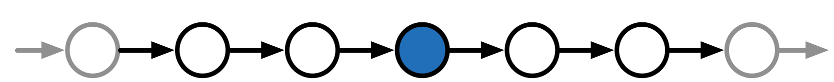
Exact bounds for distributed graph colouring

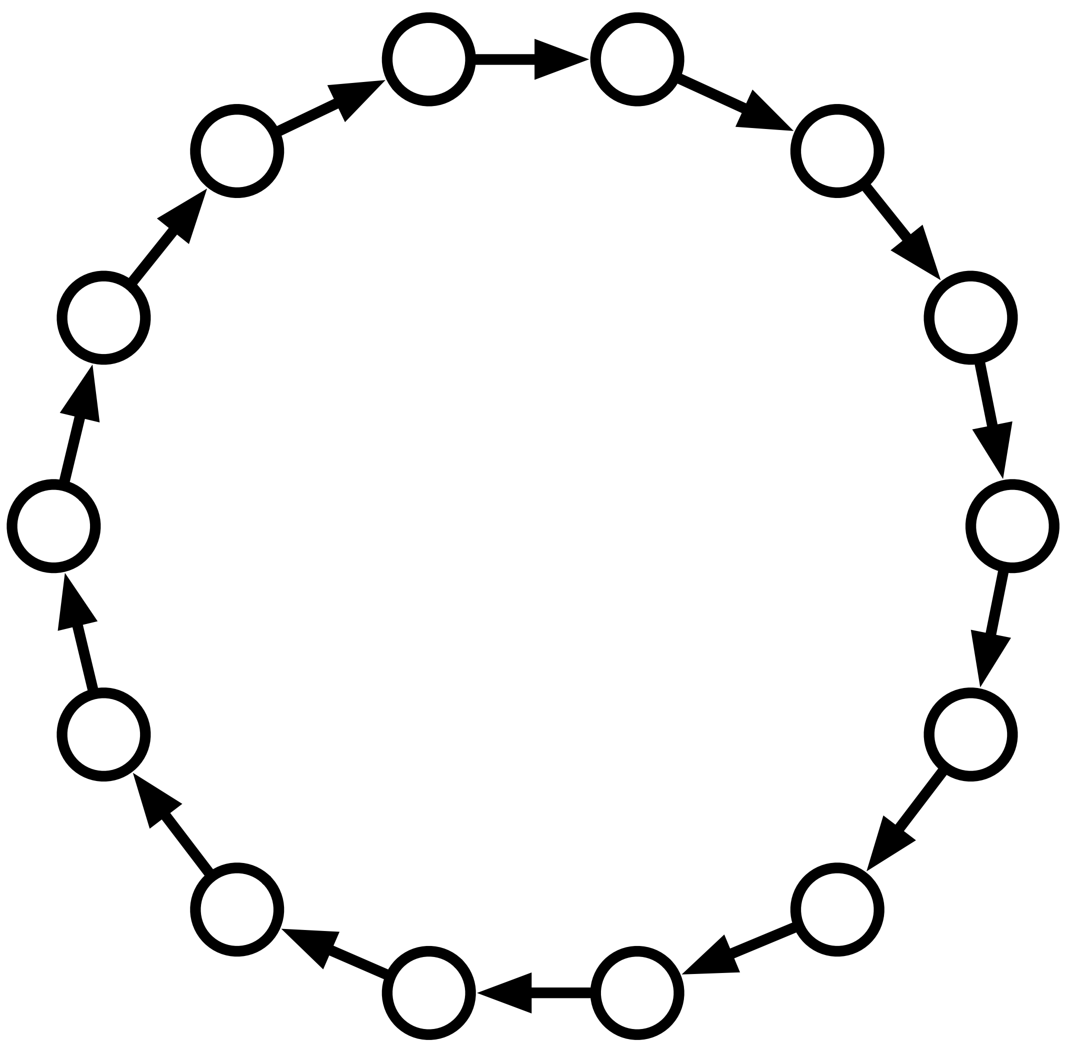
Joel Rybicki
Max Planck Institute for Informatics

Jukka Suomela
Helsinki Institute for Information Technology & Aalto University



SIROCCO 2015
July15, 2015

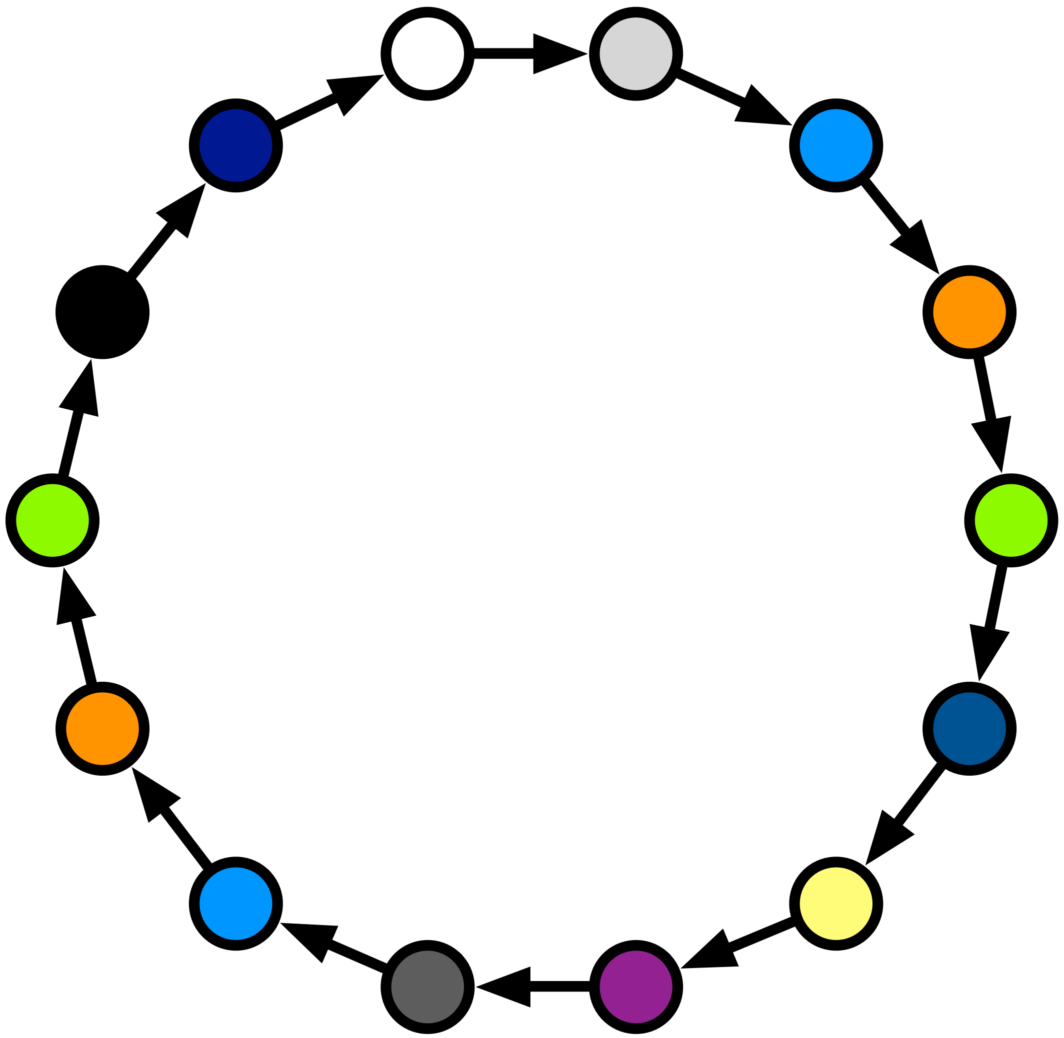
Graph colouring



$G = (V, E)$

Input: A cycle with a consistent orientation

Graph colouring



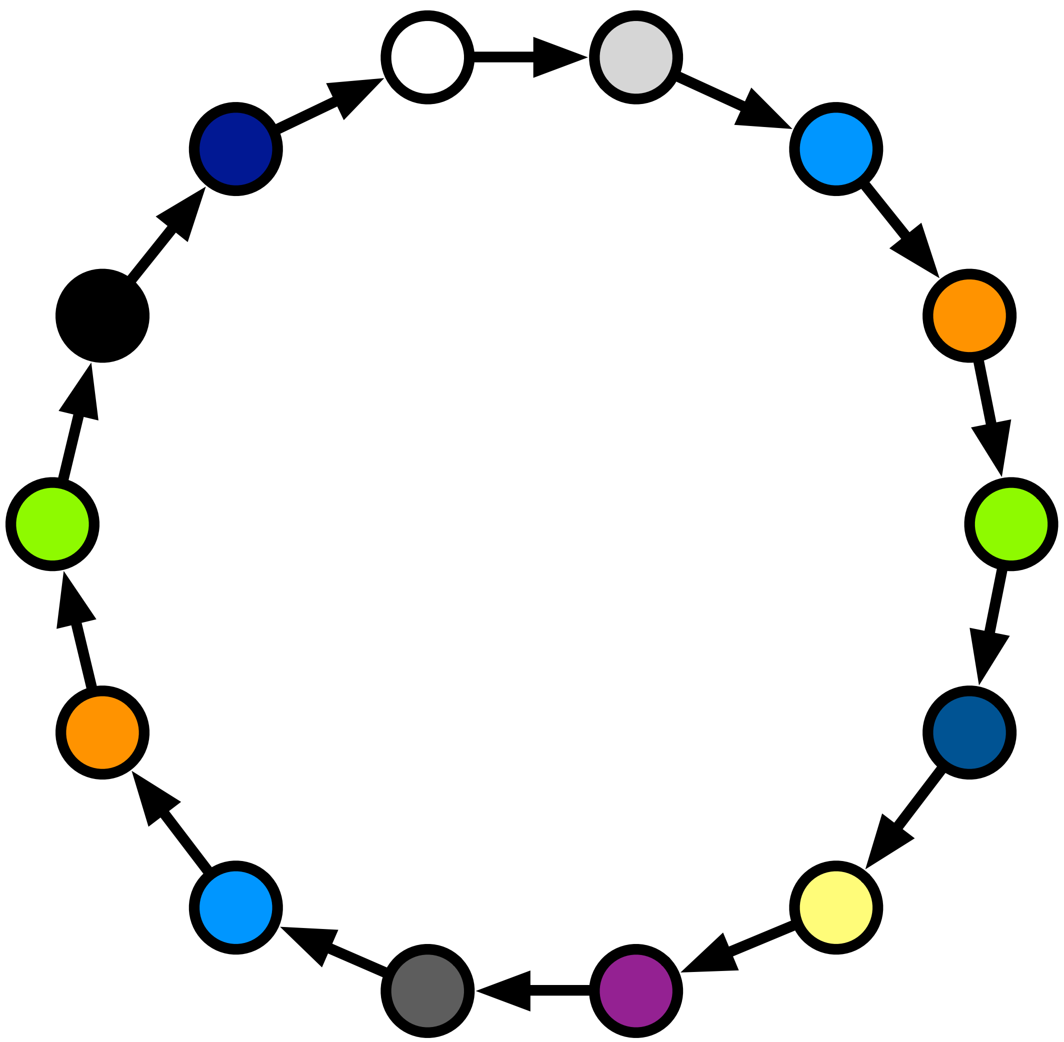
Input: A cycle with a consistent orientation

Given a colouring

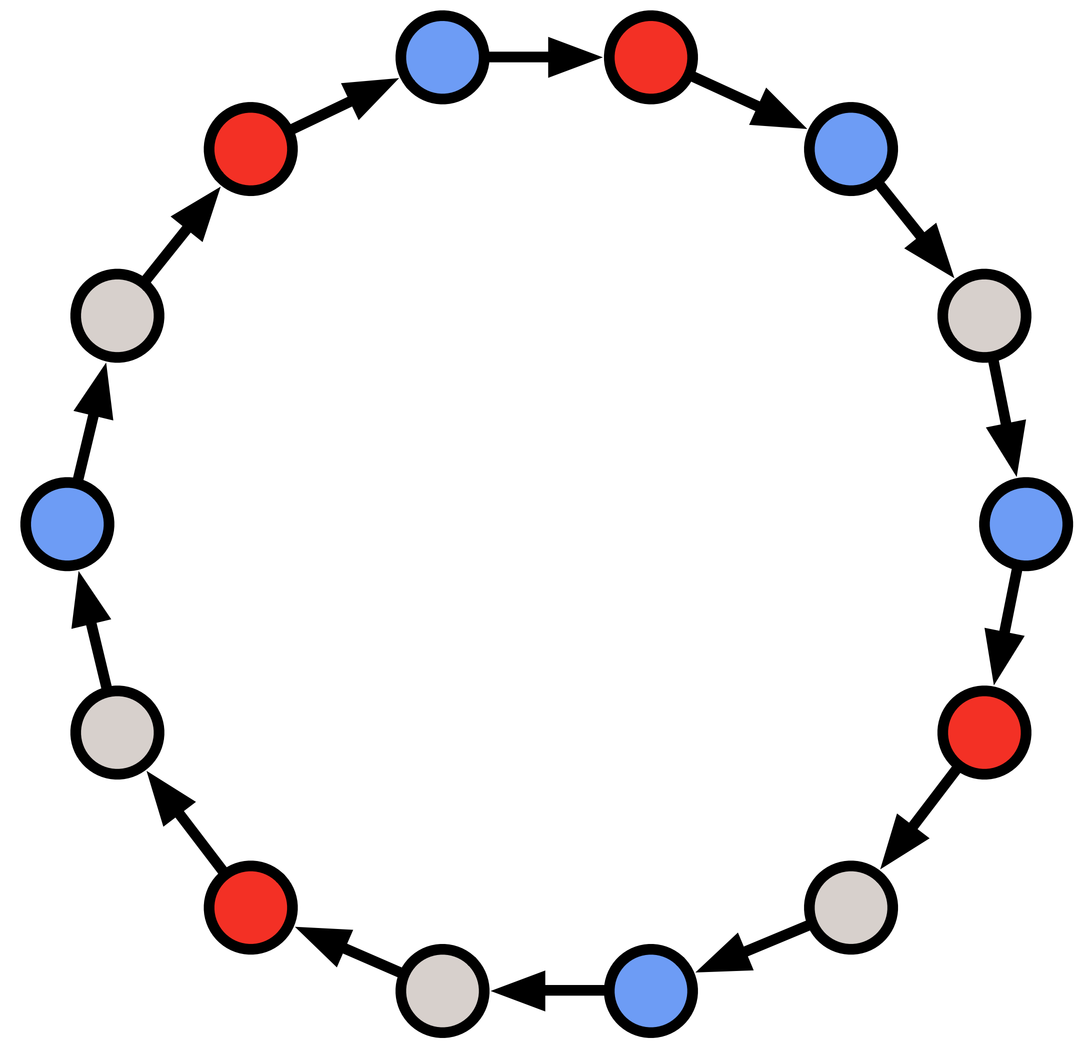
$f : V \to \{1, \ldots, n\}$

$G = (V, E)$ $\{u, v\} \in E \Rightarrow f(u) \neq f(v)$

Task: Colour reduction



Input:
n-colouring



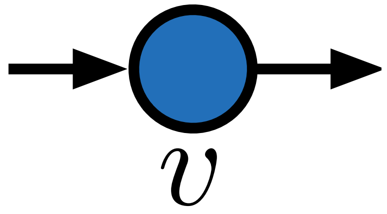
Output:
3-colouring

Model of computing

Synchronous rounds. Each node

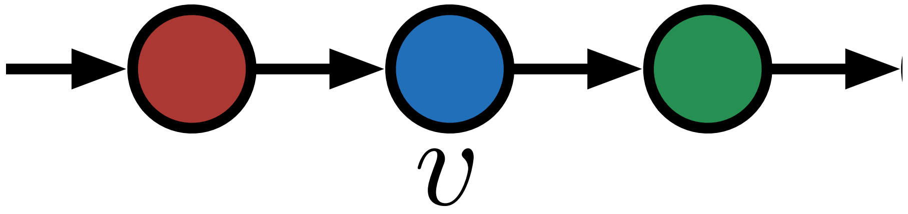
1. sends messages
2. receives messages
3. updates local state

Local views



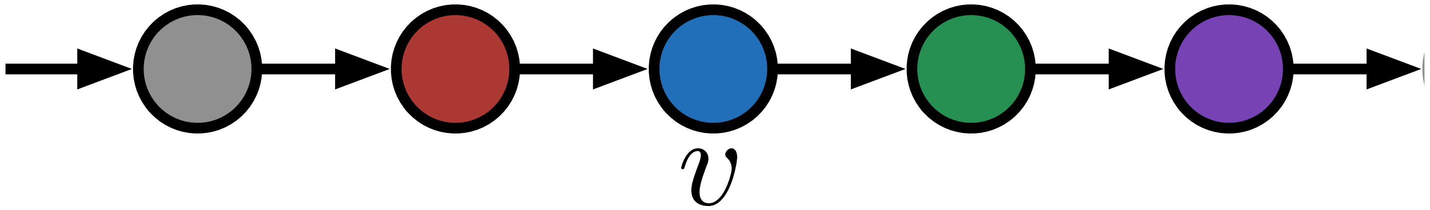
0 rounds

Local views



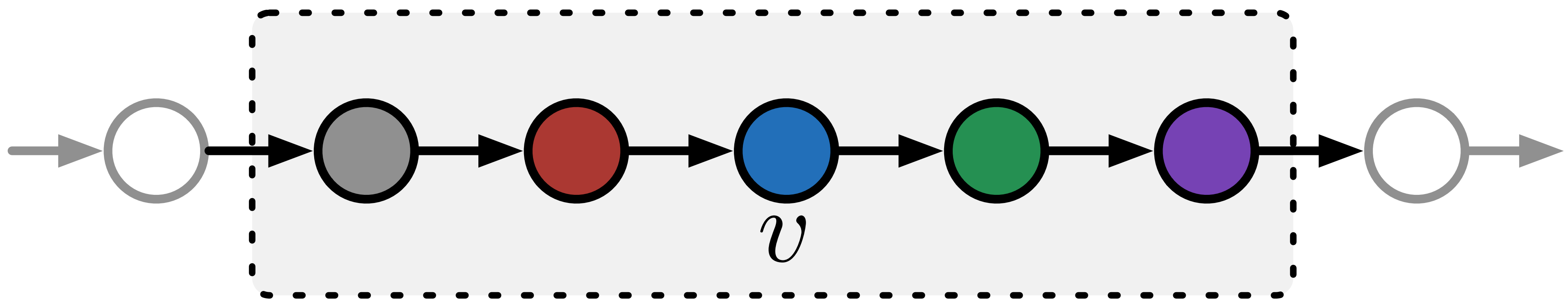
1 round

Local views



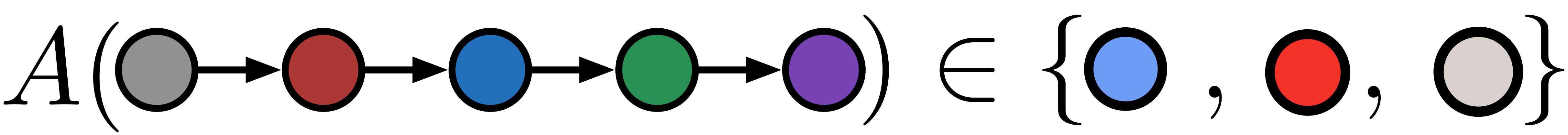
2 rounds

Local views



r rounds

An **algorithm** is a map



Time complexity

$$C(n, 3)$$

is the **exact** number of rounds it takes to 3-colour **any** *n*-coloured directed cycle

Prior work

Complexity of 3-colouring

$$\frac{1}{2}\log^* n - 1 \leq C(n,3)$$

Linial (1992)

$$\log^* n = \min\{i : \overbrace{\log \cdots \log}^{i} n \leq 1\}$$

Prior work

Complexity of 3-colouring

$$C(n,3) \le \frac{1}{2}\log^* n + 3$$

Cole & Vishkin (1987)

$$\log^* n = \min\{i : \overbrace{\log \cdots \log}^{i} n \le 1\}$$

Prior work

Complexity of 3-colouring

$$\frac{1}{2}\log^* n - 1 \le C(n,3) \le \frac{1}{2}\log^* n + 3$$

Cole & Vishkin (1987)

Linial (1992)

$$\log^* n = \min\{i : \overbrace{\log \cdots \log}^{i} n \le 1\}$$

Prior work

Complexity of 3-colouring

$$C(n,3) = \frac{1}{2}\log^* n + O(1)$$

In "practice", the additive term dominates:

$$\log^* 10^{19728} = 5$$

Our result

For infinitely many values of n,
3-colouring requires *exactly*

$$\frac{1}{2}\log^* n \quad \text{rounds.}$$

The approach

Lower bound: Tighten Linial's bound using new *computational techniques*

Upper bound: A careful analysis of Naor–Stockmeyer (1995) colour reduction

The lower bound

Step 1. Bound the complexity of finding a 16-colouring

Step 2. Show that a fast 3-colouring algorithm implies a fast 16-colouring algorithm

The lower bound

Step 1. Bound the complexity of finding a 16-colouring

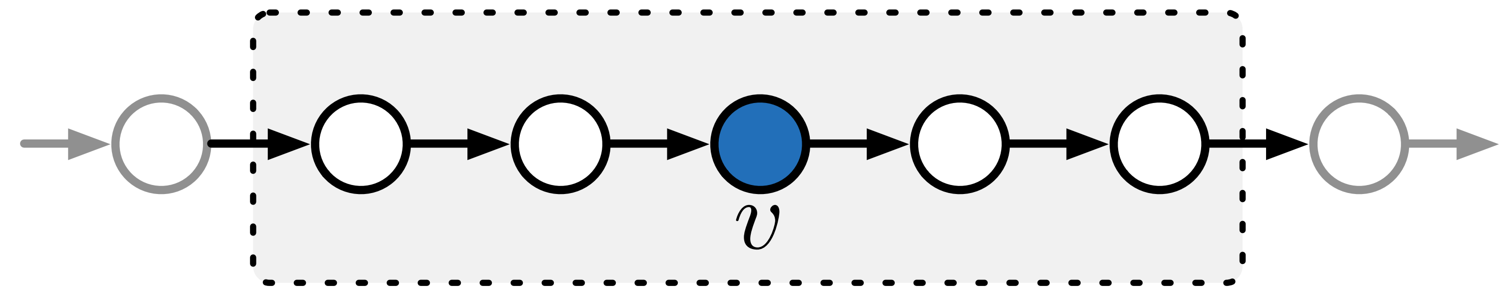
“Dependence on *n*”

Step 2. Show that a fast 3-colouring algorithm implies a fast 16-colouring algorithm

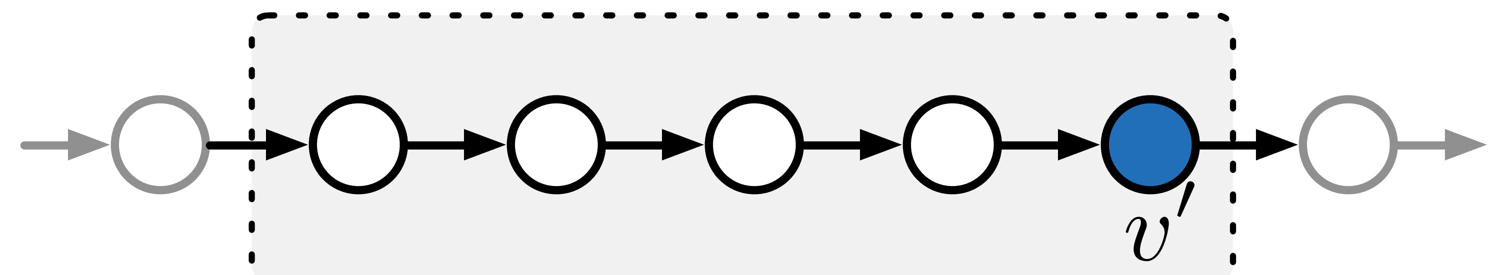
“The additive O(1) term”

Two-sided ≈ one-sided

Two-sided view
$C(n,3)$

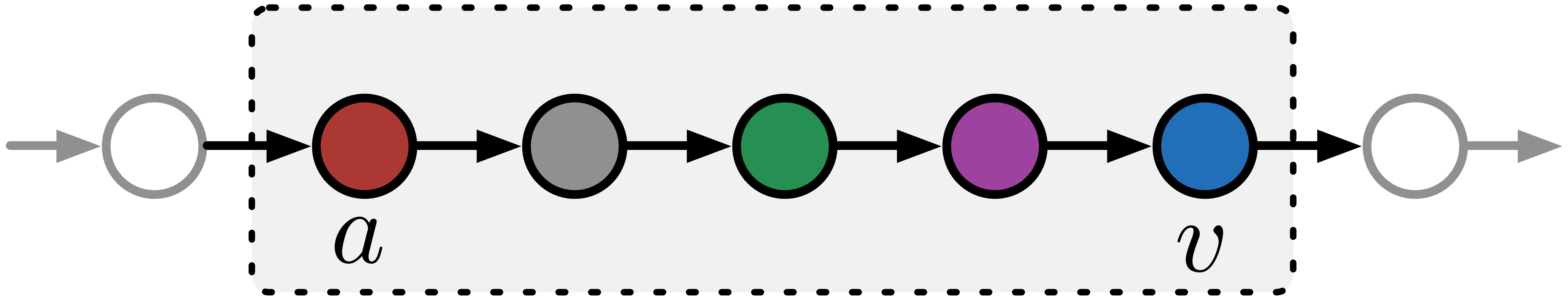


One-sided view
$T(n,3)$



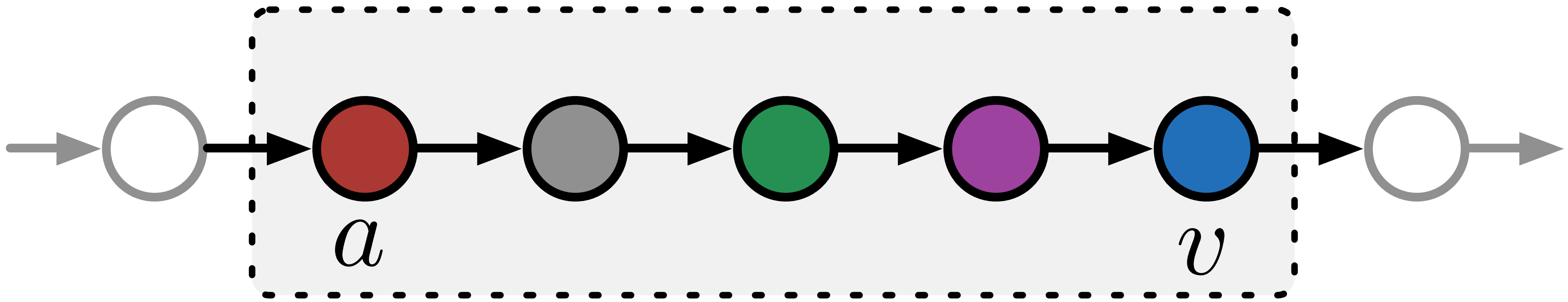
$$C(n,3) = \lceil T(n,3)/2 \rceil$$

The speed-up lemma



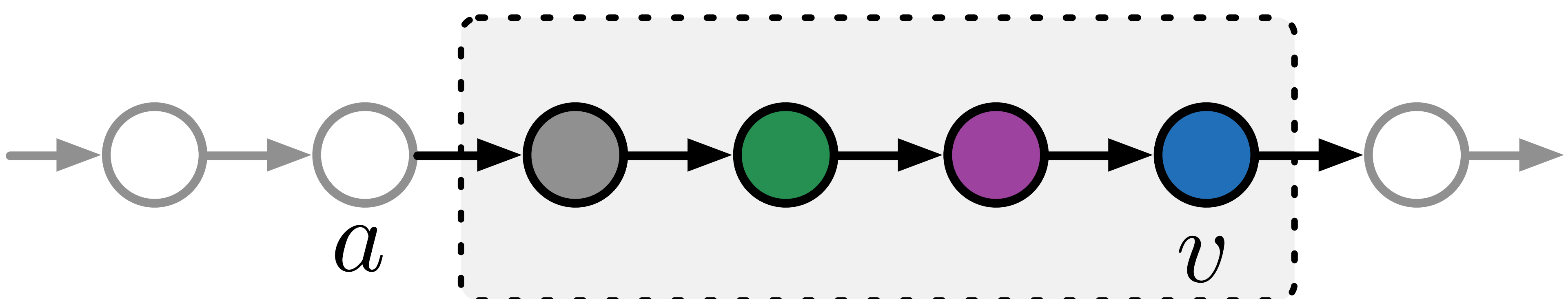
c-colouring in r rounds

The speed-up lemma



c-colouring in r rounds

$\Rightarrow$



$(2^c - 2)$-colouring in $r - 1$ rounds

New technique: Successor Graphs

Fix any (e.g. optimal) algorithm

New technique: Successor Graphs

Fix any (e.g. optimal) algorithm and apply the speed-up lemma to get

	A_0
#colours	3
#rounds	t

New technique: Successor Graphs

Fix any (e.g. optimal) algorithm and apply the speed-up lemma to get

	A_0	A_1
#colours	3	$2^3 - 2$
#rounds	t	$t - 1$

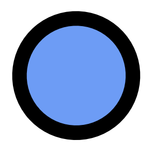
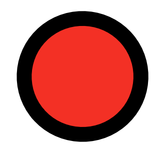
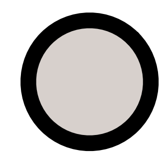
New technique: Successor Graphs

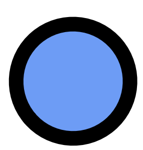
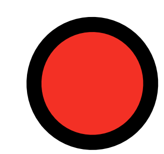
Fix any (e.g. optimal) algorithm and apply the speed-up lemma to get

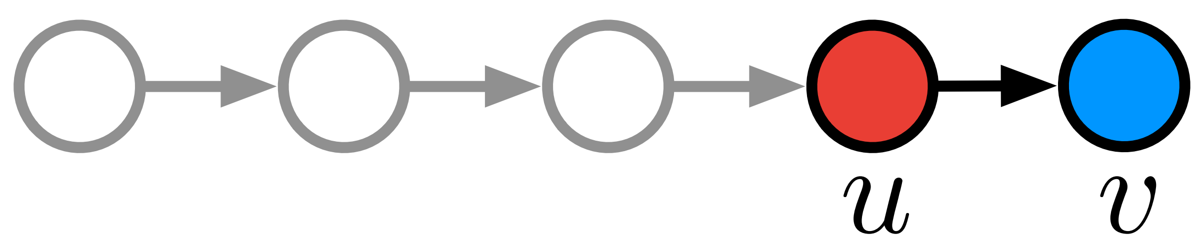
	A_0	A_1	$\cdots$	A_t
#colours	3	$2^3 - 2$	$\cdots$	$\geq n$
#rounds	t	$t - 1$	$\cdots$	0

Successor relation

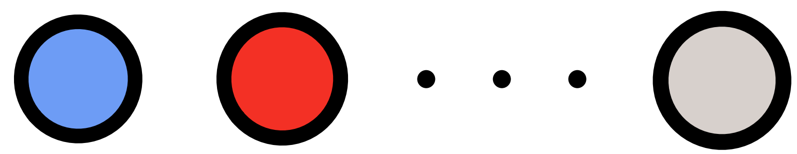
Consider A_k that outputs colours from

$C_k = \{$   $\cdots$  $\}$.

Colour  is a *successor* of colour 

if A_k outputs 

Successor graph

Nodes: $C_k = \{$  $\}$

Edges: the successor relation

Starting from any algorithm we get

Algorithm:	A_0	A_1	$\cdots$	A_t
Successor graph:	$\mathcal{S}_0$	$\mathcal{S}_1$	$\cdots$	$\mathcal{S}_t$

Colourability lemma

$\mathcal{S}_k$ is *c*-colourable

$$\Rightarrow$$

there is a *c*-colouring algorithm running in *t*-*k* rounds

A finite super graph

For all *k*, there is a **finite graph** that contains the successor graph of **any algorithm** as a subgraph.

Proving lower bounds

Super graph + colorability lemma:

Chromatic number an upper bound for all successor graphs!

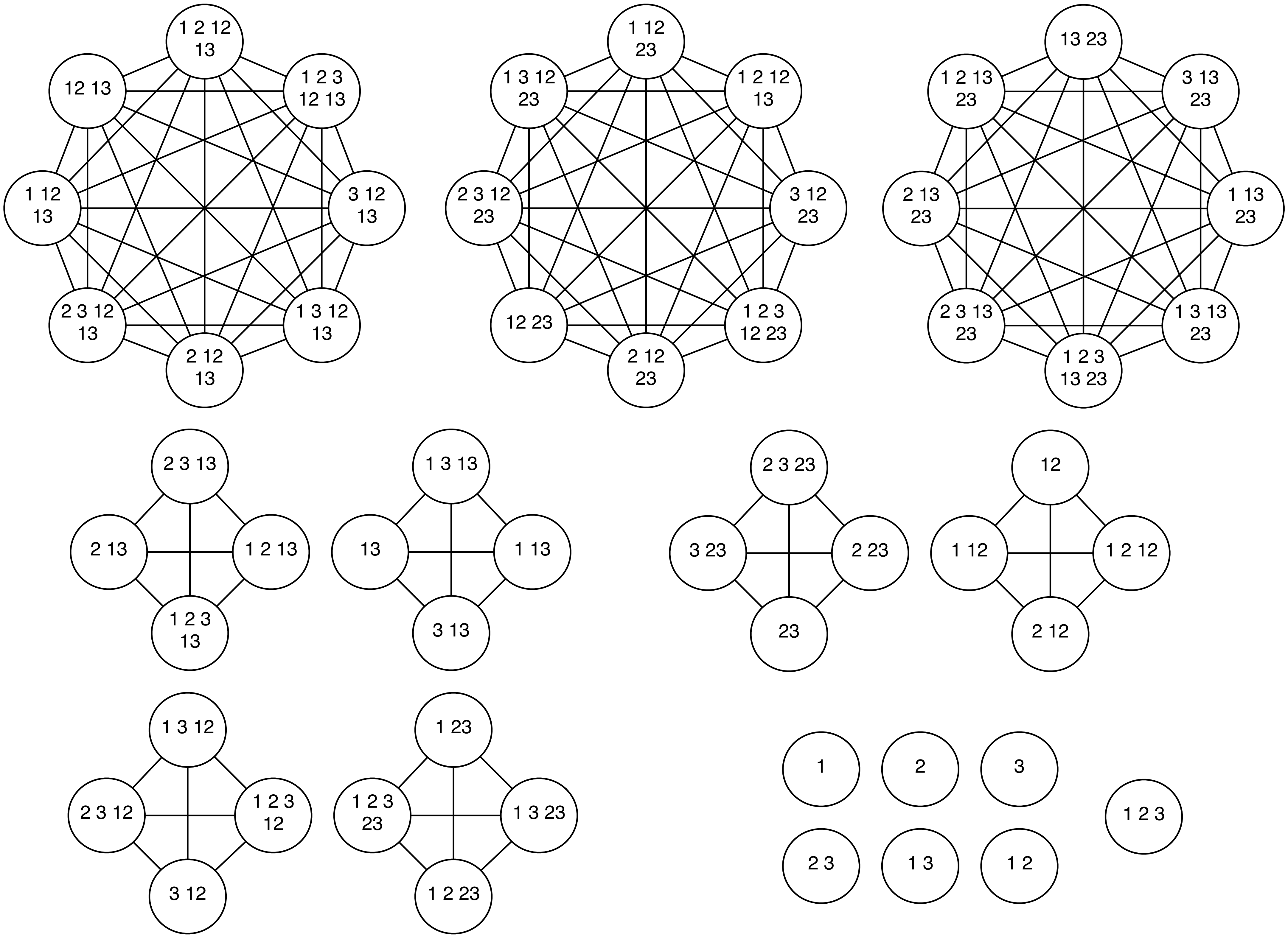
Finite super graph:

Easy to use a *computer search* for small enough super graphs!

The key result

For **any** t-time 3-colouring algorithm,
the successor graph $\mathcal{S}_2$ is **16-colourable**

Complement of S_2



The key result

For **any** t-time 3-colouring algorithm, the successor graph $\mathcal{S}_2$ is **16-colourable**

By **colourability lemma**, there exists a 16-colouring algorithm running in $t-2$ rounds

The lower bound

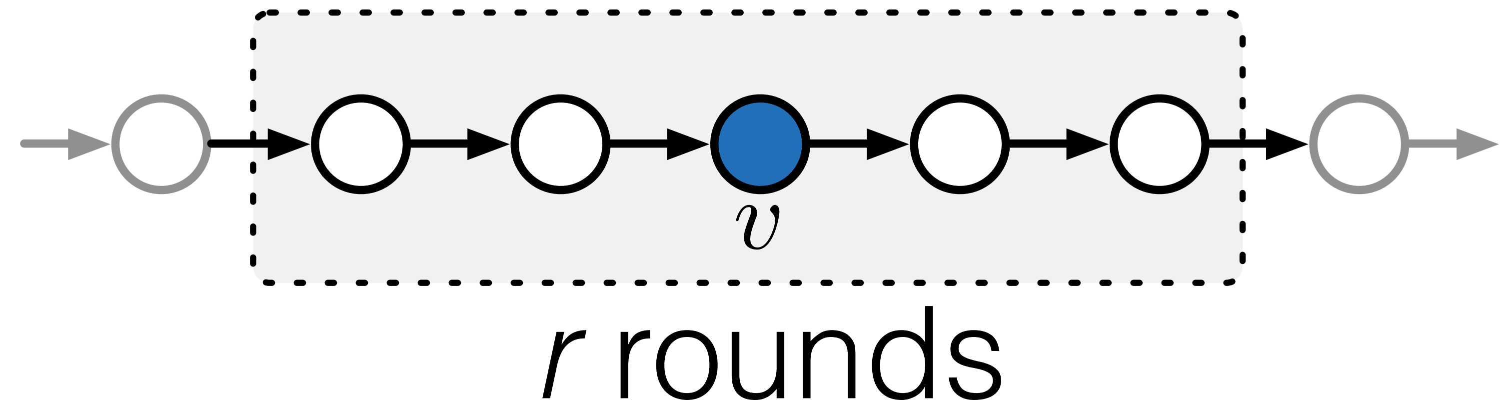
Step 1. Iterated speed-up lemma:
16-colouring takes $\log^* n - 2$ rounds

Step 2. Successor graph bound:
3-colouring takes $\log^* n$ rounds

Two-sided ≈ one-sided

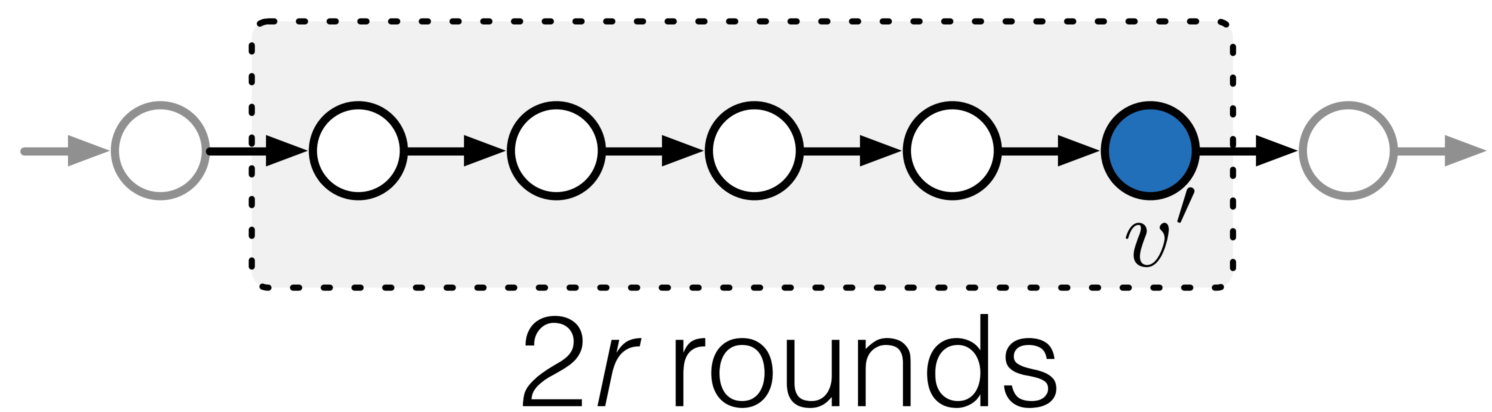
Two-sided view

$C(n, 3)$



One-sided view

$T(n, 3)$



$$C(n, 3) = \lceil T(n, 3)/2 \rceil$$

Conclusions

For infinitely many values

$$C(n,3) = \frac{1}{2}\log^* n.$$

Use **successor graphs** *and* **computers** for lower bound proofs!

Conclusions

For infinitely many values

$$C(n,3) = \frac{1}{2}\log^* n.$$

Use **successor graphs** *and* **computers** for lower bound proofs!

Thanks!