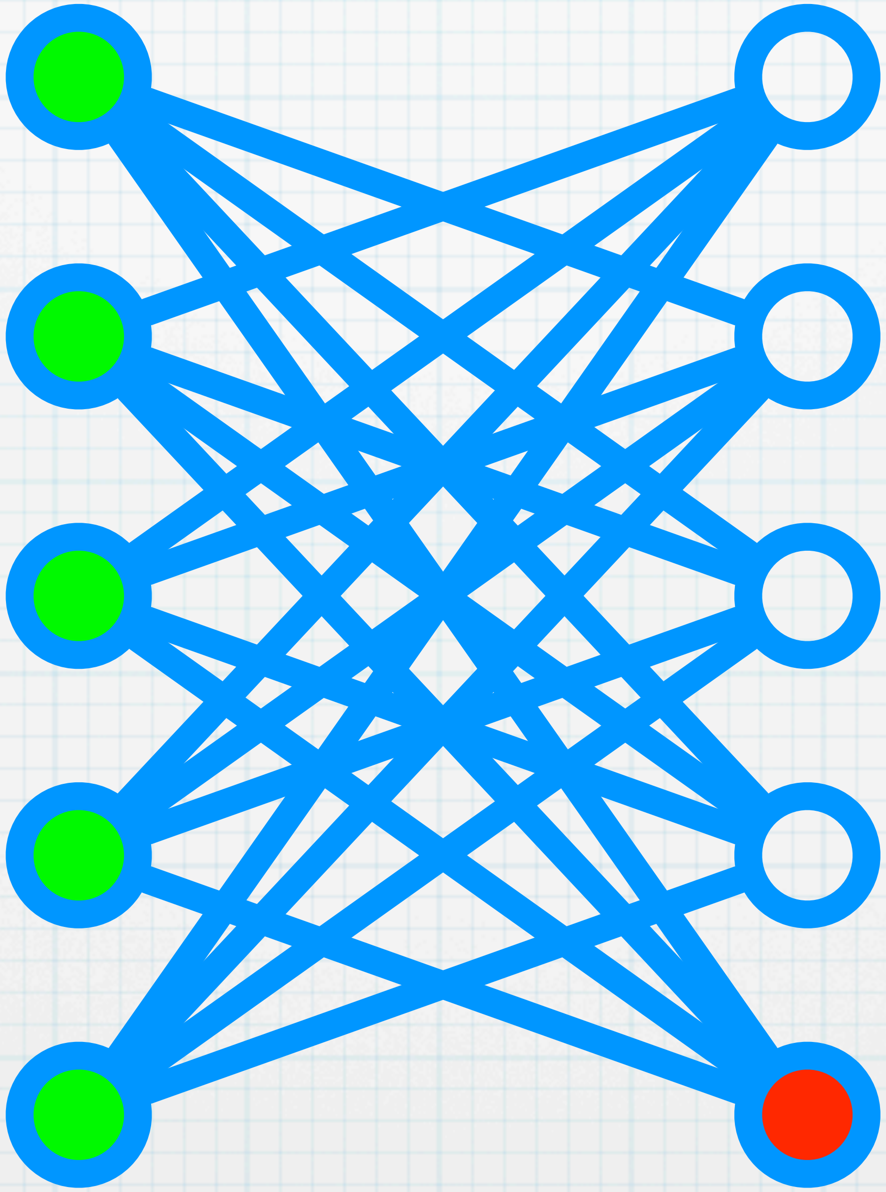
Exercise: Graph colouring



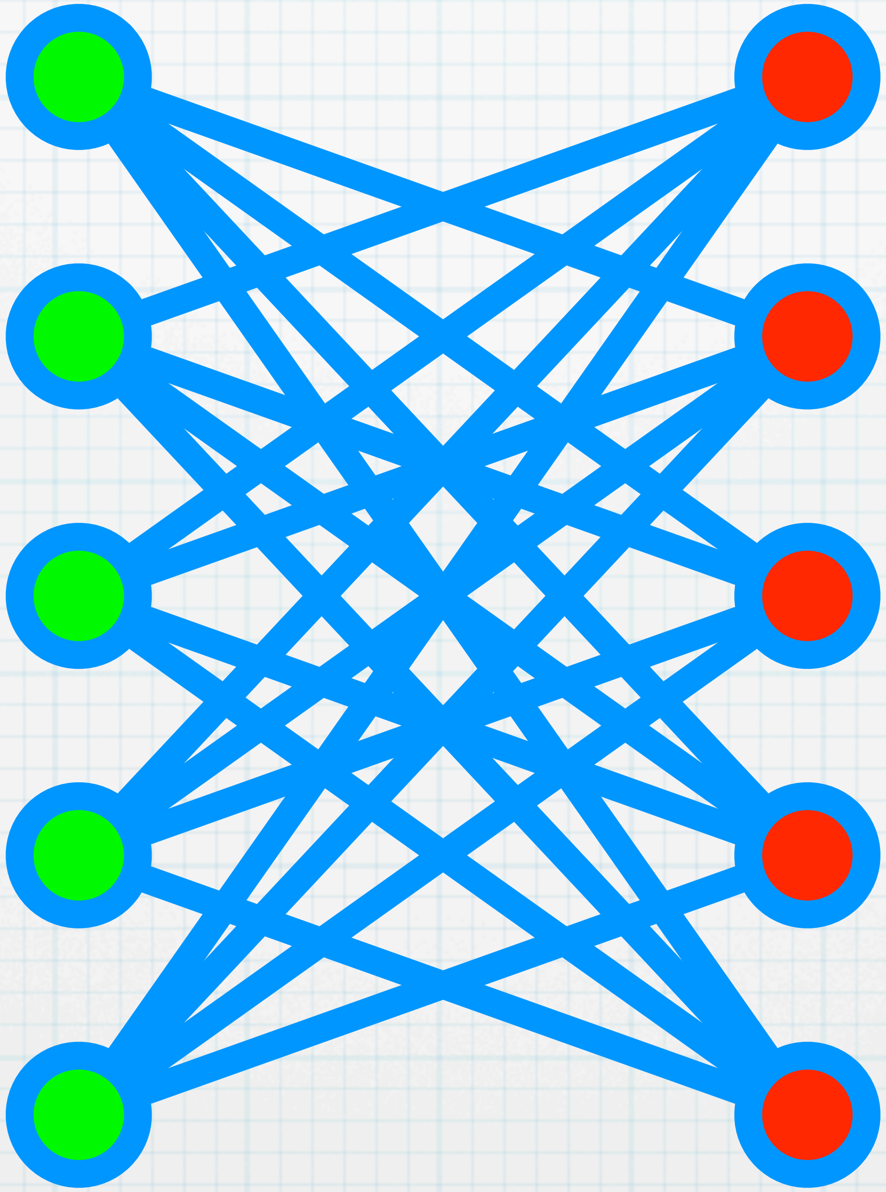
Exercise: Graph colouring



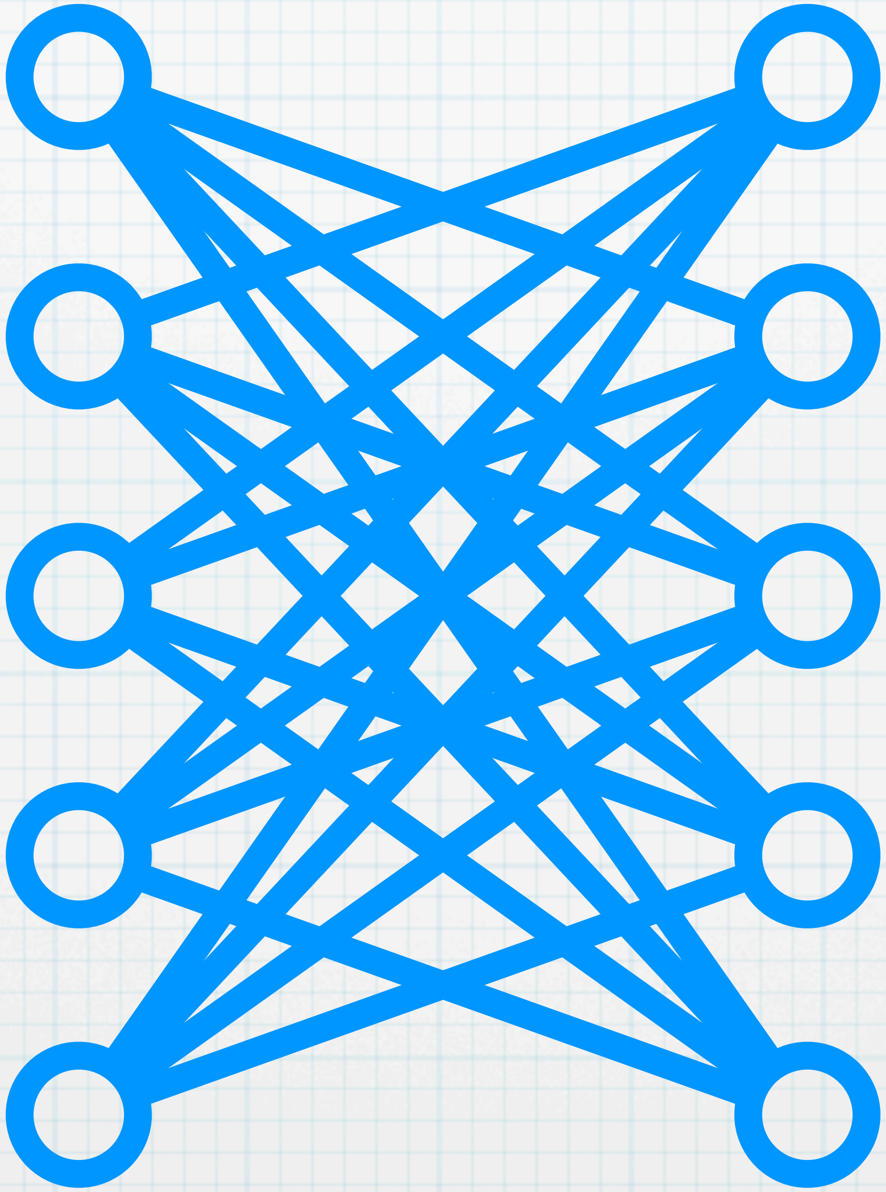
Exercise: Graph colouring



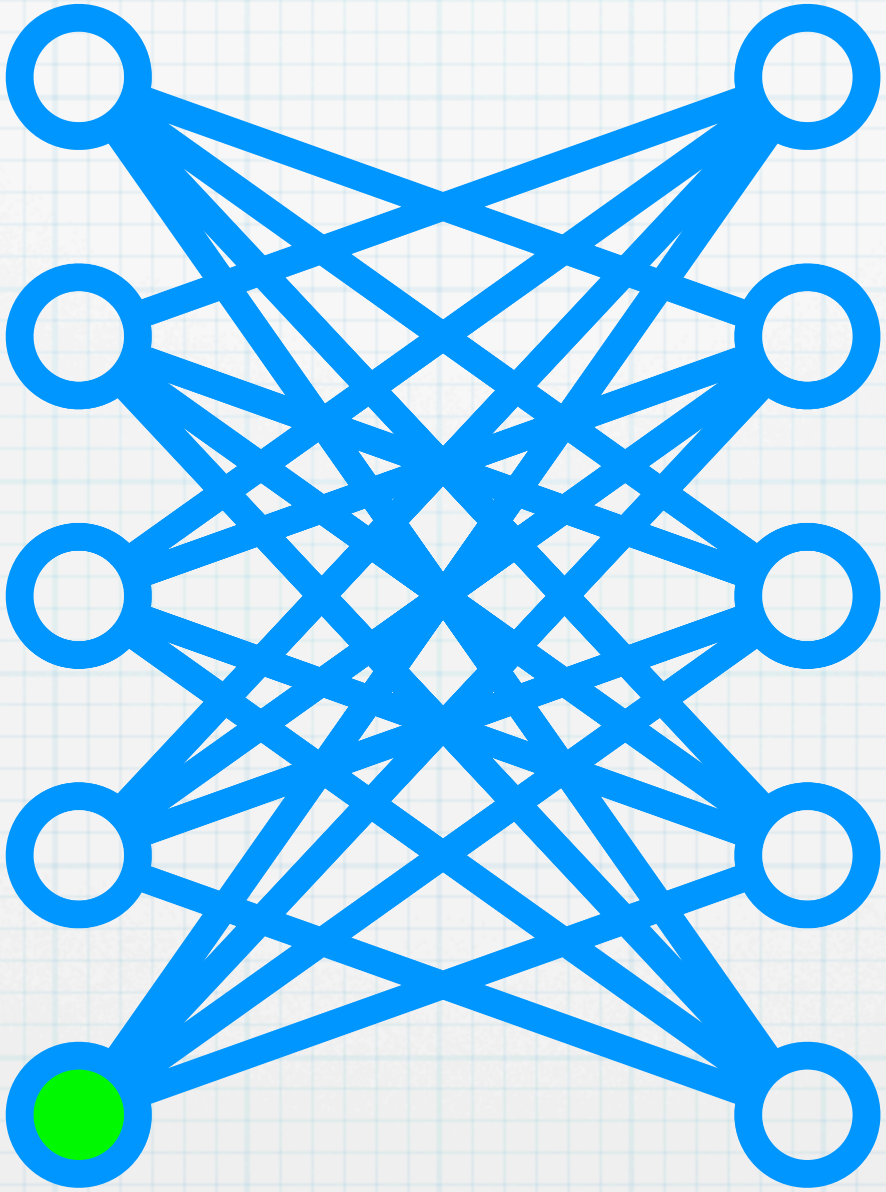
Exercise: Graph colouring



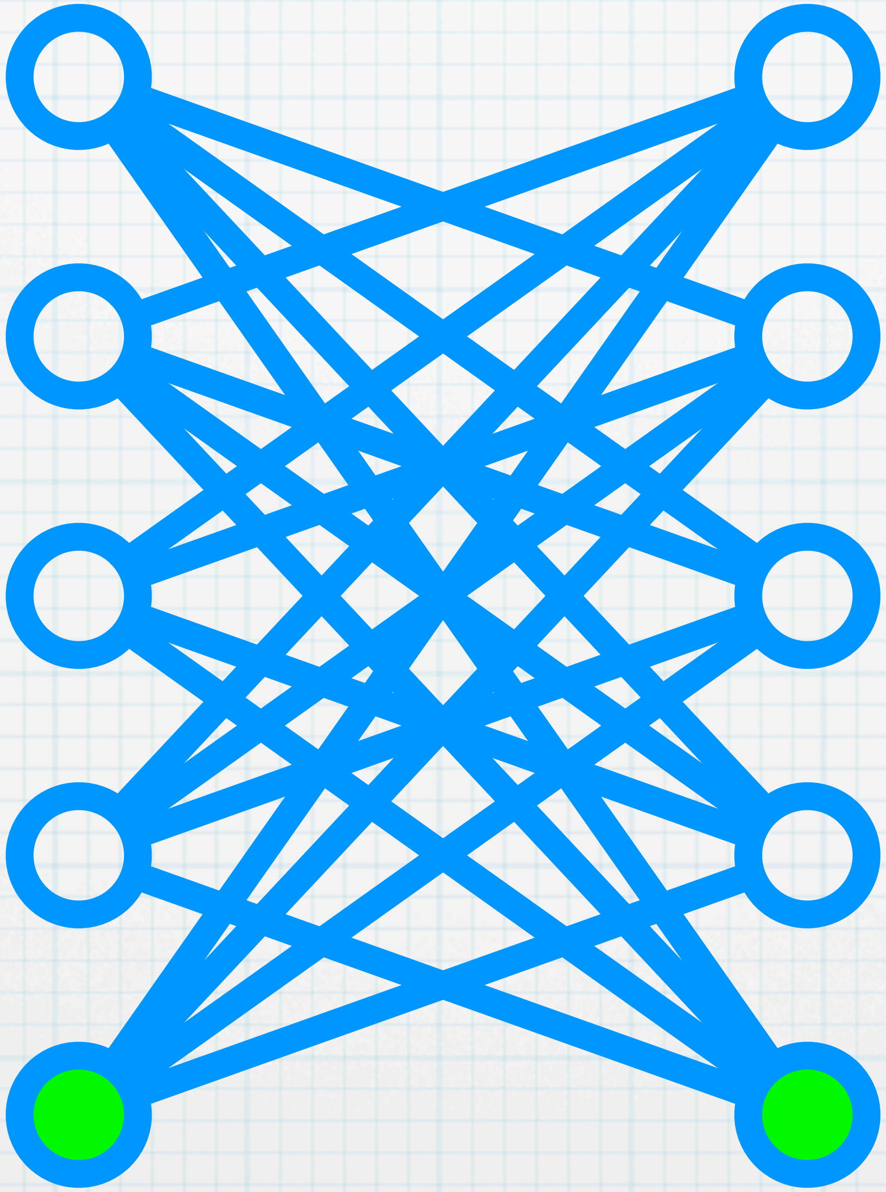
Exercise: Graph colouring



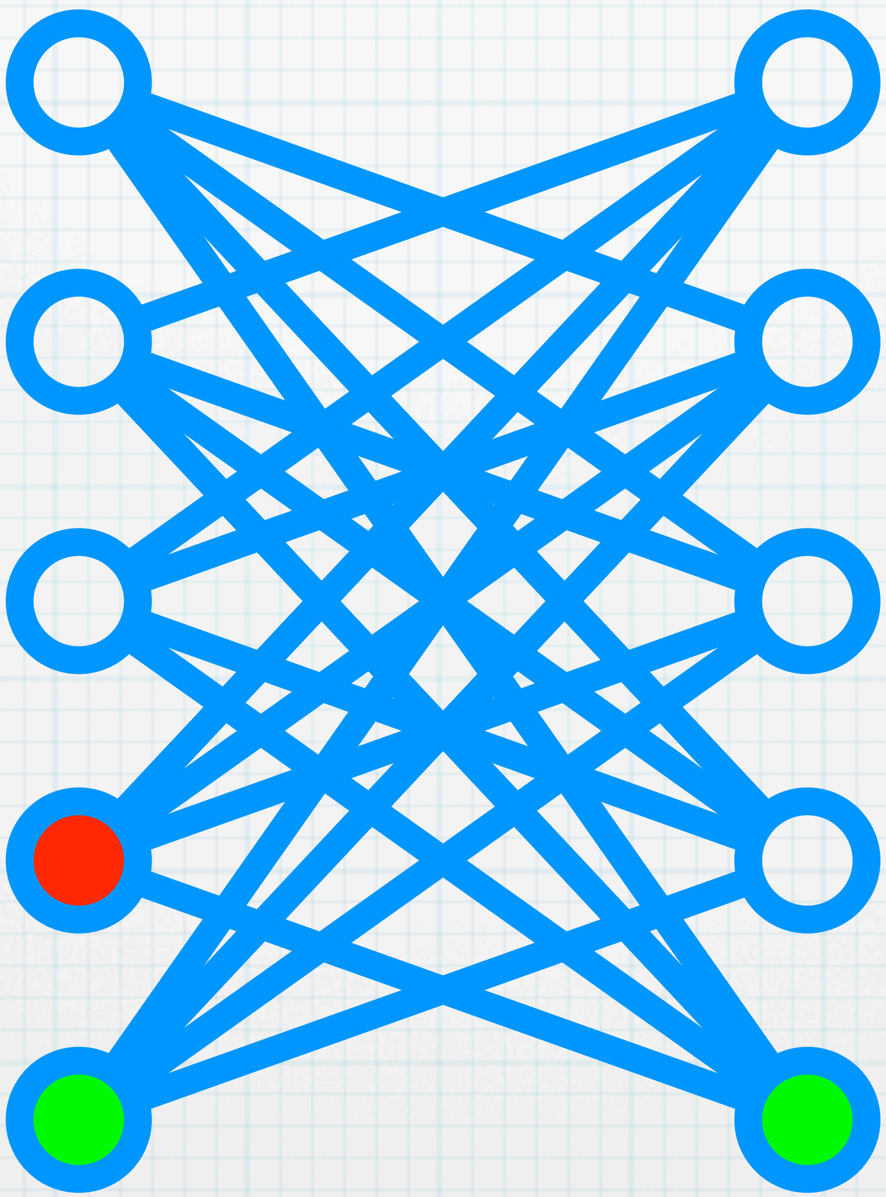
Exercise: Graph colouring



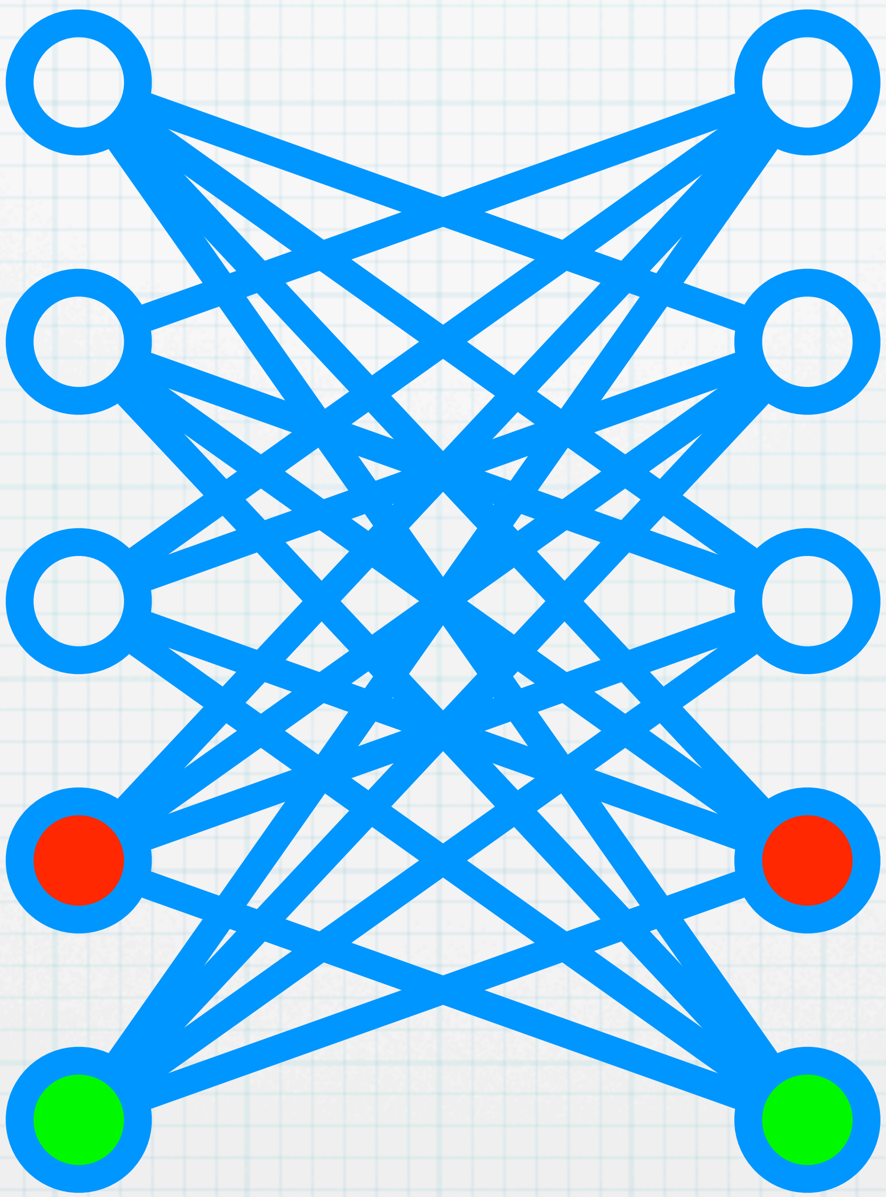
Exercise: Graph colouring



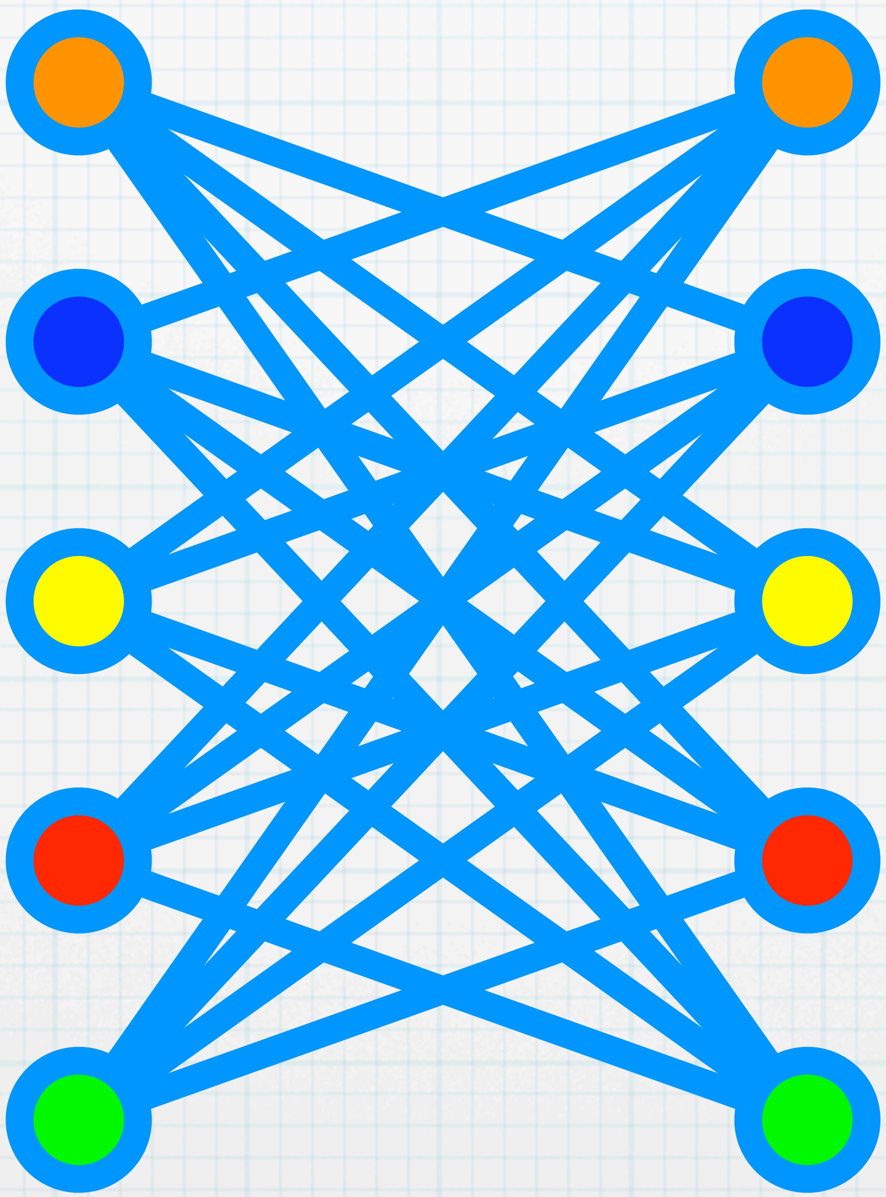
Exercise: Graph colouring



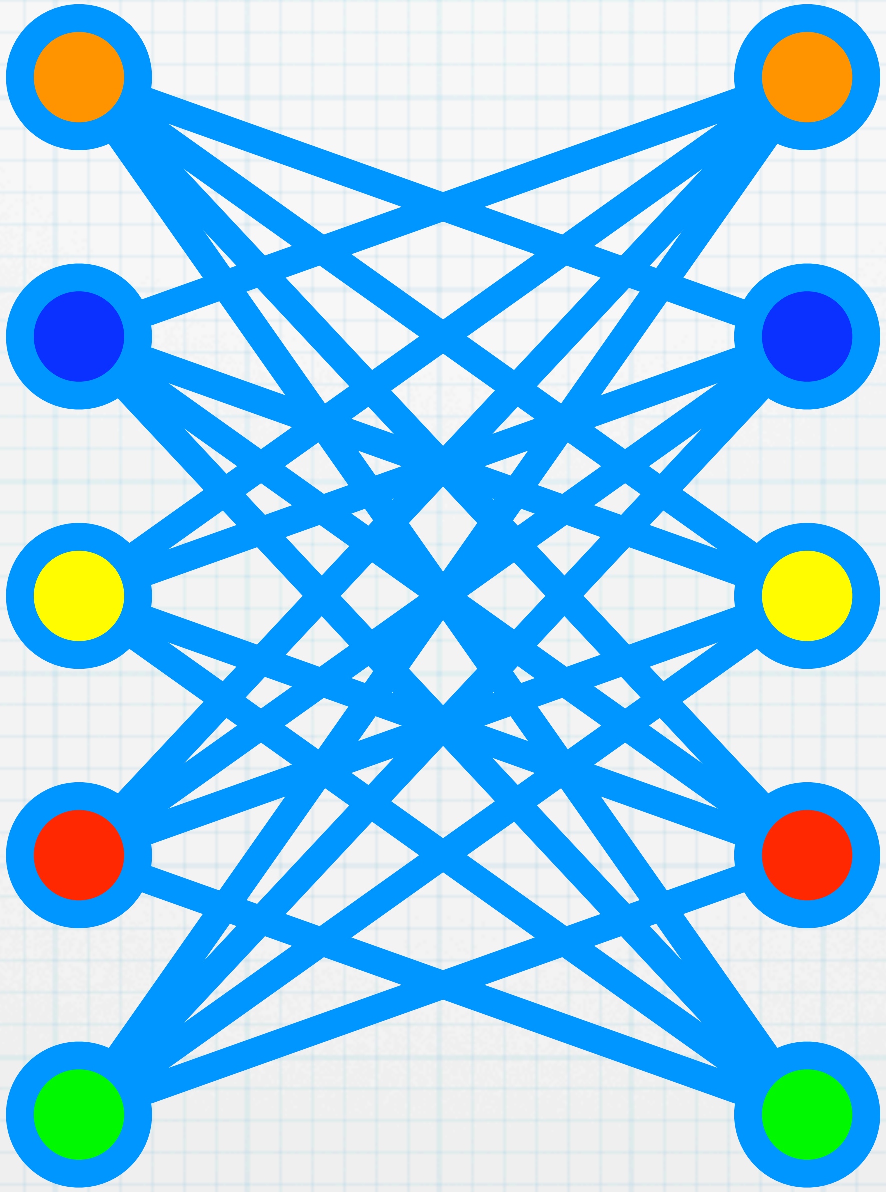
Exercise: Graph colouring



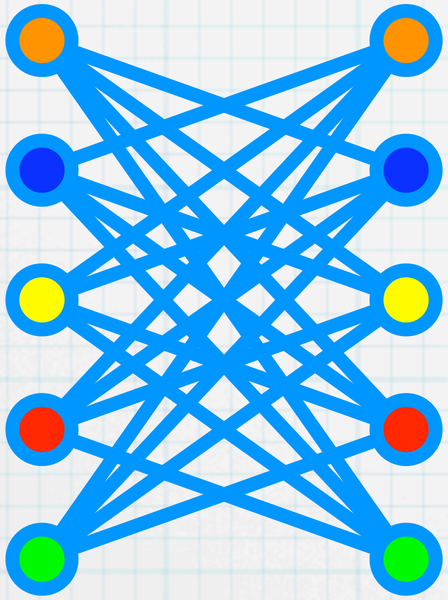
Exercise: Graph colouring



Exercise: Graph colouring



Exercise: Graph colouring



Is every k-colourable graph greedily k-colourable (for some ordering)?

Conclude: Graph colouring in O(n!)

Recursion
("Reduce to self")

Recursion
("Reduce to self")

Recursion
(“Reduce to self”)

Decrease and conquer

Divide and conquer

Recursion
("Reduce to self")

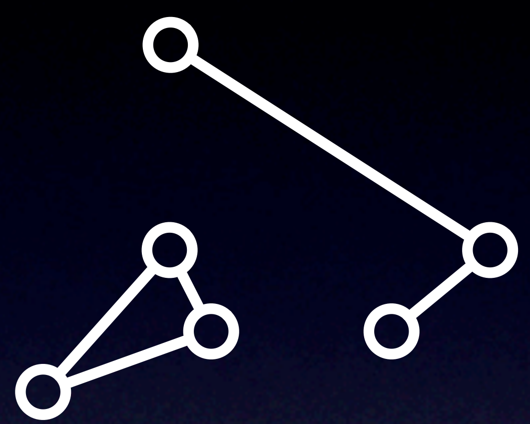
Decrease and conquer	Divide and conquer
Insertion sort	Mergesort

Recursion
("Reduce to self")

Decrease and conquer	Divide and conquer
Insertion sort	Mergesort
$a^n = a \cdot a^{n-1}$	$a^n = a^{n/2} \cdot a^{n/2}$

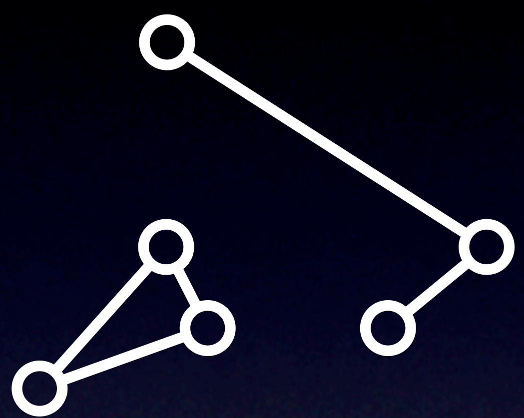
Decrease and conquer

Independent set



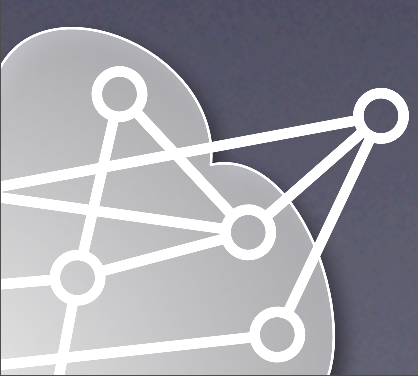
Degree ≤ 2: easy

Independent set

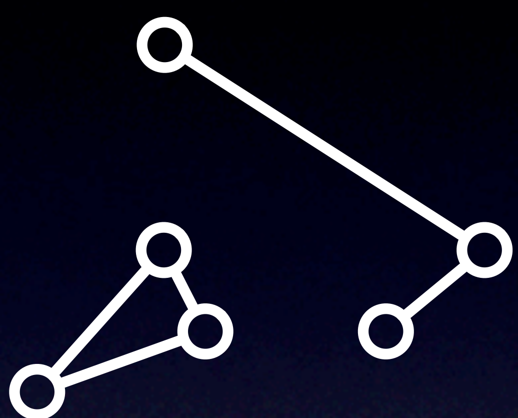


Degree ≤ 2: easy

Instance of size n



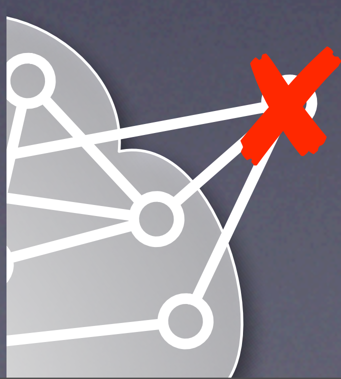
Independent set



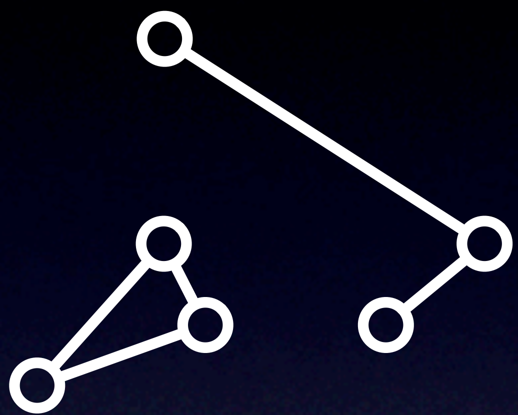
Degree ≤ 2: easy

Instance of size n

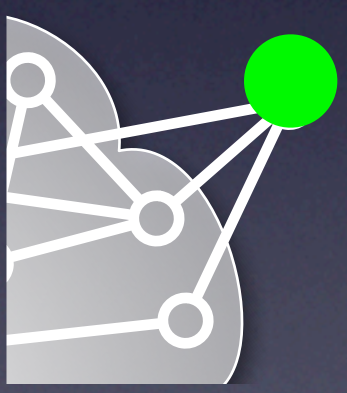
Two new instances of size $n-1$

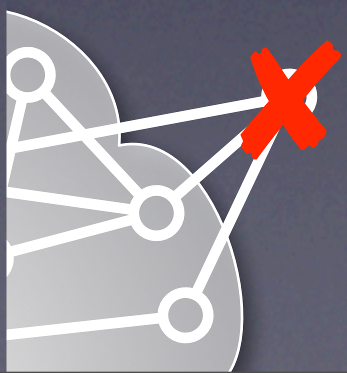


Independent set

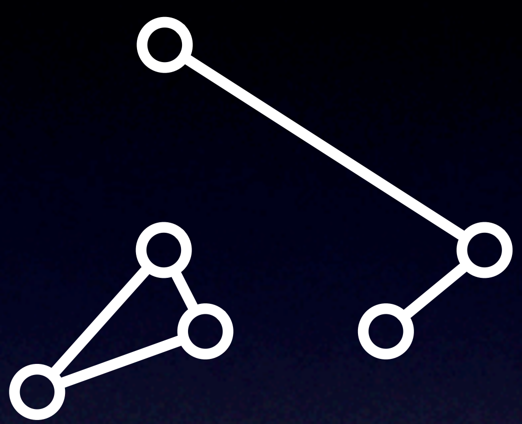


Instance of size n

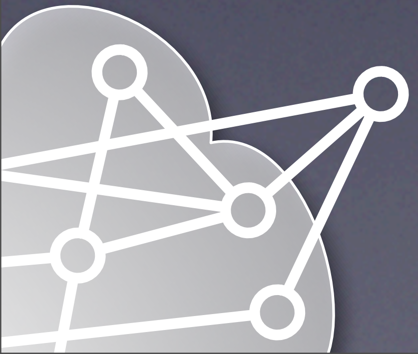


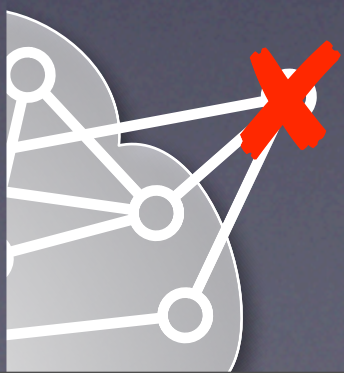


Independent set

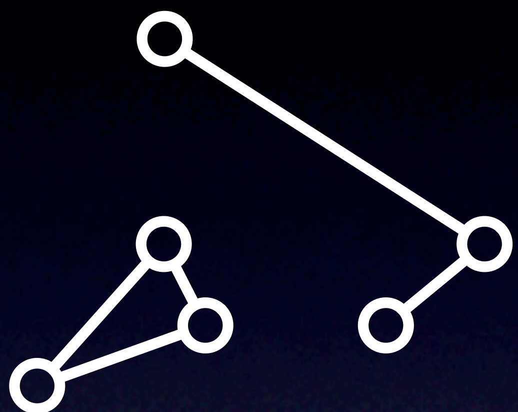


Instance of size n

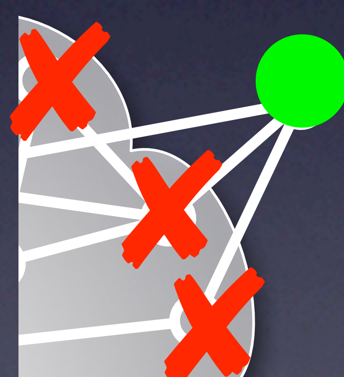




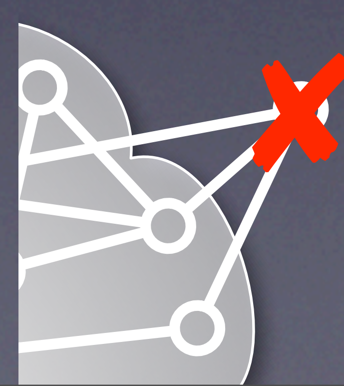
Independent set



Instance of size n



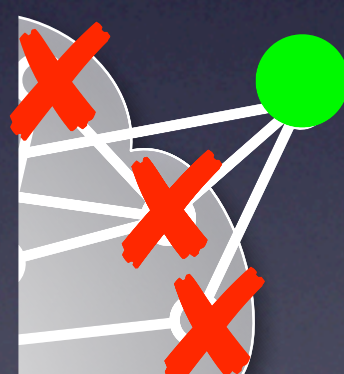
New instance of size $n-4$



New instance of size $n-1$

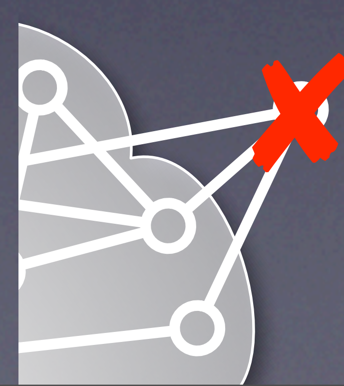
Independent set

$$T(n) = T(n-1) + T(n-4)$$



New instance of size $n-4$

Instance of size n



New instance of size $n-1$

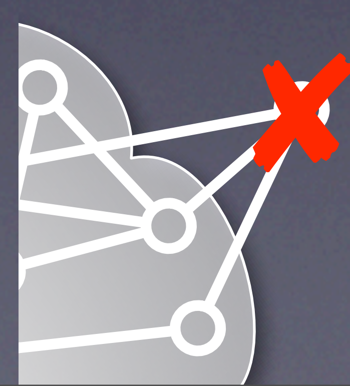
Independent set

$$T(n) = T(n-1) + T(n-4)$$

Time $O^*(1.39^n)$. Polyspace.

New instance of size $n-4$

Instance of size n



New instance of size $n-1$

3-Satisfiability

$$(\neg x \vee y \vee z) \wedge (x \vee \neg y \vee z) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

3-Satisfiability

$$(\neg x \vee y \vee z) \wedge \boxed{(x \vee \neg y \vee z)} \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

3-Satisfiability

$$(\neg x \vee y \vee z) \wedge \boxed{(x \vee \neg y \vee z)} \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

$$(\neg x \vee y \vee z) \wedge (\mathrm{T} \vee \mathrm{F} \vee \mathrm{F}) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

$$(\neg x \vee y \vee z) \wedge (x \vee \mathrm{T} \vee \mathrm{F}) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

$$(\neg x \vee y \vee z) \wedge (x \vee \neg y \vee \mathrm{T}) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

3-Satisfiability

$$T(n) = T(n-1) + T(n-2) + T(n-3)$$

$$(\neg x \vee y \vee z) \wedge \boxed{(x \vee \neg y \vee z)} \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

$$(\neg x \vee y \vee z) \wedge (\mathrm{T} \vee \mathrm{F} \vee \mathrm{F}) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

$$(\neg x \vee y \vee z) \wedge (x \vee \mathrm{T} \vee \mathrm{F}) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

$$(\neg x \vee y \vee z) \wedge (x \vee \neg y \vee \mathrm{T}) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

3-Satisfiability

$$T(n) = T(n-1) + T(n-2) + T(n-3)$$

Time $O^*(1.84^n)$. Polyspace.

$$(\neg x \vee y \vee z) \wedge \boxed{(x \vee \neg y \vee z)} \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

$$(\neg x \vee y \vee z) \wedge (\mathrm{T} \vee \mathrm{F} \vee \mathrm{F}) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

$$(\neg x \vee y \vee z) \wedge (x \vee \mathrm{T} \vee \mathrm{F}) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

$$(\neg x \vee y \vee z) \wedge (x \vee \neg y \vee \mathrm{T}) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

TSP

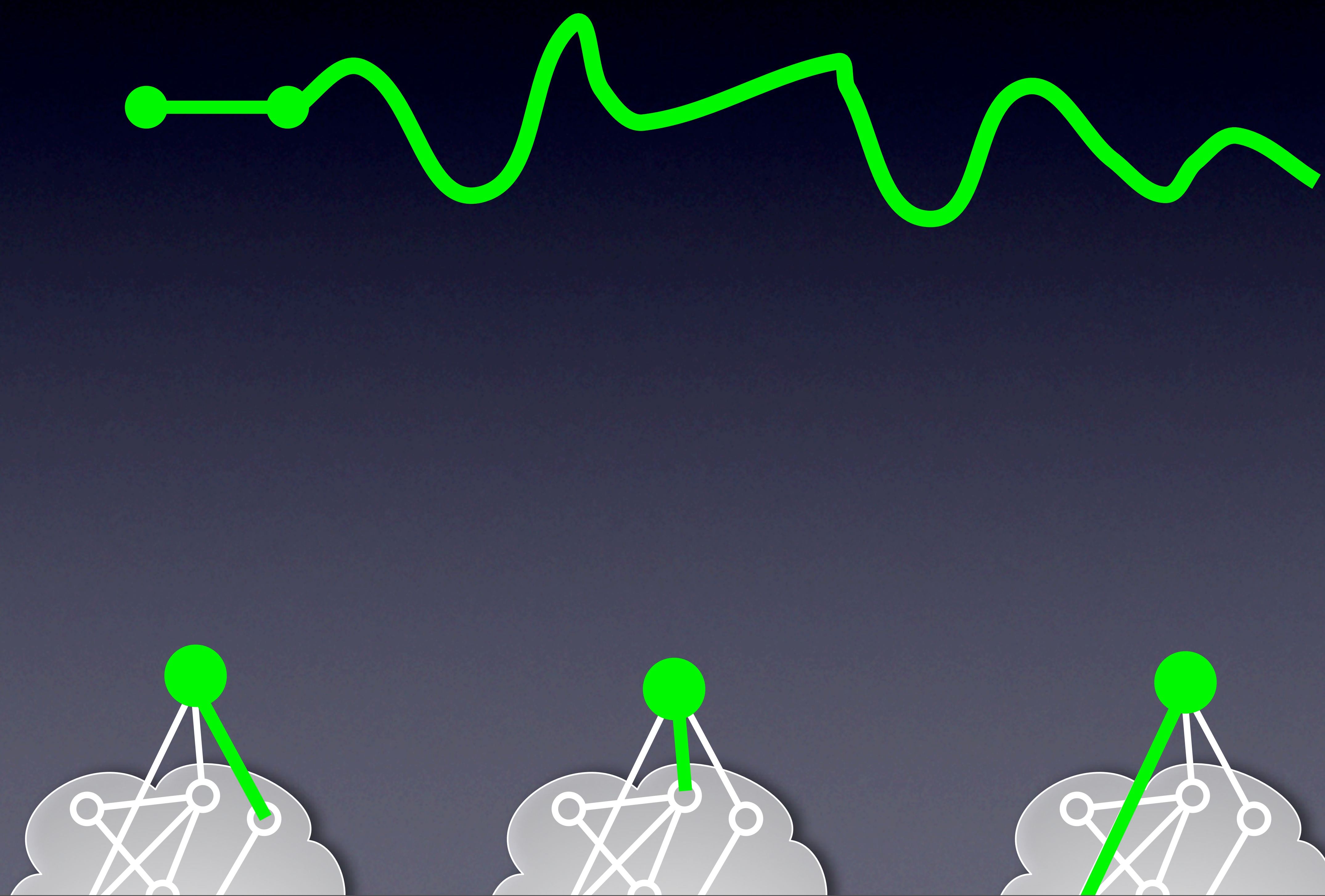
TSP

TSP

TSP

TSP

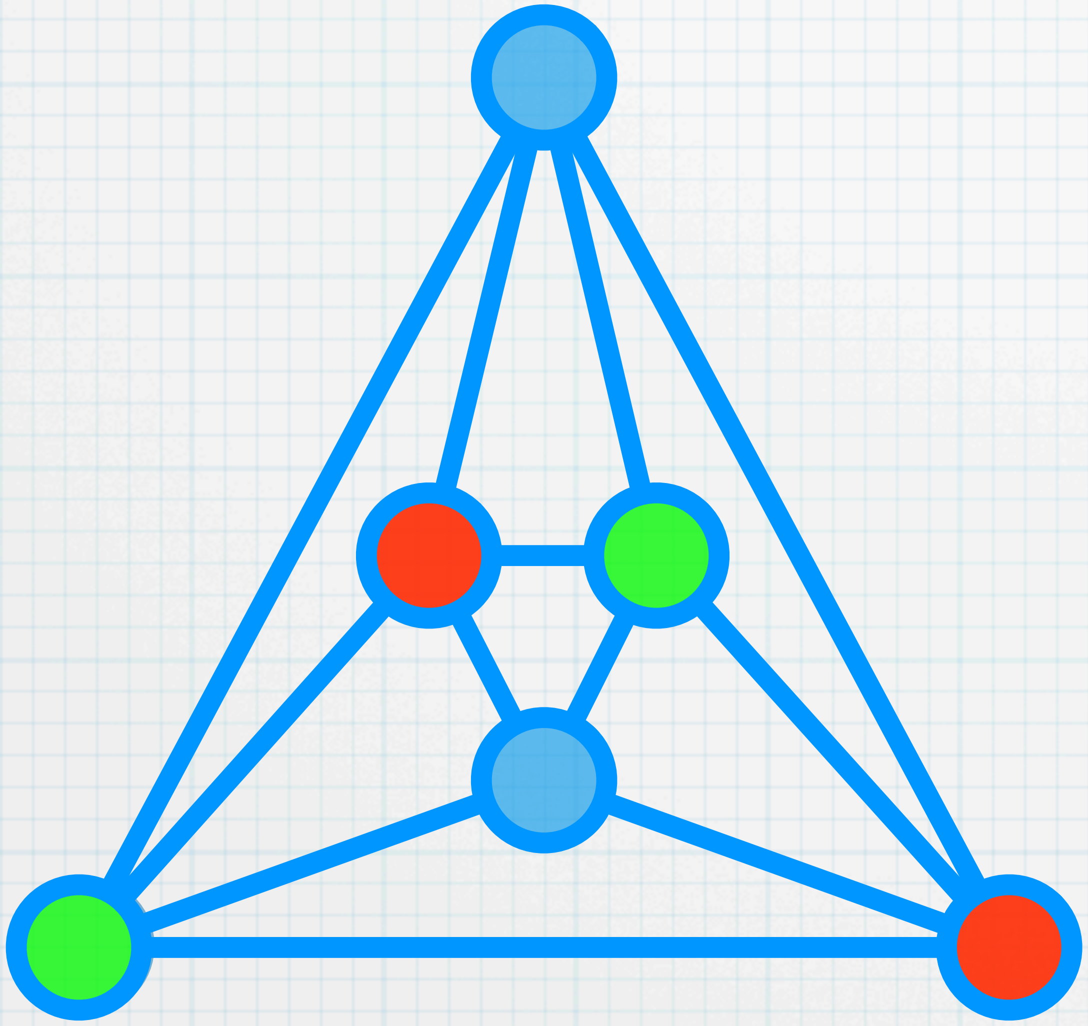
TSP



TSP

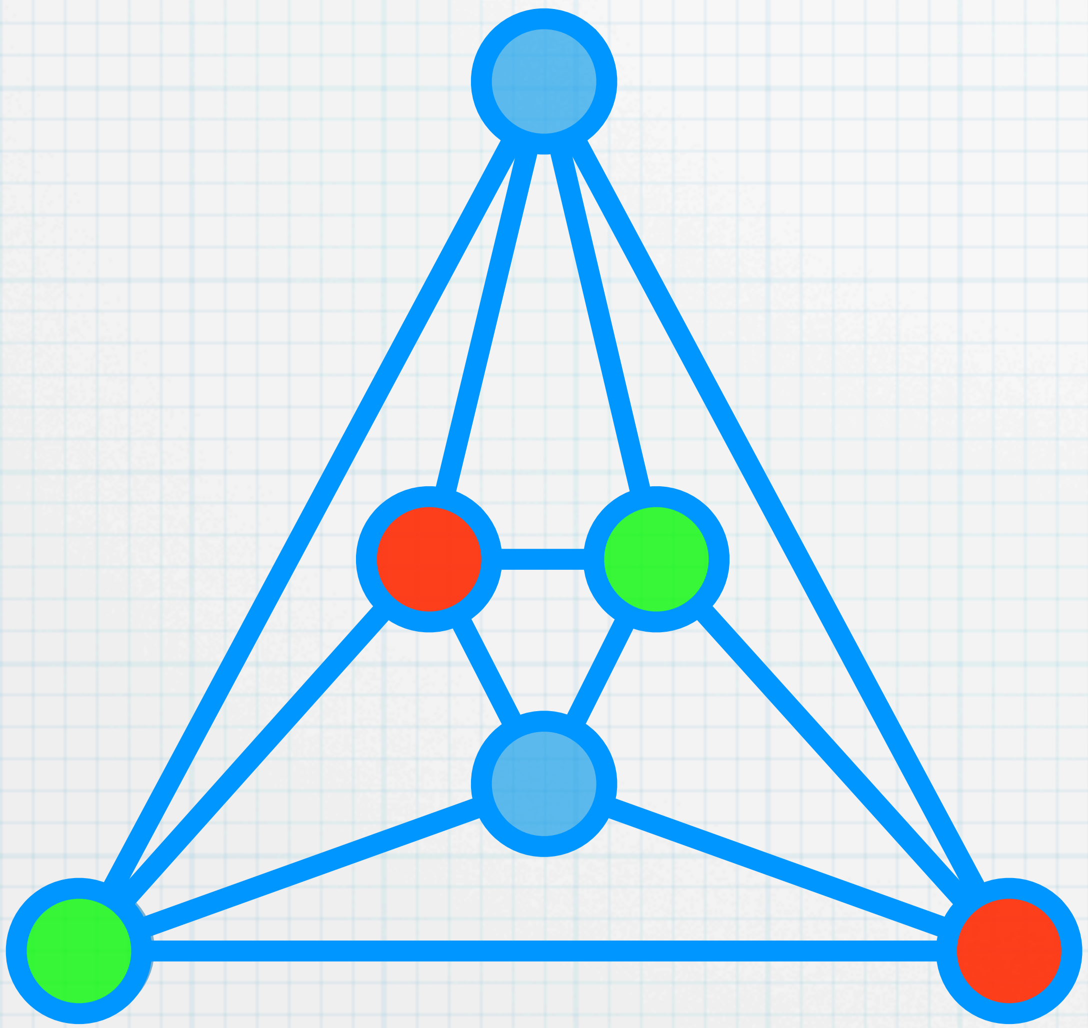
$$T(n) = n \cdot T(n-1)$$

Exercise: Graph colouring

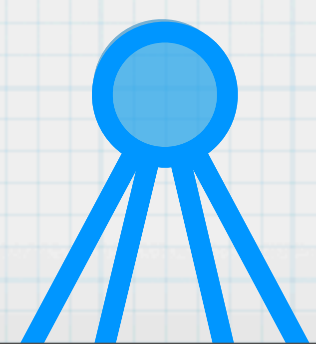


First approach:
split on vertices

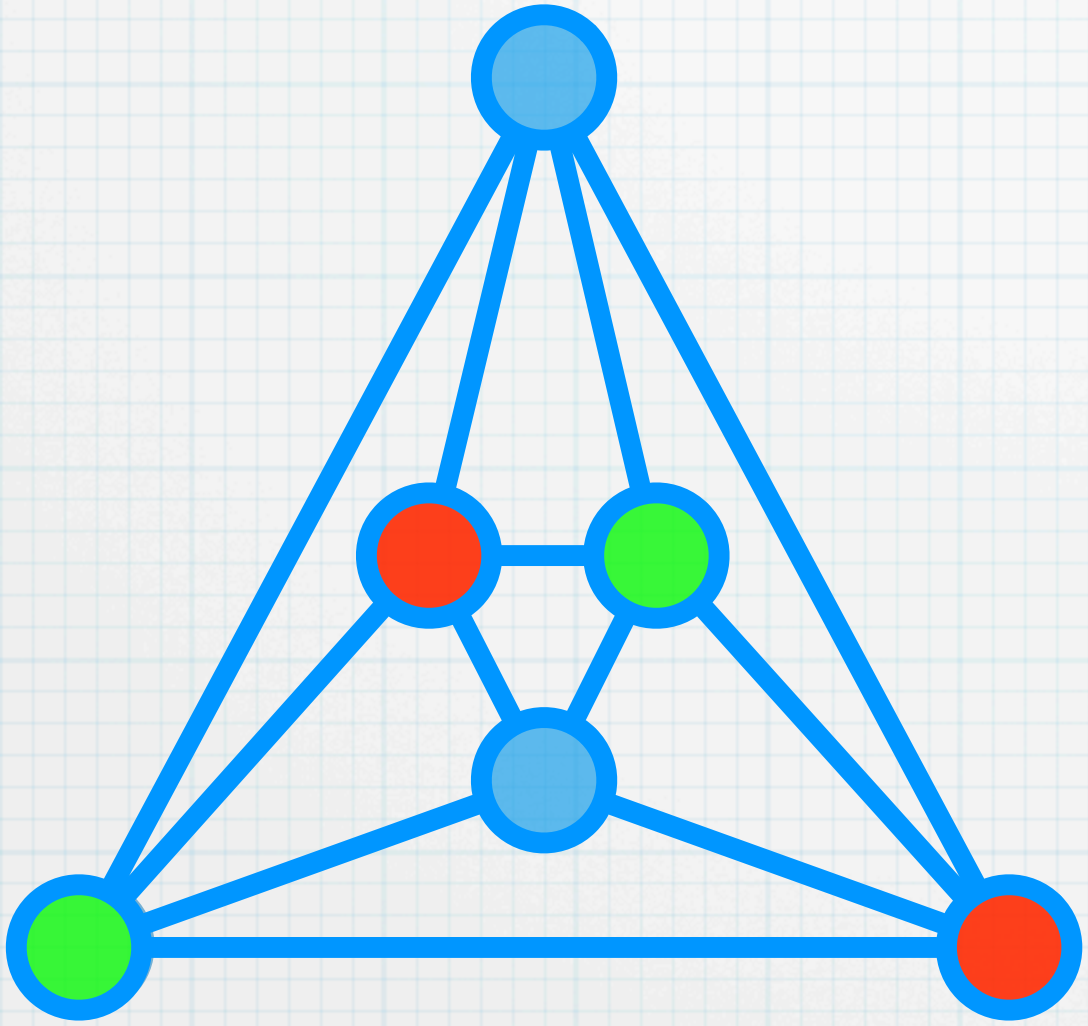
Exercise: Graph colouring



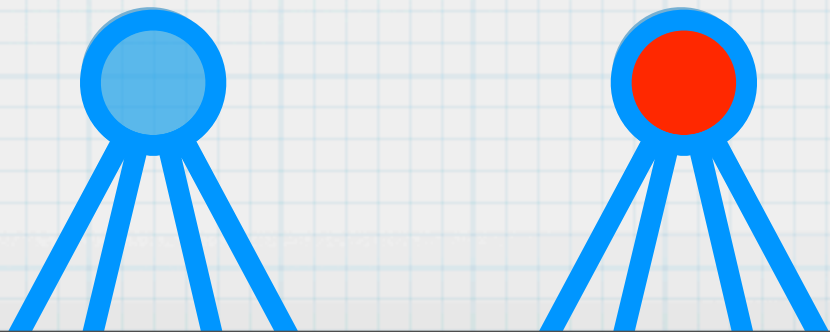
First approach:
split on vertices



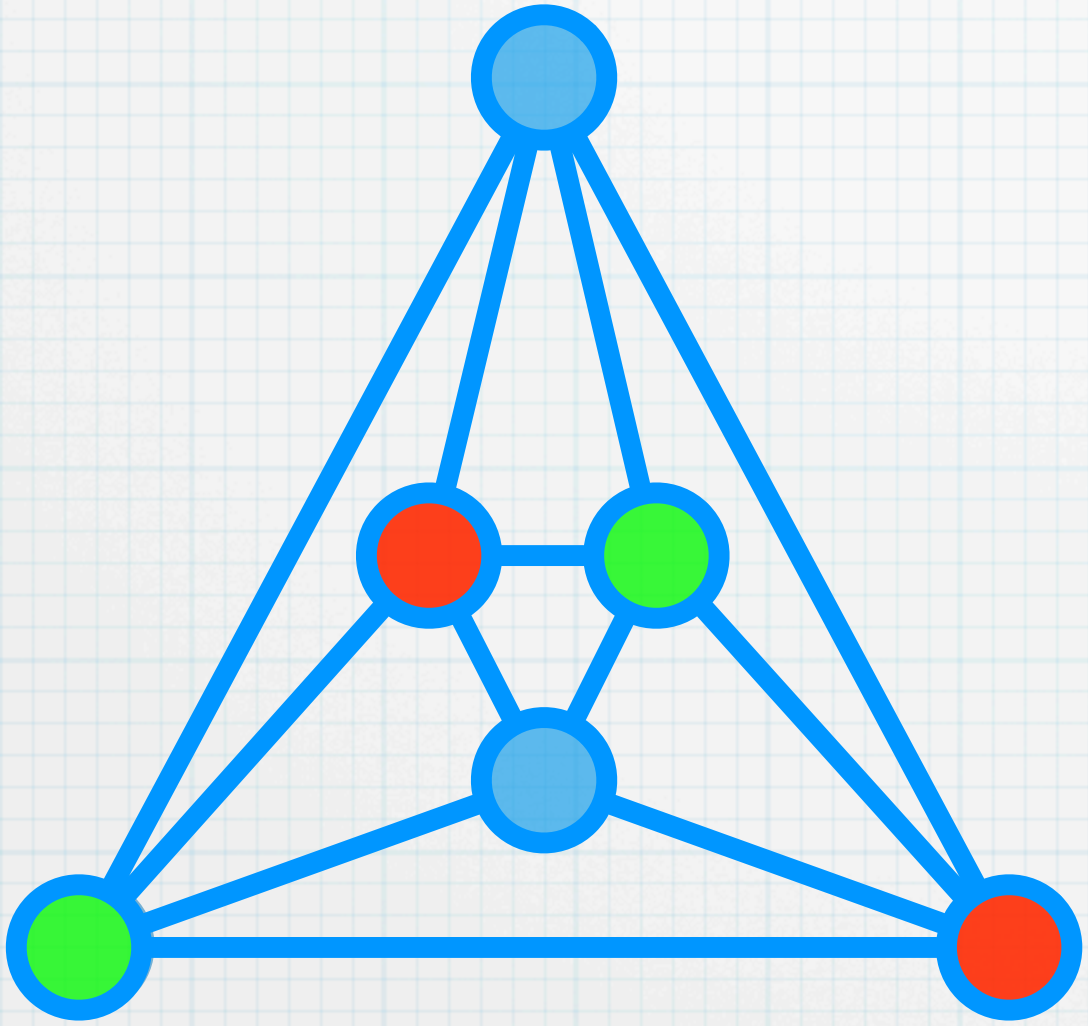
Exercise: Graph colouring



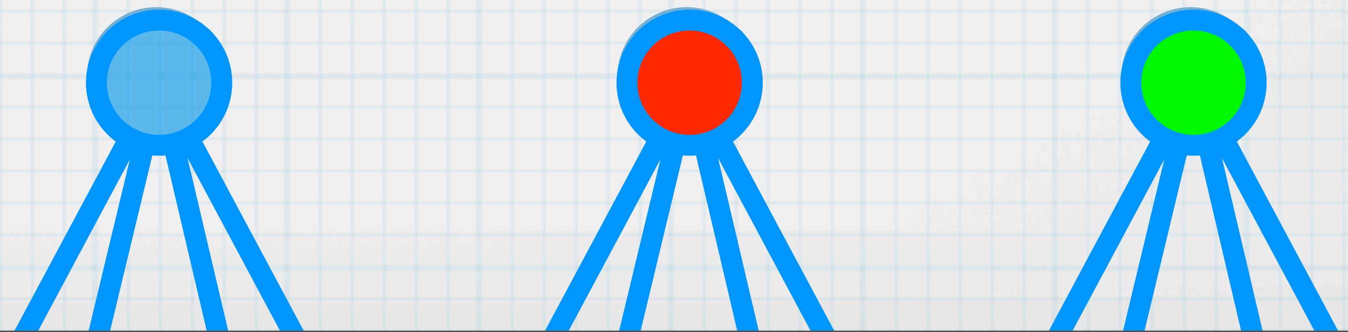
First approach:
split on vertices



Exercise: Graph colouring

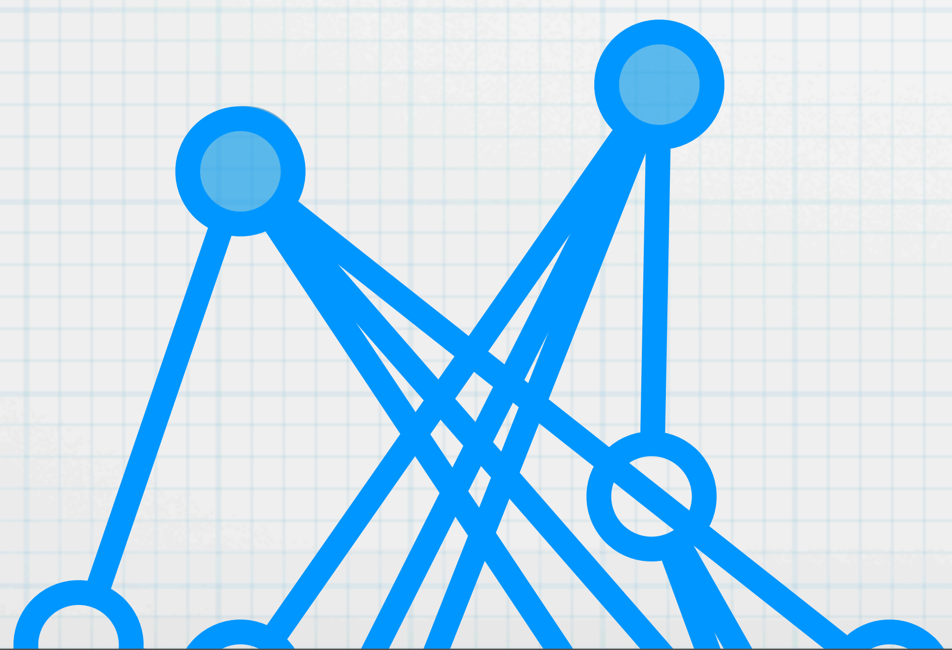


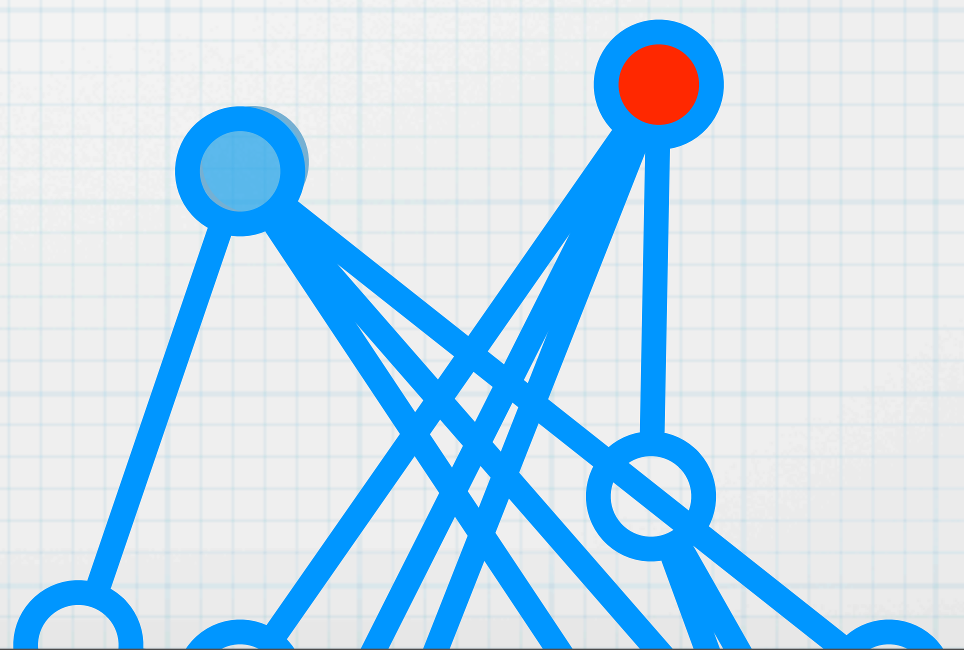
First approach:
split on vertices



Exercise: Graph colouring

Better: split on "nonedges"

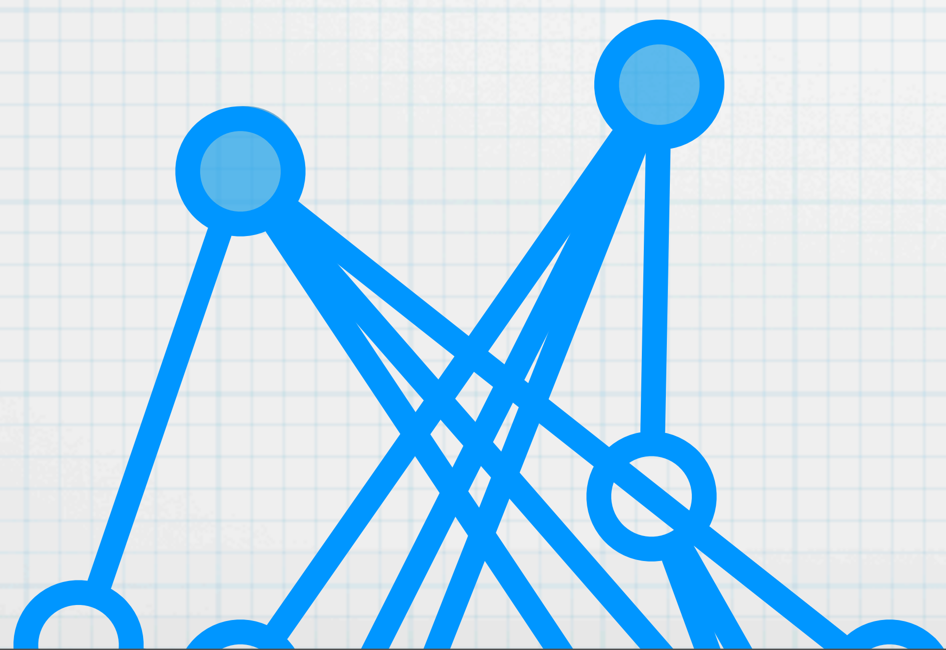


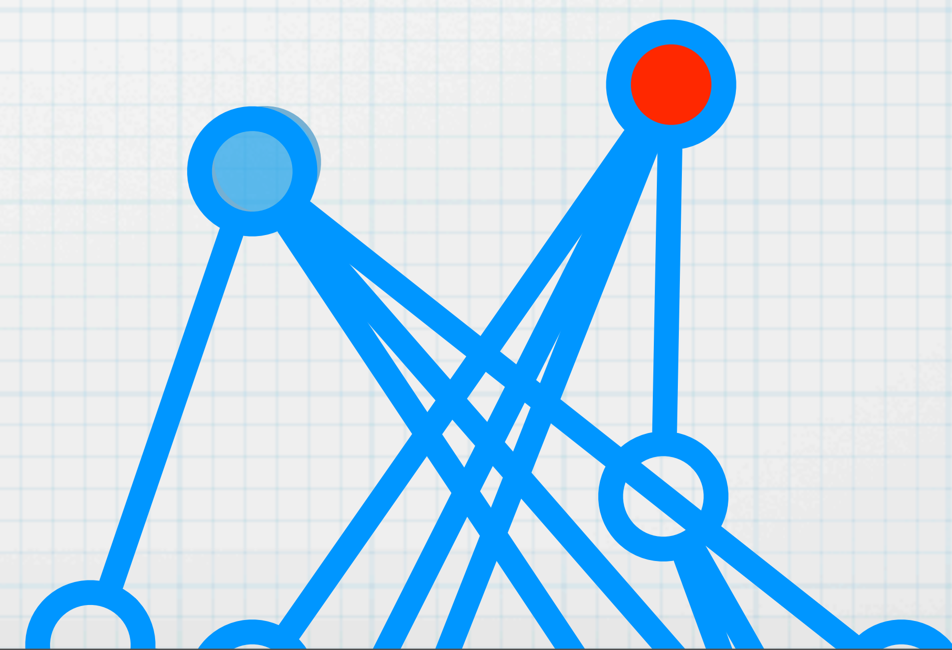


Exercise: Graph colouring

Better: split on "nonedges"

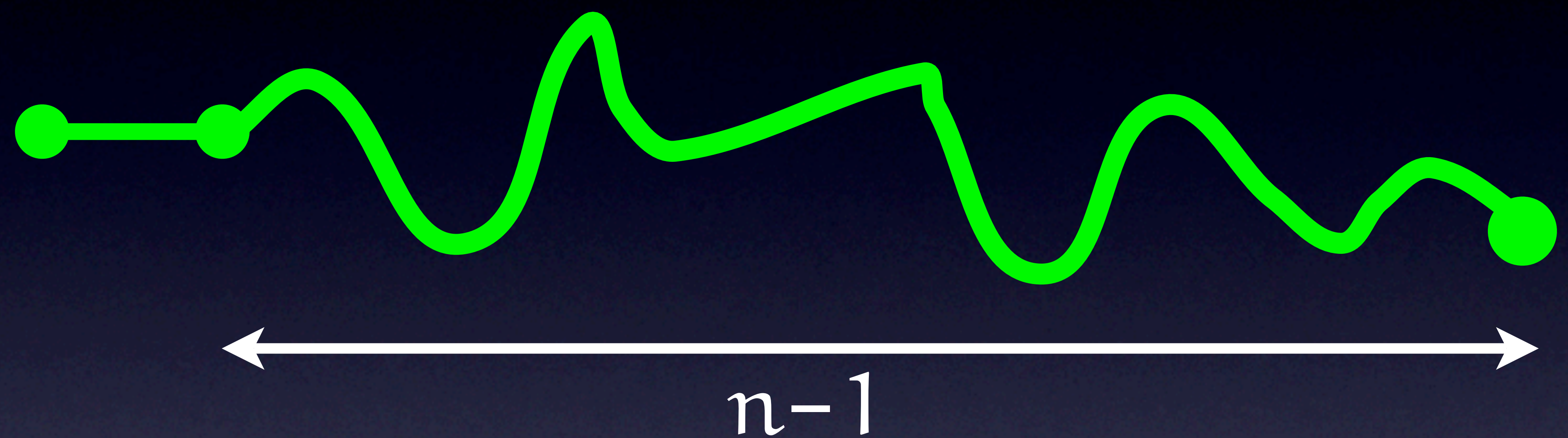
Conclude: Graph colouring in $O(1.619^{n+m'})$



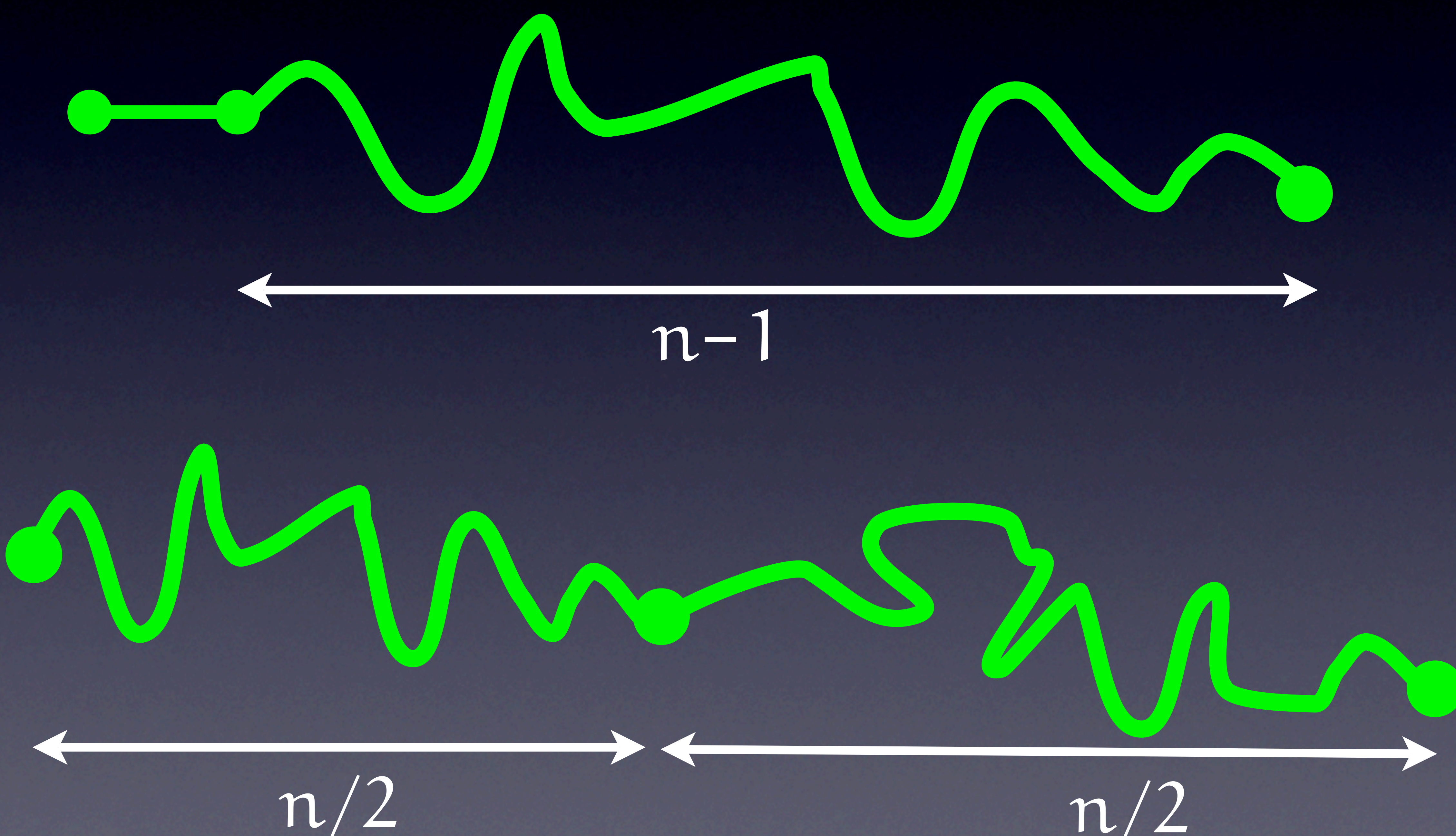


Divide and conquer

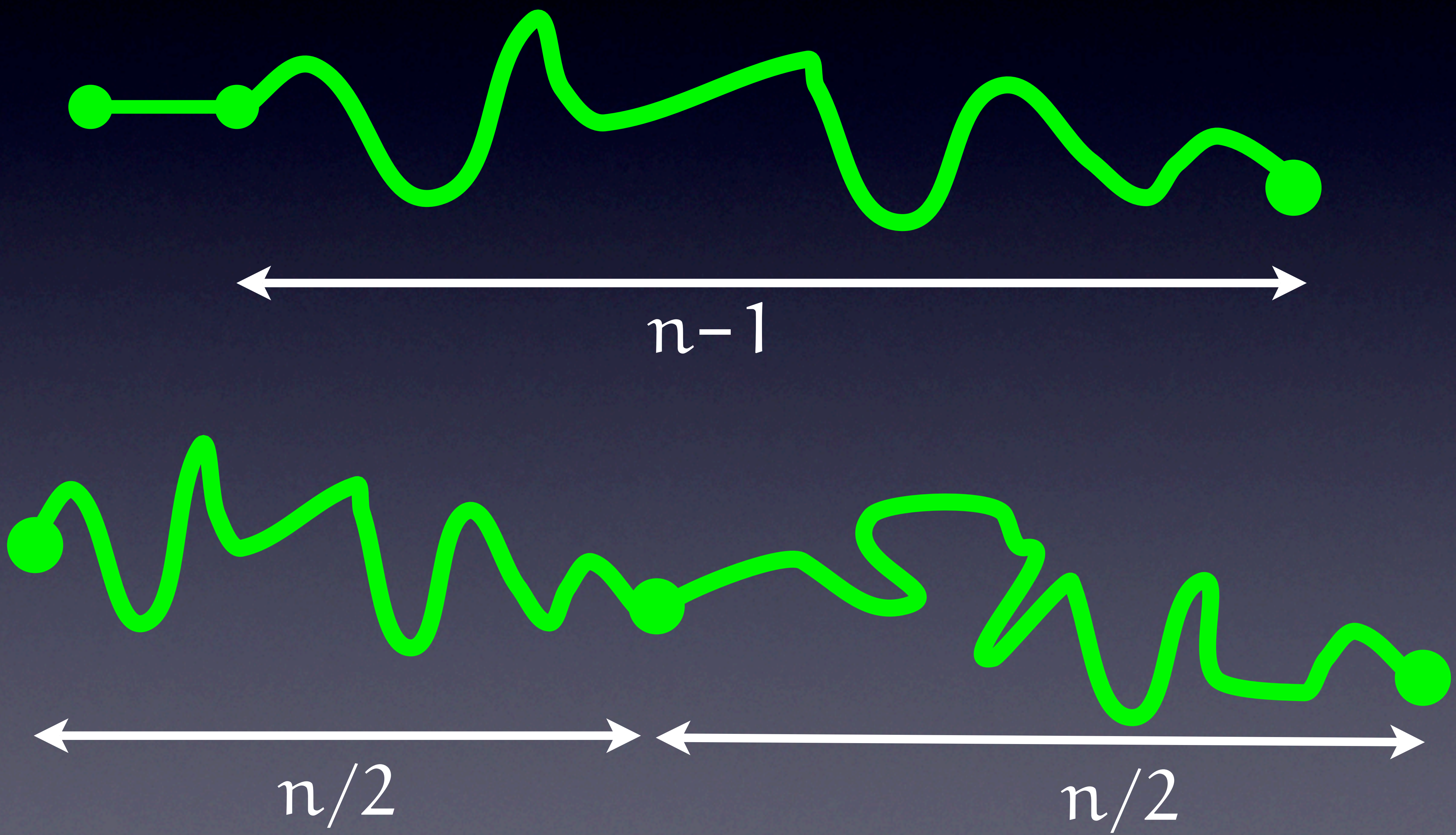
TSP



TSP

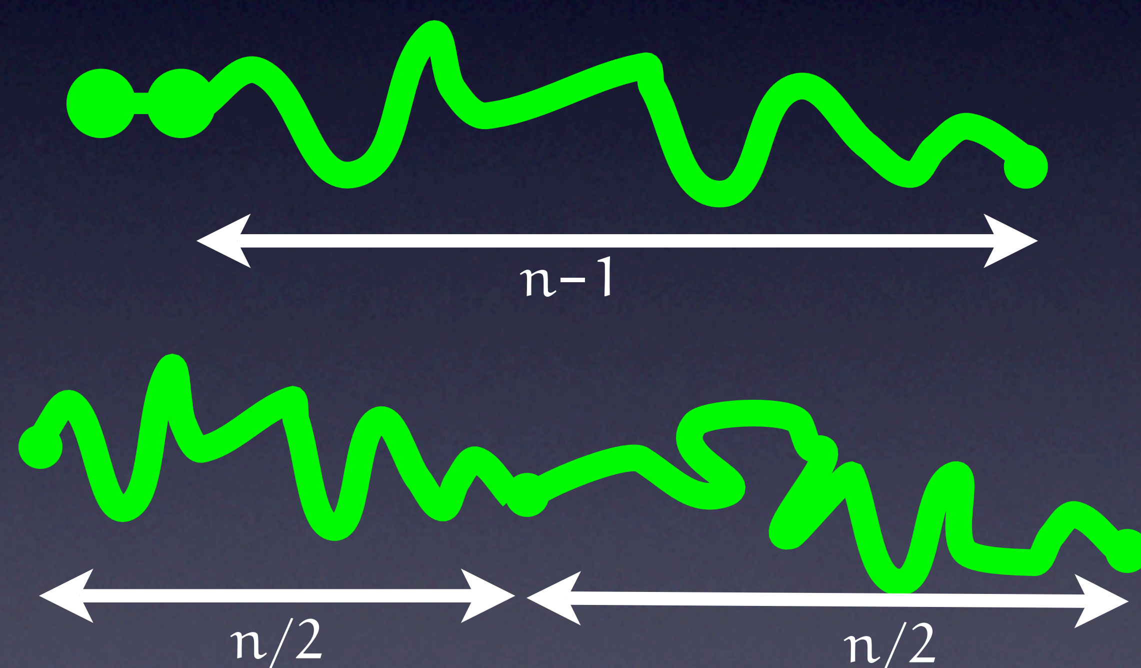


TSP



TSP

$$\mathrm{OPT}(T, v) = \min_{u \in T \setminus \{v\}} \mathrm{OPT}(T \setminus \{v\}, u) + w(u, v)$$



$$\mathrm{OPT}(U, s, t) = \min_{m, S, T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$

TSP

$$T(n) = \binom{n}{n/2} \cdot 2 \cdot T(n/2)$$

TSP

$$T(n) = \binom{n}{n/2} \cdot 2 \cdot T(n/2)$$

$$\mathrm{OPT}(U, s, t) = \min_{m,S,T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$

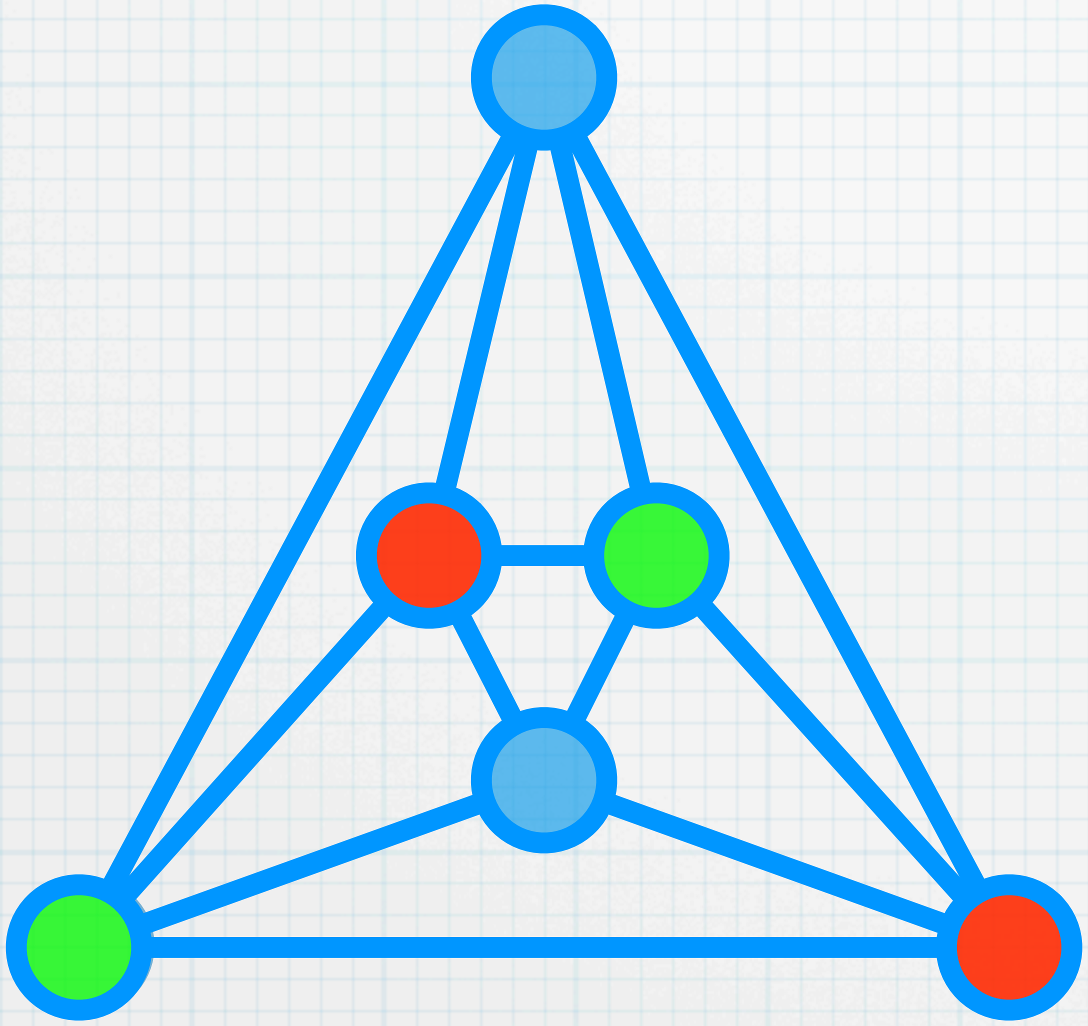
TSP

$$T(n) = \binom{n}{n/2} \cdot 2 \cdot T(n/2)$$

$$T(n) \leq 2^n \cdot T(n/2) \leq 2^n 2^{n/2} \cdots 2^0 \leq 2^{2n} = 4^n$$

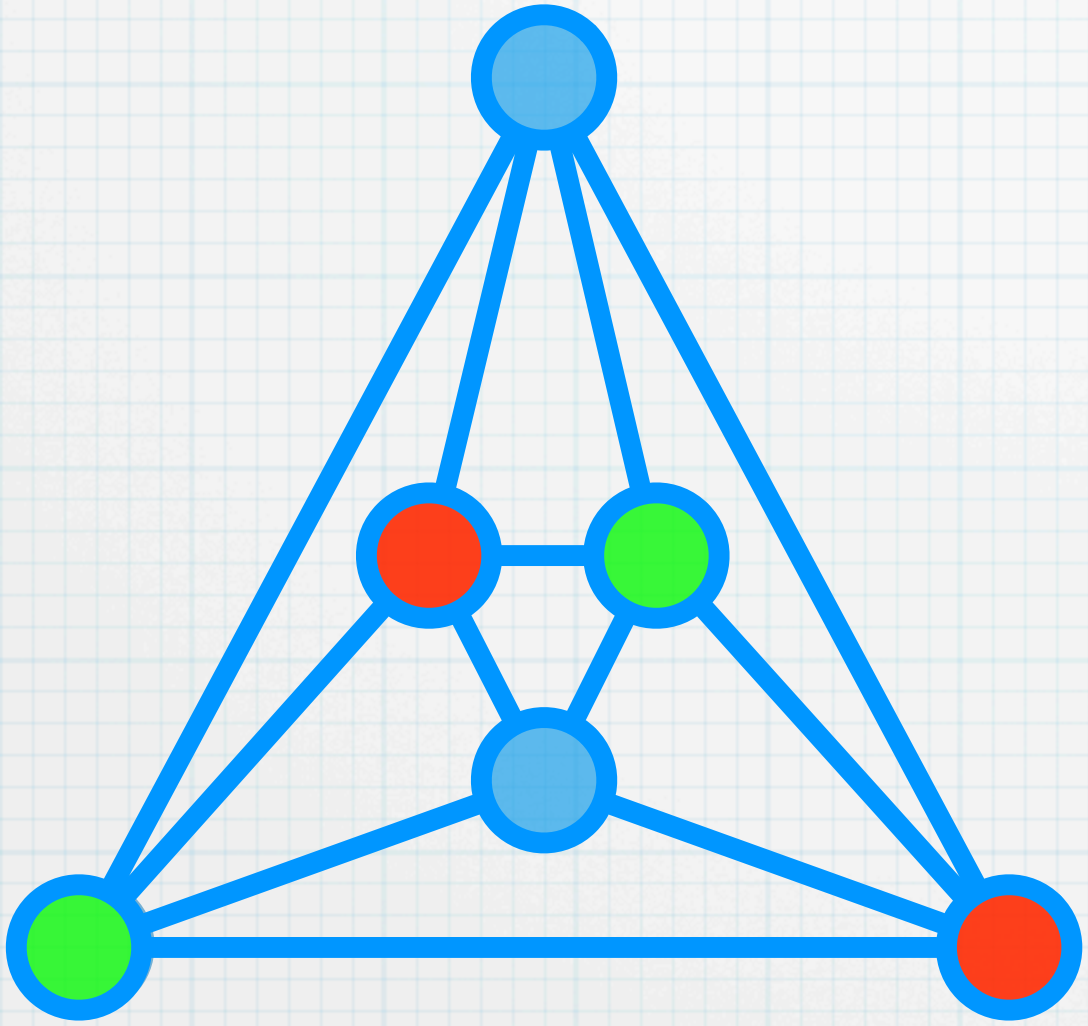
$$\mathrm{OPT}(U, s, t) = \min_{m,S,T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$

Exercise: Graph colouring



k-colouring = partition into k independent sets

Exercise: Graph colouring



k-colouring = partition into k independent sets

Conclude: Graph colouring in $O^*(9^n)$

Transformation ("Reduce to other")

$$\log a^n = n \log a$$

$$a^{(0110101)_2} = \dots$$

Transformation (“Reduce to other”)

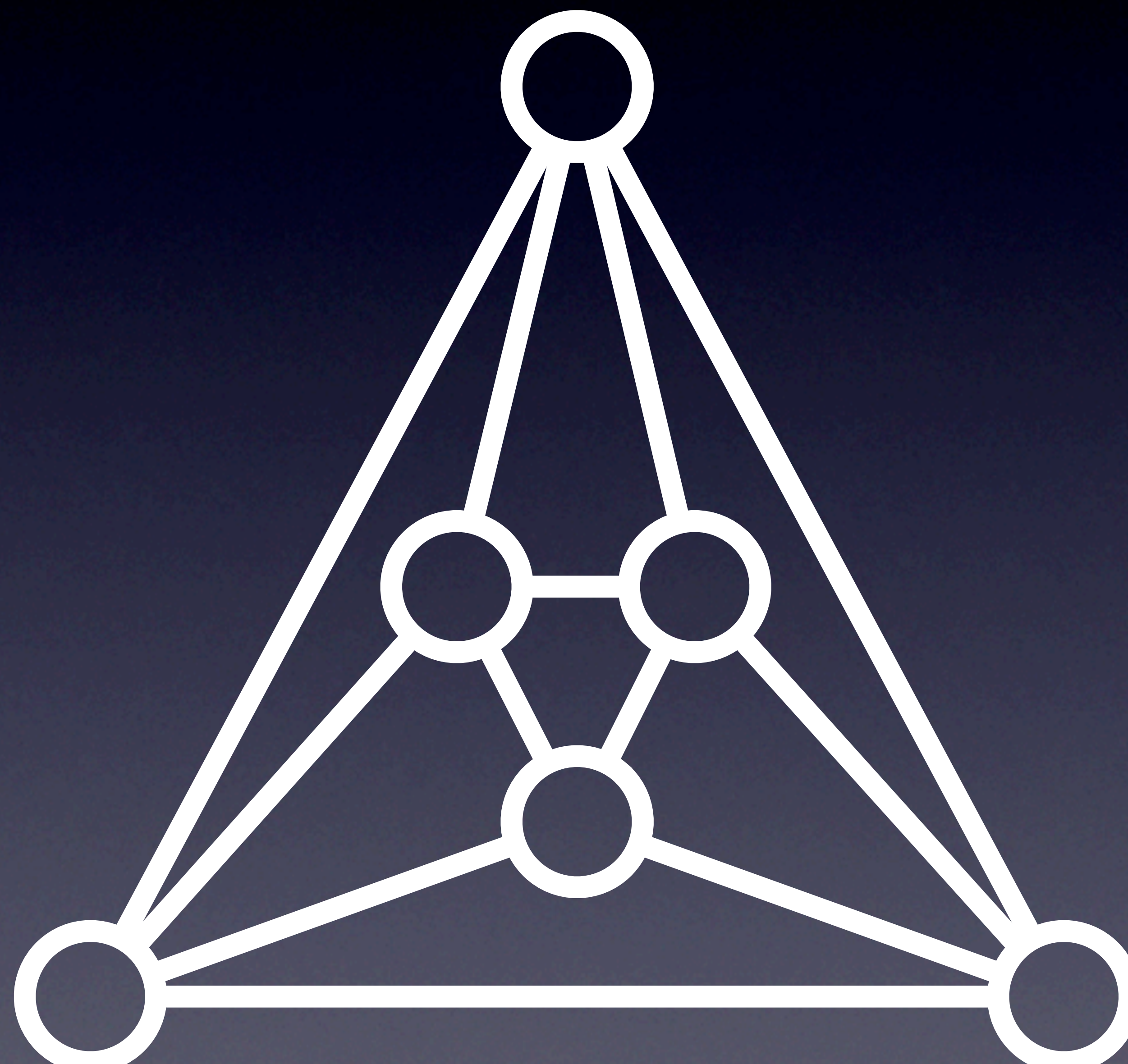
$$\log a^n = n \log a$$

$$a^{(0110101)_2} = \ldots$$

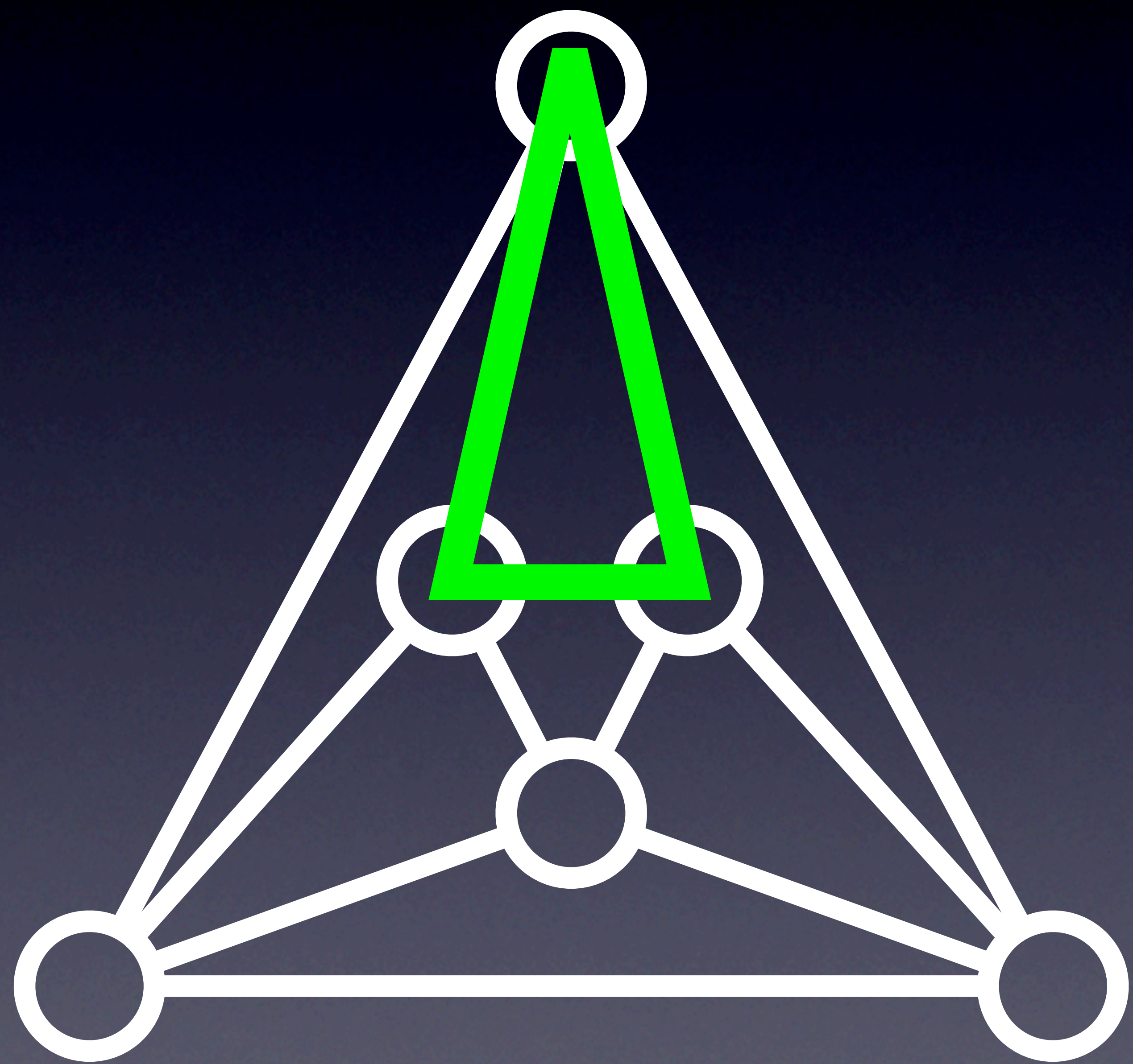
Counting triangles

Moebius transform

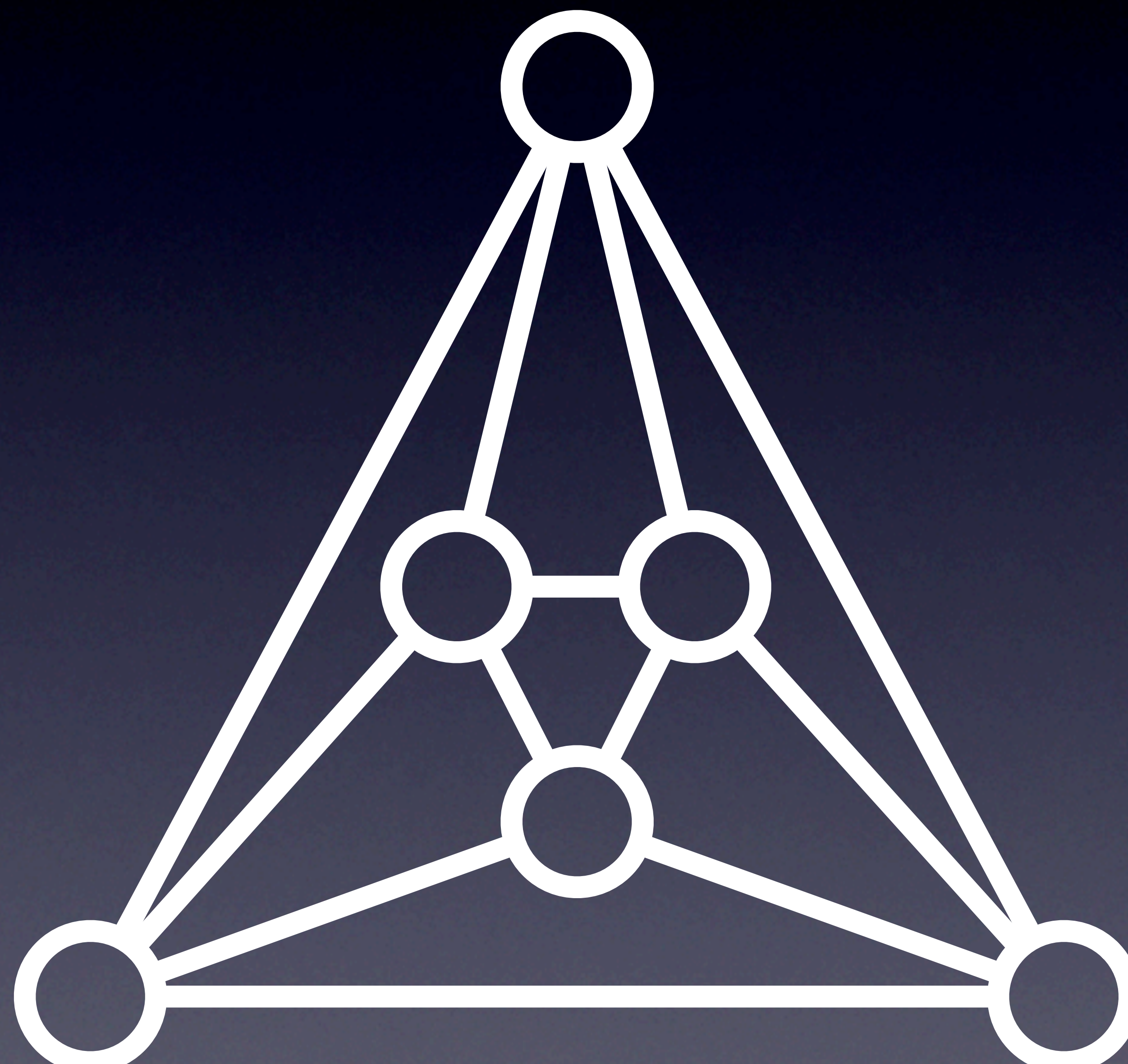
To *Counting Triangles*



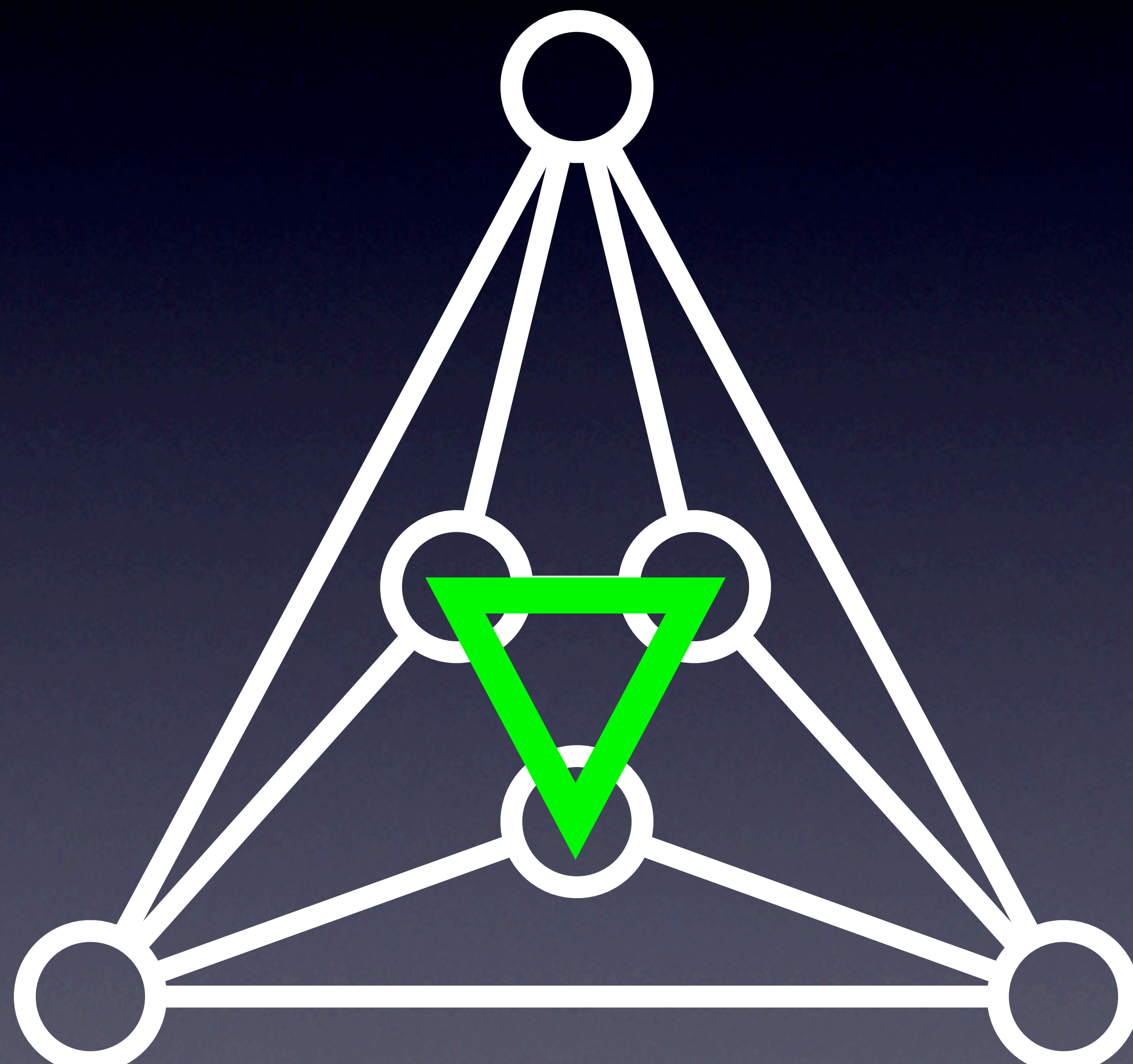
To *Counting Triangles*



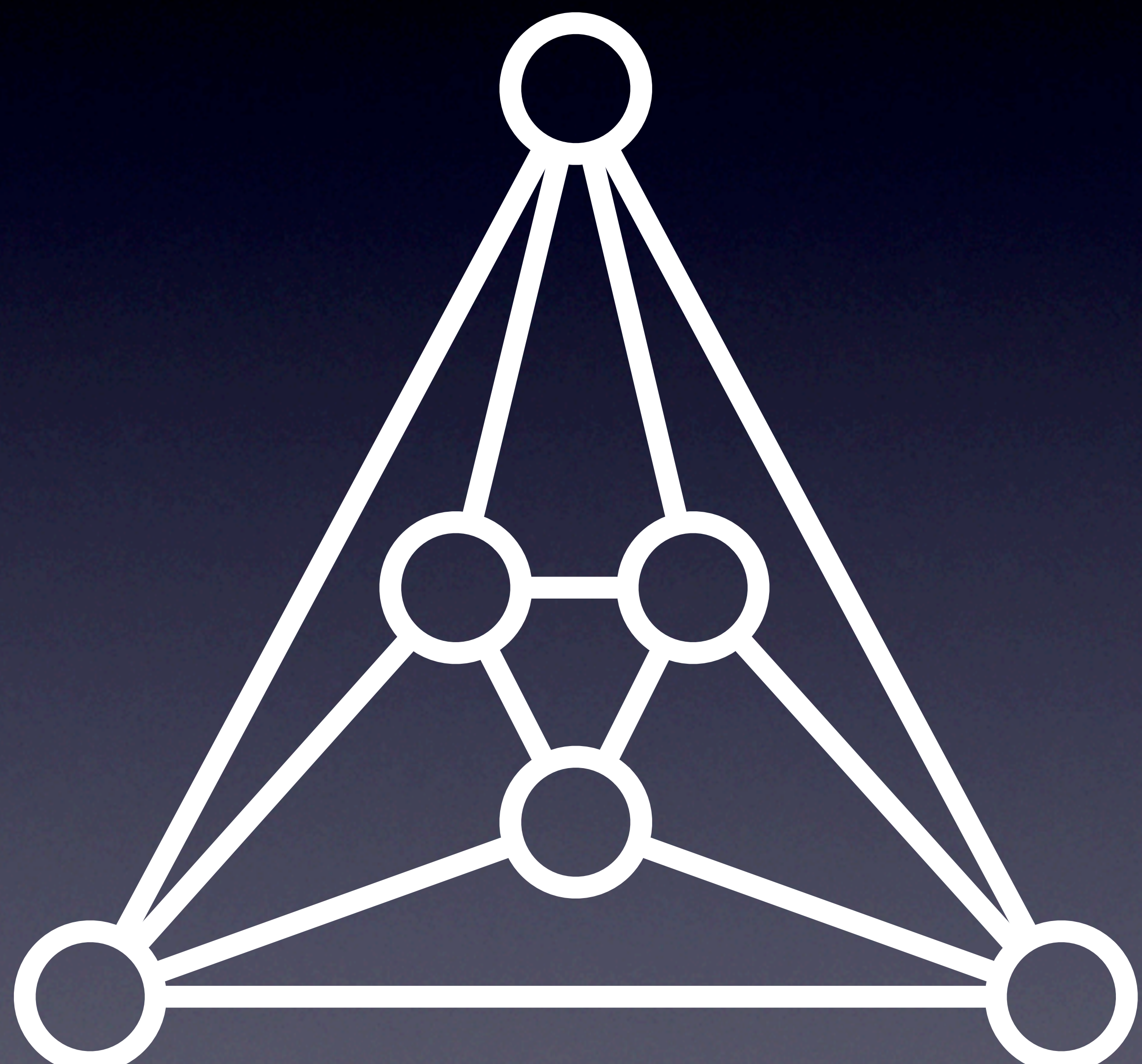
To *Counting Triangles*



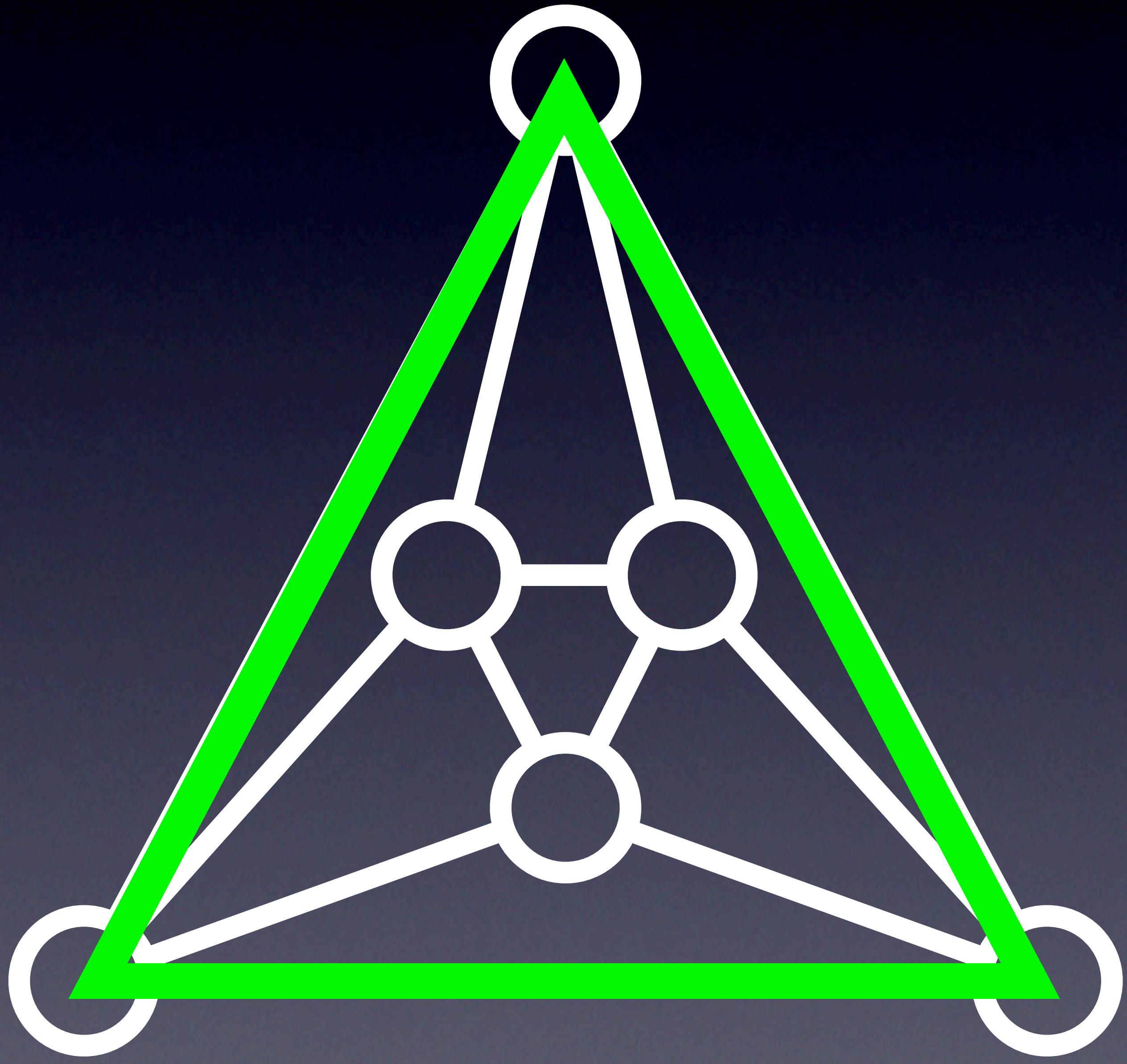
To *Counting Triangles*



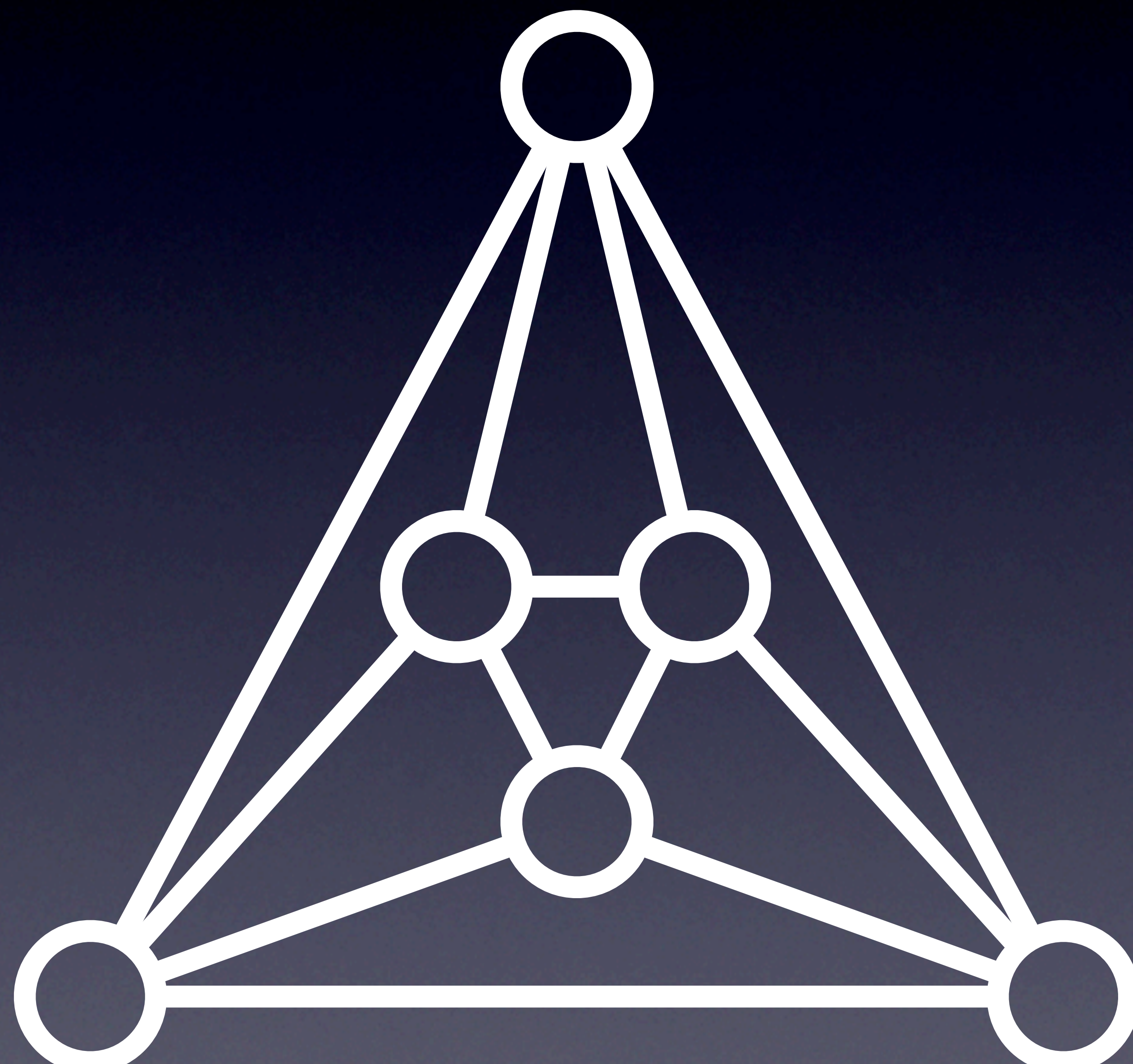
To *Counting Triangles*



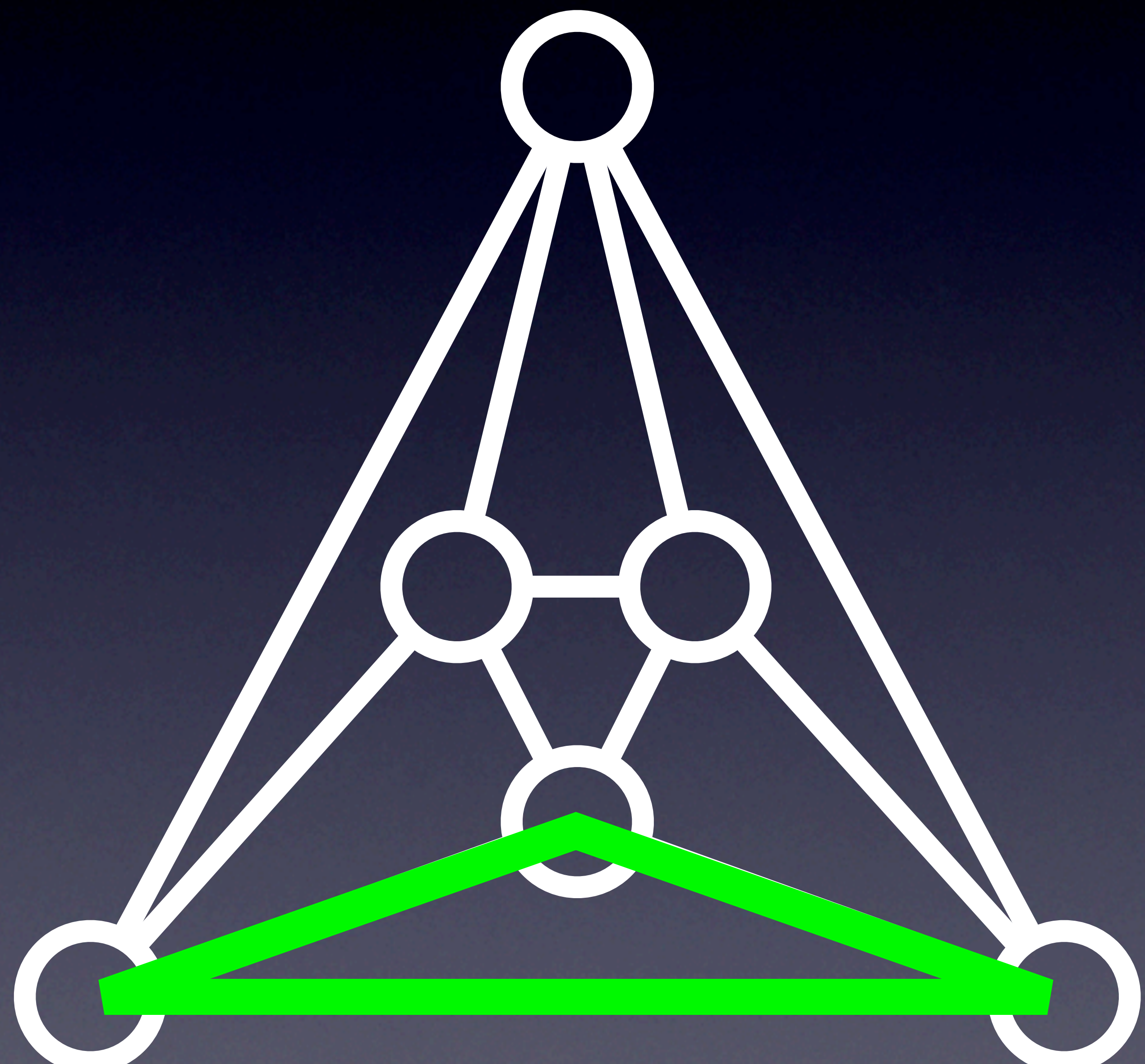
To *Counting Triangles*



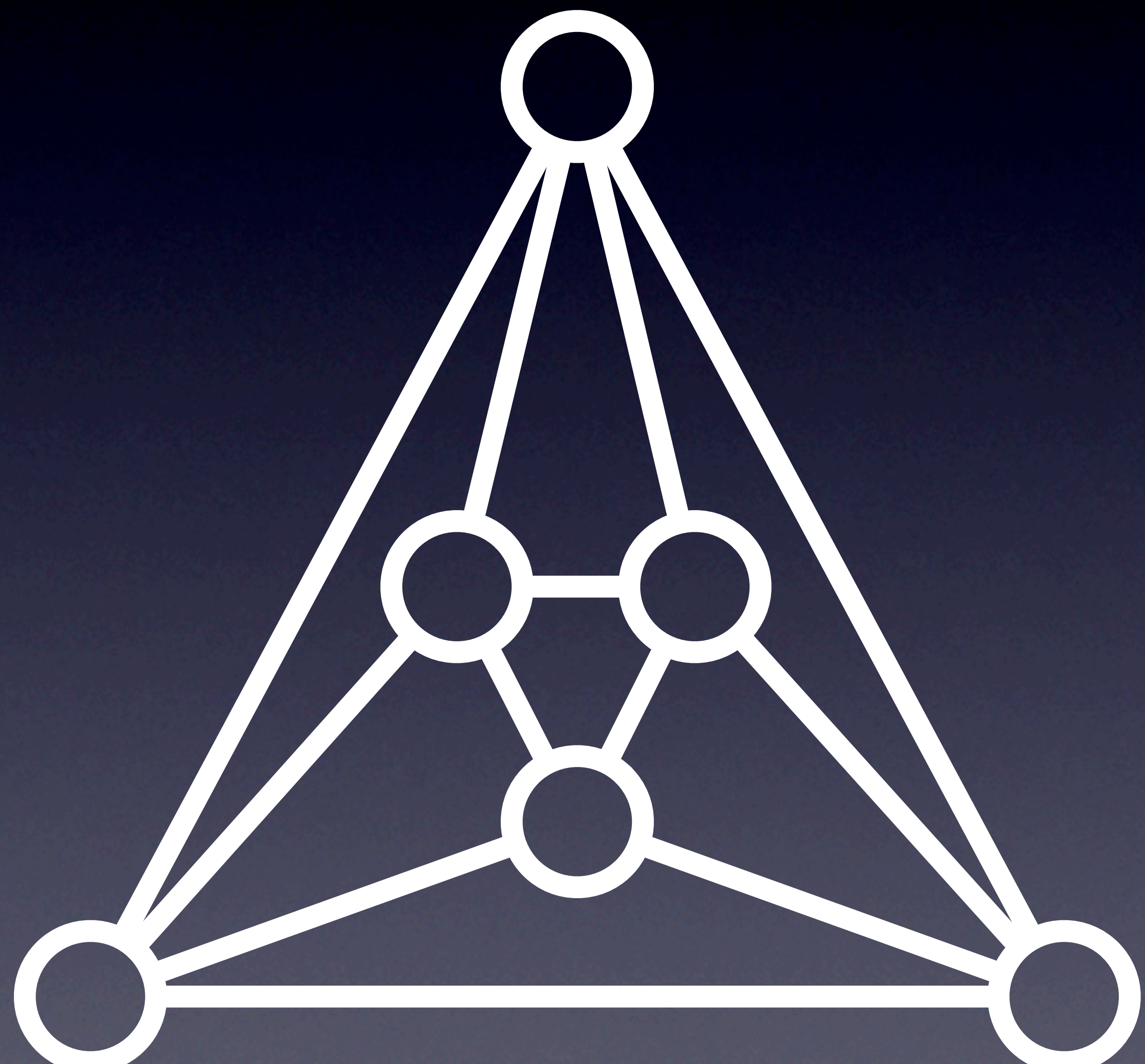
To *Counting Triangles*



To *Counting Triangles*



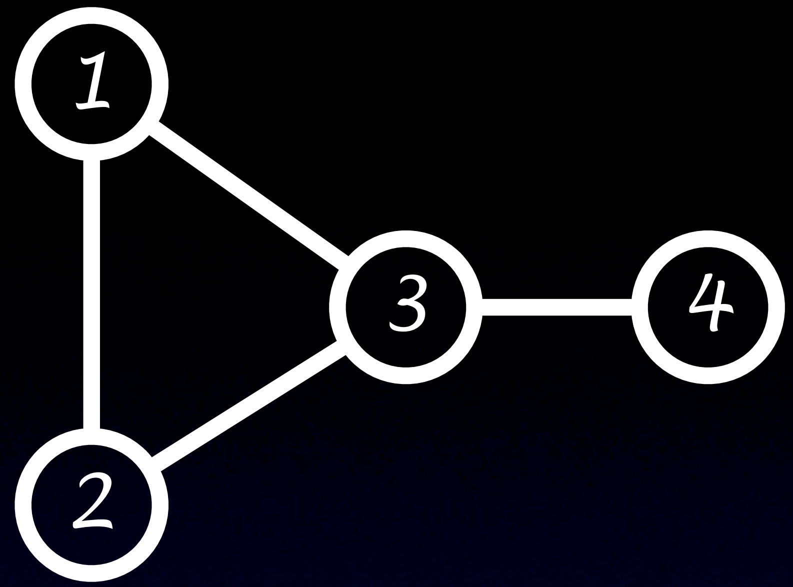
To *Counting Triangles*



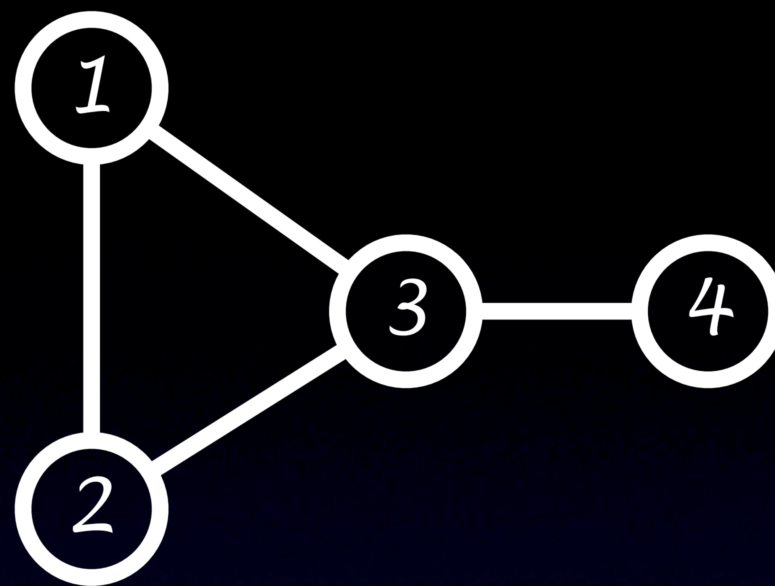
Counting triangles

Easily in $O(|V|^3)$

Surprise: can do better!
Current record: $O(|V|^{2.376})$



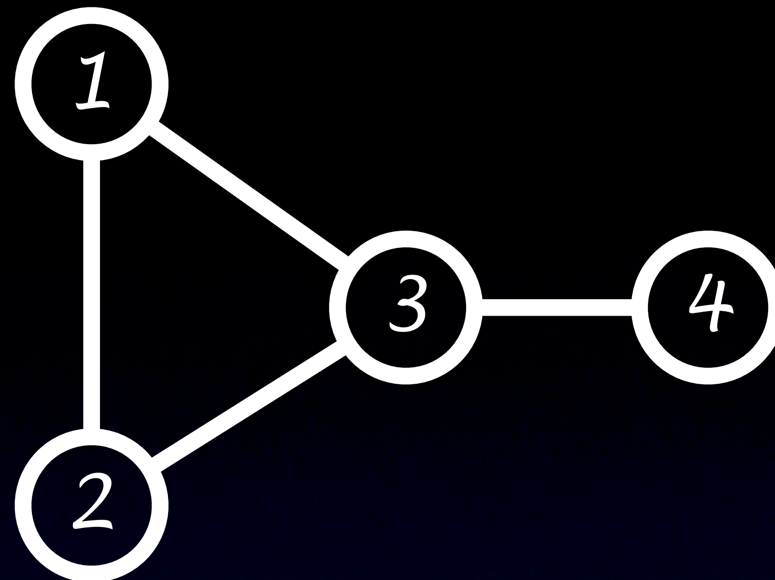
trace(A^3) = 6 times # triangles



$$A$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

trace(A^3) = 6 times # triangles



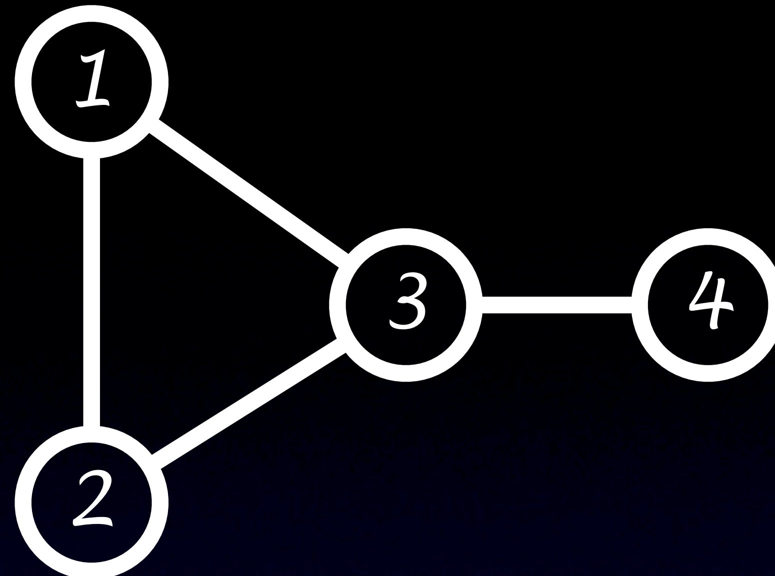
A

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

A^2

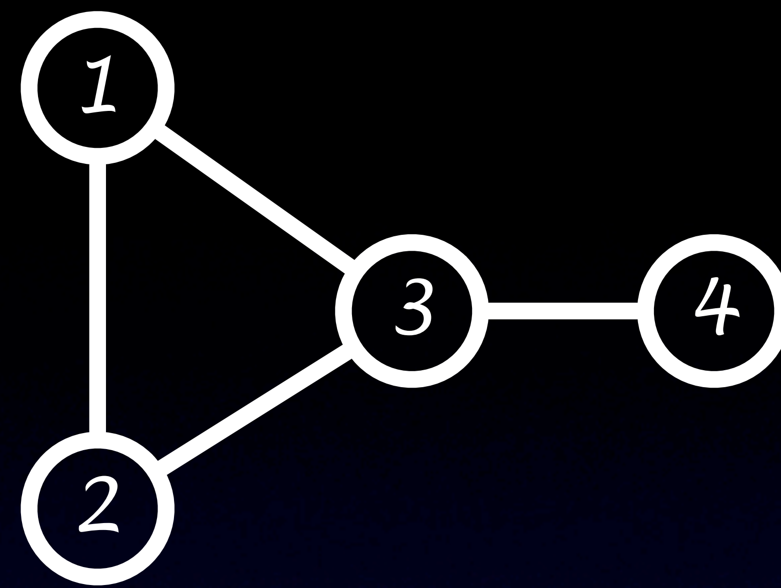
$$\begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

trace(A^3) = 6 times # triangles



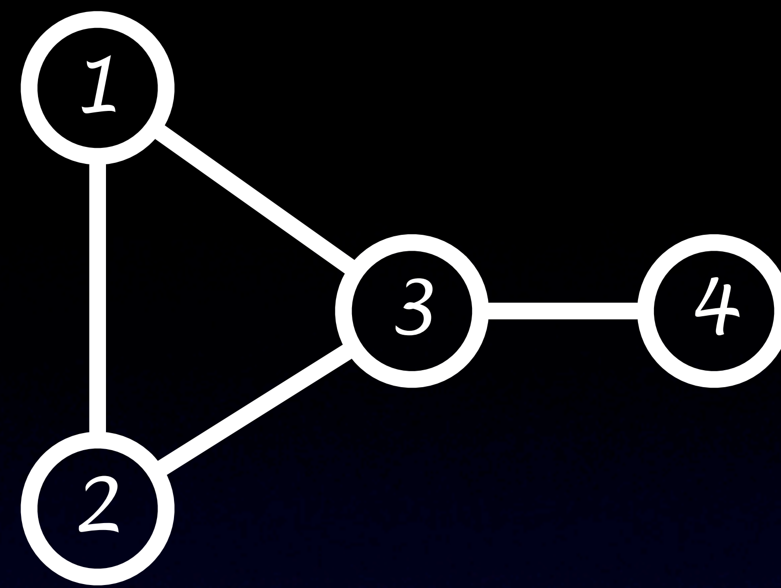
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad A^2 = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad A^3 = \begin{bmatrix} 2 & 3 & 3 & 1 \\ 3 & 2 & 3 & 1 \\ 4 & 4 & 2 & 3 \\ 1 & 1 & 2 & 0 \end{bmatrix}$$

trace(A^3) = 6 times # triangles



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad A^2 = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad A^3 = \begin{bmatrix} 2 & 3 & 3 & 1 \\ 3 & 2 & 3 & 1 \\ 4 & 4 & 2 & 3 \\ 1 & 1 & 2 & 0 \end{bmatrix}$$

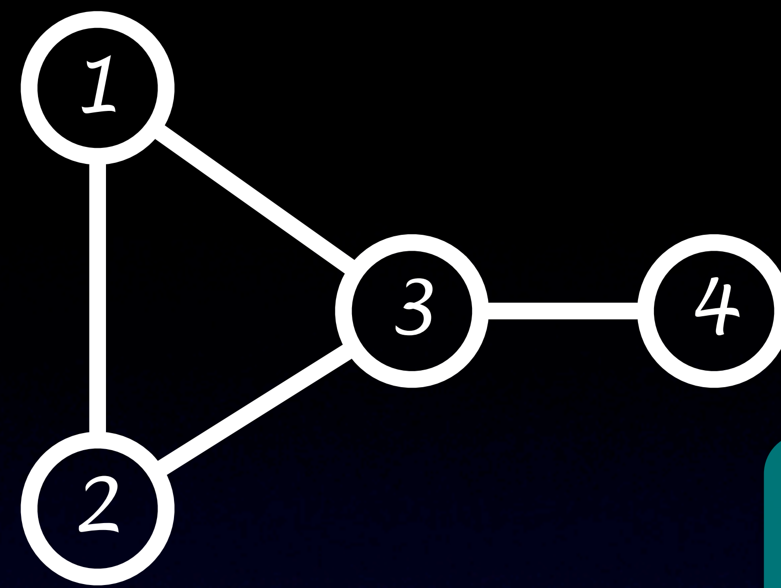
trace(A^3) = 6 times # triangles



Time $O(d^3)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad A^2 = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad A^3 = \begin{bmatrix} 2 & 3 & 3 & 1 \\ 3 & 2 & 3 & 1 \\ 4 & 4 & 2 & 3 \\ 1 & 1 & 2 & 0 \end{bmatrix}$$

trace(A^3) = 6 times # triangles

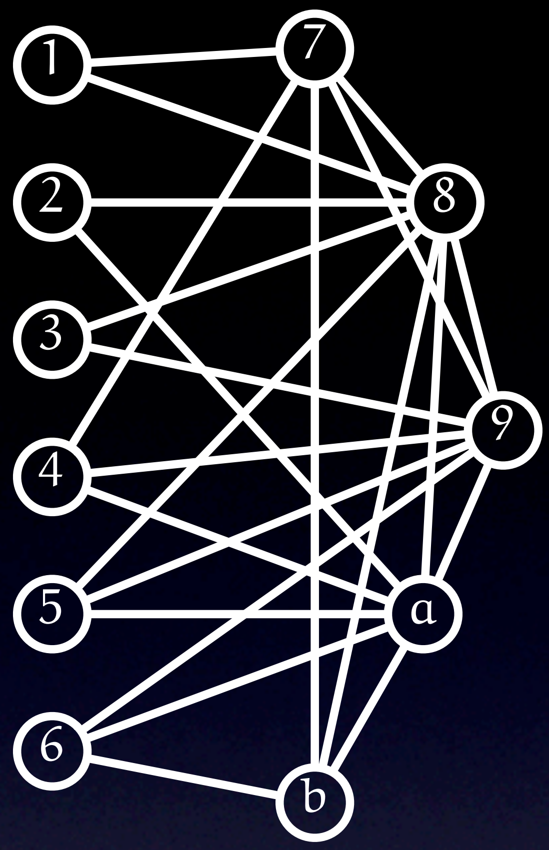


Time $O(d^{\omega})$

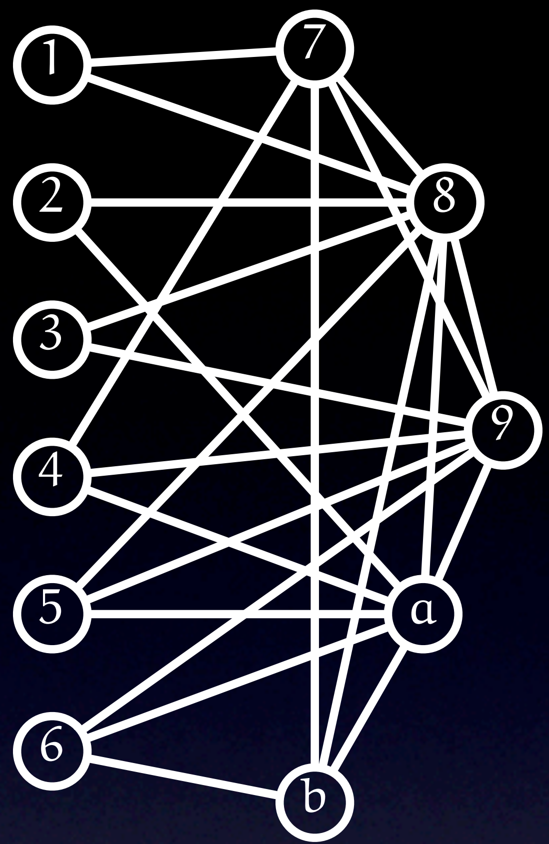
Time $O(d^3)$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad A^2 = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad A^3 = \begin{bmatrix} 2 & 3 & 3 & 1 \\ 3 & 2 & 3 & 1 \\ 4 & 4 & 2 & 3 \\ 1 & 1 & 2 & 0 \end{bmatrix}$$

trace(A^3) = 6 times # triangles

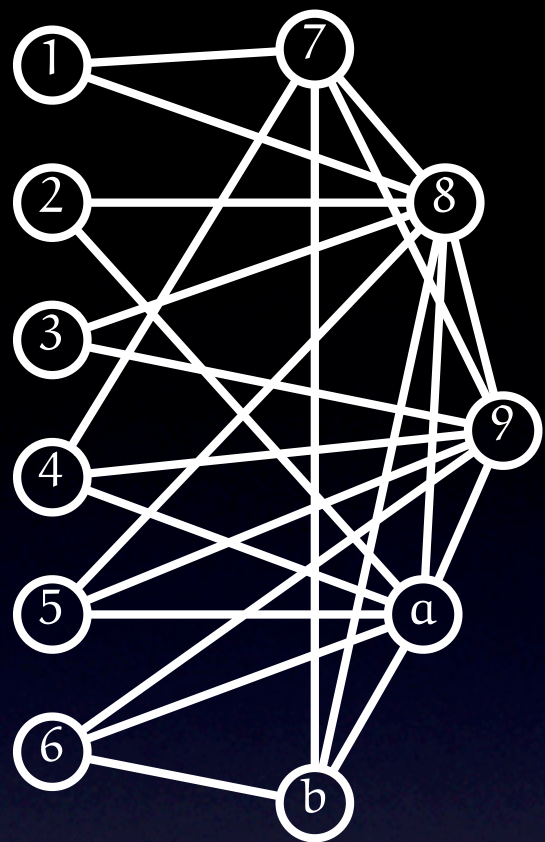


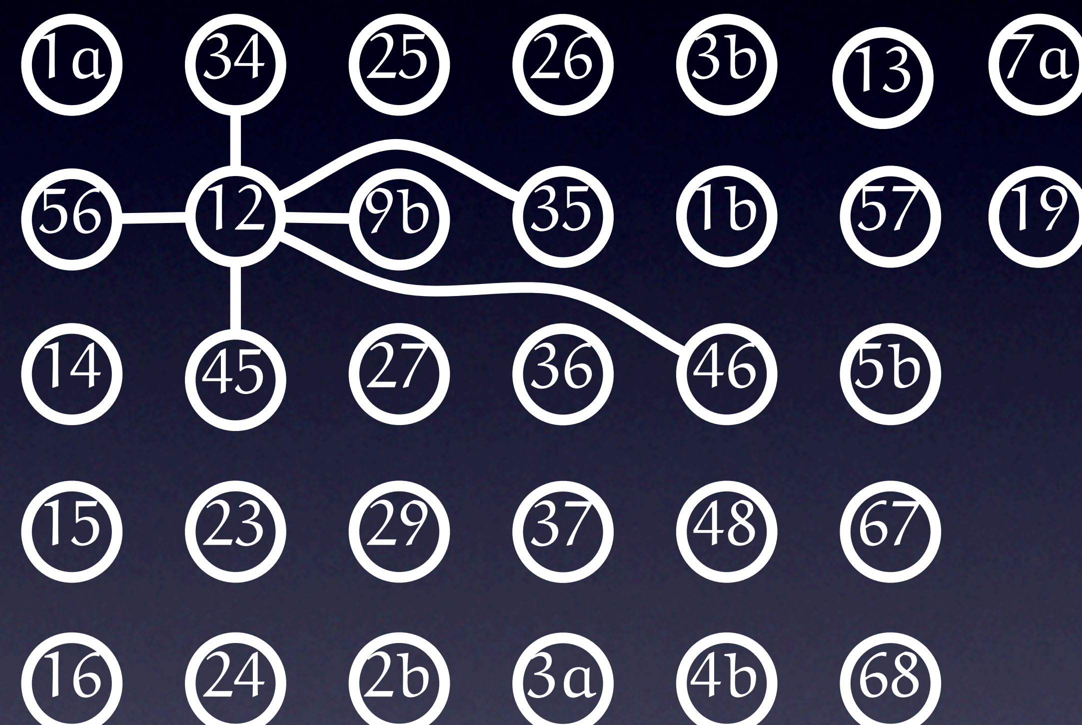
independent set of size $k=6$?



vertex for every independent subset of size $k/3$

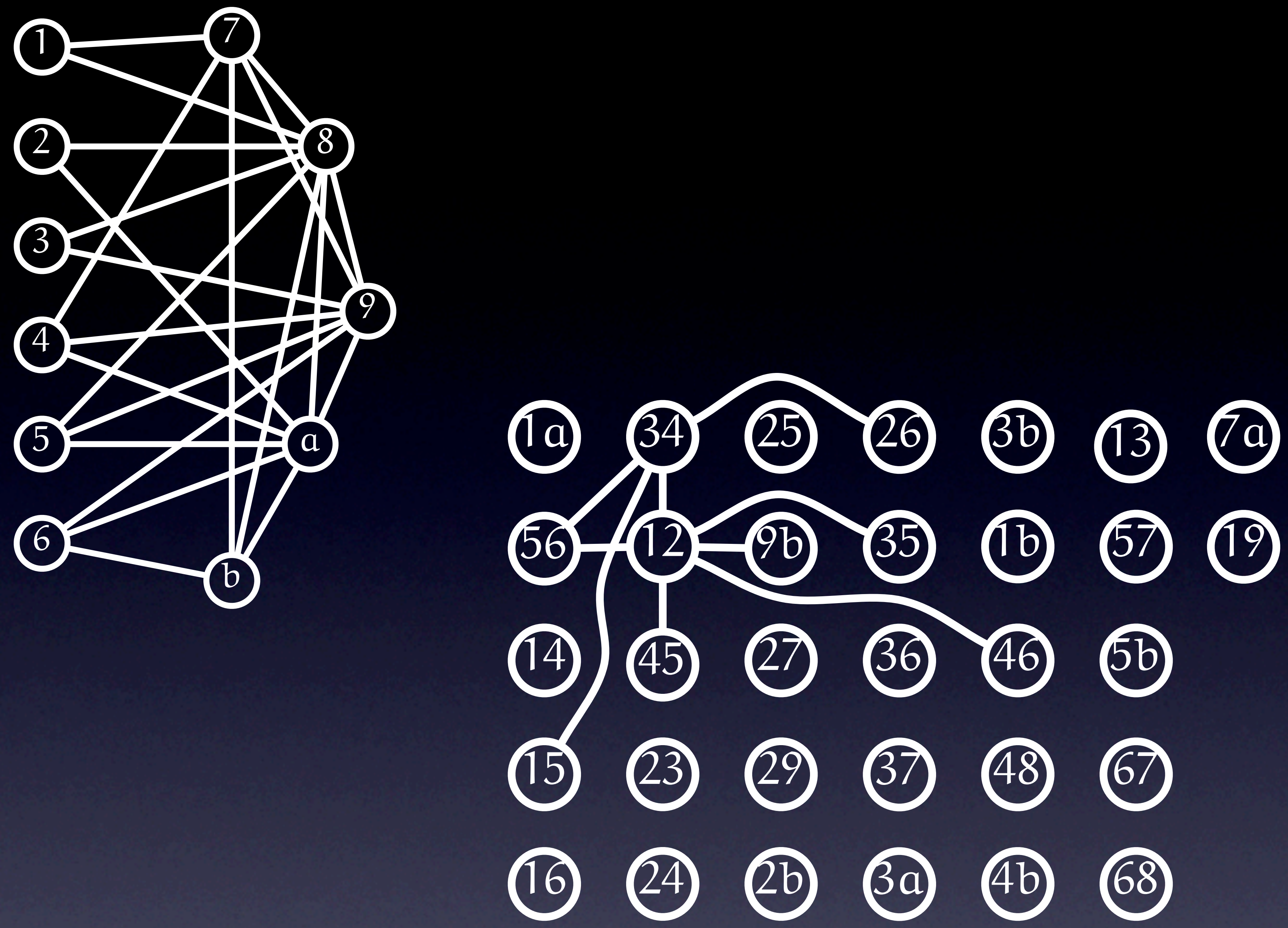
$$\binom{n}{k/3} \sim n^{k/3}$$





edge for every disjoint, independent subset

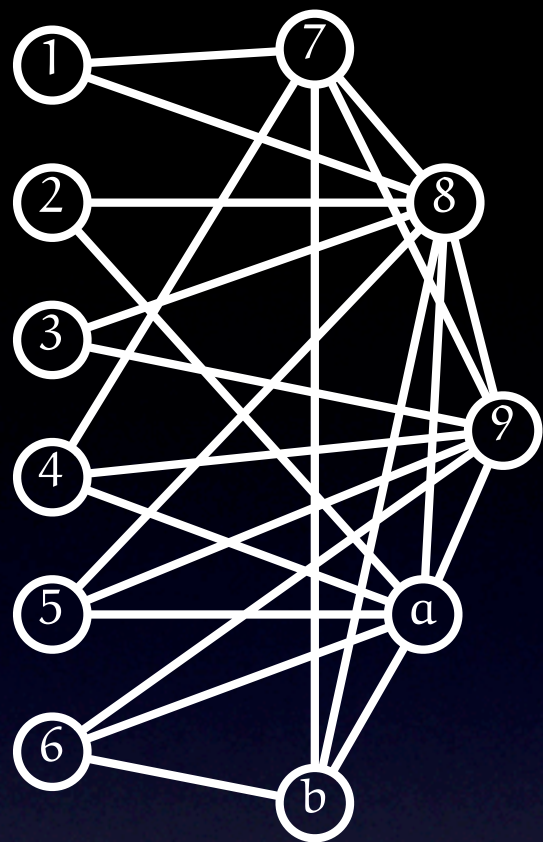
= independent subset of size $2k/3$



edge for every disjoint, independent subset
= independent subset of size $2k/3$

triangle = independent subset of size $3k/3$

Time $O((n^{k/3})^{\omega}) = O(n^{\omega k/3})$



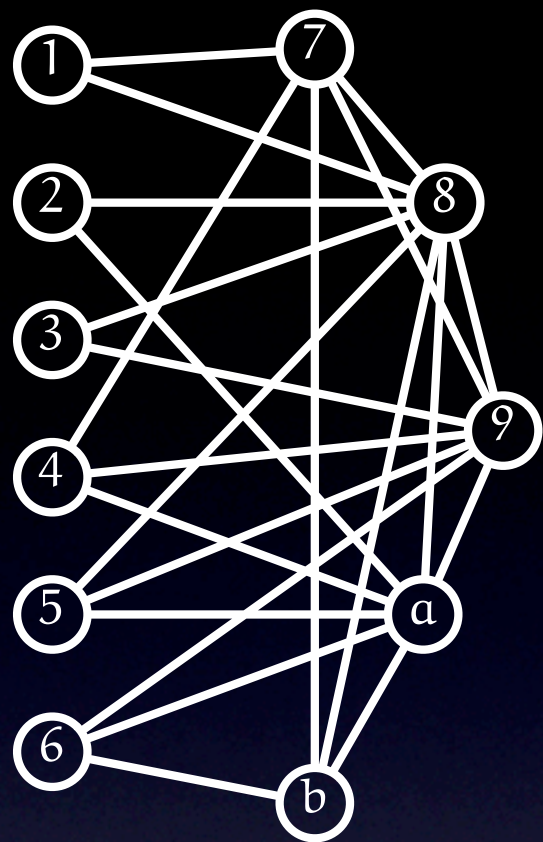
edge for every disjoint, independent subset

= independent subset of size $2k/3$

triangle = independent subset of size $3k/3$

Time $O((n^{k/3})^{\omega}) = O(n^{\omega k/3})$

Space $O(n^{k/3})$

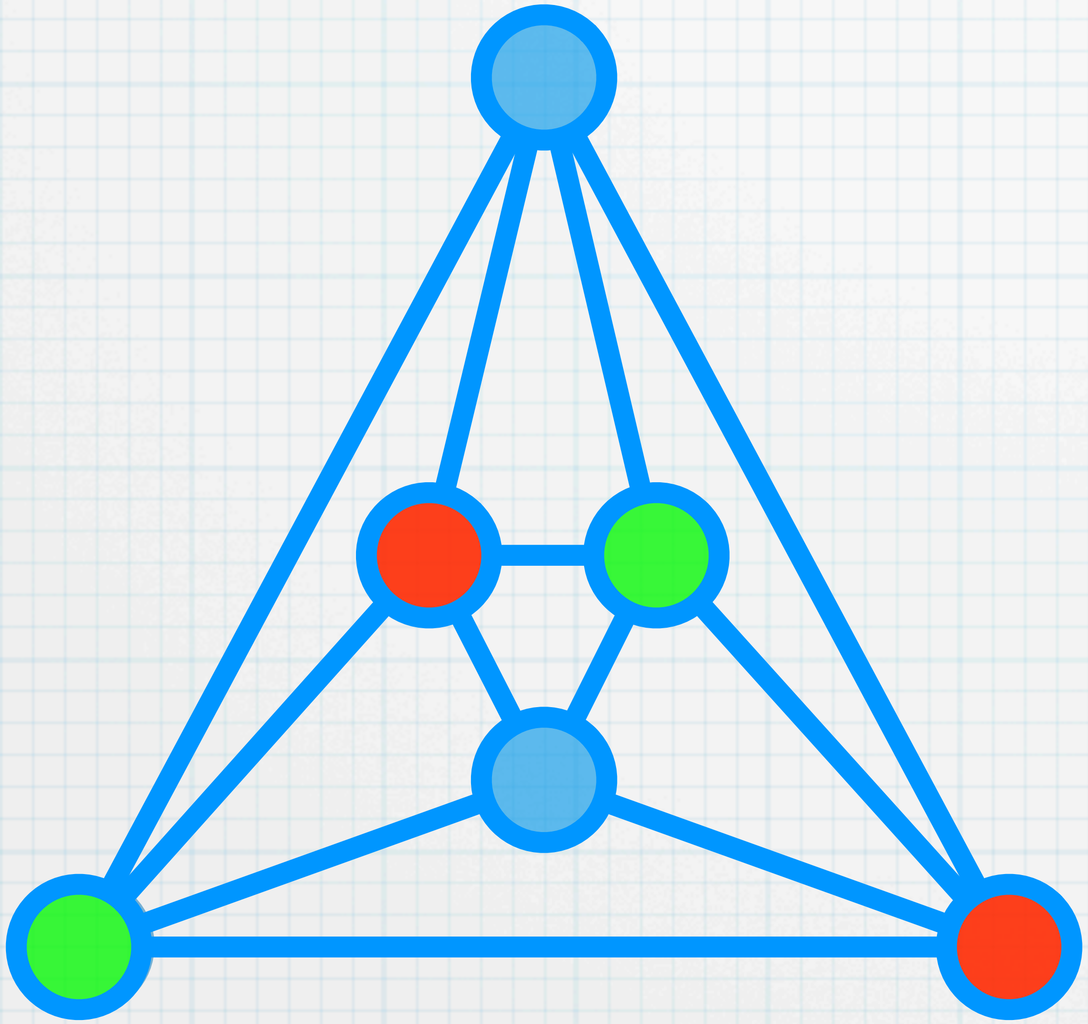


edge for every disjoint, independent subset

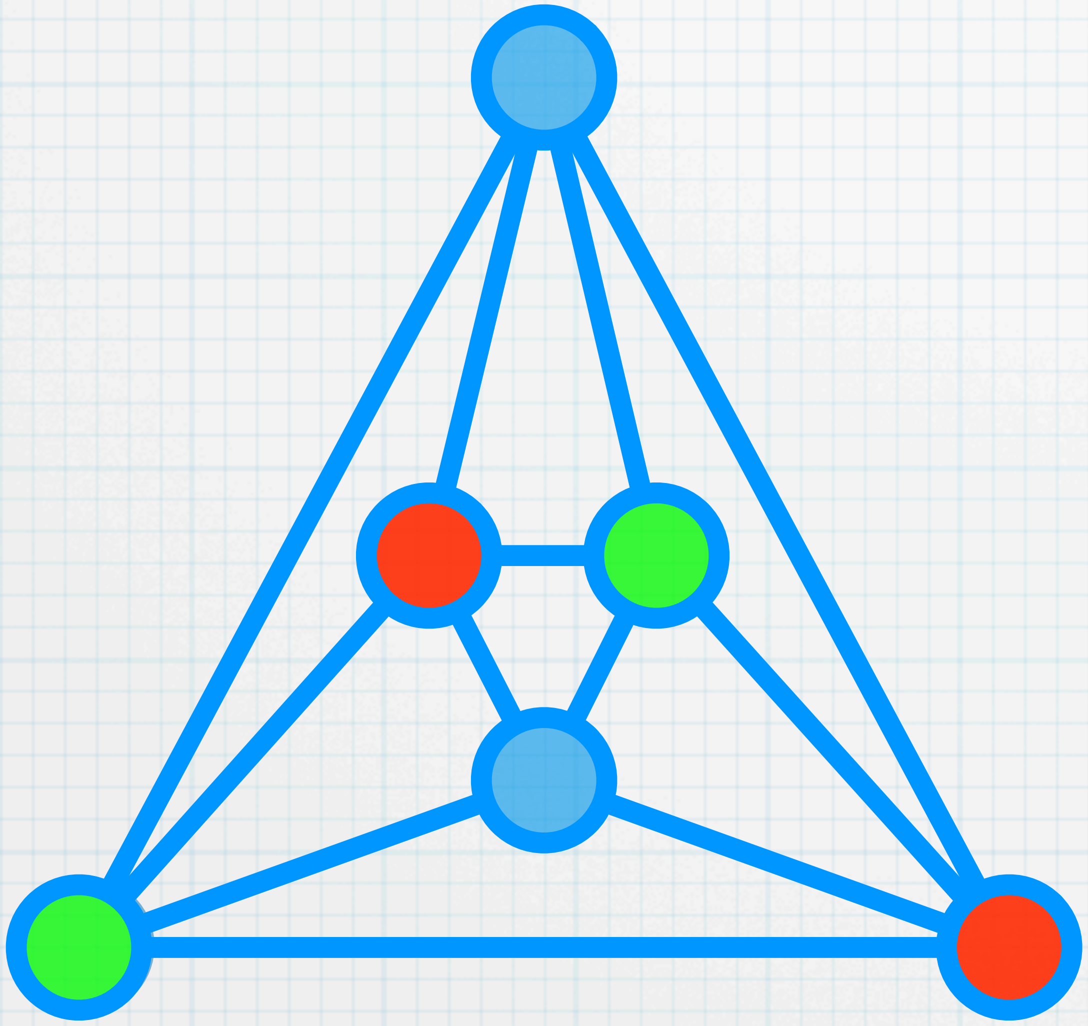
= independent subset of size $2k/3$

triangle = independent subset of size $3k/3$

Exercise: Graph colouring



Exercise: Graph colouring



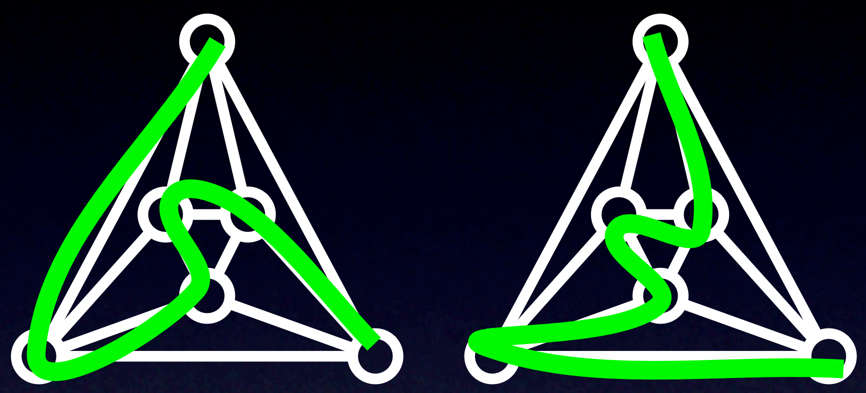
Count the number of 2-colourings

Moebius inversion

Pedestrian view: inclusion–exclusion

TSP

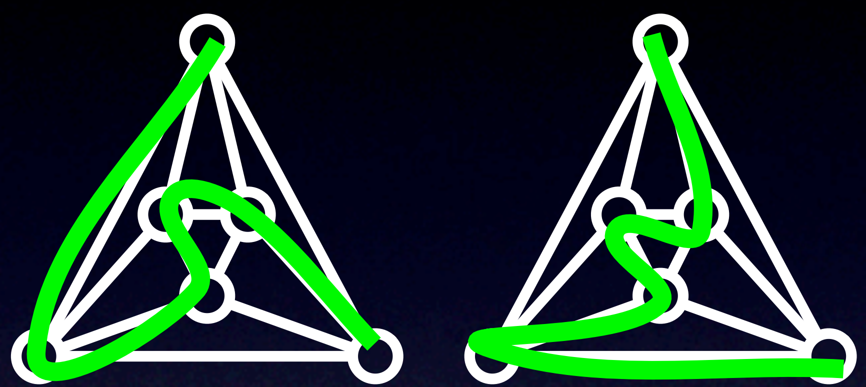
Want:



$s-t$ walks of length n that avoid no vertices

TSP

Want:



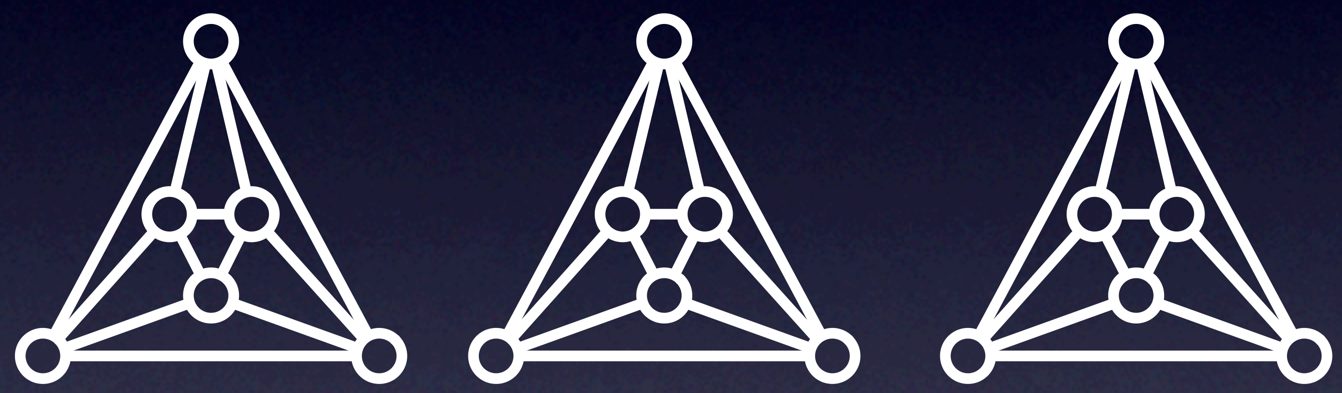
s–t walks of length n that avoid no vertices

Can count:

s–t walks of length n

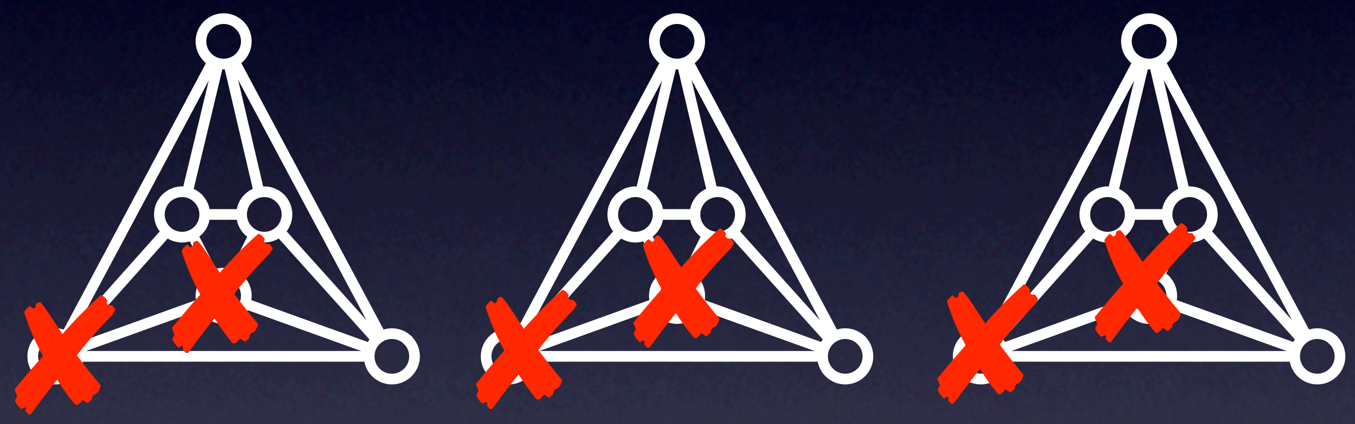
TSP

Can even explicitly forbid certain vertices:



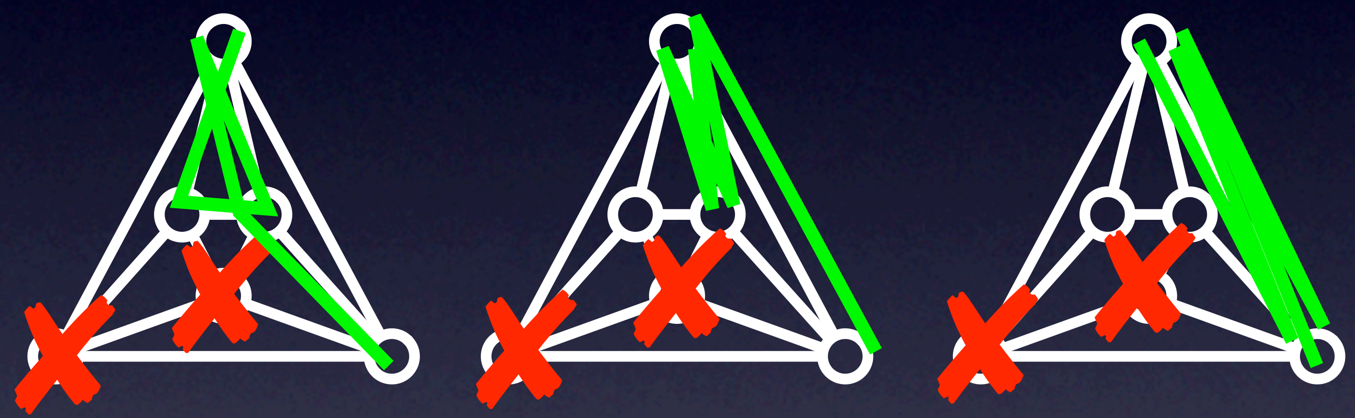
TSP

Can even explicitly forbid certain vertices:



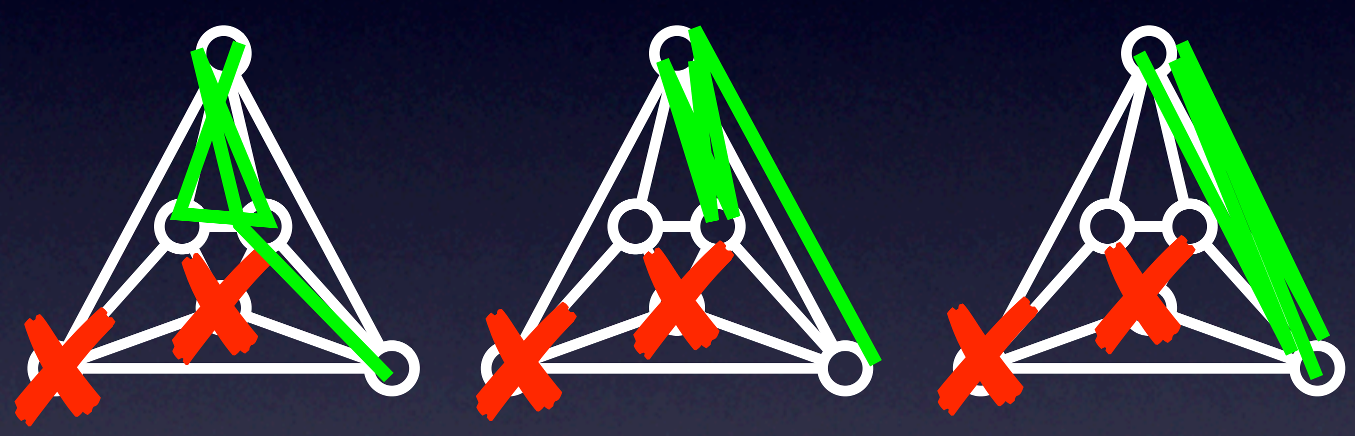
TSP

Can even explicitly forbid certain vertices:

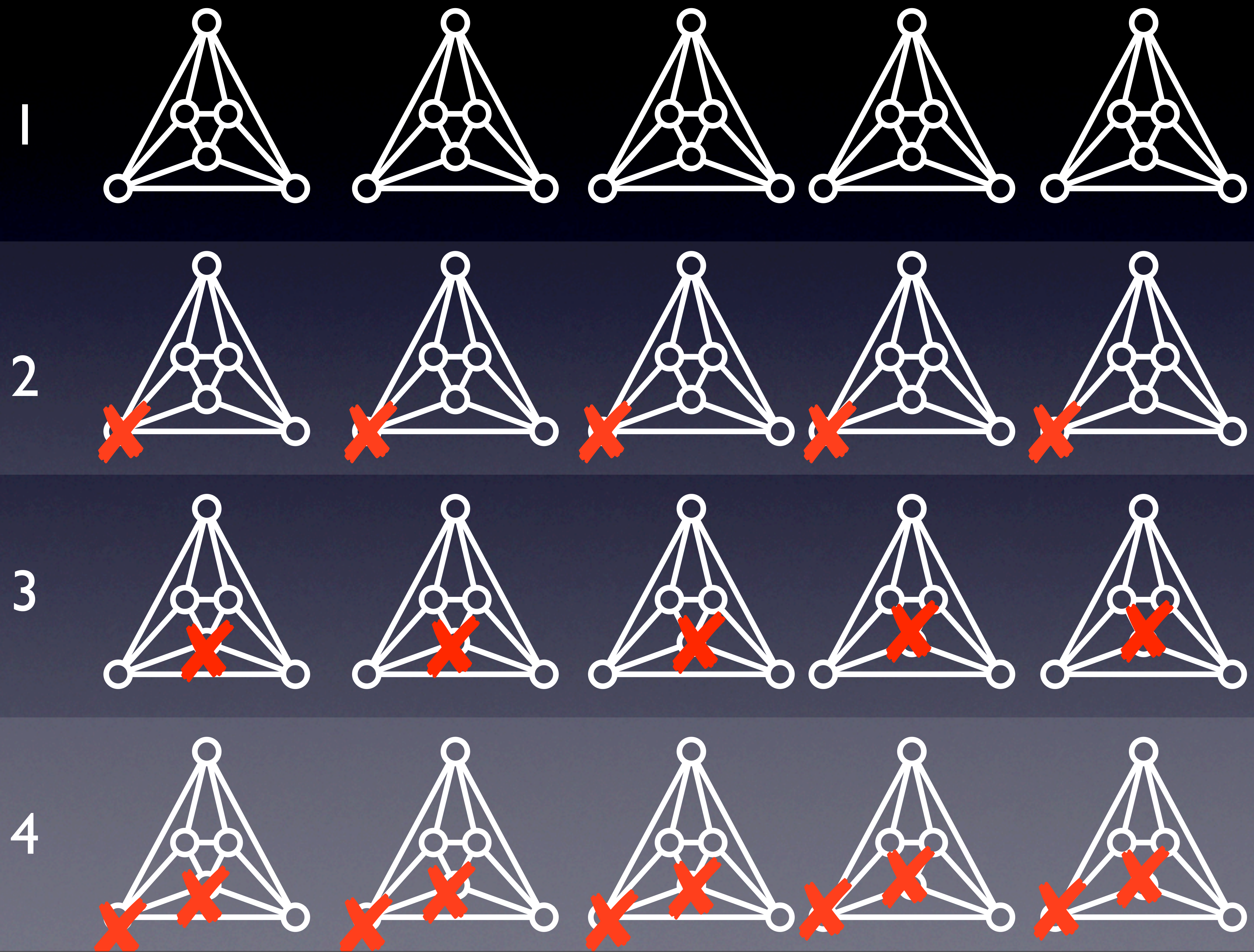


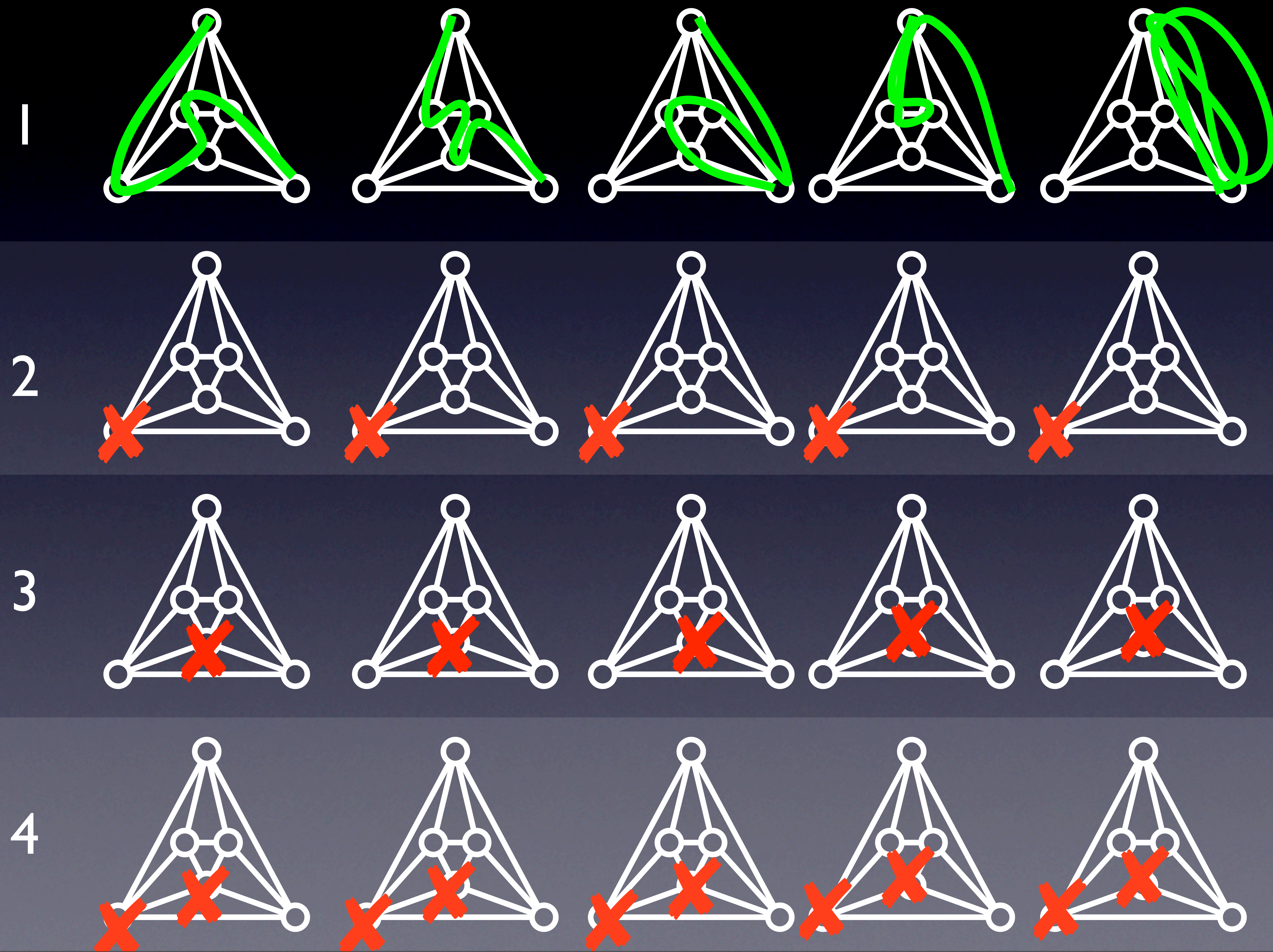
TSP

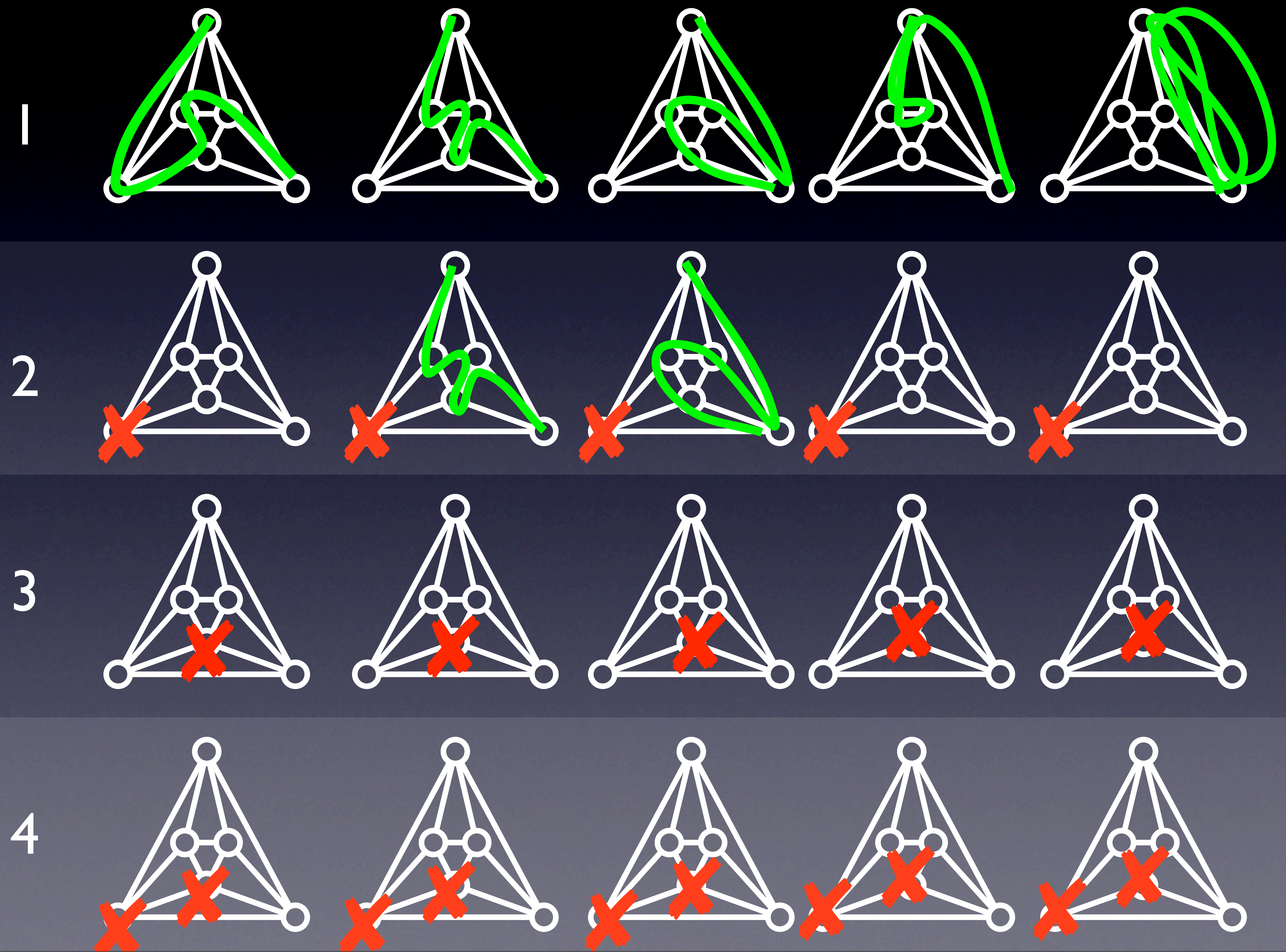
Can even explicitly forbid certain vertices:

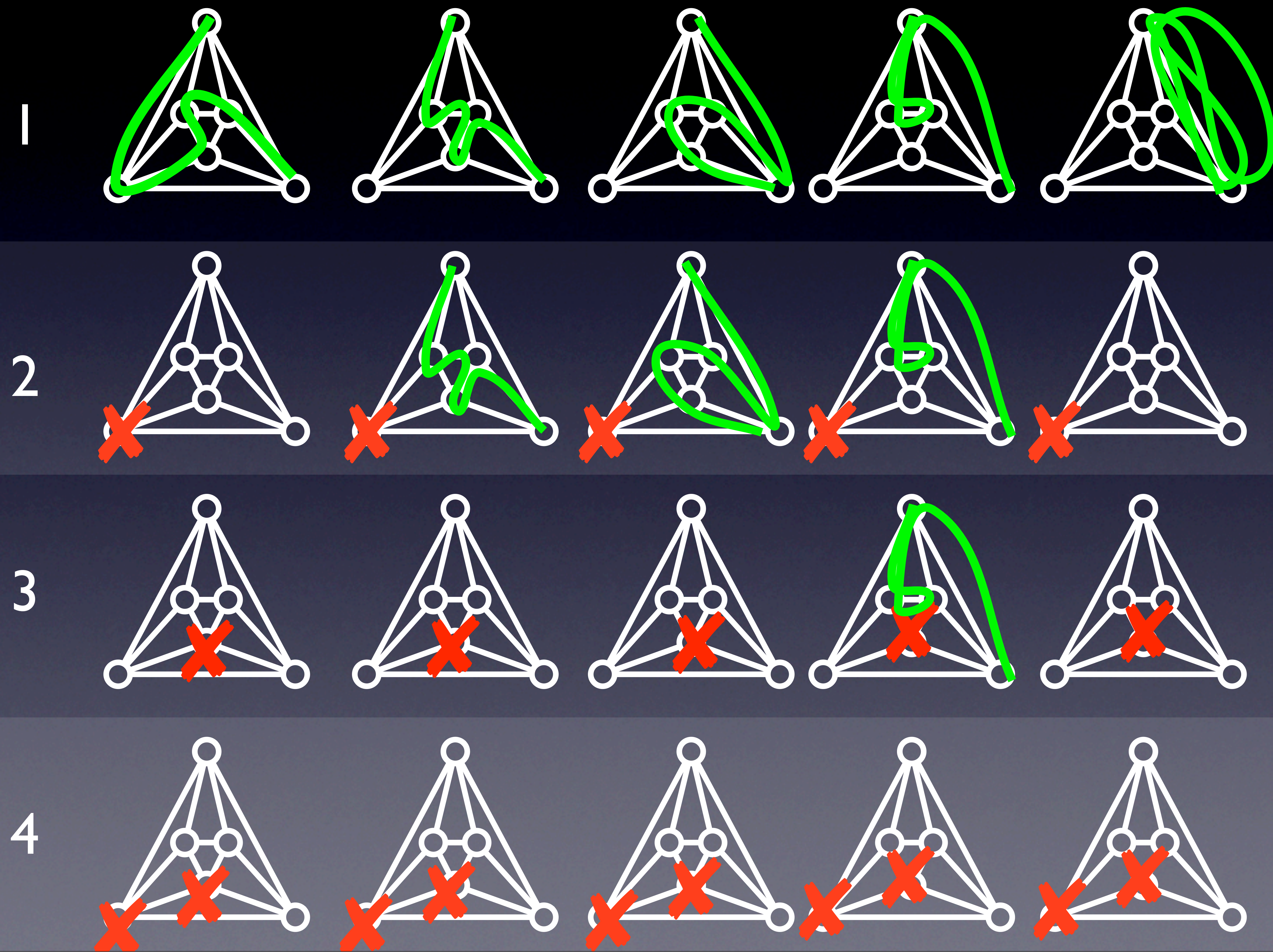


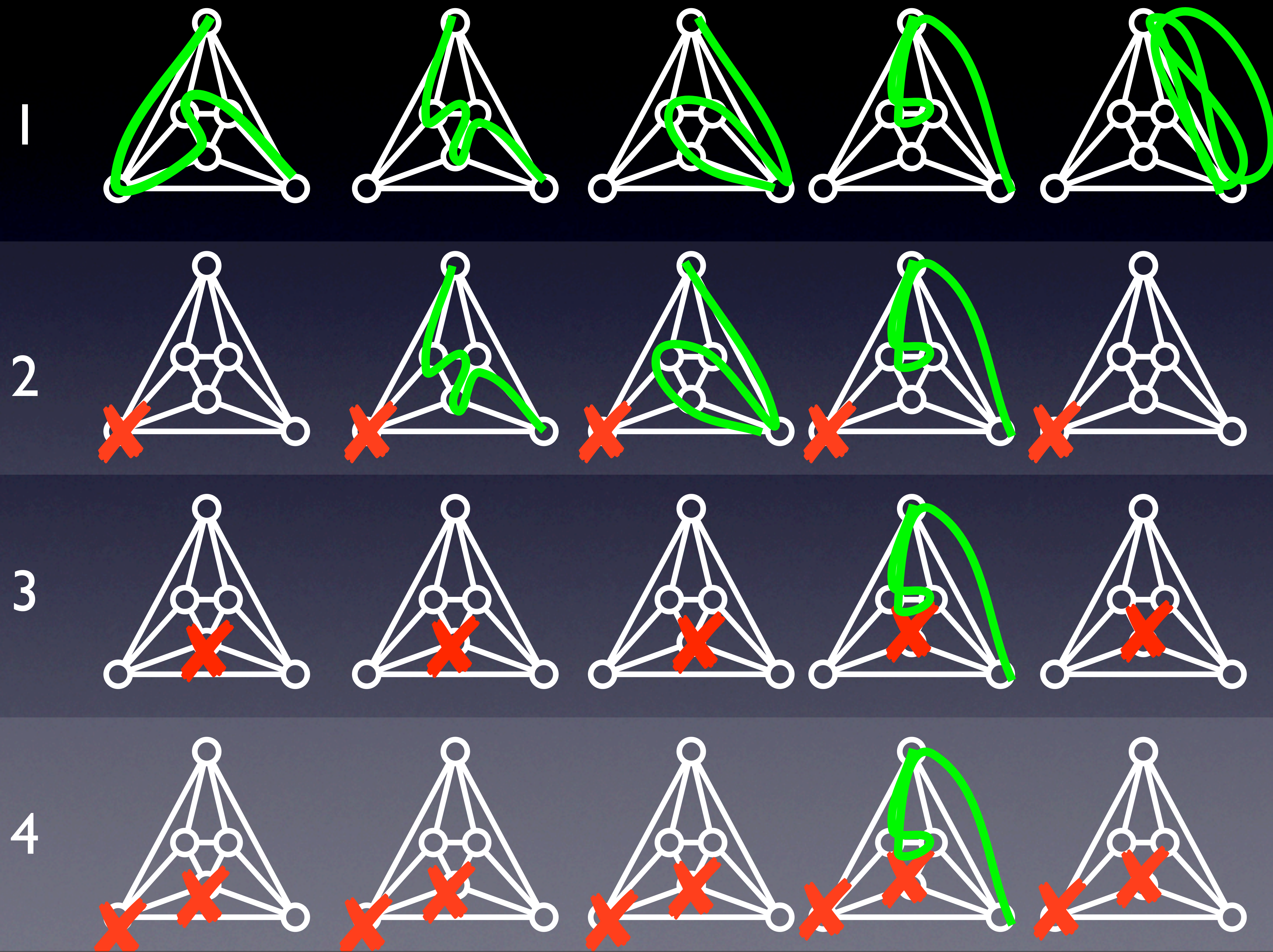
Can count $s-t$ walks of length n that avoid given subset of vertices

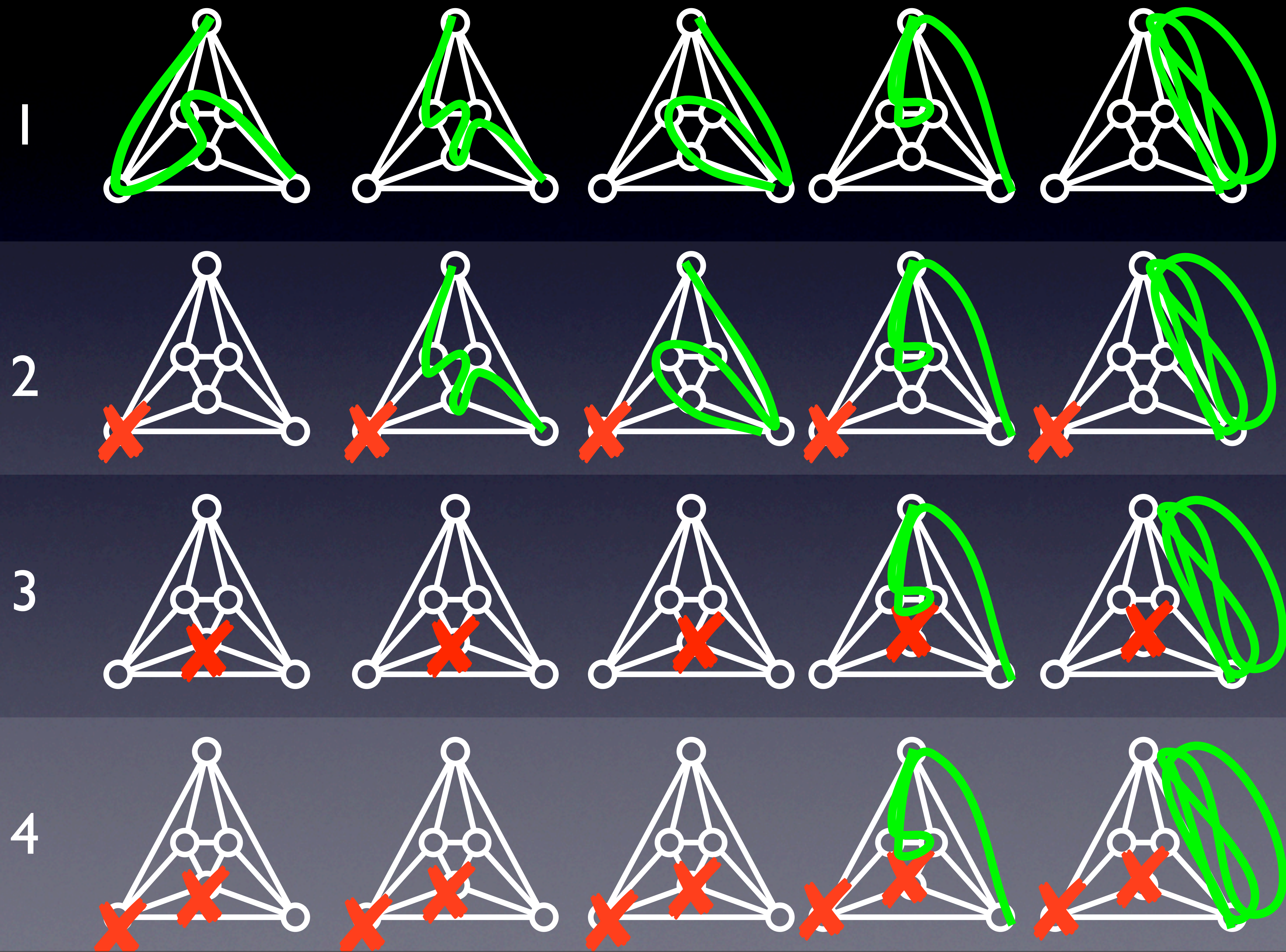

1
2
3
4


1
2
3
4


1
2
3
4


1
2
3
4


1
2
3
4


1
2
3
4

TSP

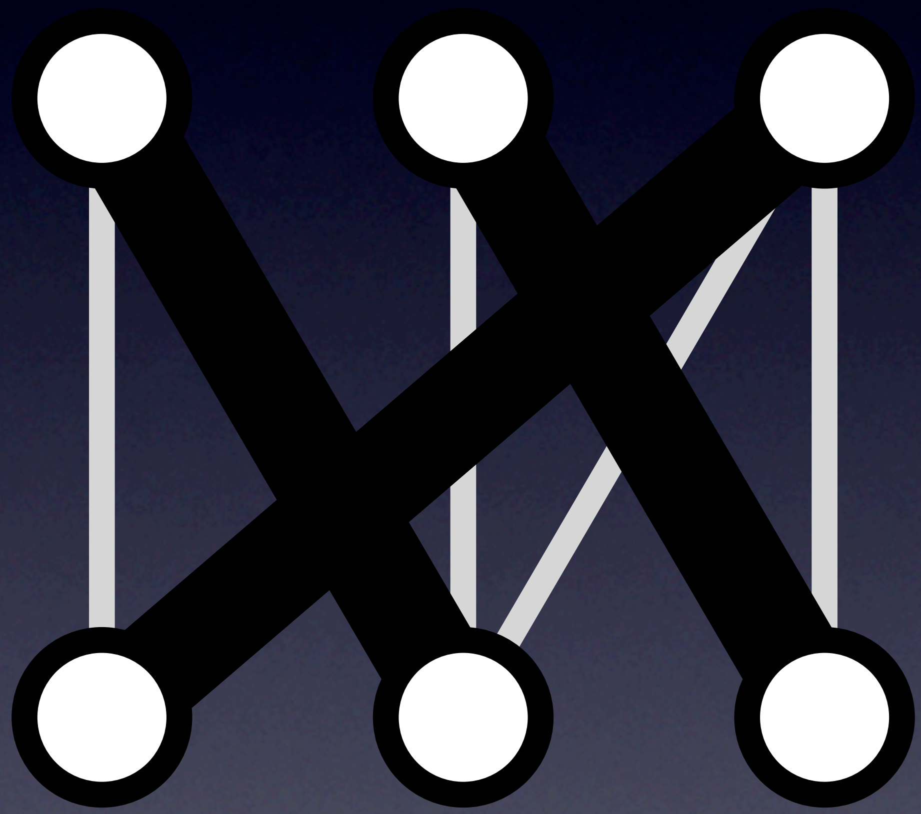
$$\sum_{X \subseteq V} (-1)^{|X|} a(X)$$

TSP

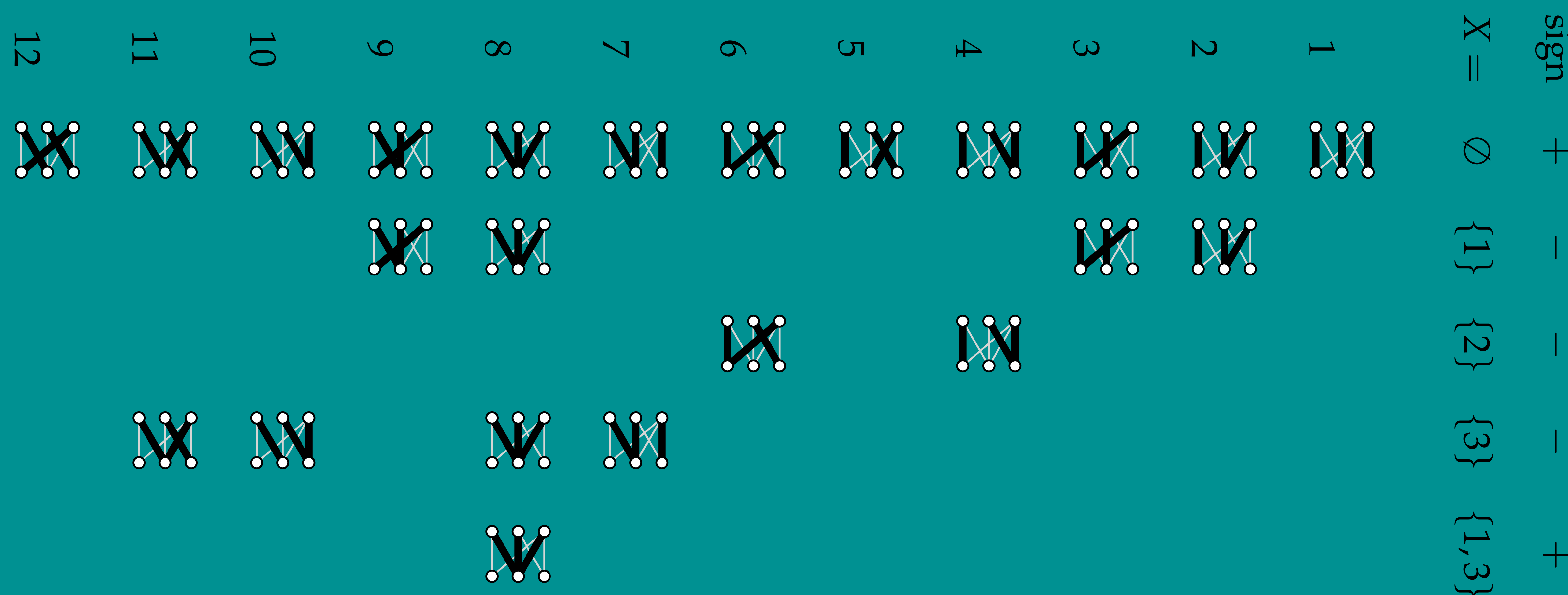
$$\sum_{X \subseteq V} (-1)^{|X|} a(X)$$

Time $O^*(2^n)$. Polyspace.

Perfect matchings

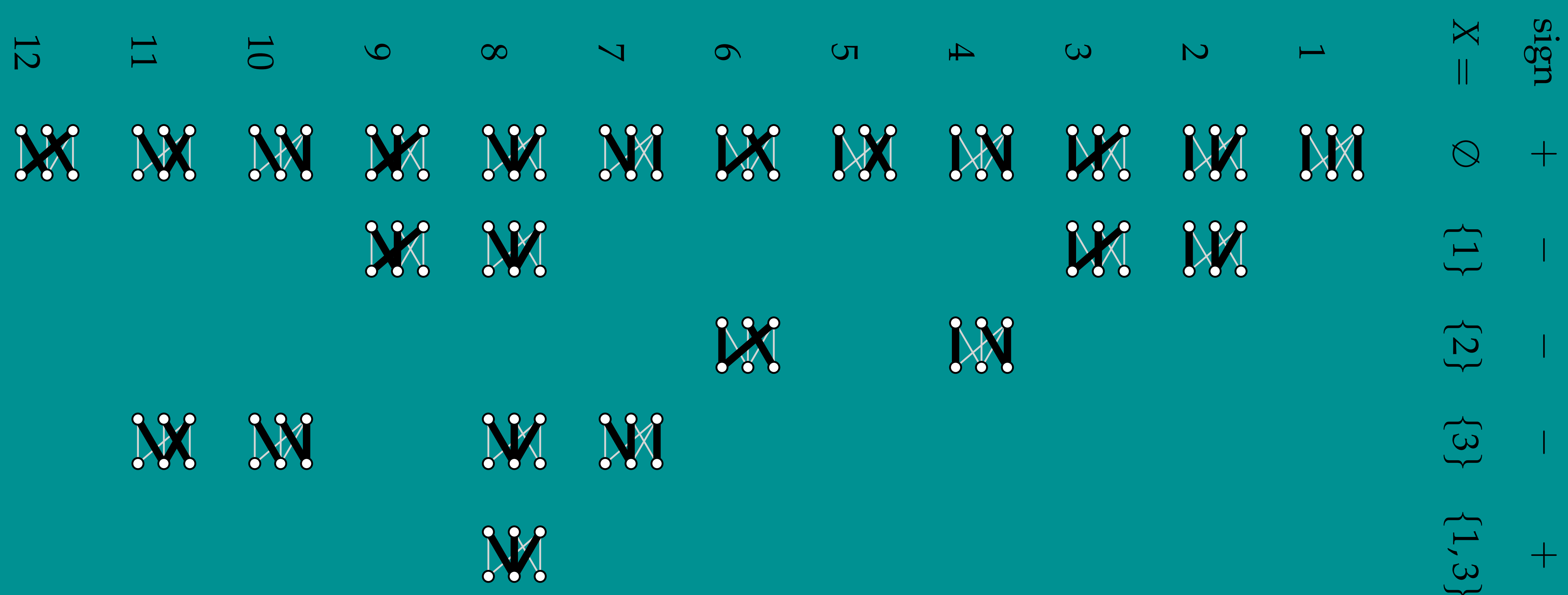


Perfect matchings



Perfect matchings

$$\sum_{X \subseteq Y} (-1)^{|X|} \prod_{i=1}^{k} \sum_{j \notin X} A_{ij}$$



Moebius inversion

Pedestrian view: inclusion–exclusion

Algebraic view: transformation on the subset lattice today

- Brute force
- Greedy
- Divide and *whatever*
- Transformation

tomorrow

- Iterative improvement (flow, simplex, local search)
- Time–space tradeoffs (dynamic programming)

Back to transformation

Moebius inversion

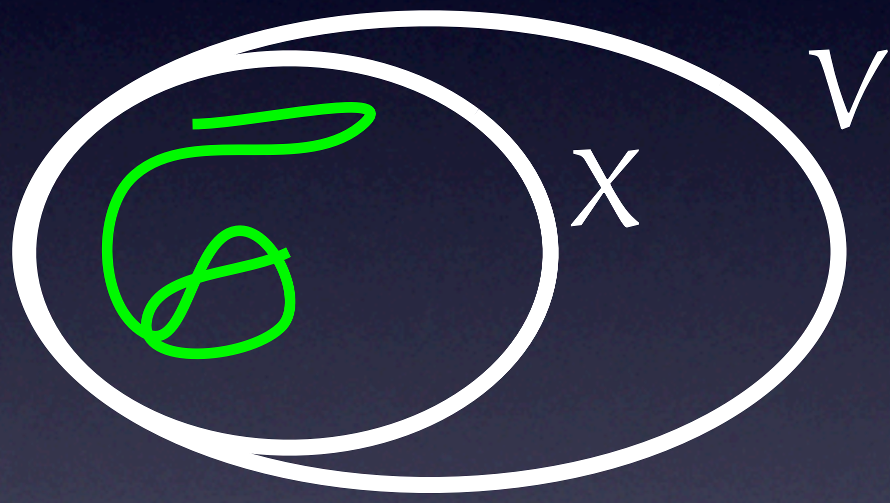
If $g(X) = \sum_{Y \subseteq X} f(Y)$ then $f(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} g(Y)$

V

Moebius inversion

If $g(X) = \sum_{Y \subseteq X} f(Y)$ then $f(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} g(Y)$

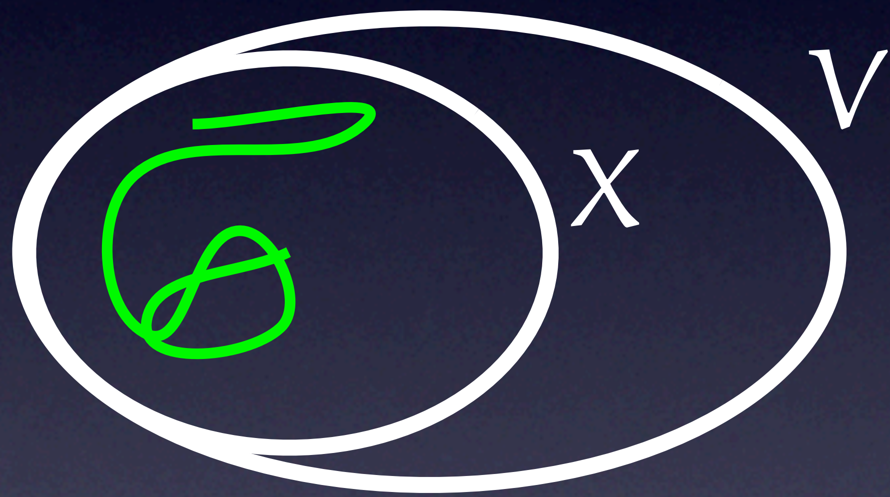
$g(X)$ = # walks
of length n from s to t
using *some* of the X



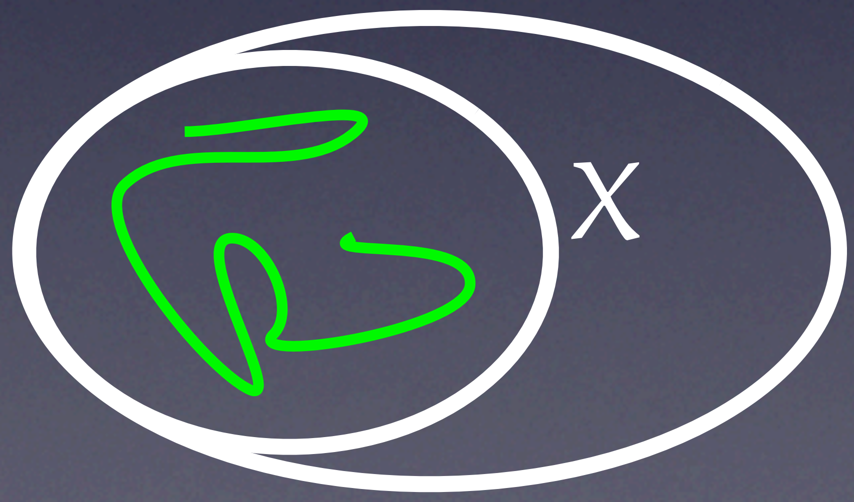
Moebius inversion

If $g(X) = \sum_{Y \subseteq X} f(Y)$ then $f(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} g(Y)$

$g(X)$ = # walks of length n from s to t using *some* of the X



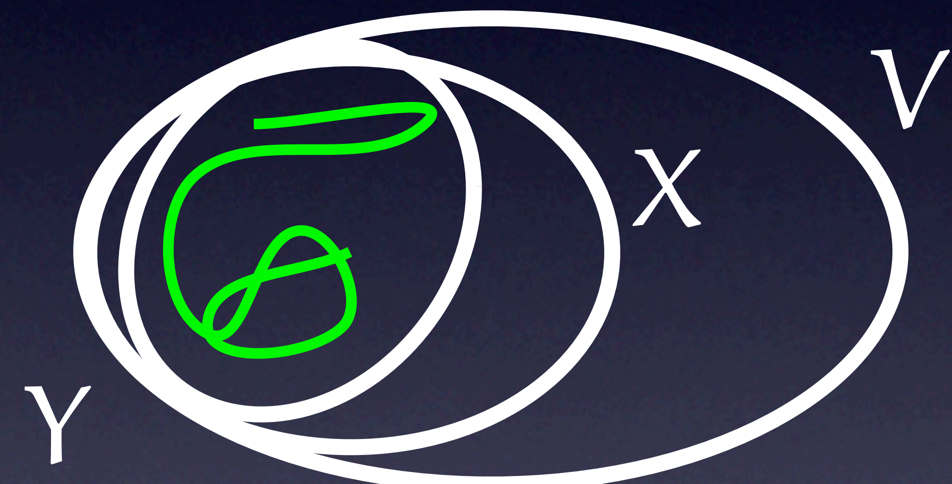
$f(X)$ = # walks of length n from s to t that using *all* of the X



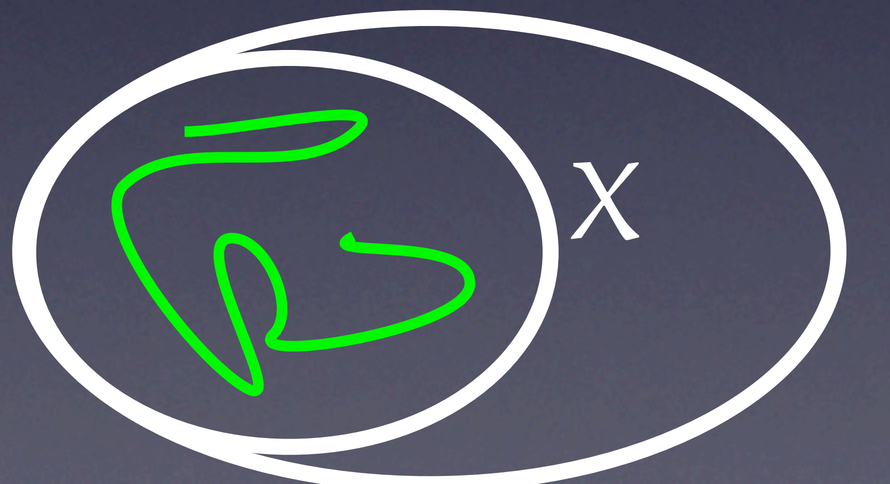
Moebius inversion

If $g(X) = \sum_{Y \subseteq X} f(Y)$ then $f(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} g(Y)$

$g(X)$ = # walks of length n from s to t using *some* of the X



$f(X)$ = # walks of length n from s to t that using *all* of the X



Moebius inversion

If $g(X) = \sum_{Y \subseteq X} f(Y)$ then $f(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} g(Y)$

Moebius inversion

If $g(X) = \sum_{Y \subseteq X} f(Y)$ then $f(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} g(Y)$

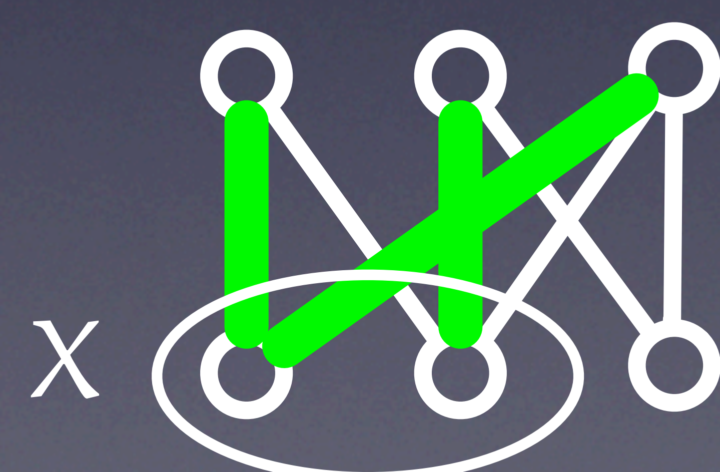
g(X) = # ways
for the boys to pick
some of the girls from X

Moebius inversion

If $g(X) = \sum_{Y \subseteq X} f(Y)$ then $f(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} g(Y)$

$g(X)$ = # ways
for the boys to pick
some of the girls from X

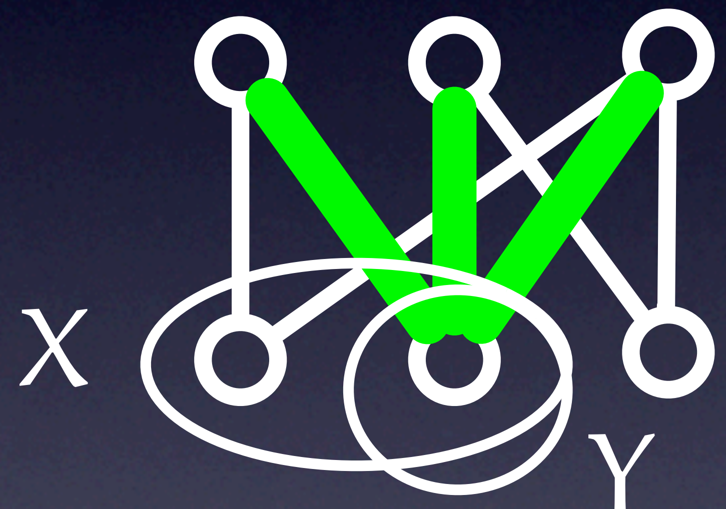
$f(X)$ = # ways
for the girls to pick
all of the girls from X



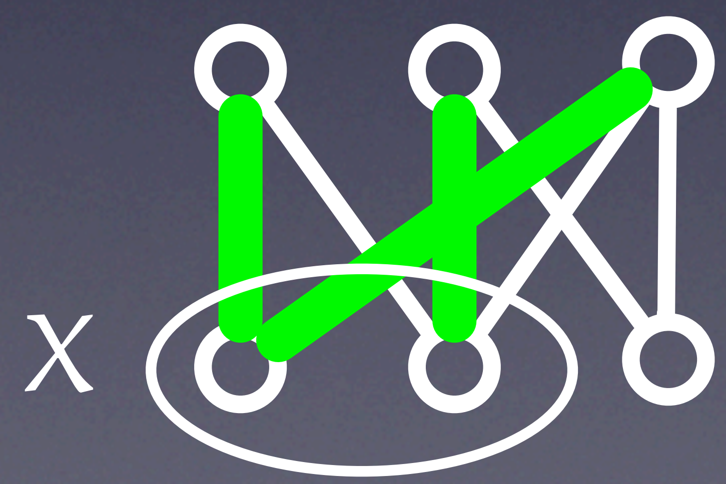
Moebius inversion

If $g(X) = \sum_{Y \subseteq X} f(Y)$ then $f(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} g(Y)$

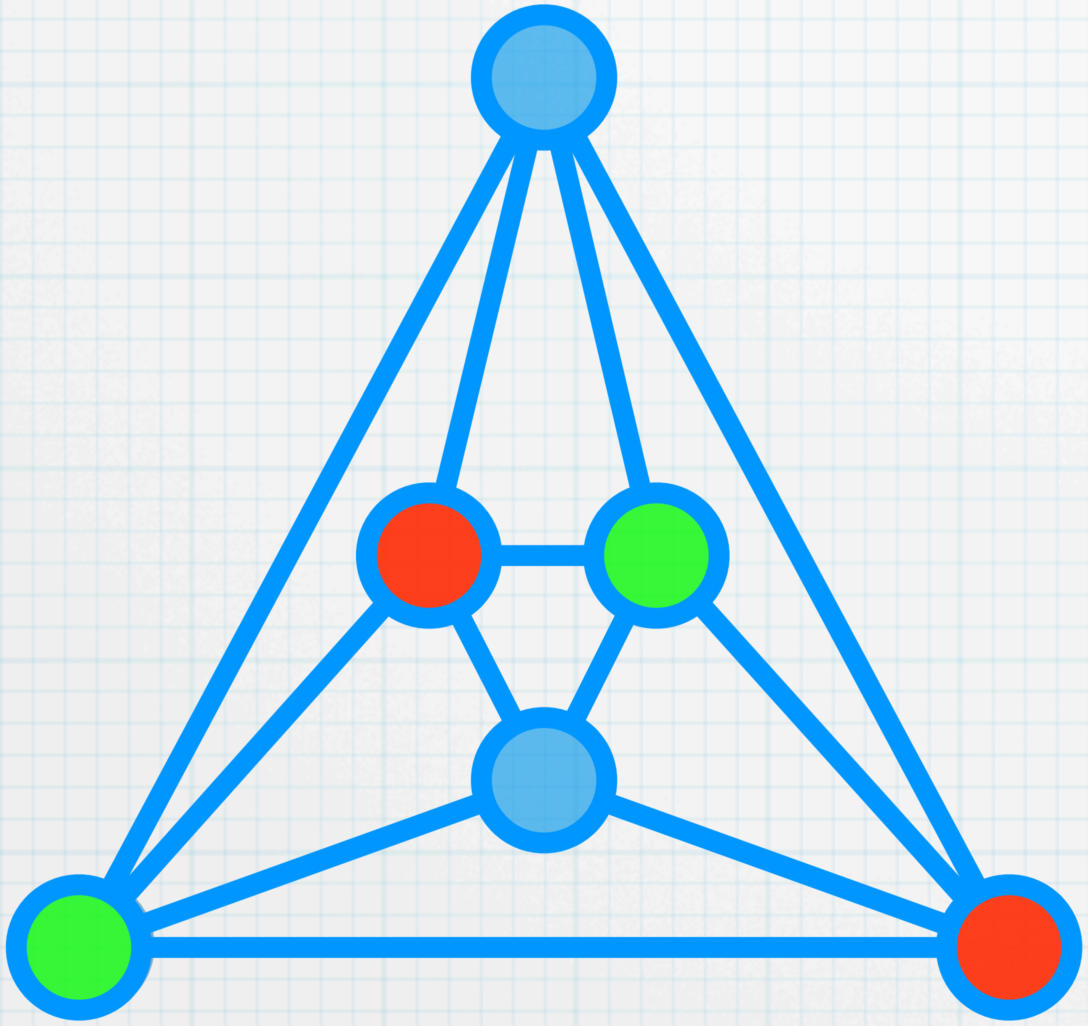
$g(X)$ = # ways
for the boys to pick
some of the girls from X



$f(X)$ = # ways
for the girls to pick
all of the girls from X

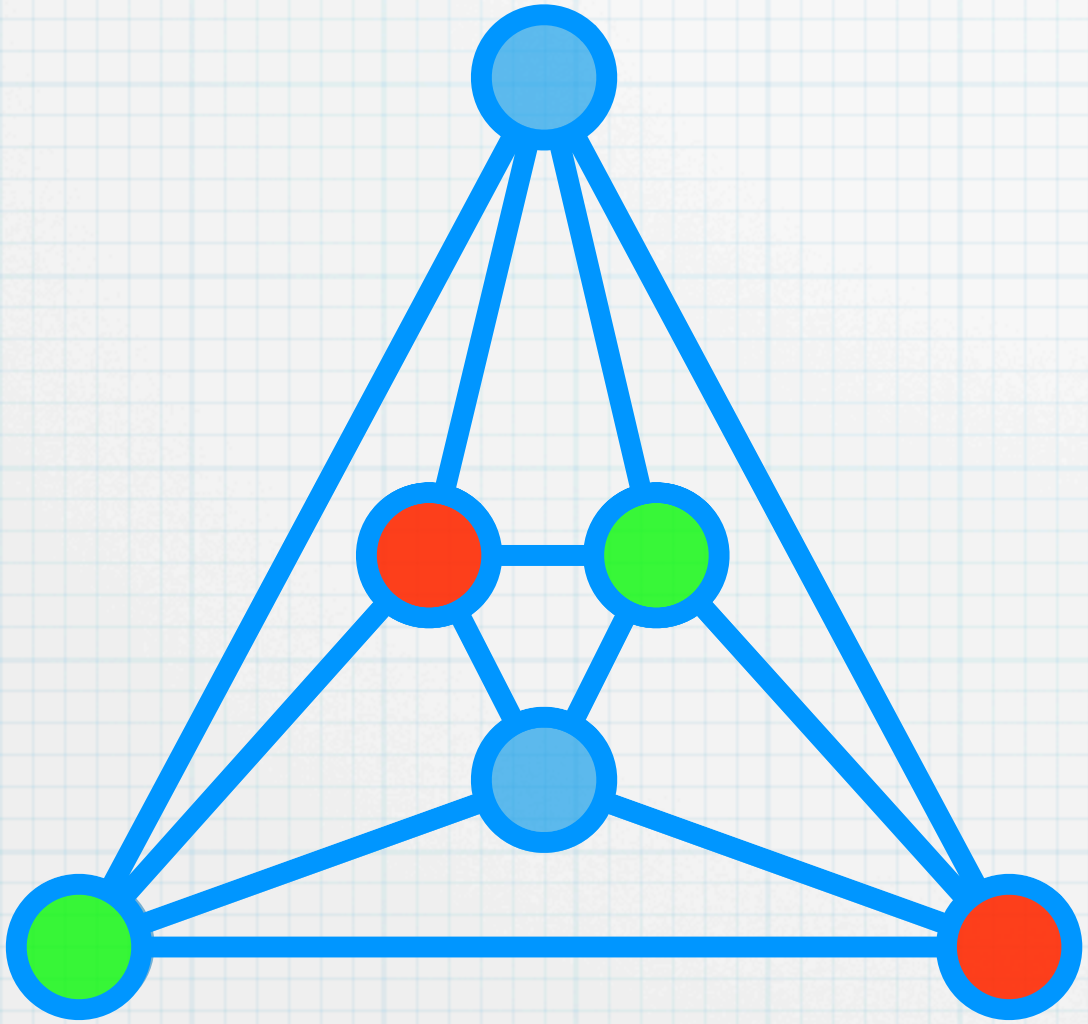


Exercise: Graph colouring



Hint: g(X)= # ways to pick
k independent sets
(not necessarily disjoint)
using some of the vertices
in X

Exercise: Graph colouring



Count the k-colourings in time $O^*(3^n)$

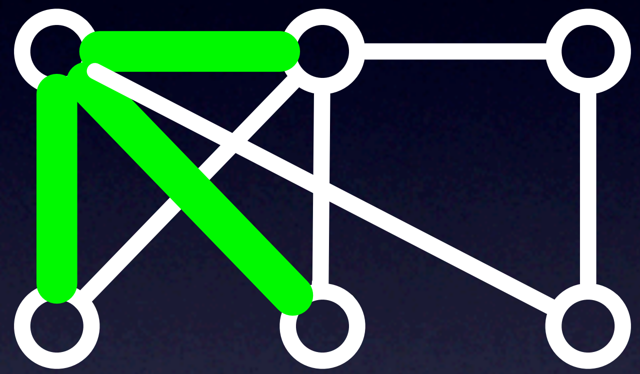
Hint: $g(X)$= # ways to pick k independent sets (not necessarily disjoint) using some of the vertices in X

If you use Yates, time becomes $O^*(2^n)$

Perfect matchings in *general* graphs

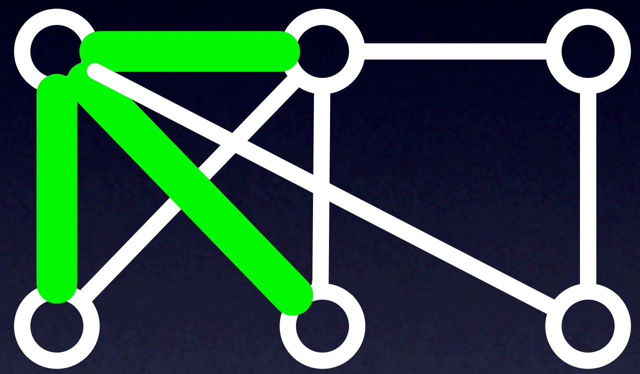
If $g(X) = \sum_{Y \subseteq X} f(Y)$ **then** $f(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} g(Y)$

$f(X)$ = # ways to pick $n/2$ edges
using *all* vertices in X

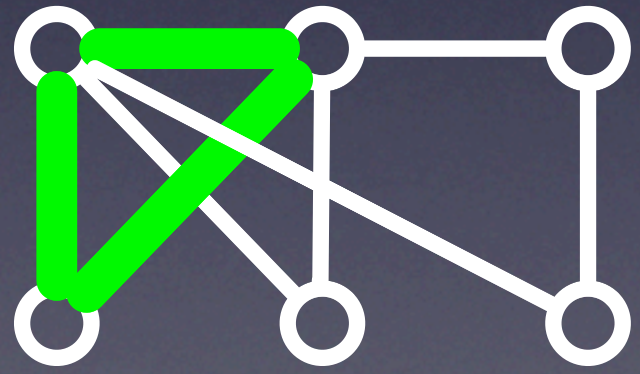


If $g(X) = \sum_{Y \subseteq X} f(Y)$ then $f(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} g(Y)$

$f(X)$ = # ways to pick $n/2$ edges using *all* vertices in X



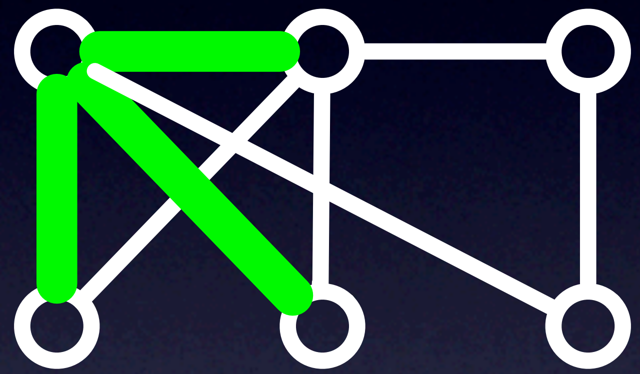
$g(X)$ = # ways to pick $n/2$ edges using *some* vertices in X



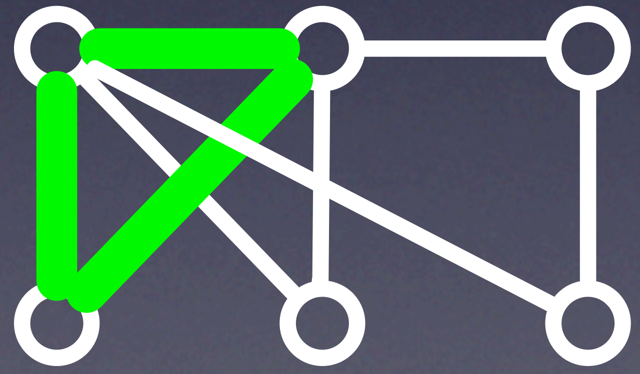
$$\text{If } g(X) = \sum_{Y \subseteq X} f(Y) \qquad \text{then } f(X) = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} g(Y)$$

Time $O^*(2^n)$. Polyspace.

$f(X)$ = # ways to pick $n/2$ edges using *all* vertices in X



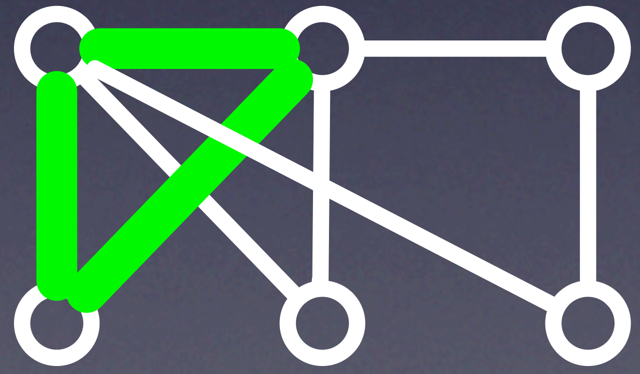
$g(X)$ = # ways to pick $n/2$ edges using *some* vertices in X



$$f(V) = \sum_{X \subseteq V} (-1)^{|V \setminus X|} g(X)$$

If $G[X]$ has k edges then $g(X)$ = ?

$g(X)$ = # ways to pick $n/2$ edges using *some* vertices in X



$$f(V) = \sum_{X \subseteq V} (-1)^{|V \setminus X|} g(X) = \sum_{X \subseteq V} (-1)^{|V \setminus X|} e(G[X])^{n/2}$$

$$= \sum_{k=0}^{m} \sum_{\substack{X \subseteq V \\ e(G[X])=k}} (-1)^{|V \setminus X|} k^{n/2}$$

$$= \sum_{k=0}^{m} \sum_{r=0}^{n} \sum_{\substack{X \subseteq V \\ e(G[X])=k \\ |X|=r}} (-1)^{|n-r|} k^{n/2}$$

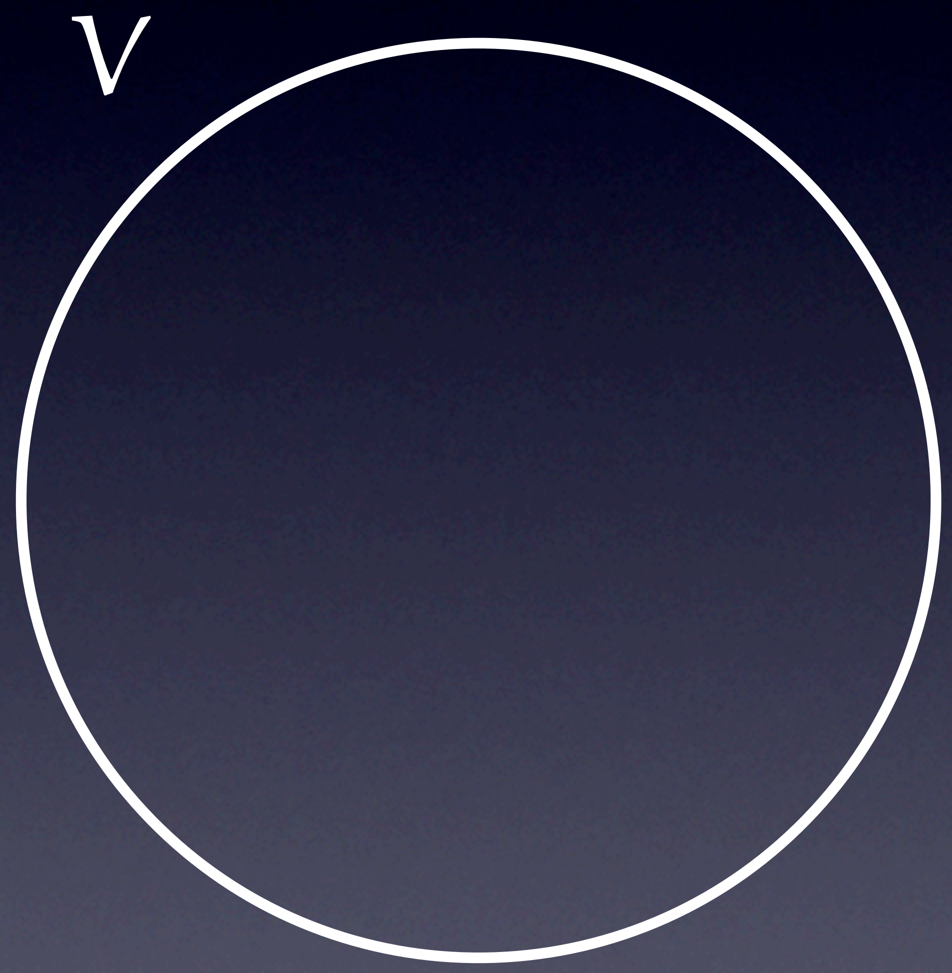
$$= \sum_{k=0}^{m} \sum_{r=0}^{n} G(n = r; m = k)(-1)^{|n-r|} k^{n/2}$$

Gist: computing

$$f(V) = \sum_{X \subseteq V} (-1)^{|V \setminus X|} g(X)$$

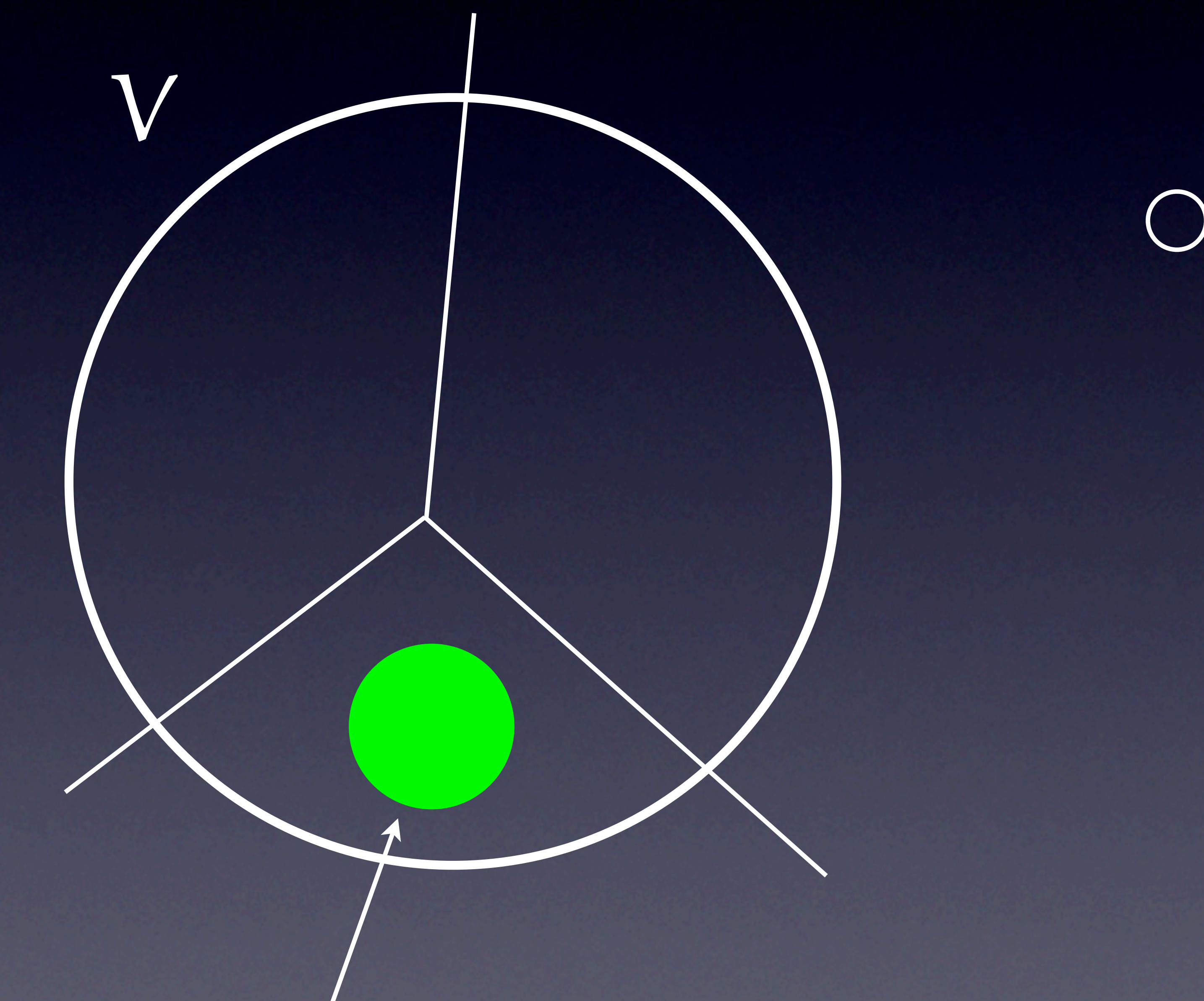
amounts to computing the number of induced subgraphs on r vertices with k edges

Counting triangles



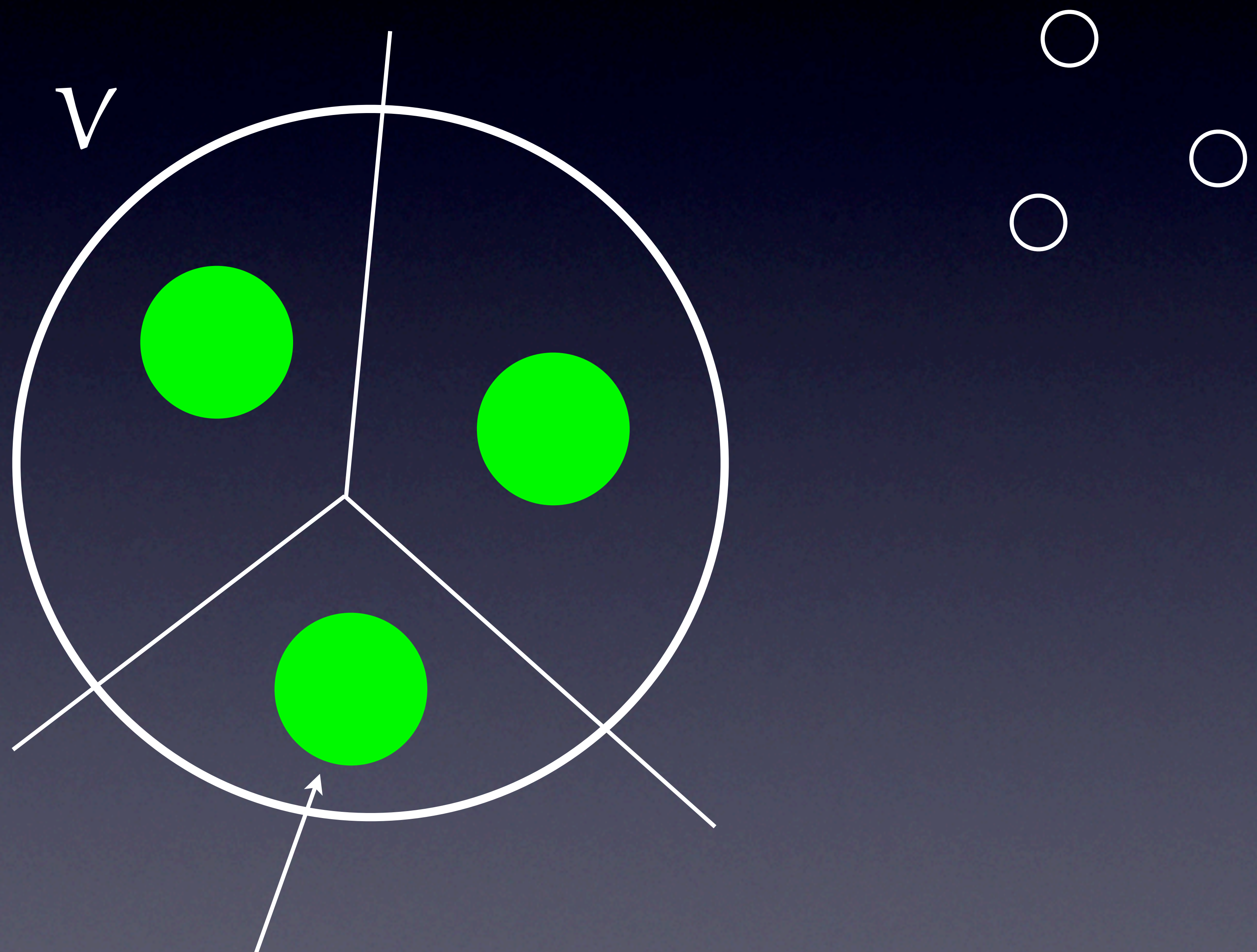
Counting triangles

Counting triangles



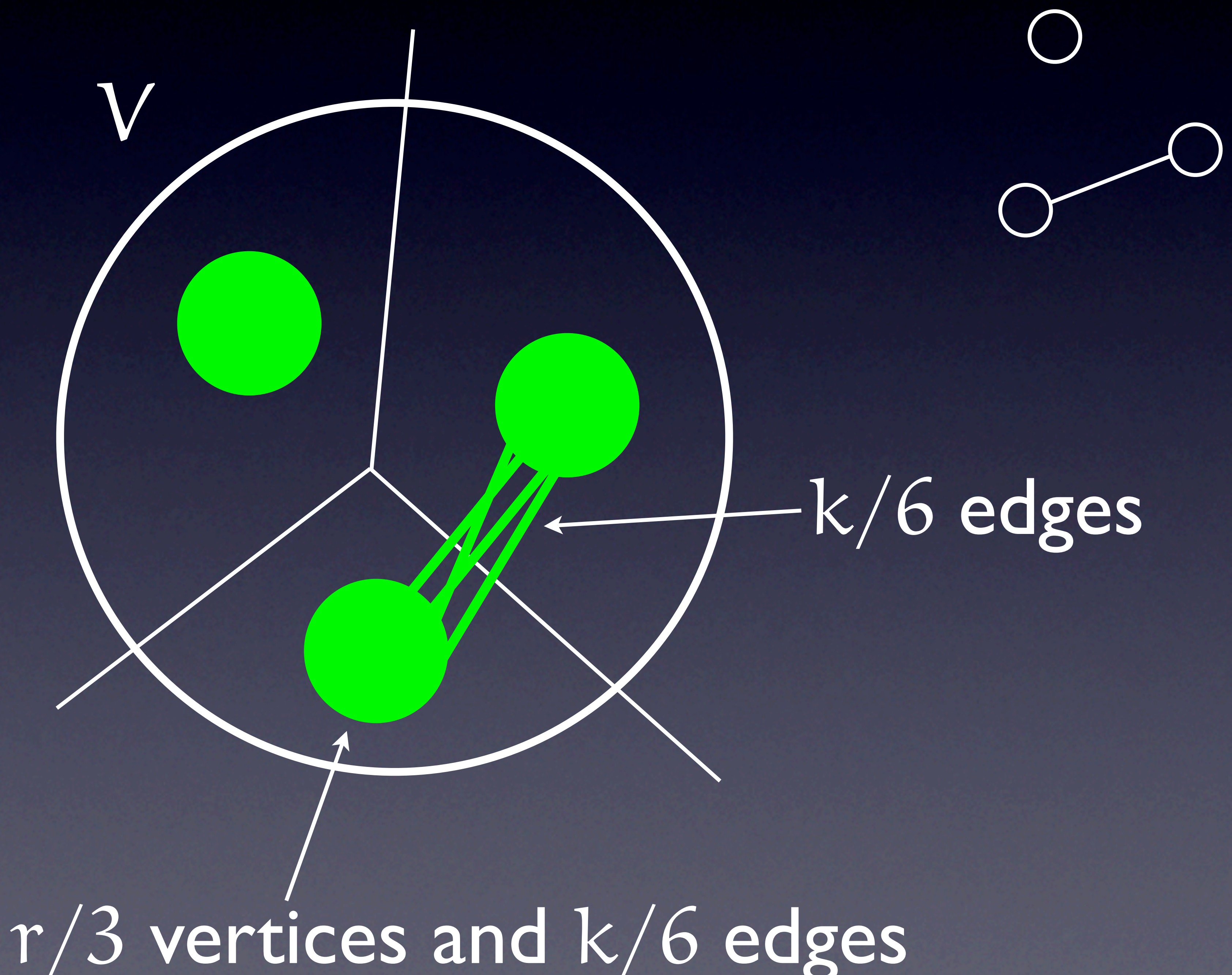
$r/3$ vertices and $k/6$ edges

Counting triangles

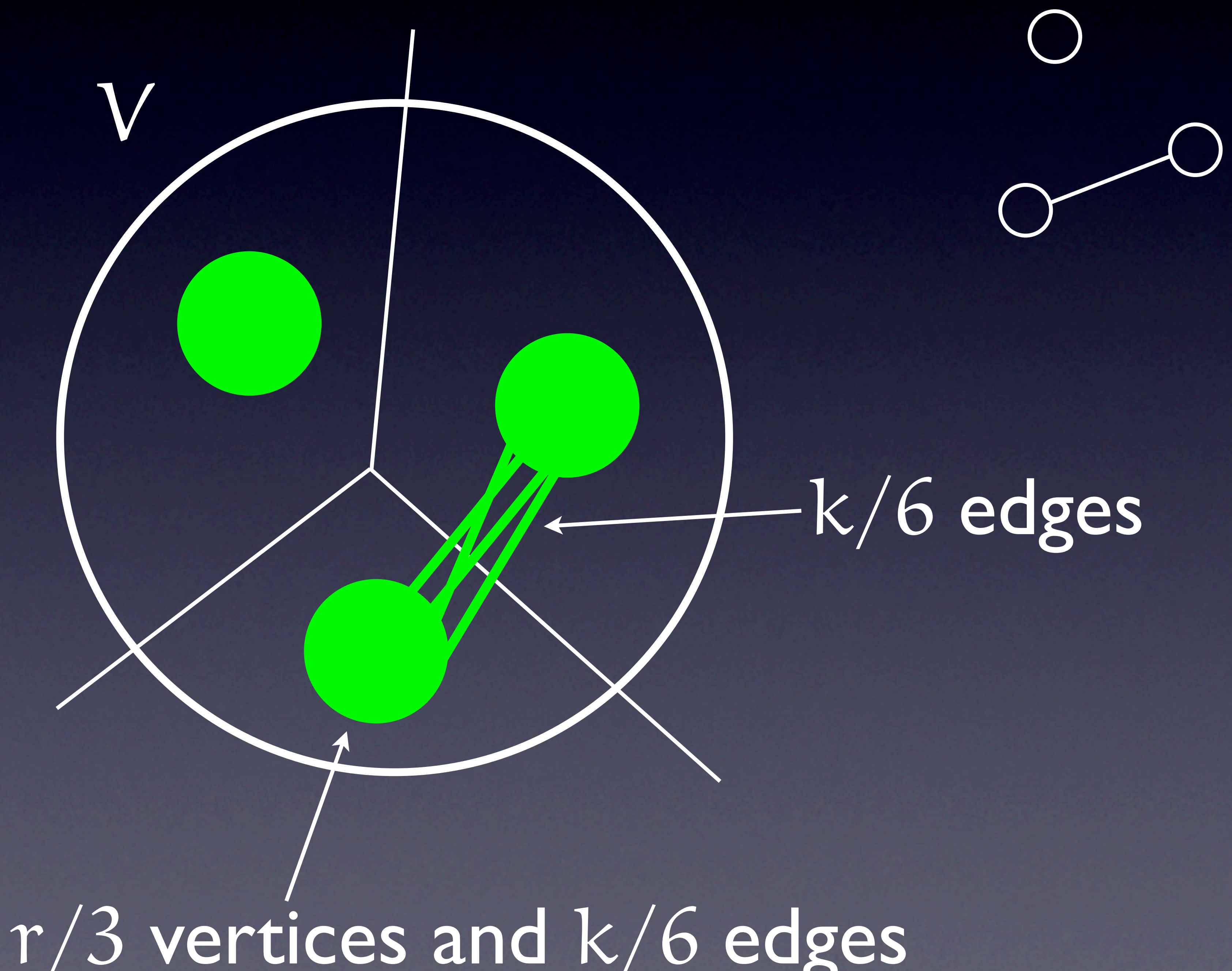


$r/3$ vertices and $k/6$ edges

Counting triangles

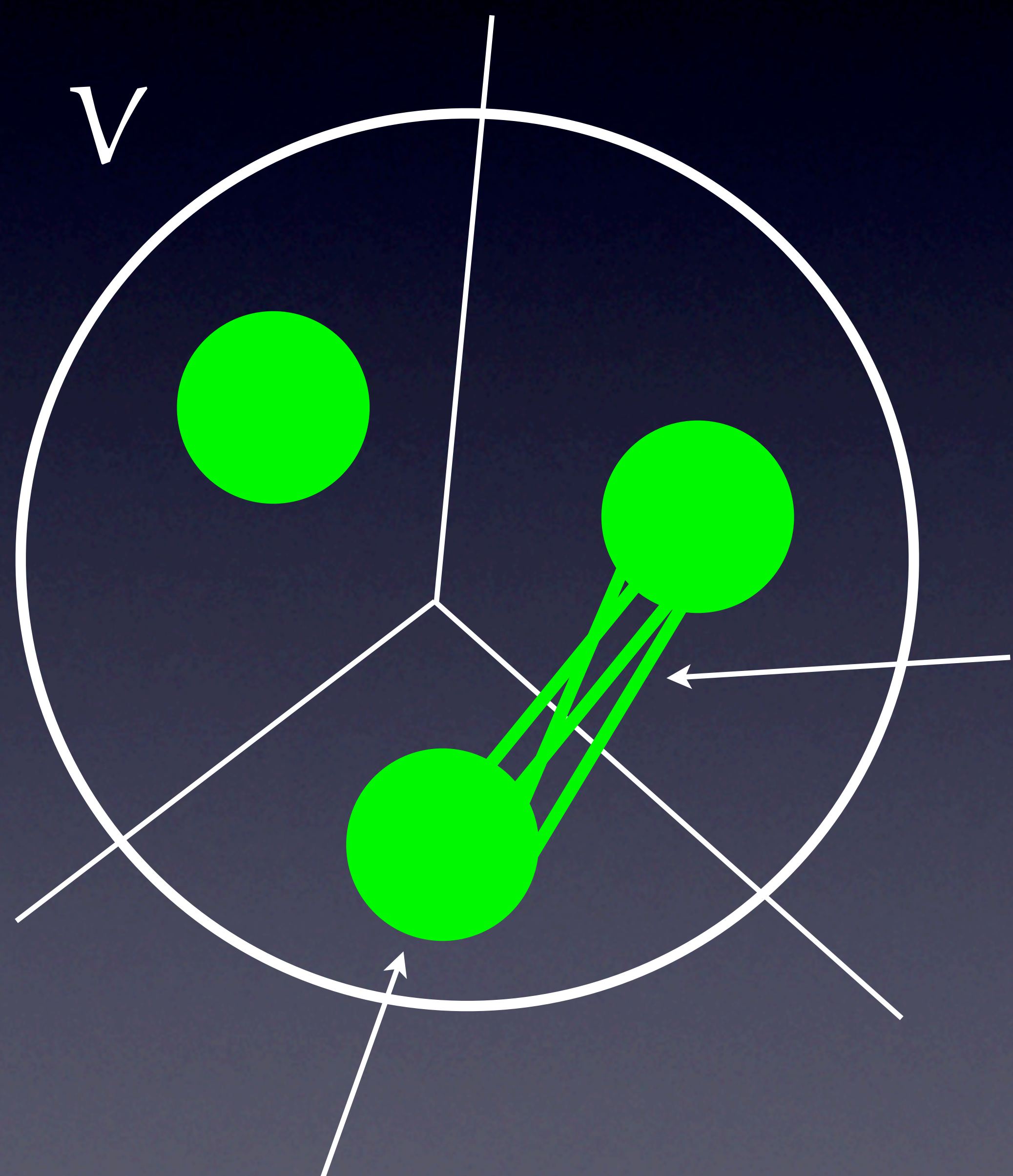


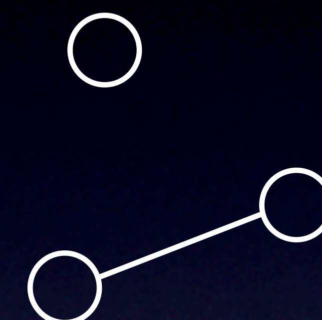
Counting triangles



Triangles correspond to subgraphs with r vertices and k edges

Counting triangles



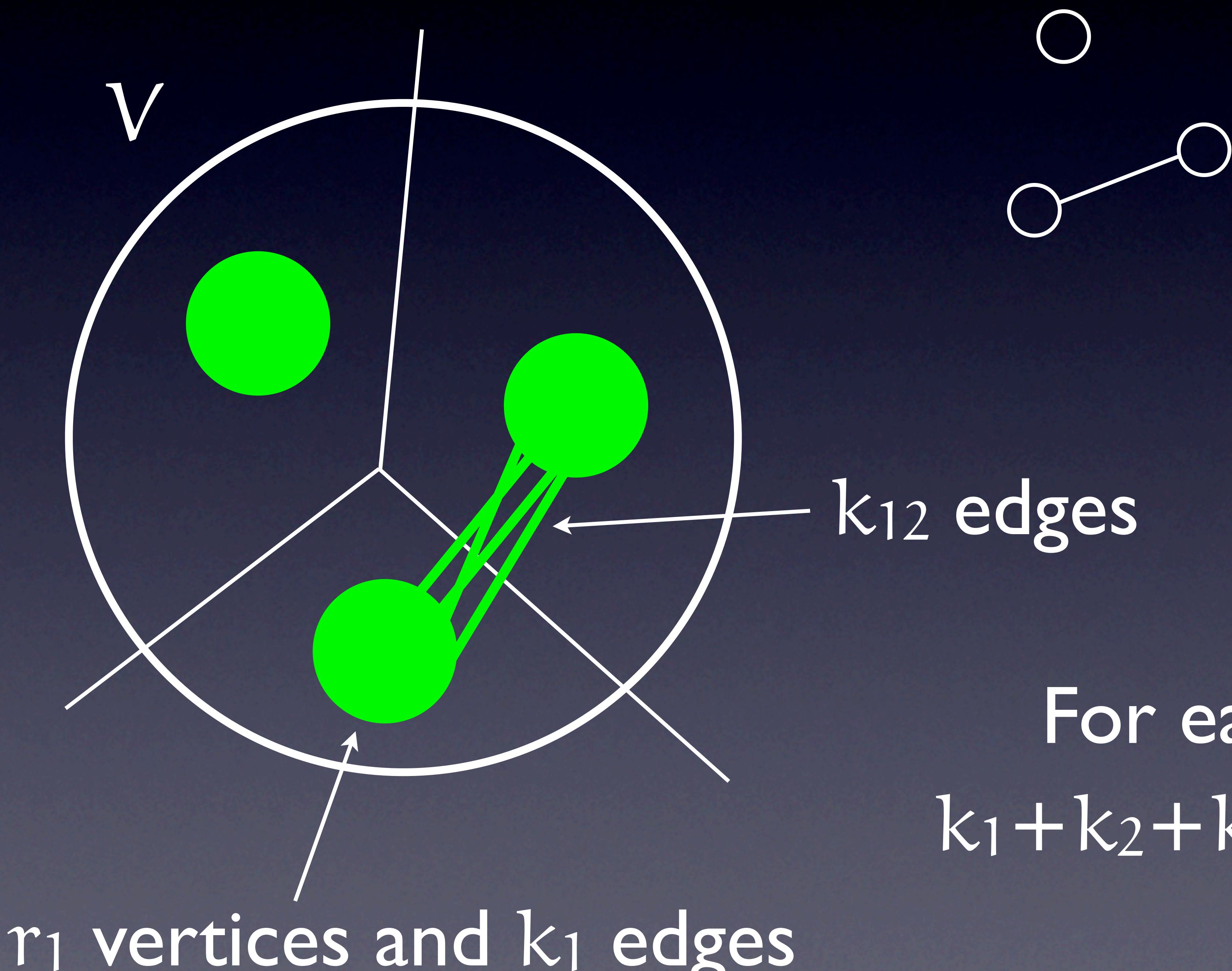


Triangles

Triangles correspond to subgraphs with r vertices and k edges

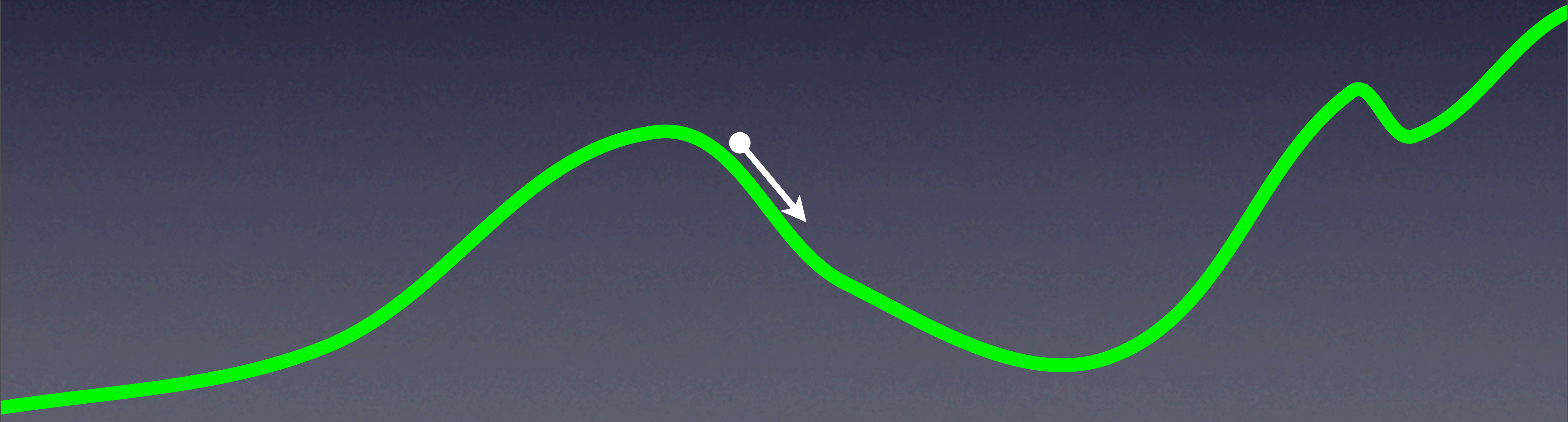
Time $O(2^{\omega n/3})$

Counting triangles



For each $r_1+r_2+r_3=r$,
$k_1+k_2+k_3+k_{12}+k_{13}+k_{23}=k$

Iterative improvement



3-Satisfiability

$$(\neg x \vee y \vee z) \wedge (x \vee \neg y \vee z) \wedge (x \vee y) \wedge (\neg x \vee \neg y \vee z)$$

Variables

1	1	1	1	0	0	1	0	0	1

Clauses

3-Satisfiability

Variables

1	1	1	1	0	0	1	0	0	1

Clauses

3-Satisfiability

1. Pick an unsatisfied clause $(L_1 \vee L_2 \vee L_3)$

Variables

Clauses

3-Satisfiability

1. Pick an unsatisfied clause $(L_1 \vee L_2 \vee L_3)$
2. Pick one of its 3 literals

3-Satisfiability

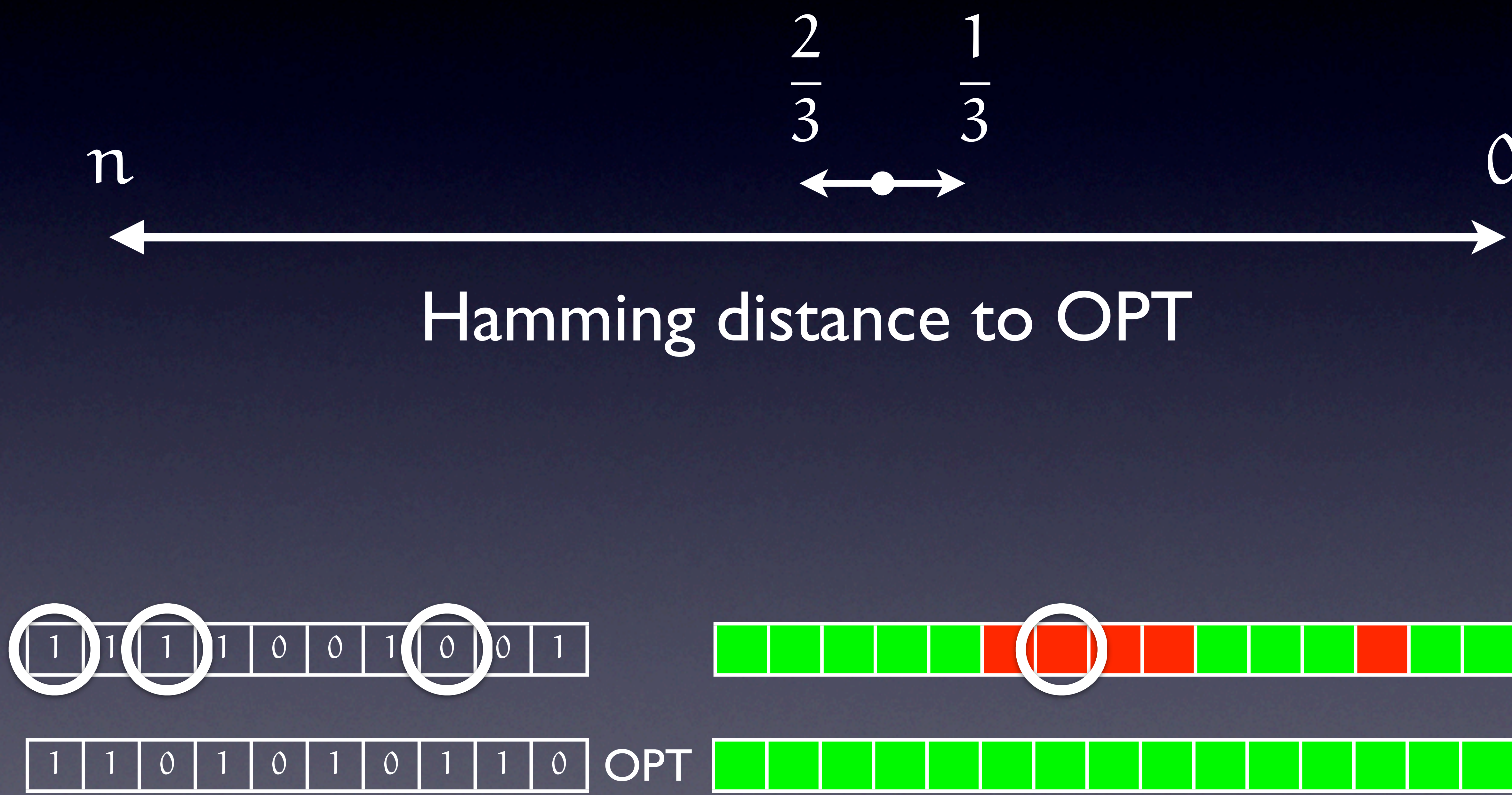
1. Pick an unsatisfied clause ($L_1 \vee L_2 \vee L_3$)
2. Pick one of its 3 literals
3. Flip the corresponding variable

3-Satisfiability

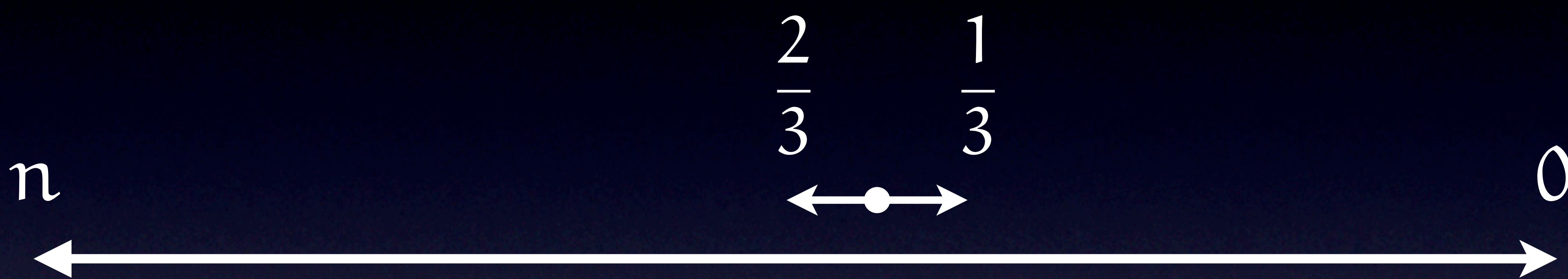
1. Pick an unsatisfied clause $(L_1 \vee L_2 \vee L_3)$
2. Pick one of its 3 literals
3. Flip the corresponding variable

3-Satisfiability

3-Satisfiability



3-Satisfiability



Hamming distance to OPT

$\Pr(\text{from } i \text{ to } 0) = 2^{-i}$

3-Satisfiability

1. Pick an unsatisfied clause ($L_1 \vee L_2 \vee L_3$)
2. Pick one of its 3 literals
3. Flip the corresponding variable

1. Pick an unsatisfied clause $(L_1 \vee L_2 \vee L_3)$
2. Pick one of its 3 literals
3. Flip the corresponding variable

Repeat $3n$ times

1. Pick an unsatisfied clause $(L_1 \vee L_2 \vee L_3)$
2. Pick one of its 3 literals
3. Flip the corresponding variable

Repeat $3n$ times

1. Pick an unsatisfied clause $(L_1 \vee L_2 \vee L_3)$
2. Pick one of its 3 literals
3. Flip the corresponding variable

$$\Pr(\text{from } i \text{ to } 0) = 2^{-i}$$

Repeat a bunch of times

Pick a random assignment

Repeat $3n$ times

1. Pick an unsatisfied clause $(L_1 \vee L_2 \vee L_3)$
2. Pick one of its 3 literals
3. Flip the corresponding variable

$$\Pr(\text{from } i \text{ to } 0) = 2^{-i}$$

Repeat a bunch of times

Pick a random assignment

Repeat $3n$ times

1. Pick an unsatisfied clause $(L_1 \vee L_2 \vee L_3)$
2. Pick one of its 3 literals
3. Flip the corresponding variable

$$\Pr(\text{from } i \text{ to } 0) = 2^{-i}$$

$$\Pr(\text{random assignment has distance } i) = \binom{n}{i} 2^{-n}$$

Repeat a bunch of times

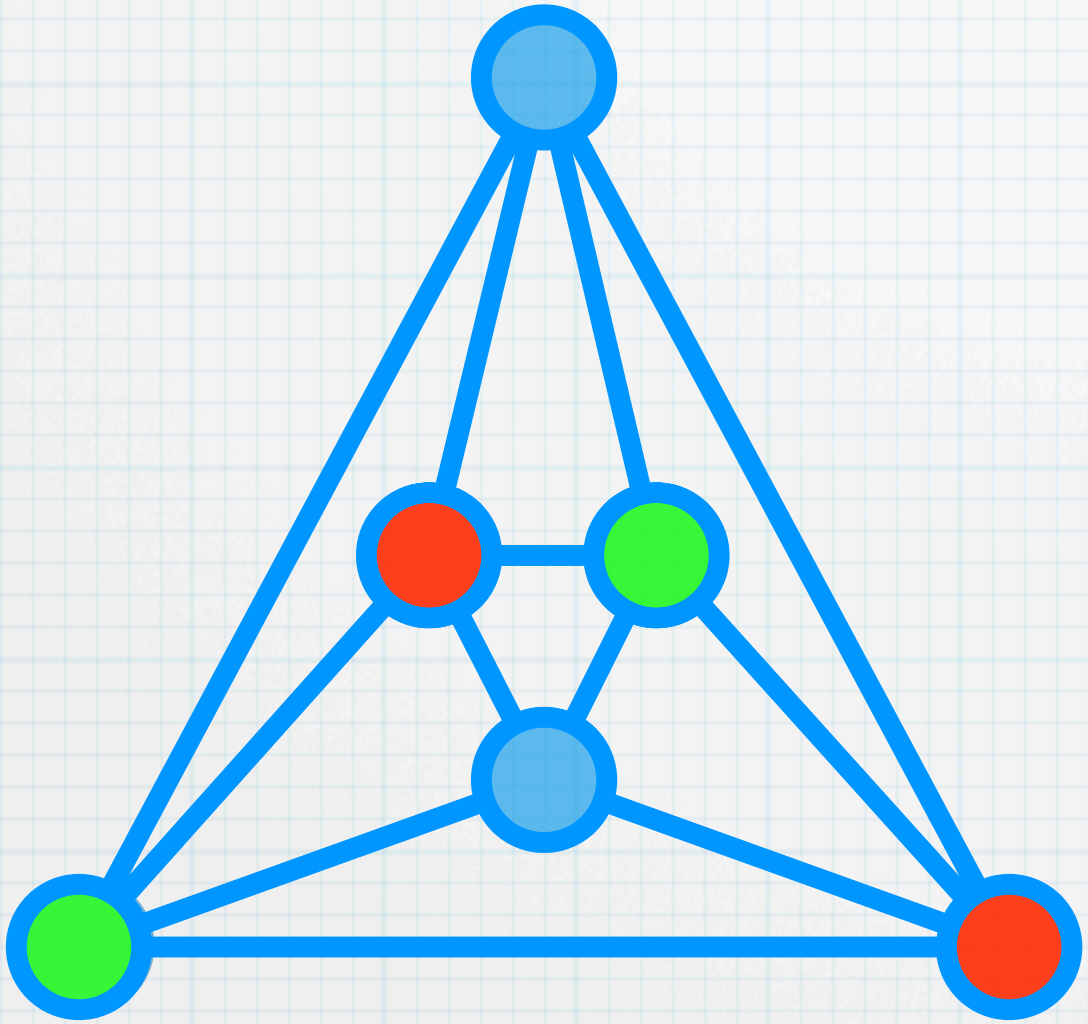
- Pick a random assignment
- Repeat $3n$ times
  1. Pick an unsatisfied clause $(L_1 \vee L_2 \vee L_3)$
  2. Pick one of its 3 literals
  3. Flip the corresponding variable

$$\Pr(\text{from } i \text{ to } 0) = 2^{-i}$$

$$\Pr(\text{random assignment has distance } i) = \binom{n}{i} 2^{-n}$$

$$\Pr(\text{success}) = \sum_{i=0}^{n} \binom{n}{i} 2^{-n} 2^{-i} = \cdots = \left(\frac{3}{4}\right)^n$$

Exercise: Graph colouring



See the Perfect Talk

Time–Space Tradeoffs

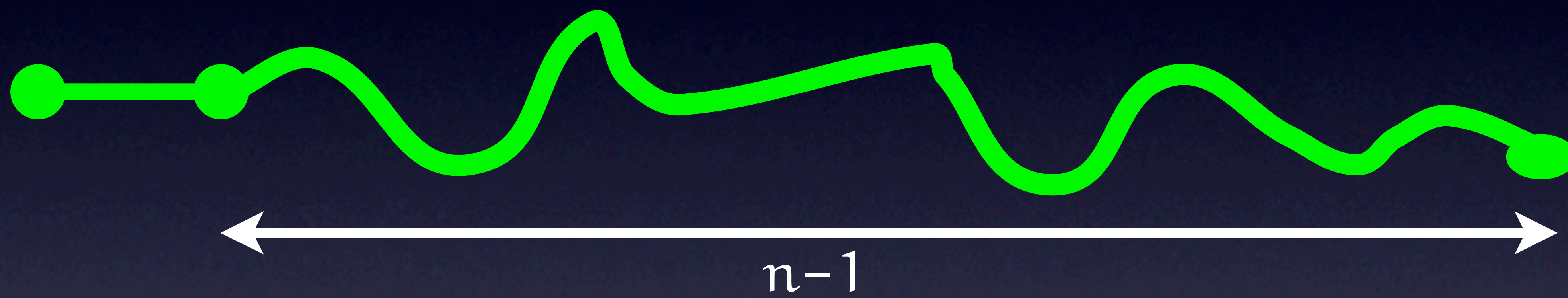
Time–Space Tradeoffs

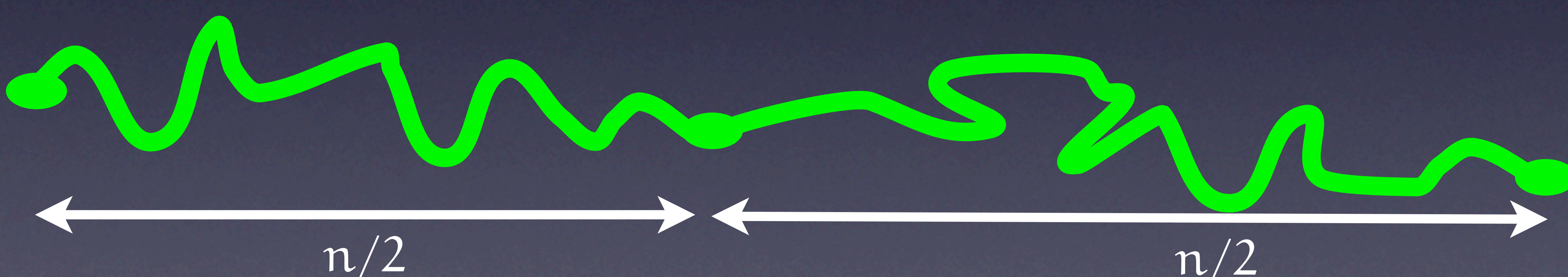
- Dynamic programming
  - over the subsets
  - over a tree-decomposition
- Meet in the middle

Meet in the middle

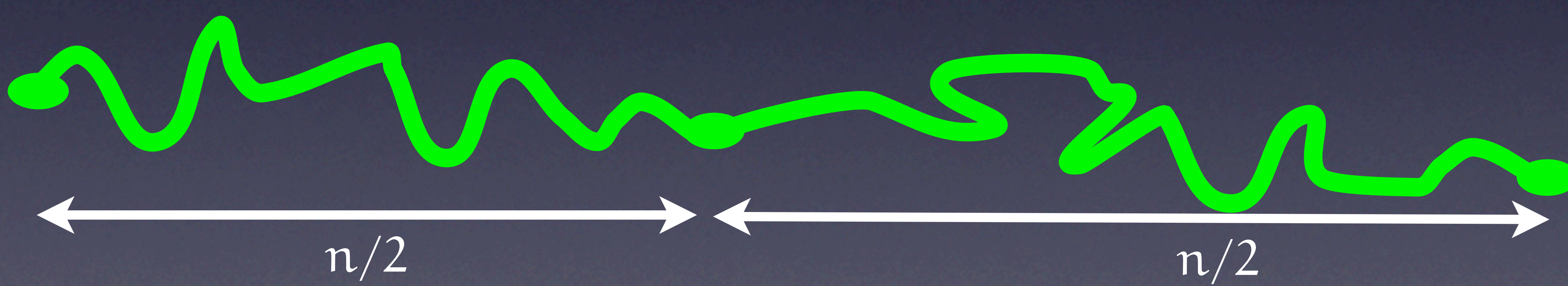
TSP, degree 4

$$\mathrm{OPT}(T, v) = \min_{u \in T \setminus \{v\}} \mathrm{OPT}(T \setminus \{v\}, u) + w(u, v)$$





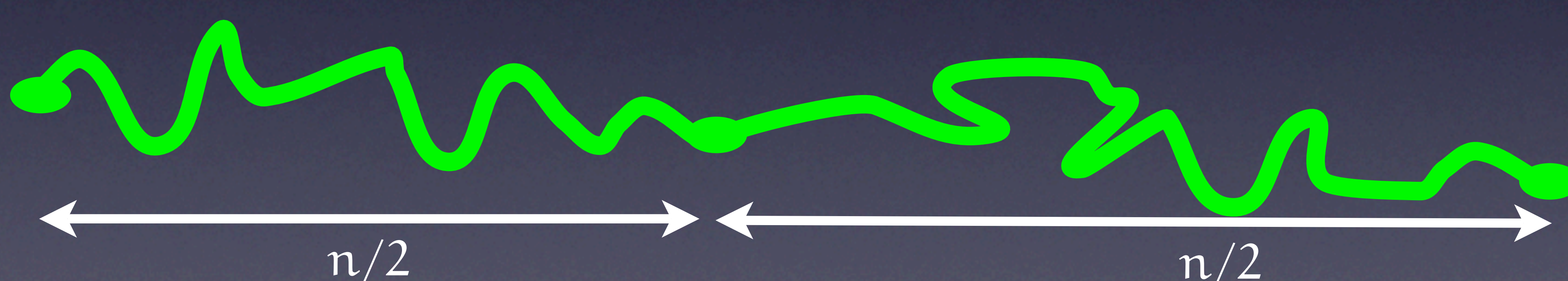
$$\mathrm{OPT}(U, s, t) = \min_{m, S, T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$



$$\mathrm{OPT}(U, s, t) = \min_{m, S, T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$

1. Compute all OPT(T,m,t), store them

T	m	OPT(T,m,t)
...	...	...
$\{v_4, v_{16}, \ldots.\}$	v_{63}	43673
...	...	...

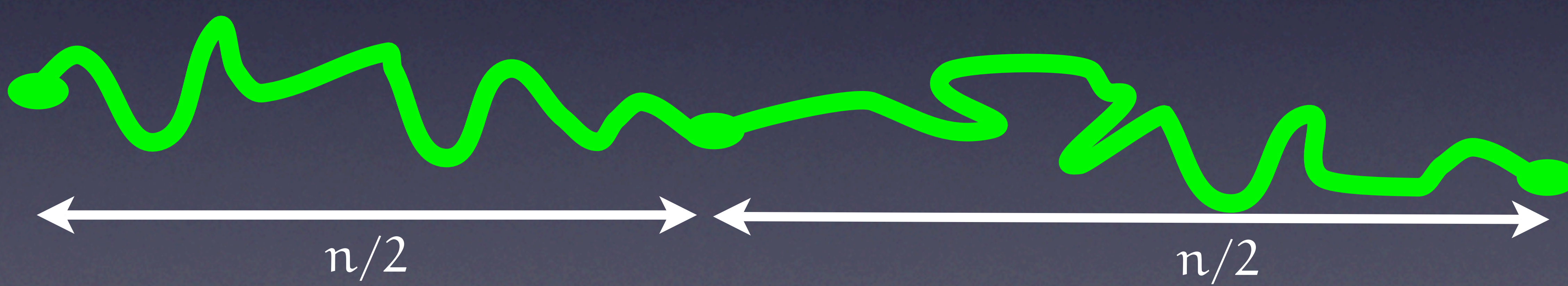


$$\mathrm{OPT}(U, s, t) = \min_{m,S,T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$

2. Compute all $OPT(S,m,s)$, look up corresponding $OPT(V-S,m,t)$

1. Compute all $OPT(T,m,t)$, store them

T	m	$OPT(T,m,t)$
...	...	...
$\{v_4, v_{16}, \ldots.\}$	v_{63}	43673
...	...	...



$$OPT(U,s,t) = \min_{m,S,T} OPT(S,s,m) + OPT(T,m,t)$$

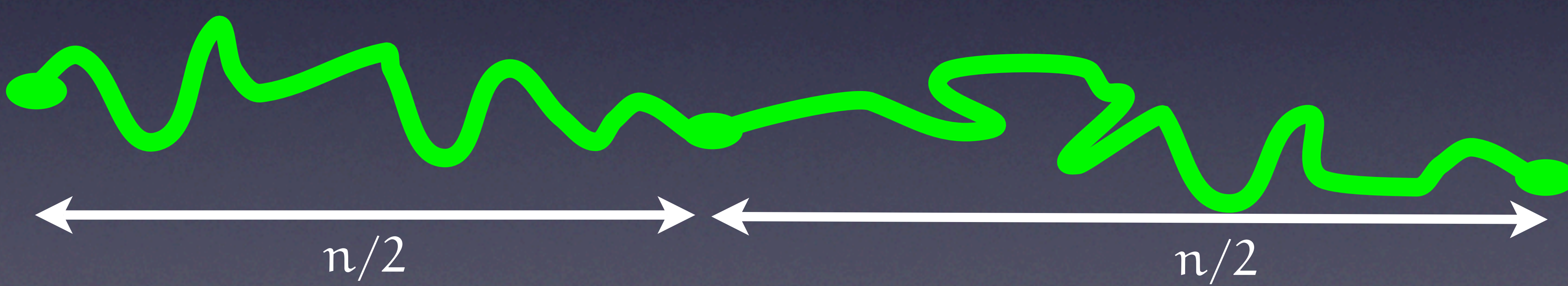
1. Compute all $\mathrm{OPT}(T,m,t)$, store them

T	m	$\mathrm{OPT}(T,m,t)$
…	…	…
$\{v_4, v_{16}, \ldots.\}$	v_{63}	43673
…	…	…

2. Compute all $\mathrm{OPT}(S,m,s)$, look up corresponding $\mathrm{OPT}(V-S,m,t)$

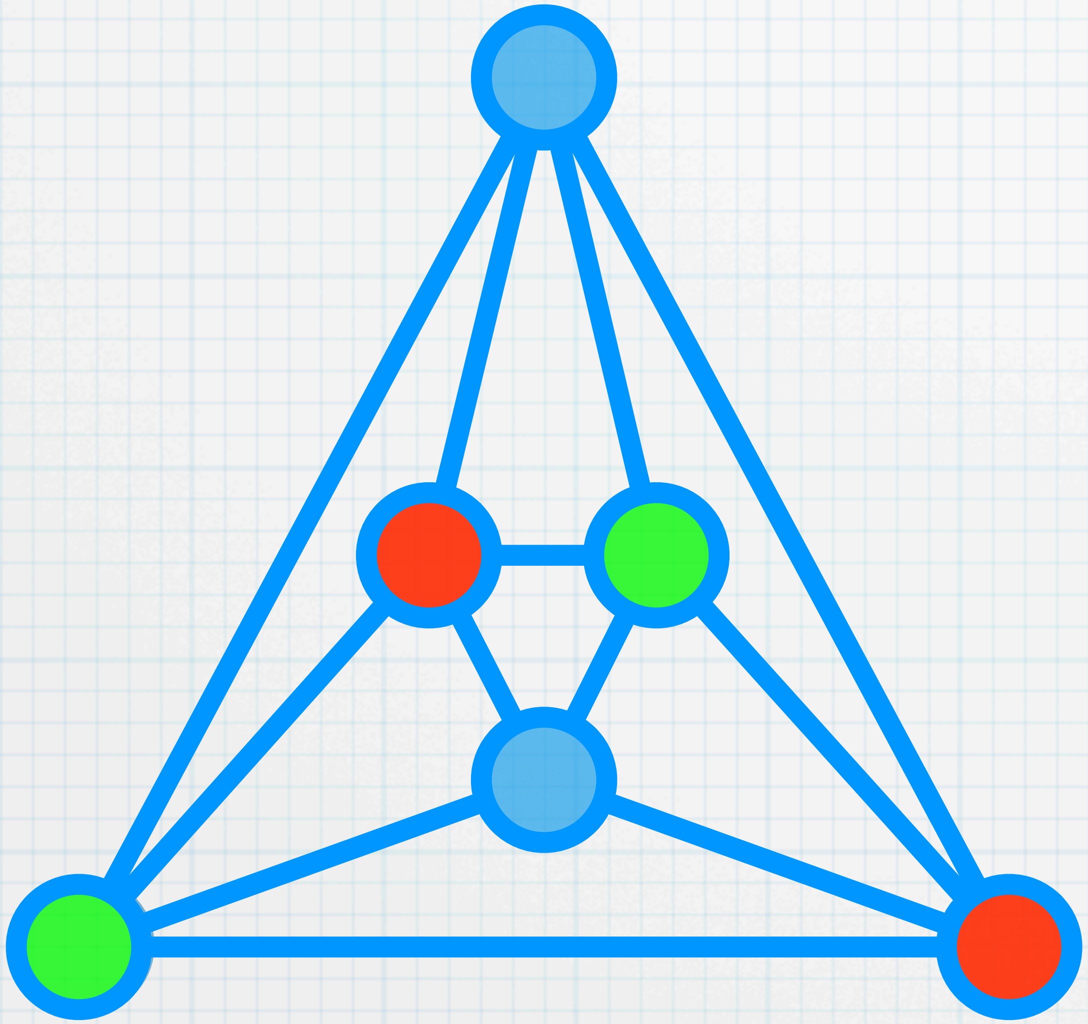
Time $O(3^{n/2})$

Space $O(3^{n/2})$



$$\mathrm{OPT}(U,s,t) = \min_{m,S,T} \mathrm{OPT}(S,s,m) + \mathrm{OPT}(T,m,t)$$

Exercise: Graph colouring

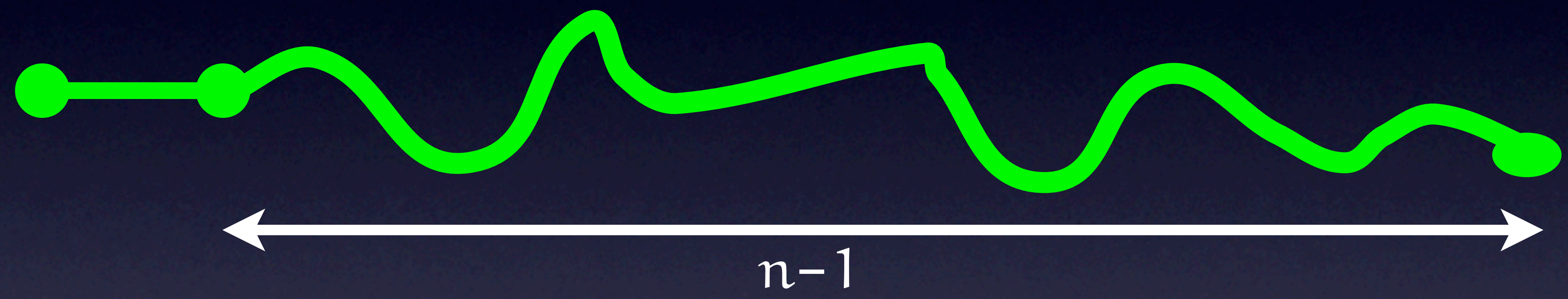


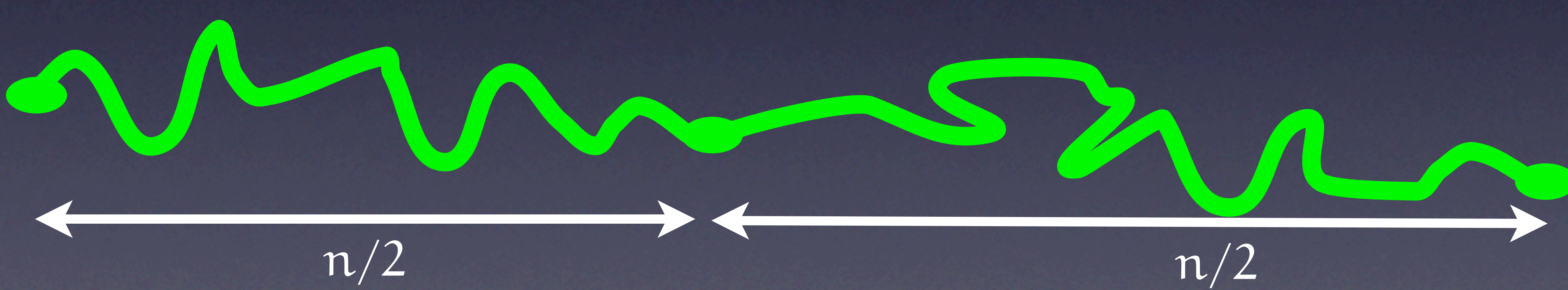
See the Perfect Talk

Dynamic programming over the subsets

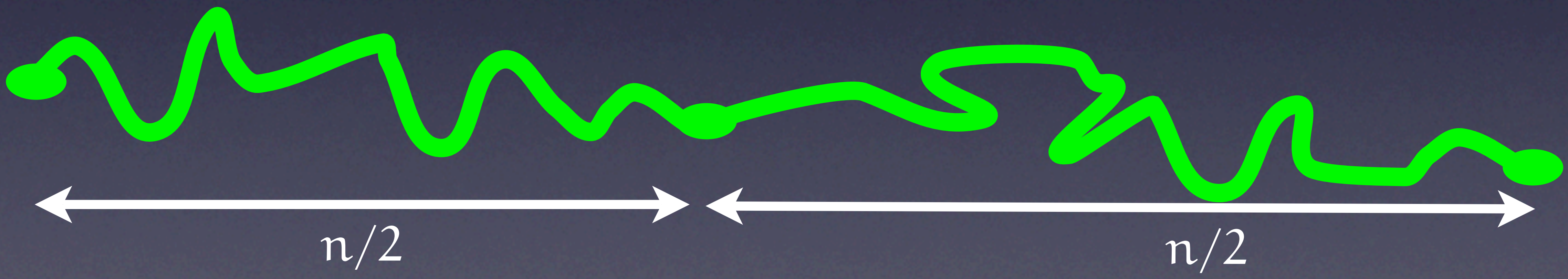
TSP

$$\mathrm{OPT}(T, v) = \min_{u \in T \setminus \{v\}} \mathrm{OPT}(T \setminus \{v\}, u) + w(u, v)$$



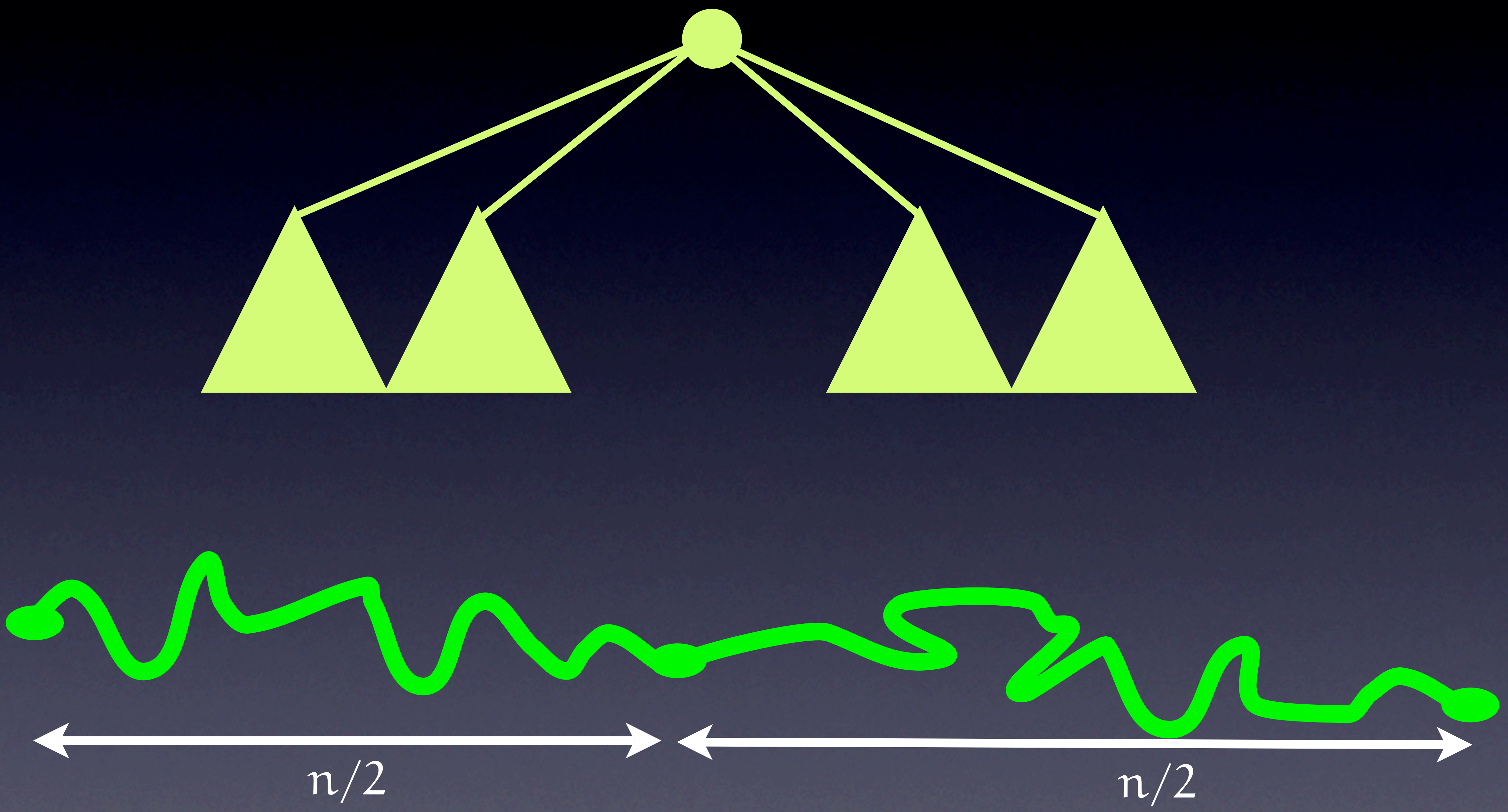


$$\mathrm{OPT}(U, s, t) = \min_{m, S, T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$



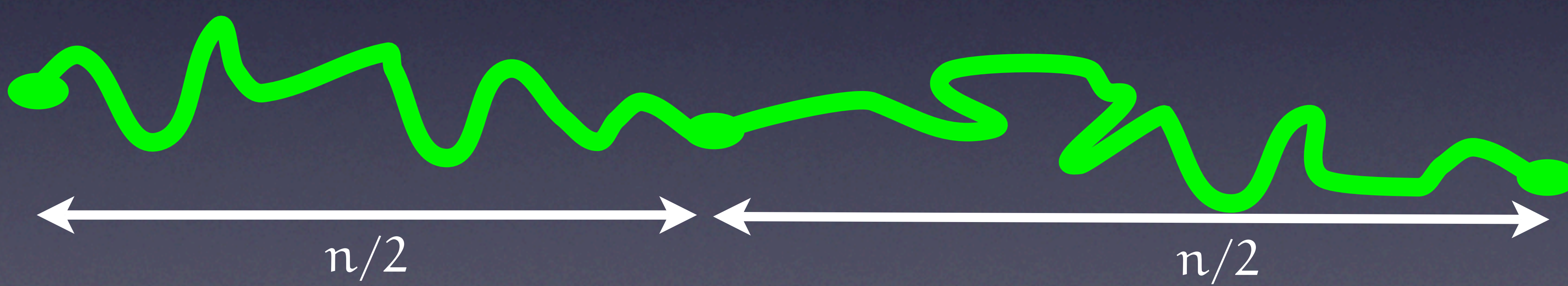
$$\mathrm{OPT}(U, s, t) = \min_{m,S,T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$

Exponential divide and conquer



$$\mathrm{OPT}(U, s, t) = \min_{m, S, T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$

Exponential divide and conquer

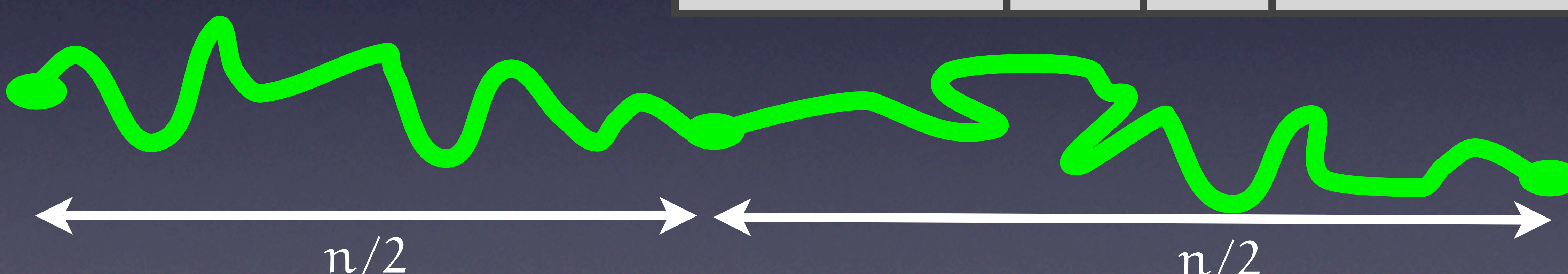


$$\mathrm{OPT}(U, s, t) = \min_{m,S,T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$

Compute all $OPT(X,u,v)$, store them

Exponential divide and conquer

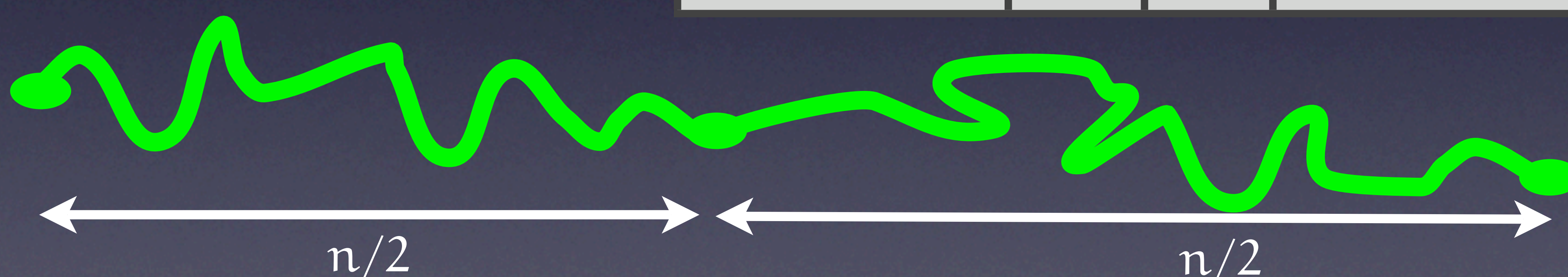
X	u	v	$OPT(X,u,v)$
...	...		...
$\{v_4, v_{16}, \ldots.\}$	v_{63}	v_{23}	43673
...	...		...



$$OPT(U,s,t) = \min_{m,S,T} OPT(S,s,m) + OPT(T,m,t)$$

Compute all $\mathrm{OPT}(X,u,v)$, store them

X	u	v	$\mathrm{OPT}(X,u,v)$
...	...		...
$\{v_4, v_{16}, \ldots.\}$	v_{63}	v_{23}	43673
...	...		...

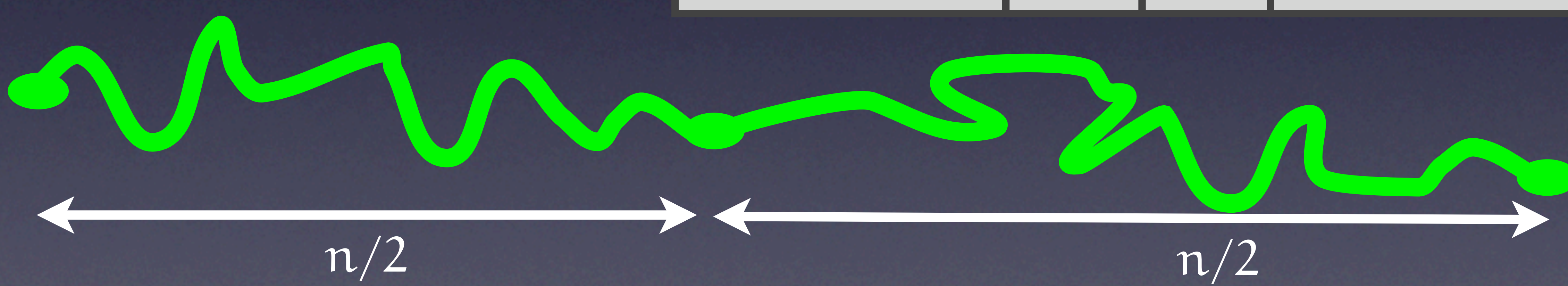


$$\mathrm{OPT}(U, s, t) = \min_{m,S,T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$

Compute all $\mathrm{OPT}(X,u,v)$, store them

2^n entries.
Entry for X
takes $2^{|X|}$ time.

X	u	v	$\mathrm{OPT}(X,u,v)$
...	...		...
$\{v_4, v_{16}, \ldots.\}$	v_{63}	v_{23}	43673
...	...		...



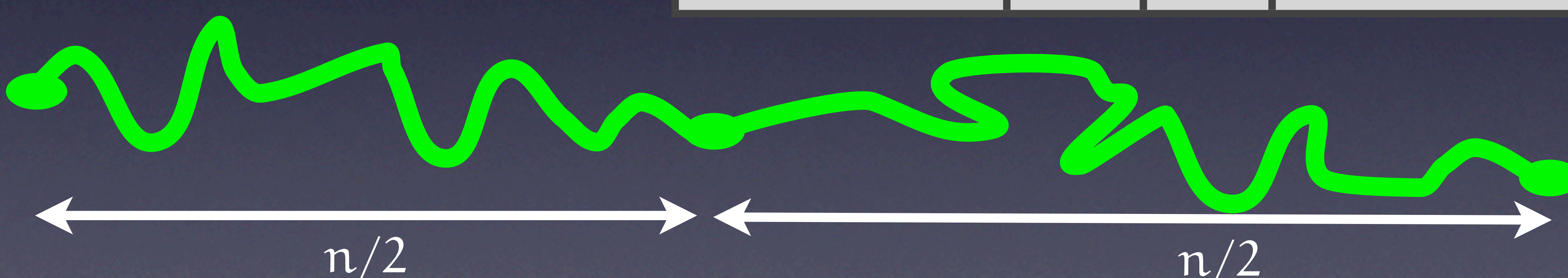
$$\mathrm{OPT}(U, s, t) = \min_{m,S,T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$

Compute all $\mathrm{OPT}(X,u,v)$, store them

2^n entries.
Entry for X takes $2^{|X|}$ time.

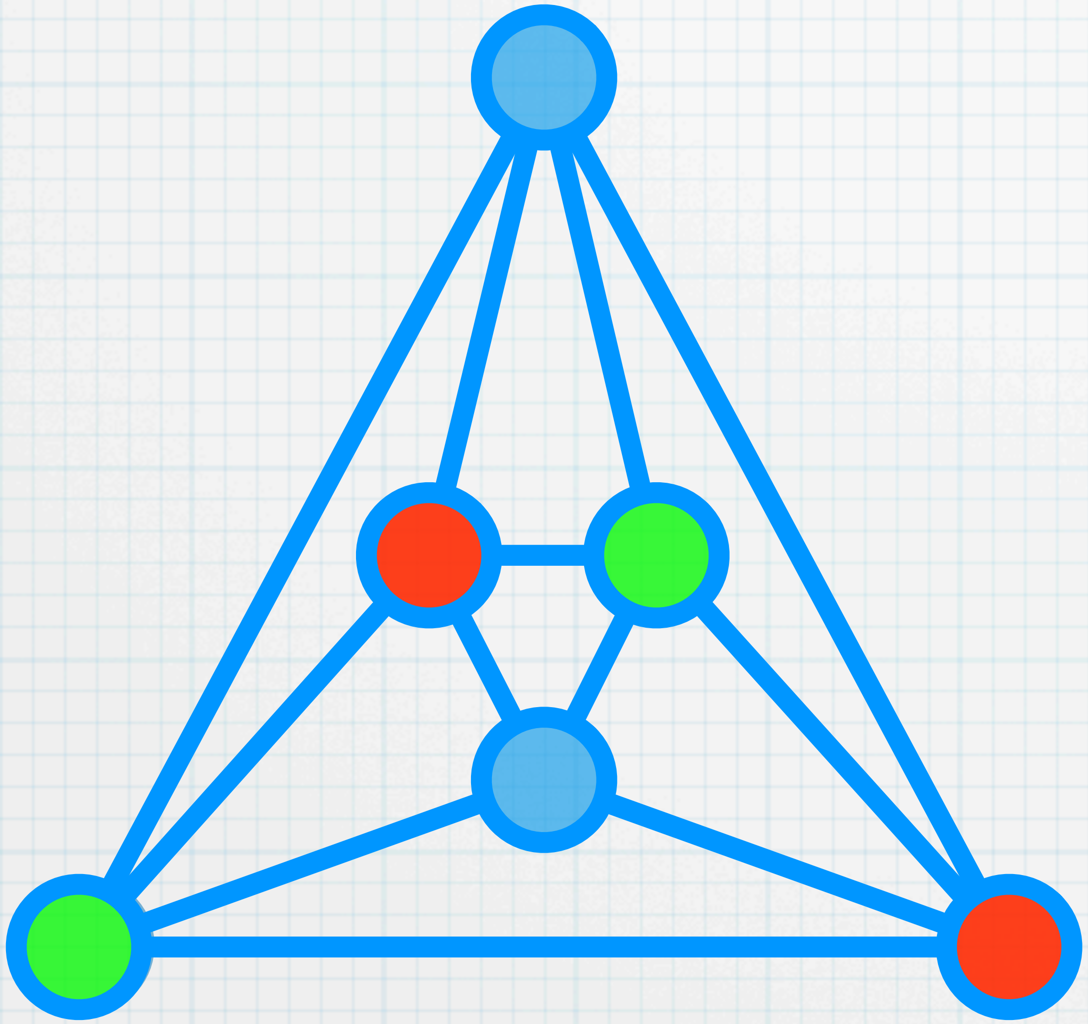
Time $O^*(3^n)$

X	u	v	$\mathrm{OPT}(X,u,v)$
...	...		...
$\{v_4, v_{16}, \ldots.\}$	v_{63}	v_{23}	43673
...	...		...

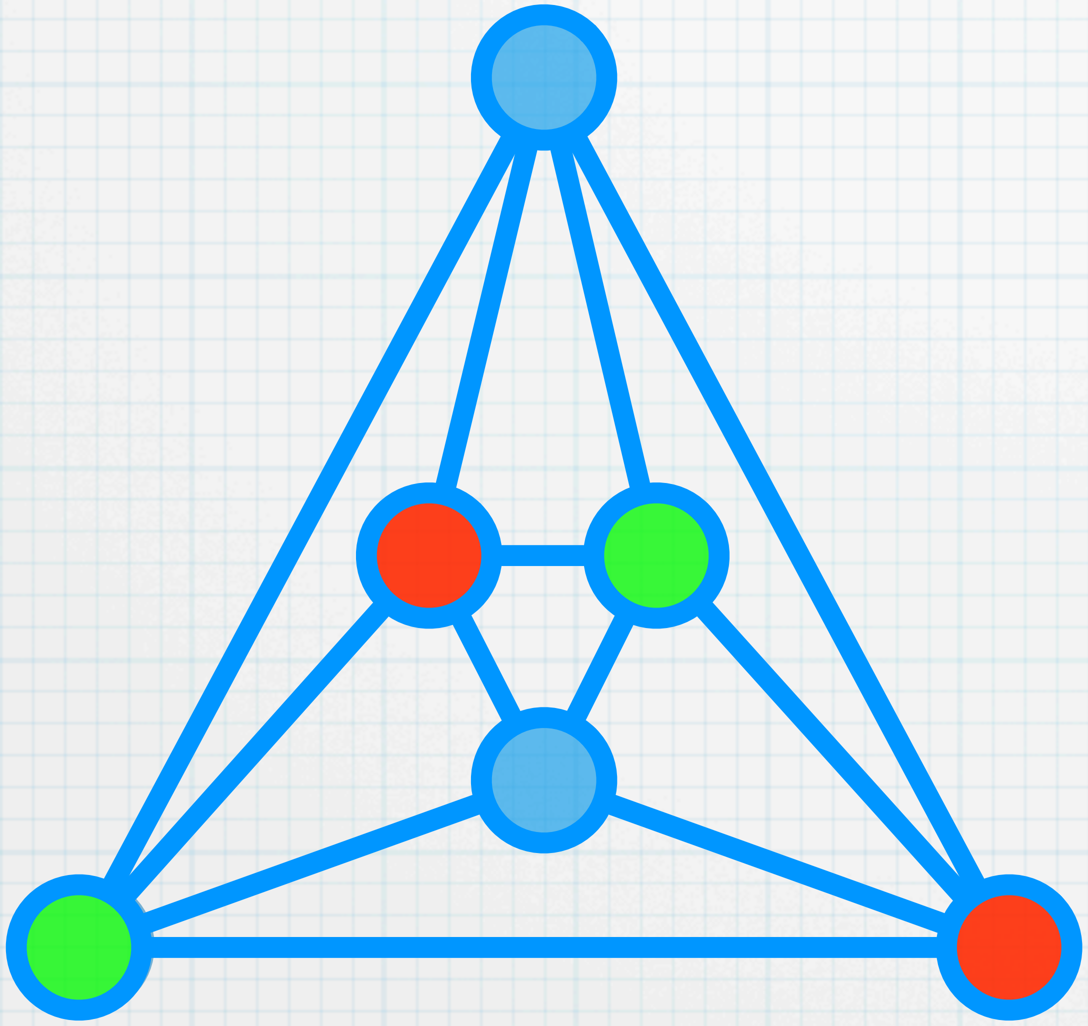


$$\mathrm{OPT}(U,s,t) = \min_{m,S,T} \mathrm{OPT}(S,s,m) + \mathrm{OPT}(T,m,t)$$

Exercise: Graph colouring



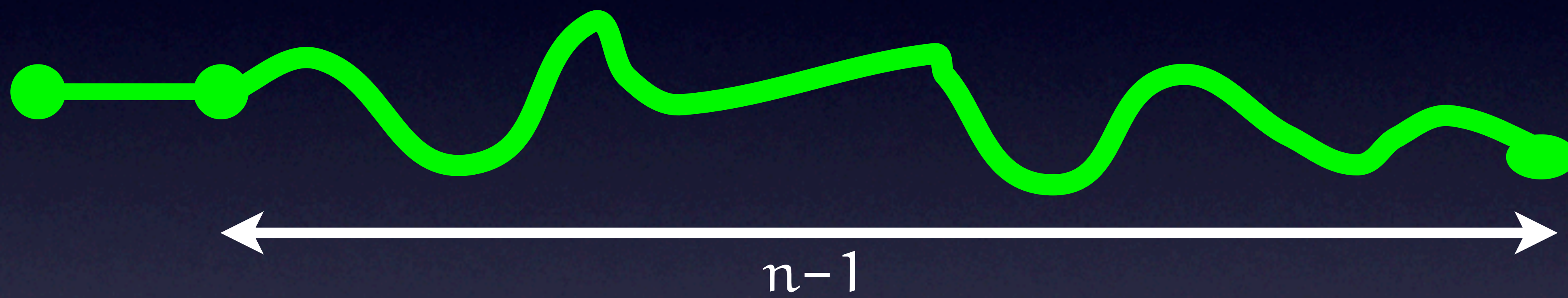
Exercise: Graph colouring

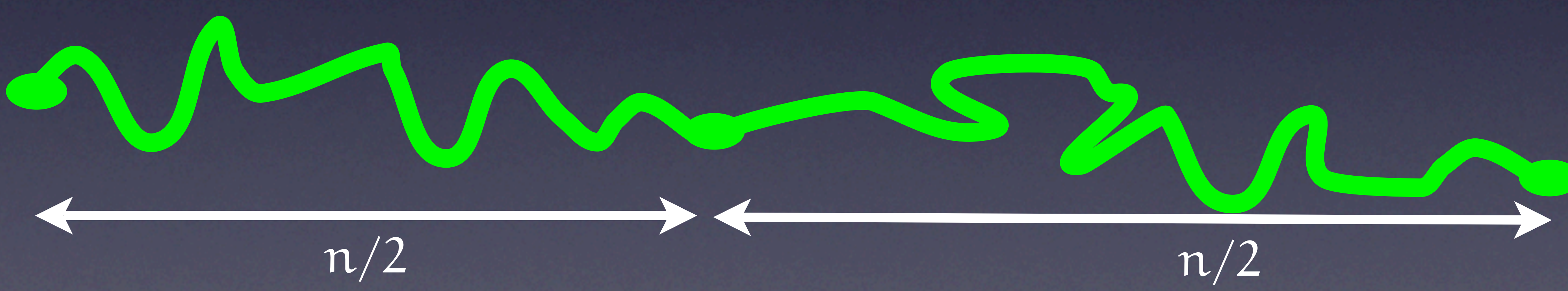


Count the k-colourings in time $O^*(3^n)$

TSP

$$\mathrm{OPT}(T, v) = \min_{u \in T \setminus \{v\}} \mathrm{OPT}(T \setminus \{v\}, u) + w(u, v)$$

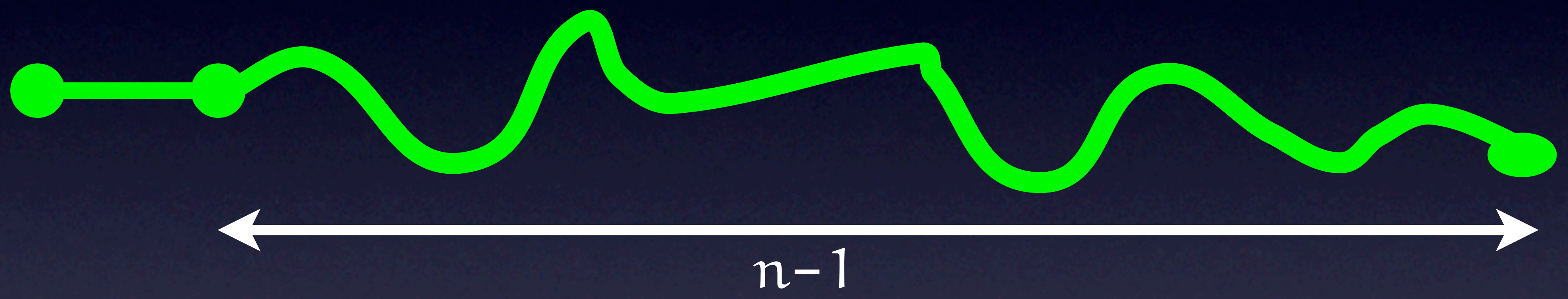




$$\mathrm{OPT}(U, s, t) = \min_{m, S, T} \mathrm{OPT}(S, s, m) + \mathrm{OPT}(T, m, t)$$

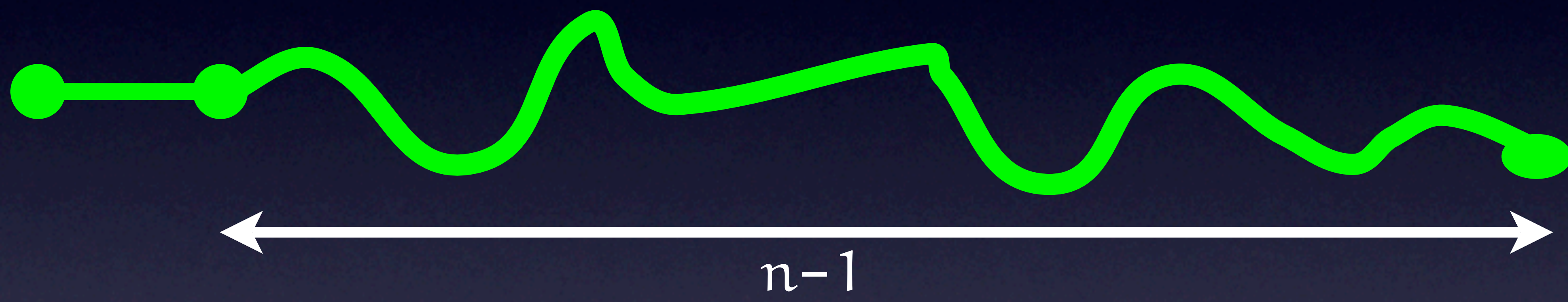
TSP

$$\mathrm{OPT}(T, v) = \min_{u \in T \setminus \{v\}} \mathrm{OPT}(T \setminus \{v\}, u) + w(u, v)$$



TSP

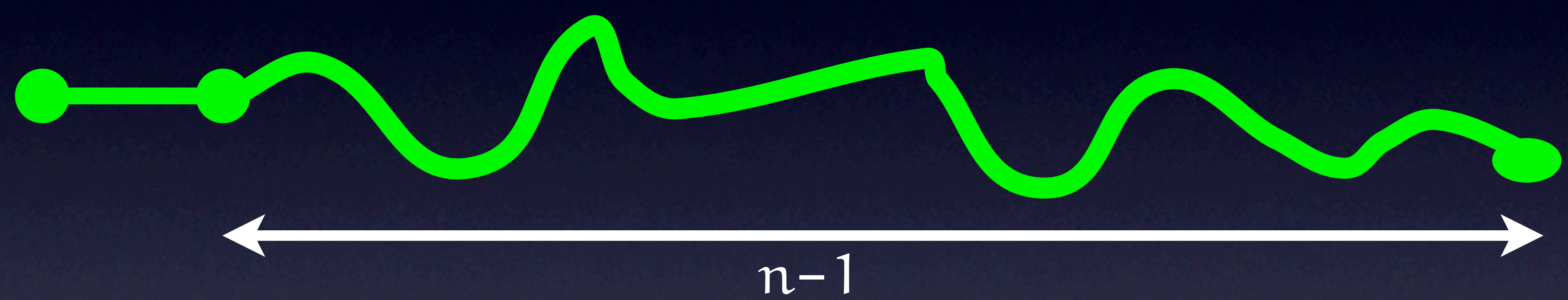
$$\mathrm{OPT}(T, v) = \min_{u \in T \setminus \{v\}} \mathrm{OPT}(T \setminus \{v\}, u) + w(u, v)$$



X	u	OPT(X,u)
...	...	...
$\{v_4, v_{16}, \ldots.\}$	v_{63}	43673
...	...	...

TSP

$$\mathrm{OPT}(T, v) = \min_{u \in T \setminus \{v\}} \mathrm{OPT}(T \setminus \{v\}, u) + w(u, v)$$

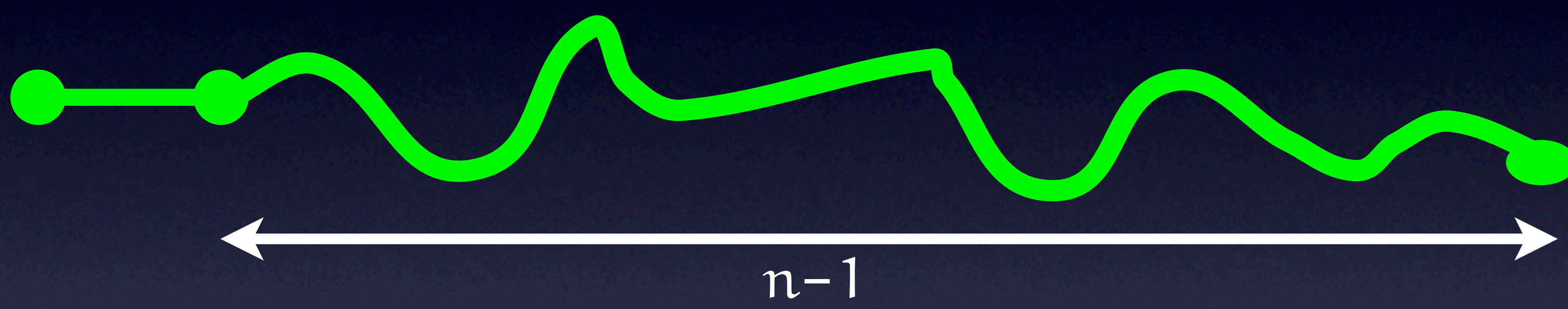


2^n entries.
Entry for X
takes n time.

X	u	OPT(X,u)
...	...	...
$\{v_4, v_{16}, \ldots.\}$	v_{63}	43673
...	...	...

TSP

$$\mathrm{OPT}(T, v) = \min_{u \in T \setminus \{v\}} \mathrm{OPT}(T \setminus \{v\}, u) + w(u, v)$$

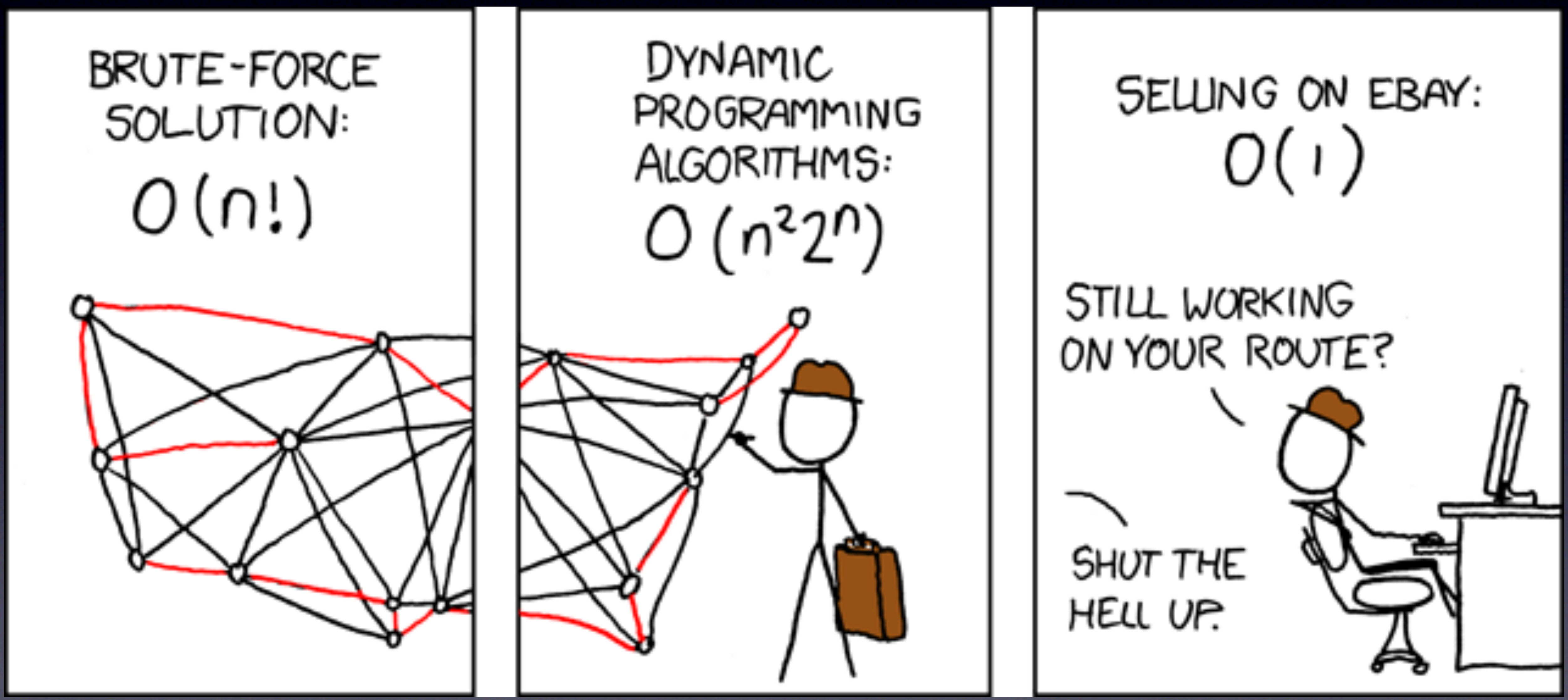


2^n entries.
Entry for X
takes n time.

Time $O^*(2^n)$

X	u	OPT(X,u)
...	...	...
$\{v_4, v_{16}, \ldots.\}$	v_{63}	43673
...	...	...

Part of popular geek culture



[xkcd #399]

	TSP/HC	3-Sat	Independent set	#perfect matchings	colouring

	TSP/HC	3-Sat	Independent set	#perfect matchings	colouring
Brute force	$n!$	2^n	2^n	2^m, $n!$	k^n

	TSP/HC	3-Sat	Independent set	#perfect matchings	colouring
Brute force	$n!$	2^n	2^n	2^m, $n!$	k^n
Greedy					$n!$

	TSP/HC	3-Sat	Independent set	#perfect matchings	colouring
Brute force	$n!$	2^n	2^n	2^m, $n!$	k^n
Greedy					$n!$
Decrease and conquer	$n!$	1.83^n	1.39^n		1.62^{n+m}

	TSP/HC	3-Sat	Independent set	#perfect matchings	colouring
Brute force	$n!$	2^n	2^n	2^m, $n!$	k^n
Greedy					$n!$
Decrease and conquer	$n!$	1.83^n	1.39^n		1.62^{n+m}
Divide and conquer	4^n				9^n

	TSP/HC	3-Sat	Independent set	#perfect matchings	colouring
Brute force	$n!$	2^n	2^n	2^m, $n!$	k^n
Greedy					$n!$
Decrease and conquer	$n!$	1.83^n	1.39^n		1.62^{n+m}
Divide and conquer	4^n				9^n
Triangle counting			$2^{2.38k/3}$		1.73^n

	TSP/HC	3-Sat	Independent set	#perfect matchings	colouring
Brute force	$n!$	2^n	2^n	2^m, $n!$	k^n
Greedy					$n!$
Decrease and conquer	$n!$	1.83^n	1.39^n		1.62^{n+m}
Divide and conquer	4^n				9^n
Triangle counting			$2^{2.38k/3}$		1.73^n
Moebius transformation	2^n			1.41^n, 1.73^n	3^n, 2^n

	TSP/HC	3-Sat	Independent set	#perfect matchings	colouring
Brute force	$n!$	2^n	2^n	2^m, $n!$	k^n
Greedy					$n!$
Decrease and conquer	$n!$	1.83^n	1.39^n		1.62^{n+m}
Divide and conquer	4^n				9^n
Triangle counting			$2^{2.38k/3}$		1.73^n
Moebius transformation	2^n			1.41^n, 1.73^n	3^n, 2^n
Local search		$(4/3)^n$			

	TSP/HC	3-Sat	Independent set	#perfect matchings	colouring
Brute force	$n!$	2^n	2^n	2^m, $n!$	k^n
Greedy					$n!$
Decrease and conquer	$n!$	1.83^n	1.39^n		1.62^{n+m}
Divide and conquer	4^n				9^n
Triangle counting			$2^{2.38k/3}$		1.73^n
Moebius transformation	2^n			1.41^n, 1.73^n	3^n, 2^n
Local search		$(4/3)^n$			
Meet in the middle	$3^{n/2}$				

	TSP/HC	3-Sat	Independent set	#perfect matchings	colouring
Brute force	$n!$	2^n	2^n	2^m, $n!$	k^n
Greedy					$n!$
Decrease and conquer	$n!$	1.83^n	1.39^n		1.62^{n+m}
Divide and conquer	4^n				9^n
Triangle counting			$2^{2.38k/3}$		1.73^n
Moebius transformation	2^n			1.41^n, 1.73^n	3^n, 2^n
Local search		$(4/3)^n$			
Meet in the middle	$3^{n/2}$				
Dynamic programming	2^n				3^n