Reliability/Robustness in Distributed Computing

David Ilcinkas

CNRS and Univ. Bordeaux (LaBRI), France

AlgoTel 2020
October 1st

In this talk

Main message

Reliable/robust algorithms for harsh environments

$\Longrightarrow$

Reliable/robust research papers

Outline

- The specificity of distributed computing
- Improving the confidence in the results

In this talk

Main message

Reliable/robust algorithms for harsh environments

$\Longrightarrow$

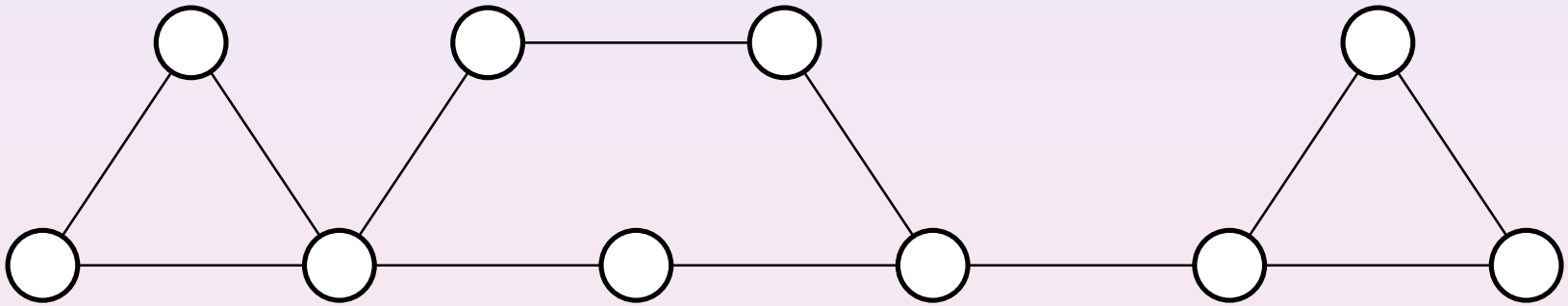
Reliable/robust research papers

Outline

- The specificity of distributed computing
- Improving the confidence in the results

Rooted shortest-path tree

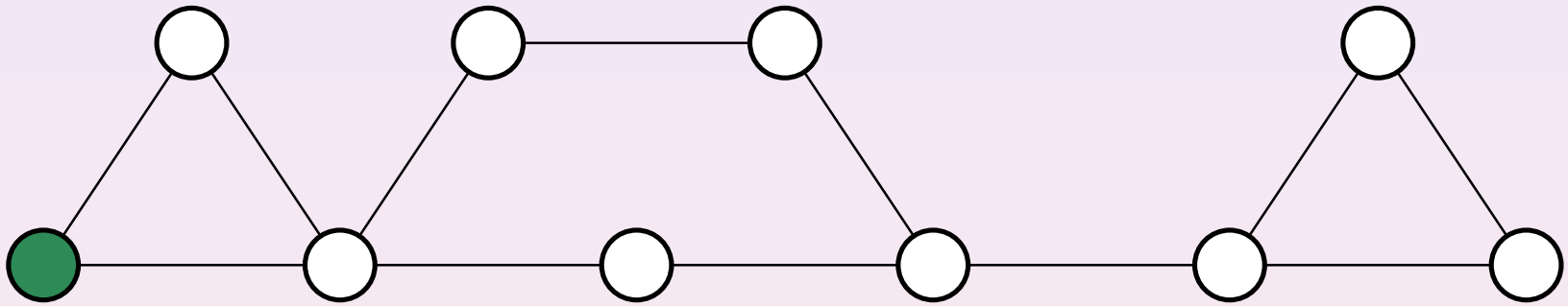
(Framework: distributed computing in networks.)



Basically, local knowledge of a shortest path to a fixed node.

Rooted shortest-path tree

(Framework: distributed computing in networks.)

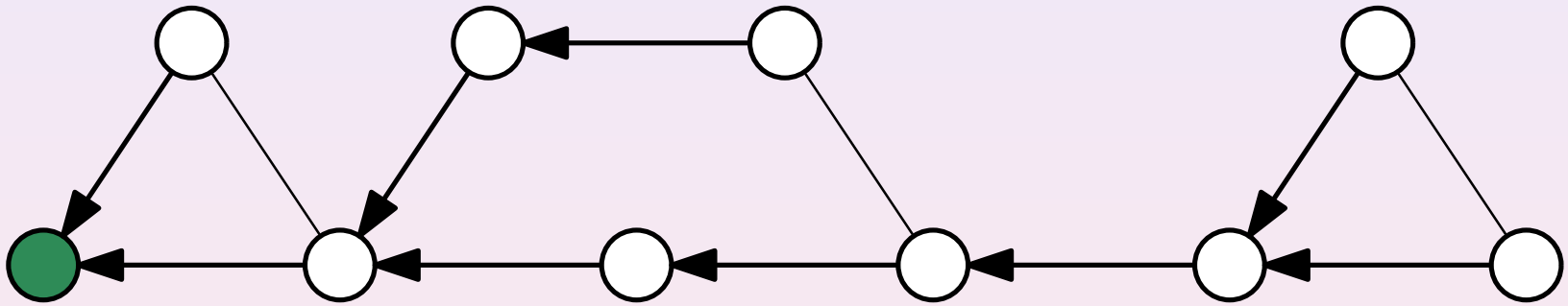


root r

Basically, local knowledge of a shortest path to a fixed node.

Rooted shortest-path tree

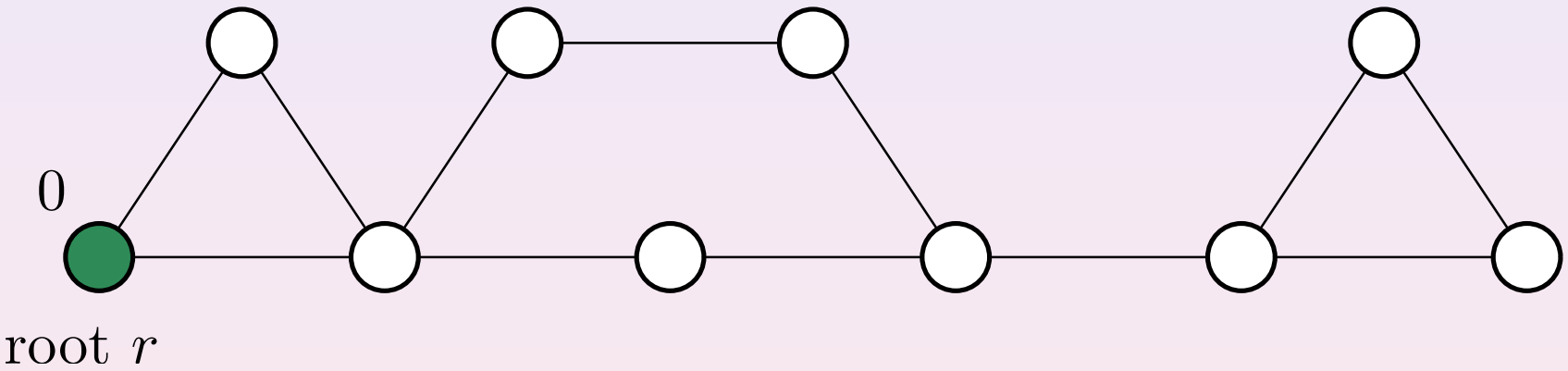
(Framework: distributed computing in networks.)



root r

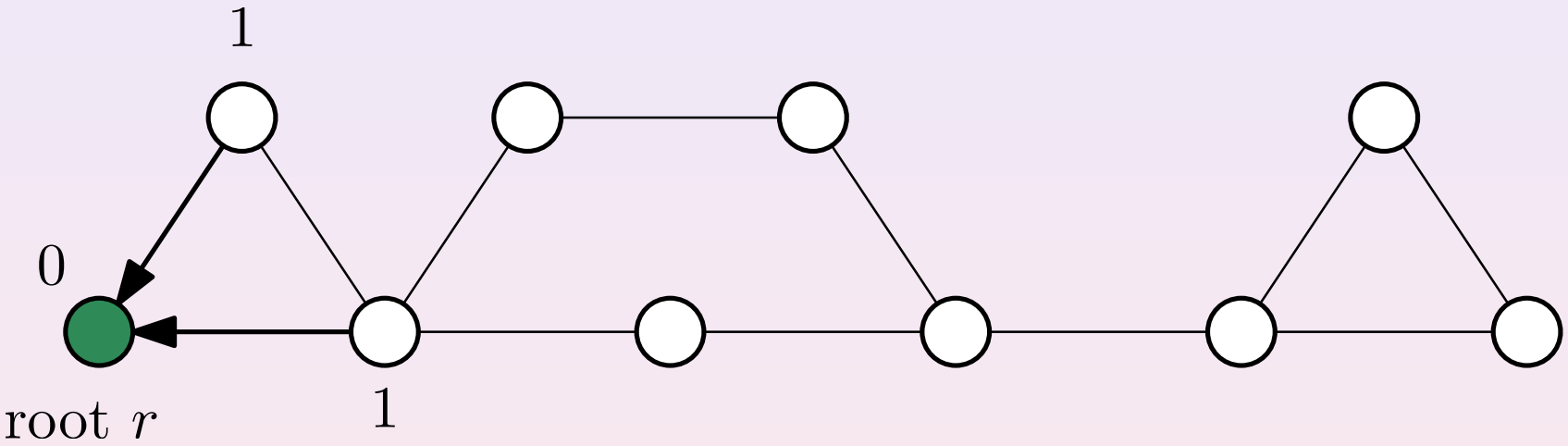
Basically, local knowledge of a shortest path to a fixed node.

Bellman-Ford Algorithm



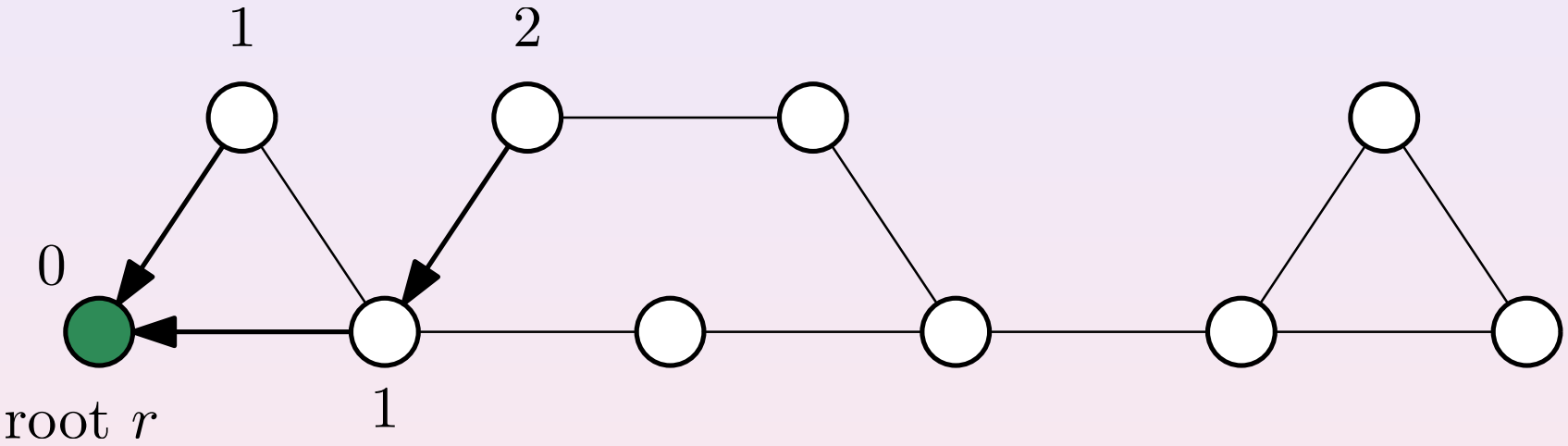
Distributed propagation of distances from the root.

Bellman-Ford Algorithm



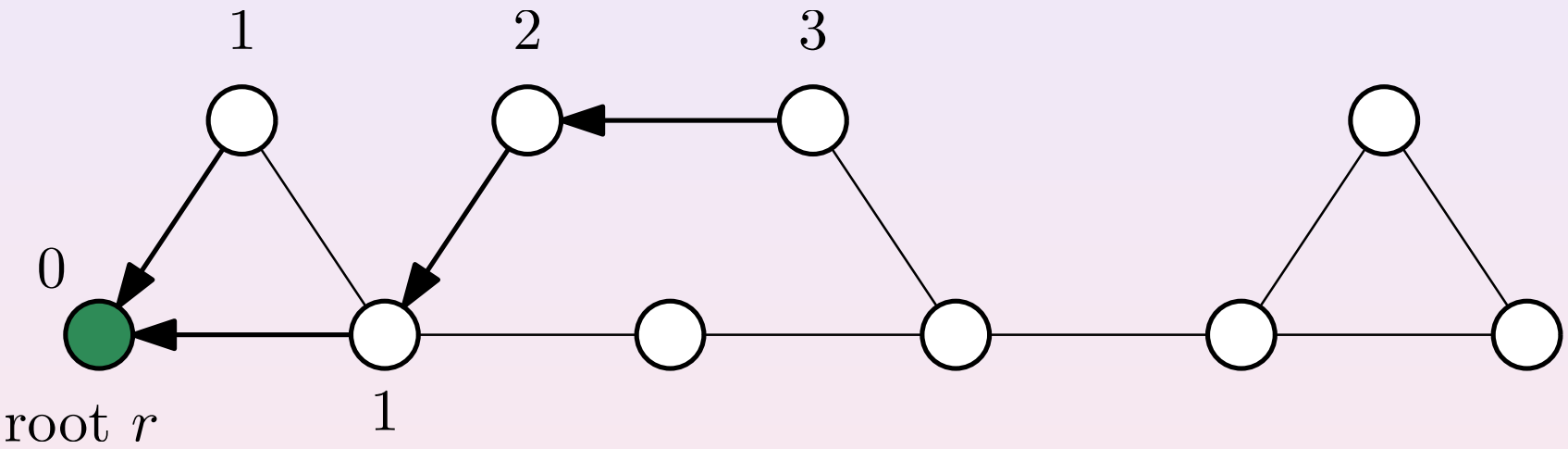
Distributed propagation of distances from the root.

Bellman-Ford Algorithm



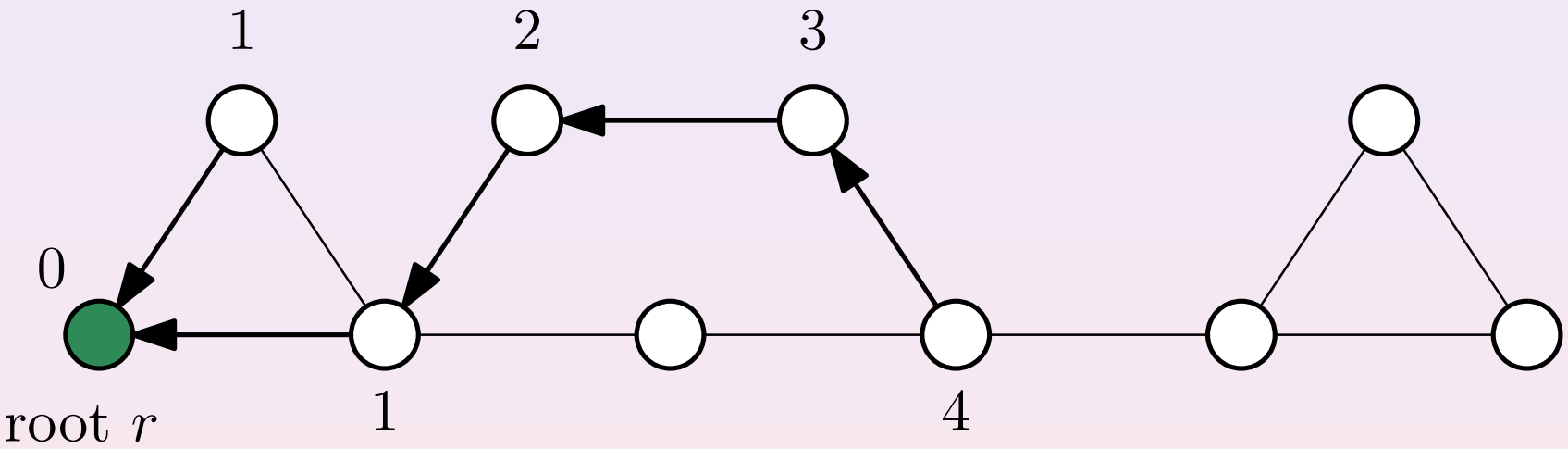
Distributed propagation of distances from the root.

Bellman-Ford Algorithm



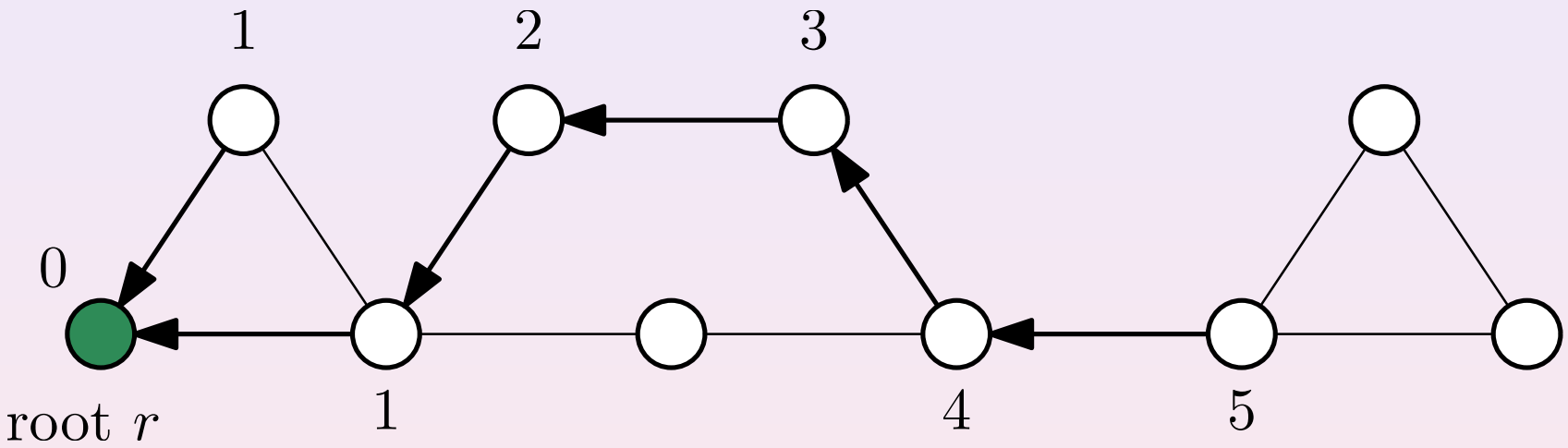
Distributed propagation of distances from the root.

Bellman-Ford Algorithm



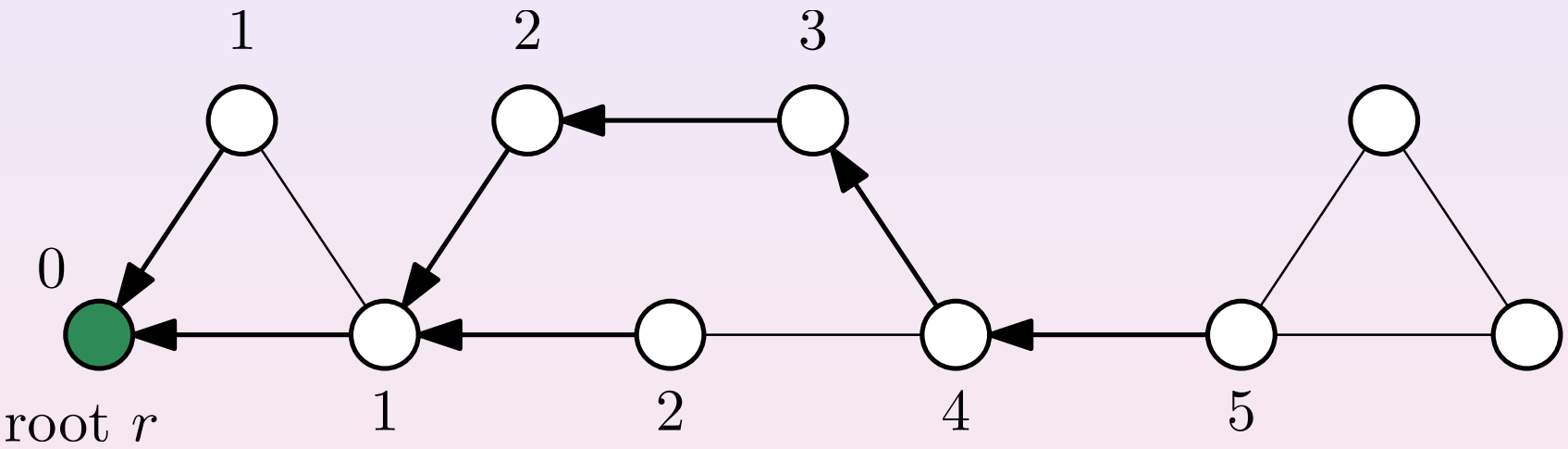
Distributed propagation of distances from the root.

Bellman-Ford Algorithm



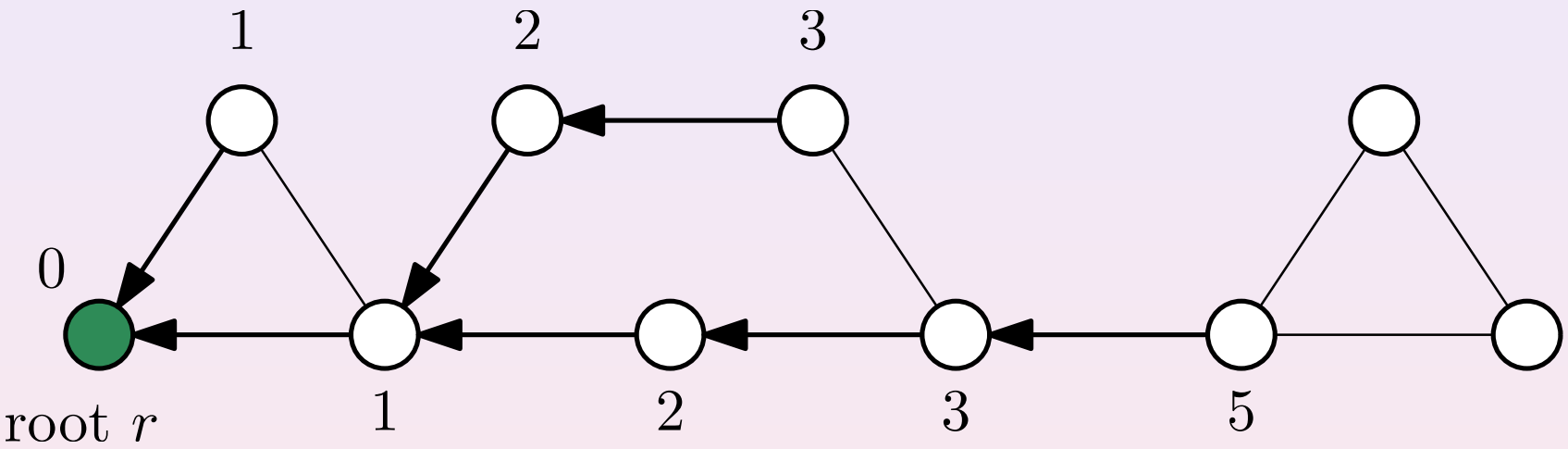
Distributed propagation of distances from the root.

Bellman-Ford Algorithm



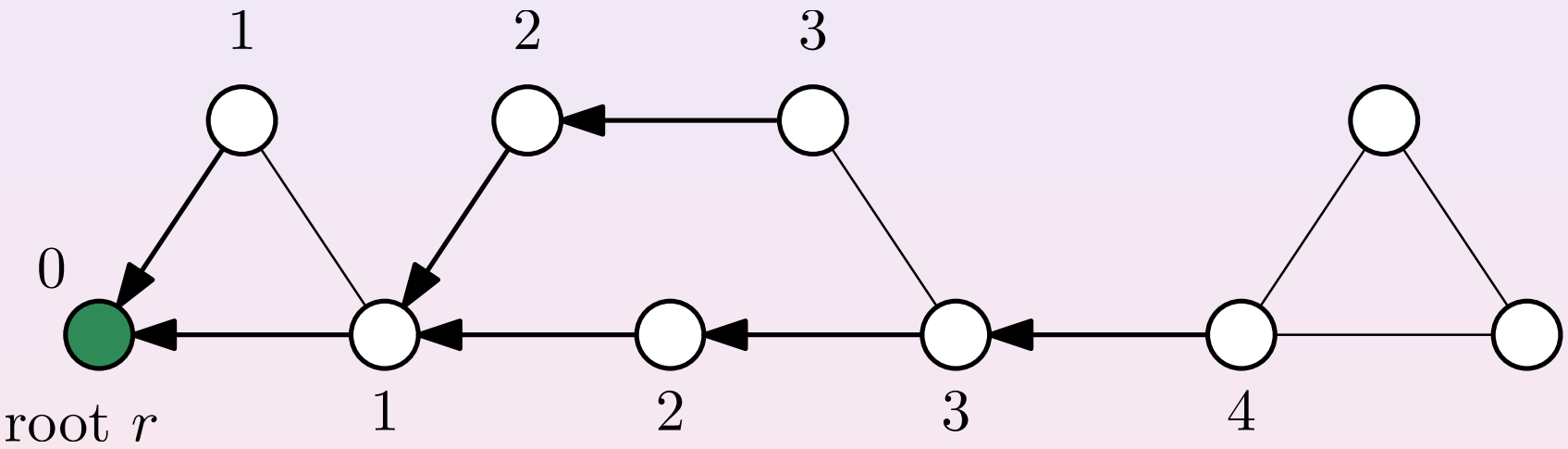
Distributed propagation of distances from the root.

Bellman-Ford Algorithm



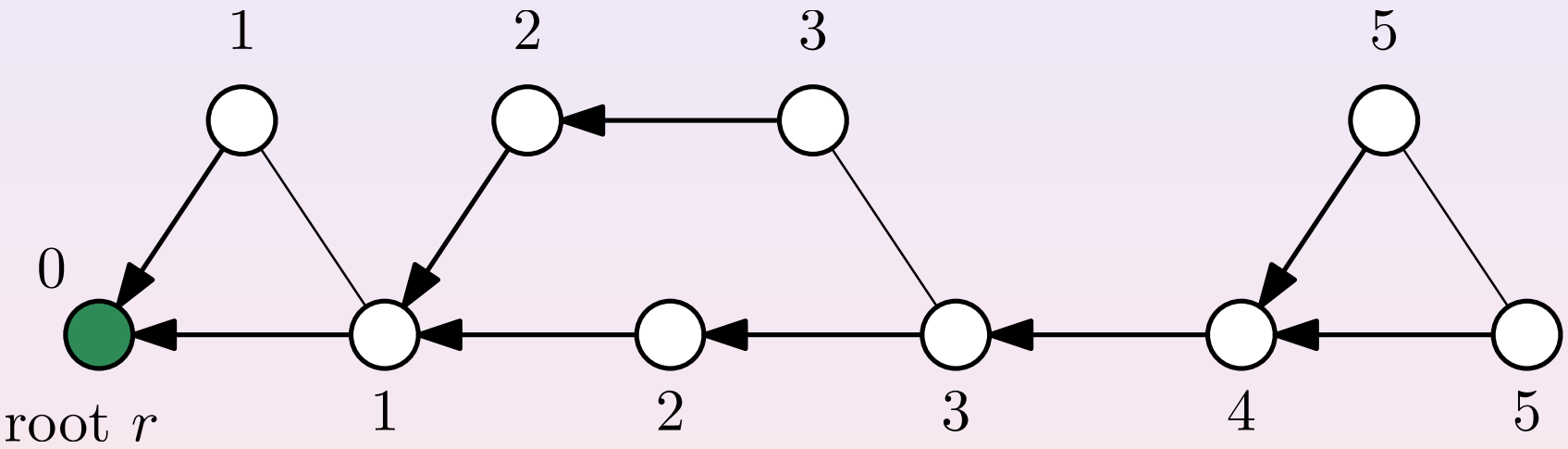
Distributed propagation of distances from the root.

Bellman-Ford Algorithm



Distributed propagation of distances from the root.

Bellman-Ford Algorithm



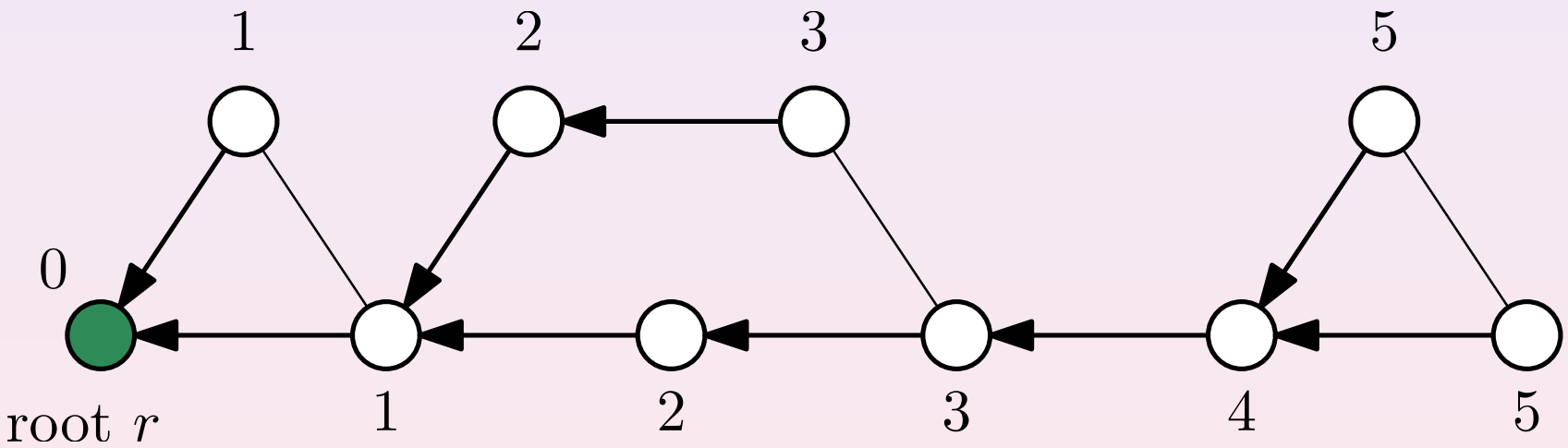
Distributed propagation of distances from the root.

Count-to-infinity problem

The network may be dynamic.

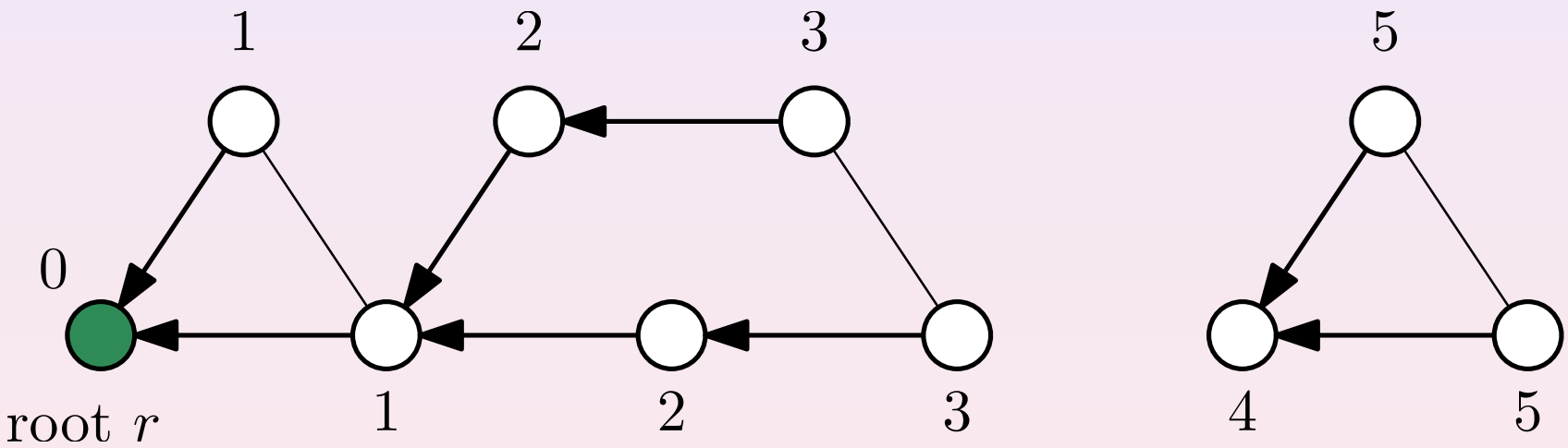
Count-to-infinity problem

The network may be dynamic.



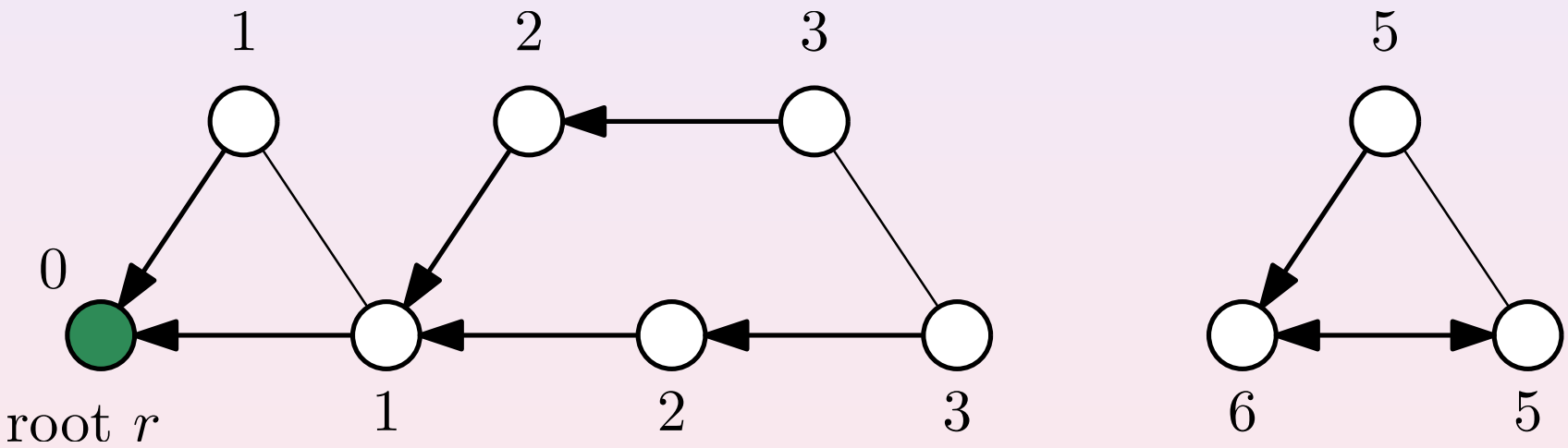
Count-to-infinity problem

The network may be dynamic.



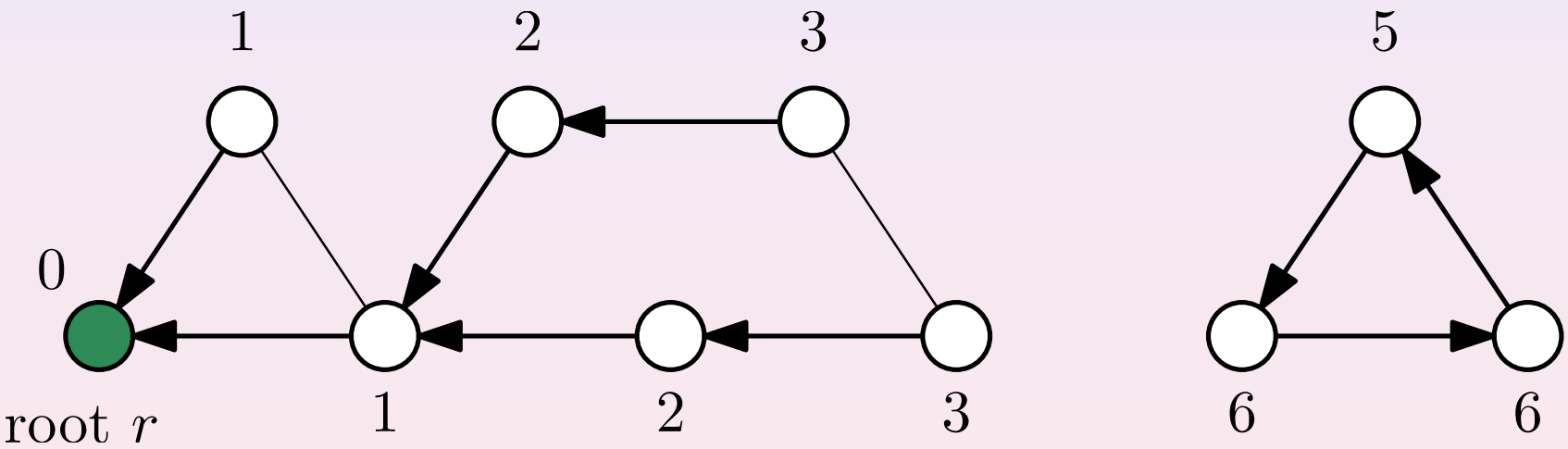
Count-to-infinity problem

The network may be dynamic.



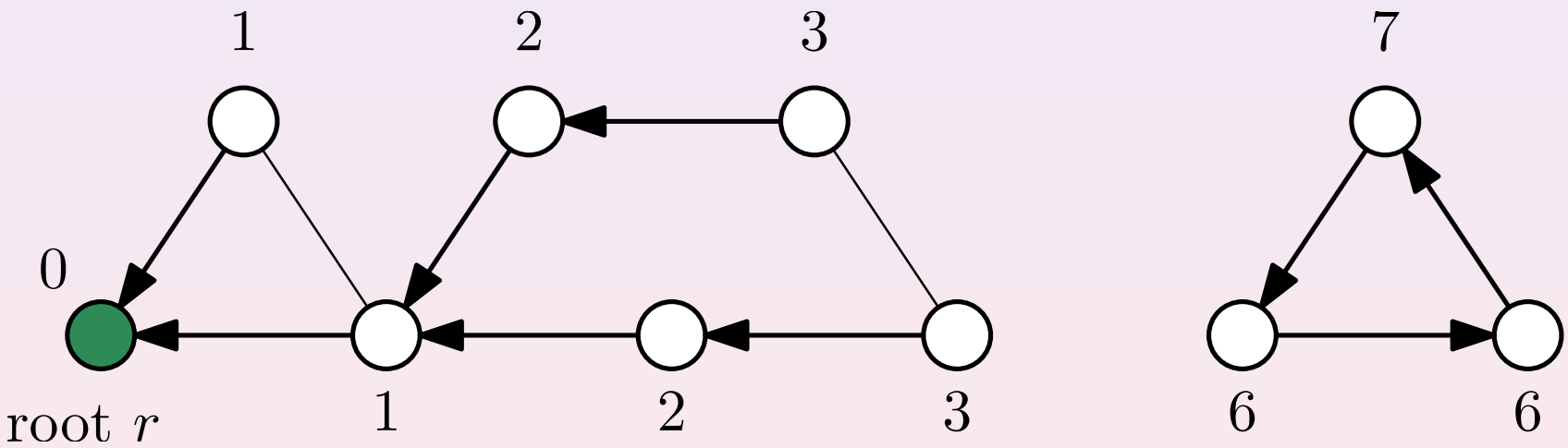
Count-to-infinity problem

The network may be dynamic.



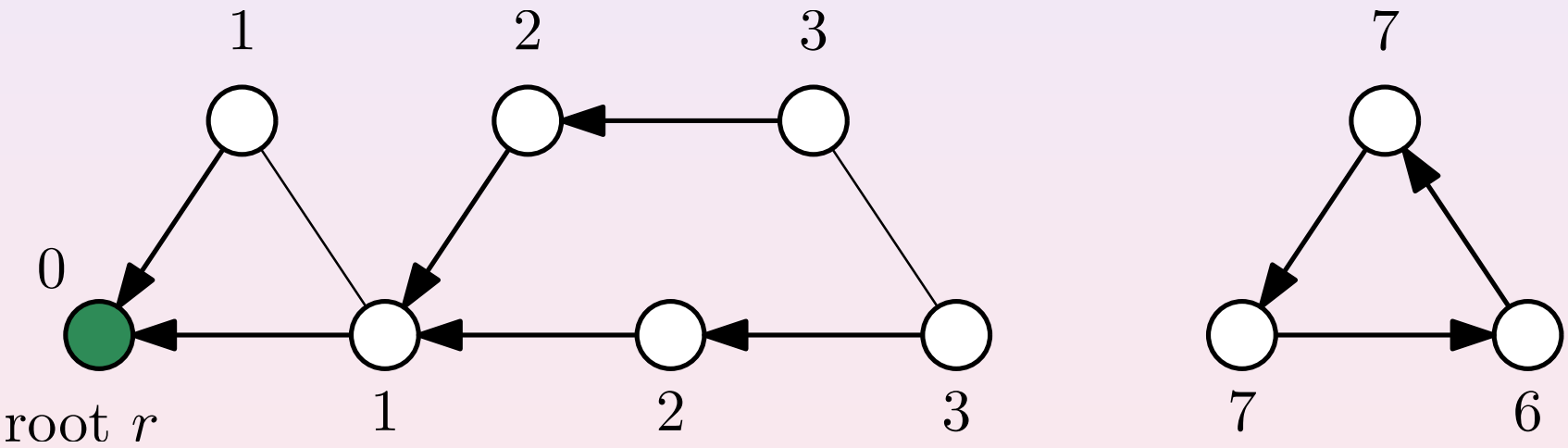
Count-to-infinity problem

The network may be dynamic.



Count-to-infinity problem

The network may be dynamic.



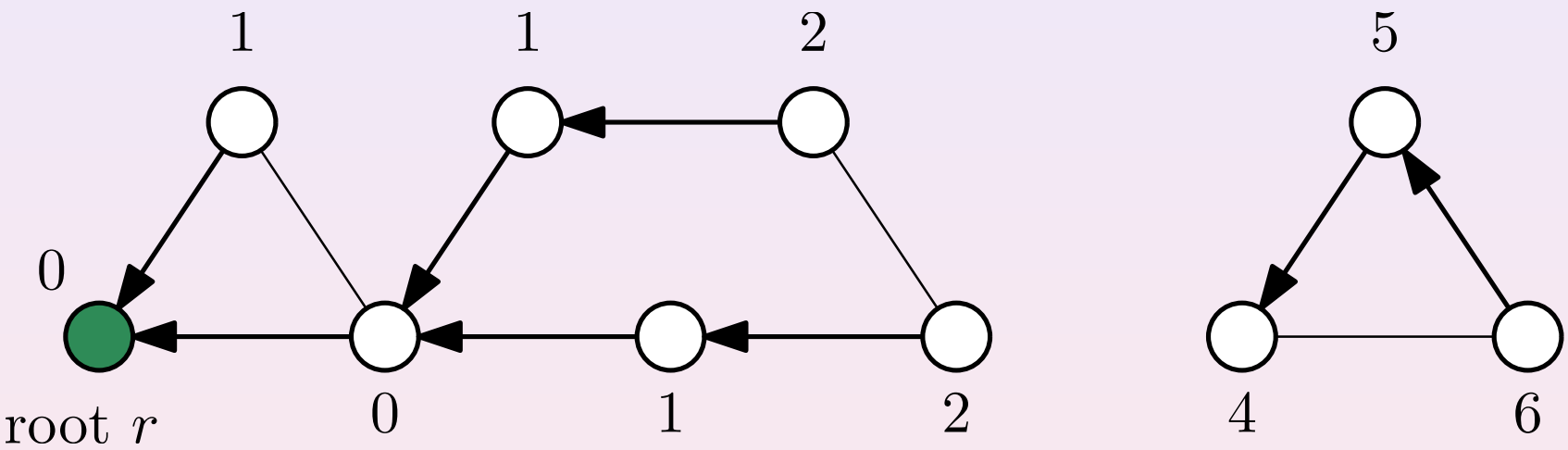
A self-stabilizing version

Self-stabilizing: Correct whatever the initial configuration

To be pretty cautious. . . . . . but not too much.

A self-stabilizing version

Self-stabilizing: Correct whatever the initial configuration

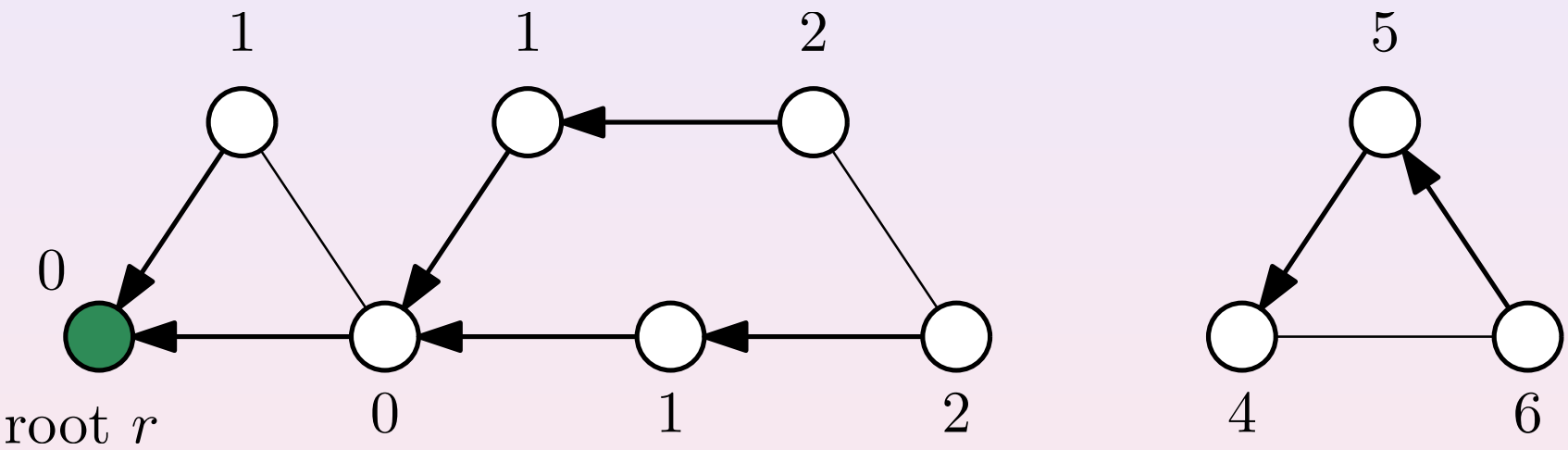


To be pretty cautious...

...but not too much.

A self-stabilizing version

Self-stabilizing: Correct whatever the initial configuration

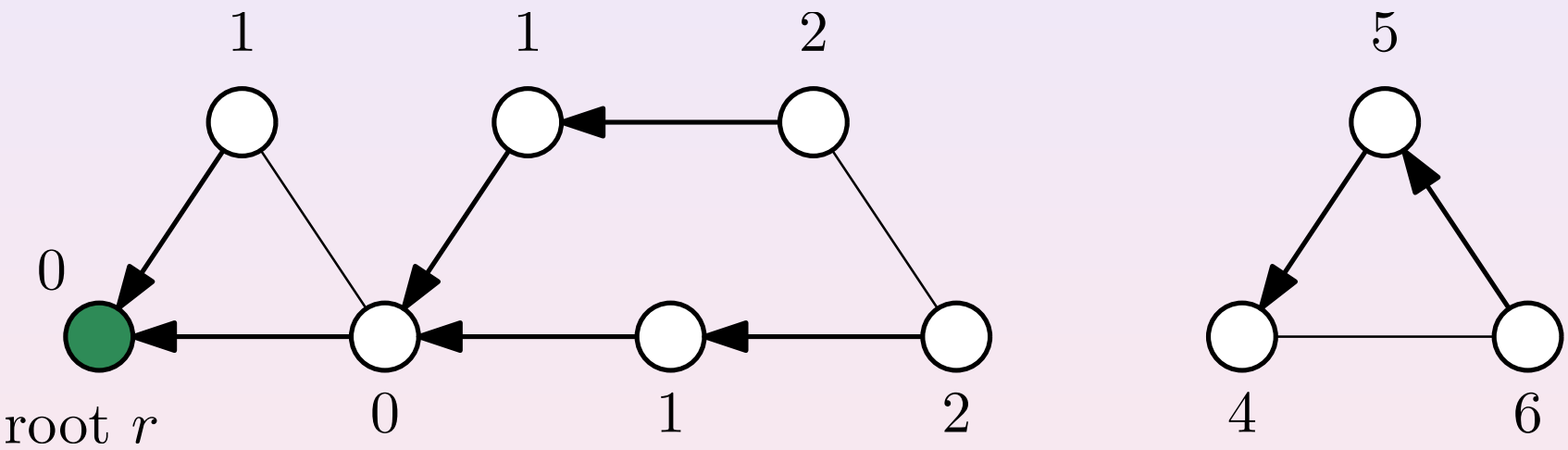


To be pretty cautious...

...but not too much.

A self-stabilizing version

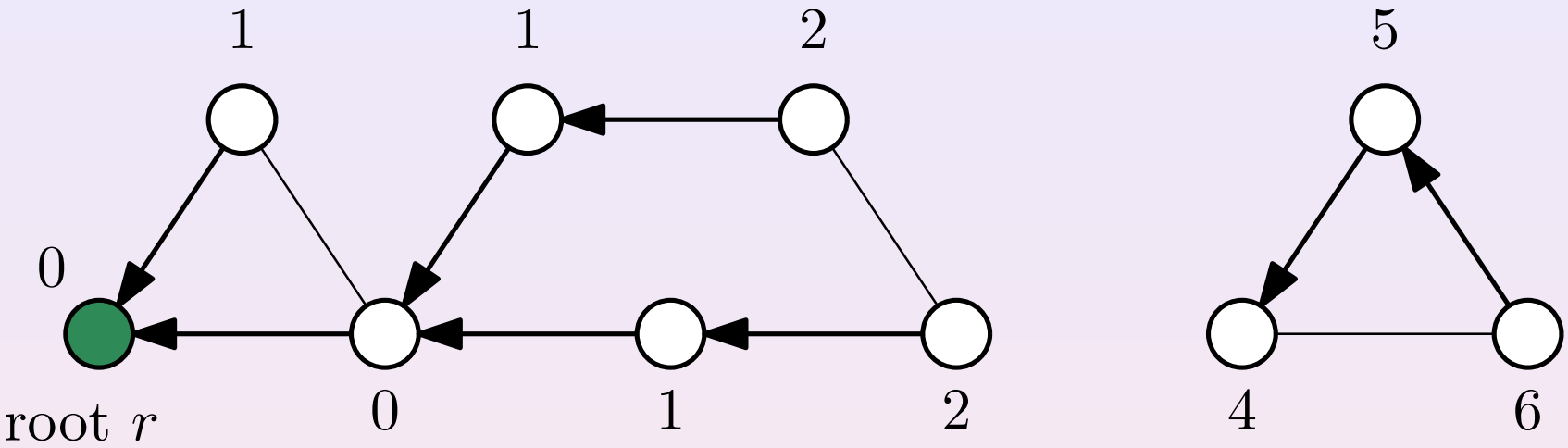
Self-stabilizing: Correct whatever the initial configuration



To be pretty cautious...

...but not too much.

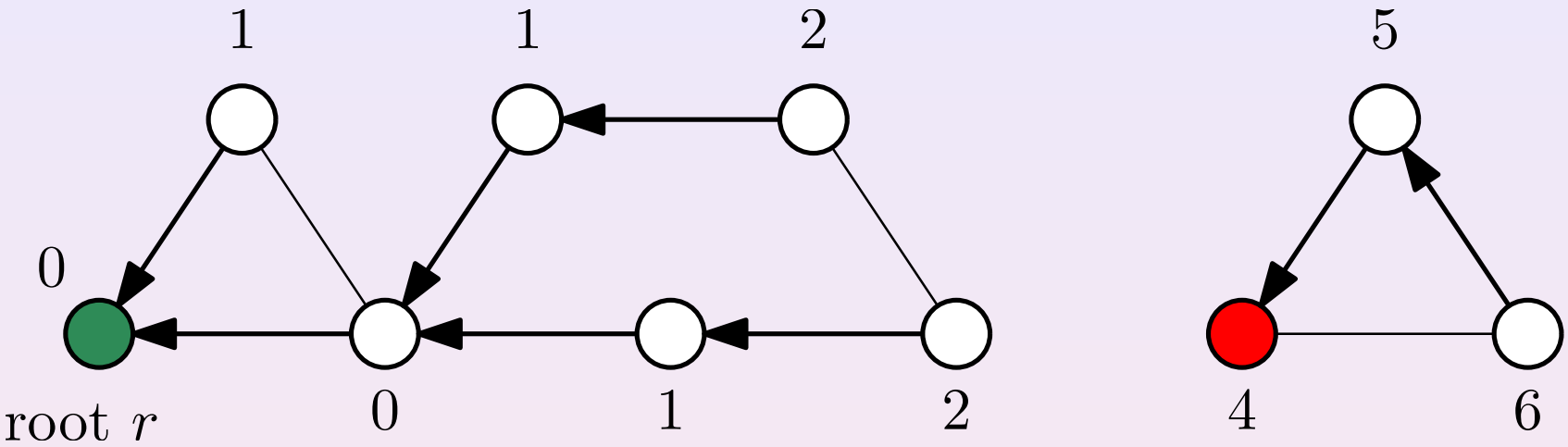
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

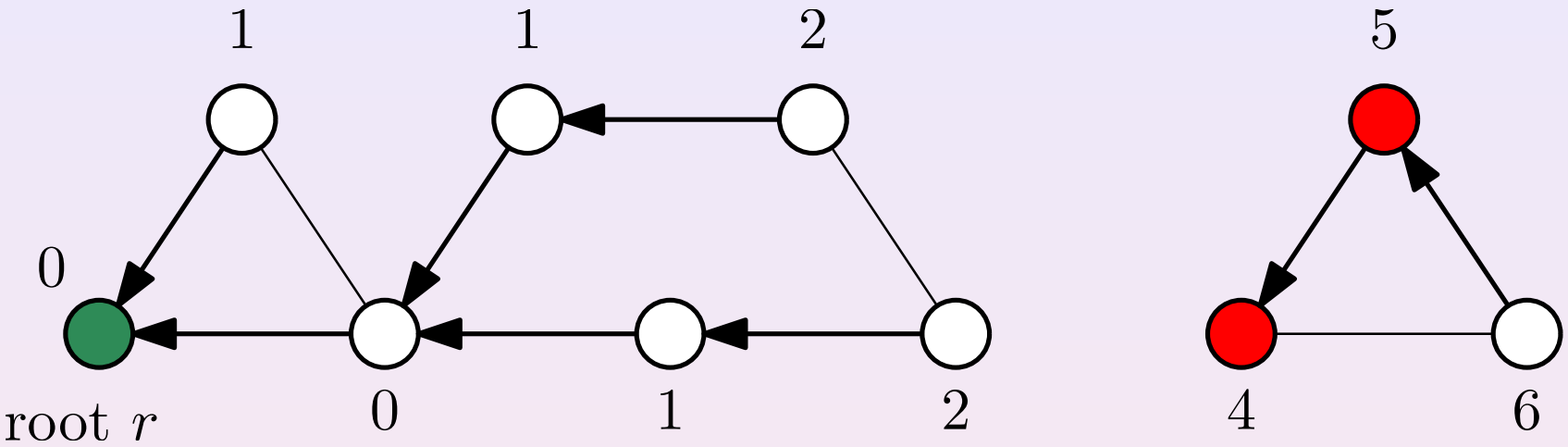
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

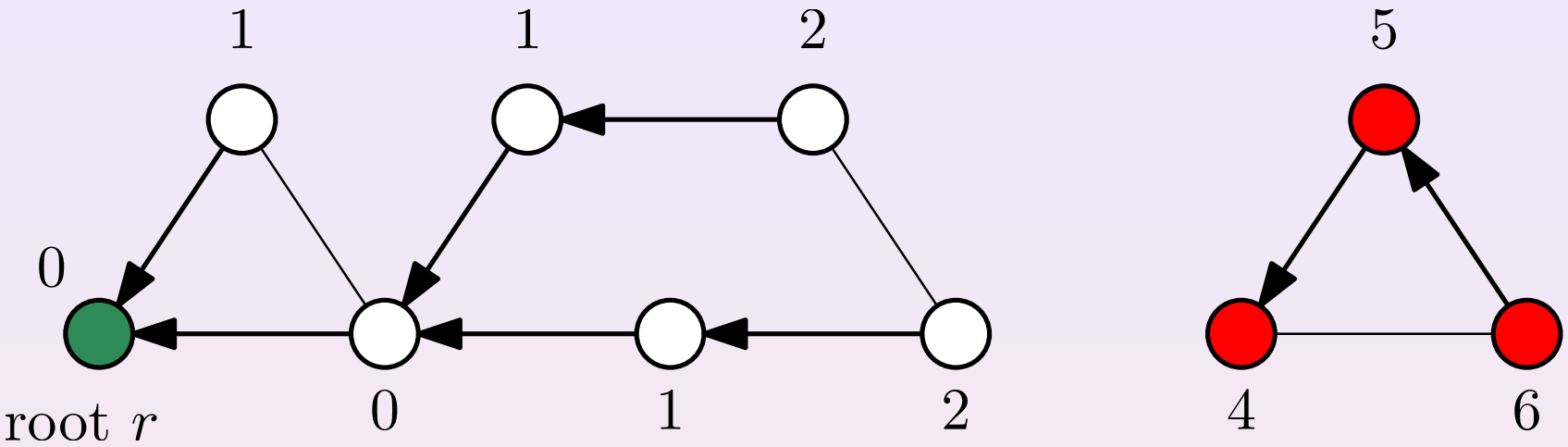
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

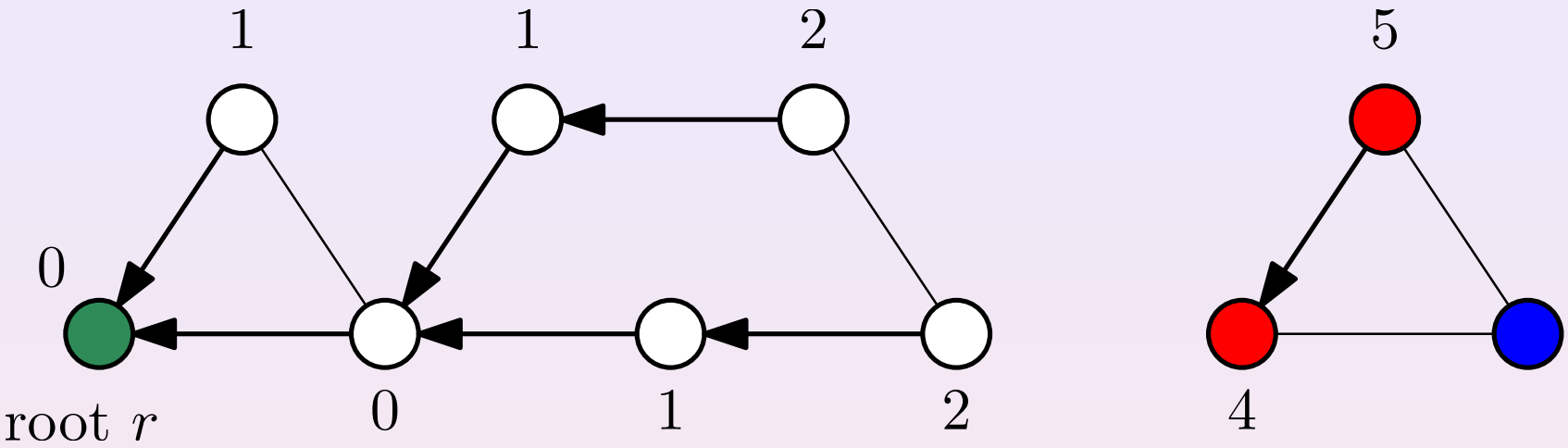
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

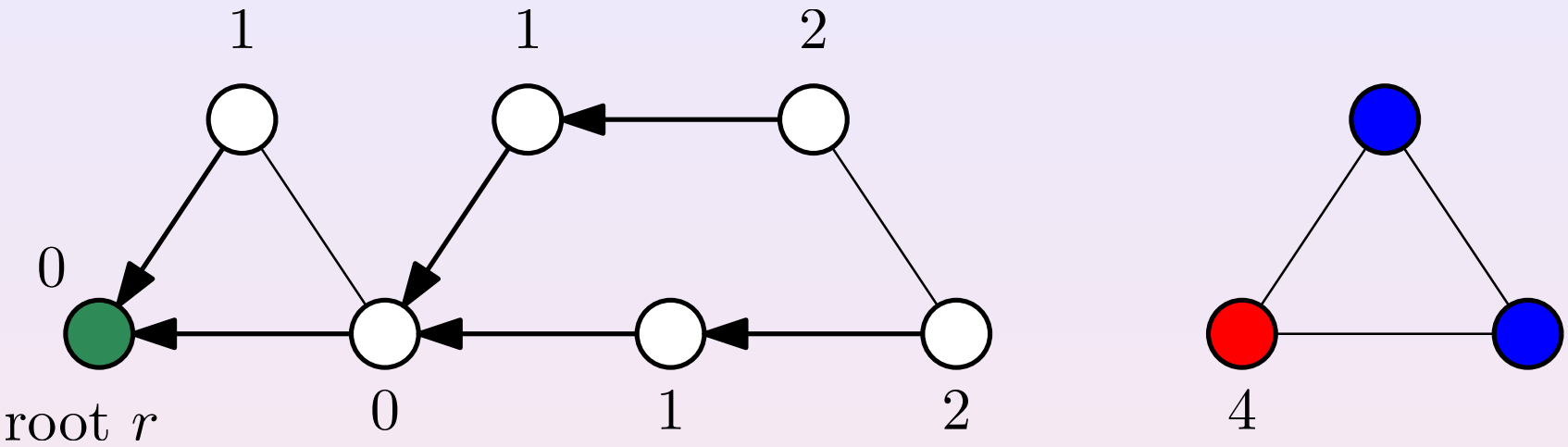
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

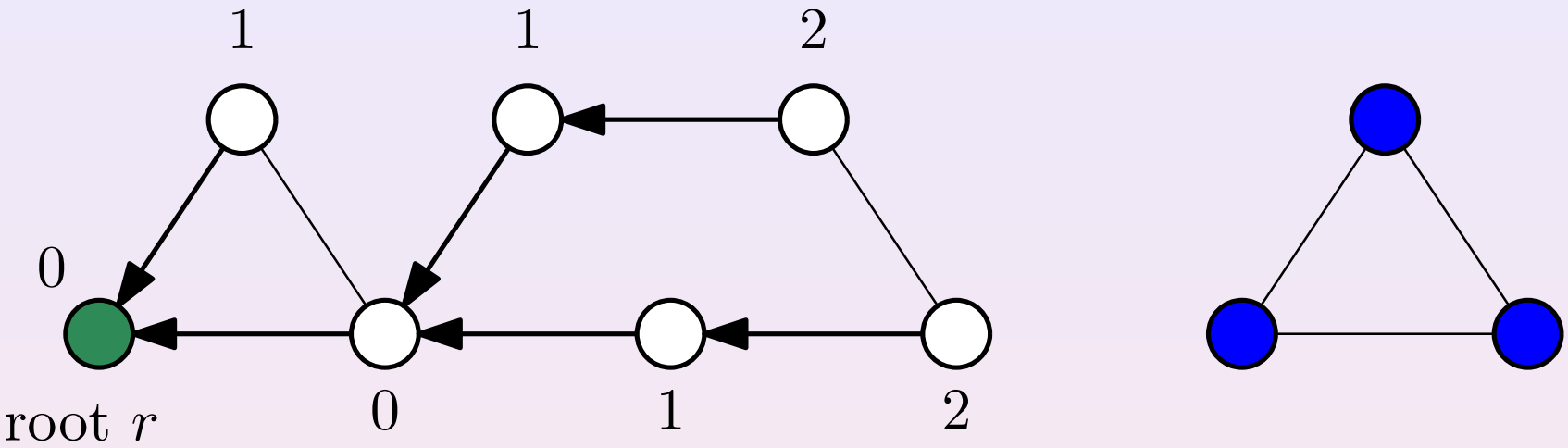
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

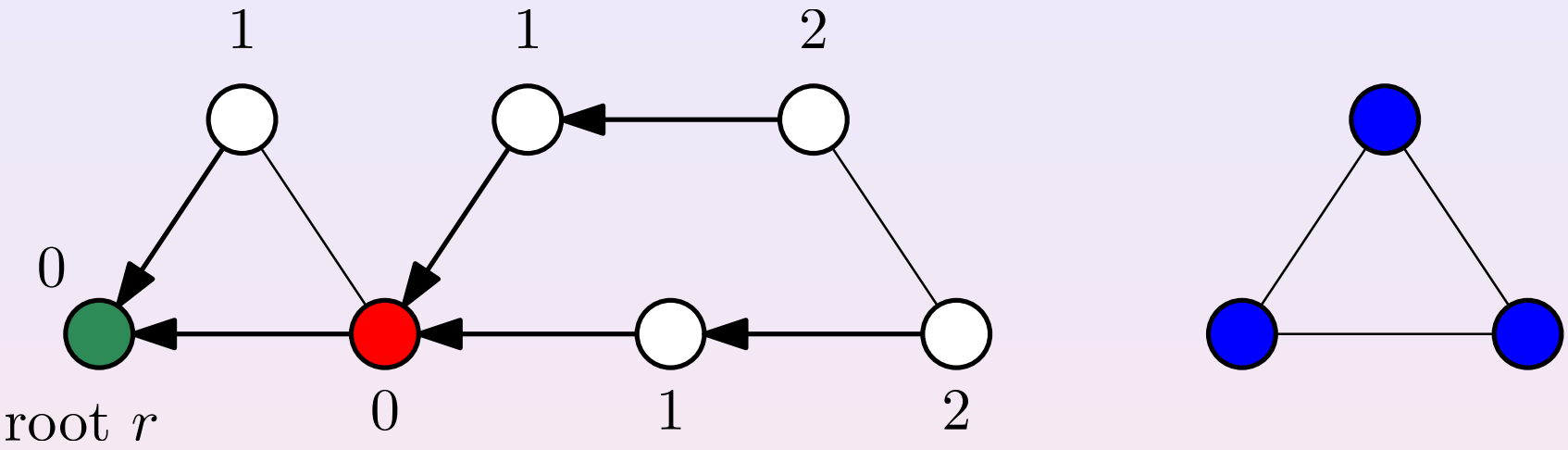
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

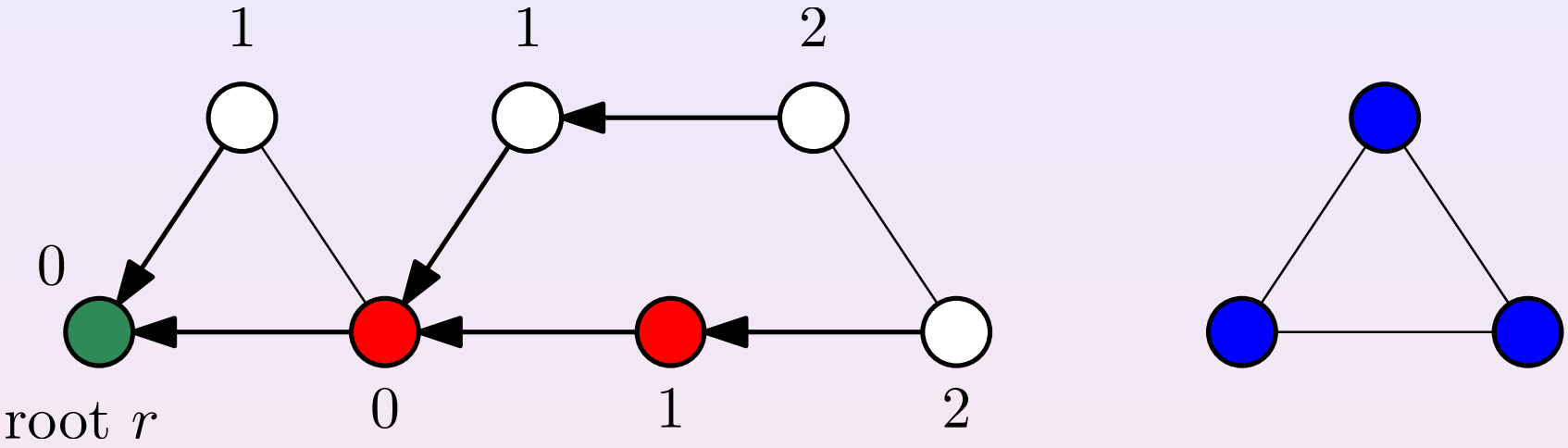
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm [Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps. [Devismes, I., Johnen, 2016]

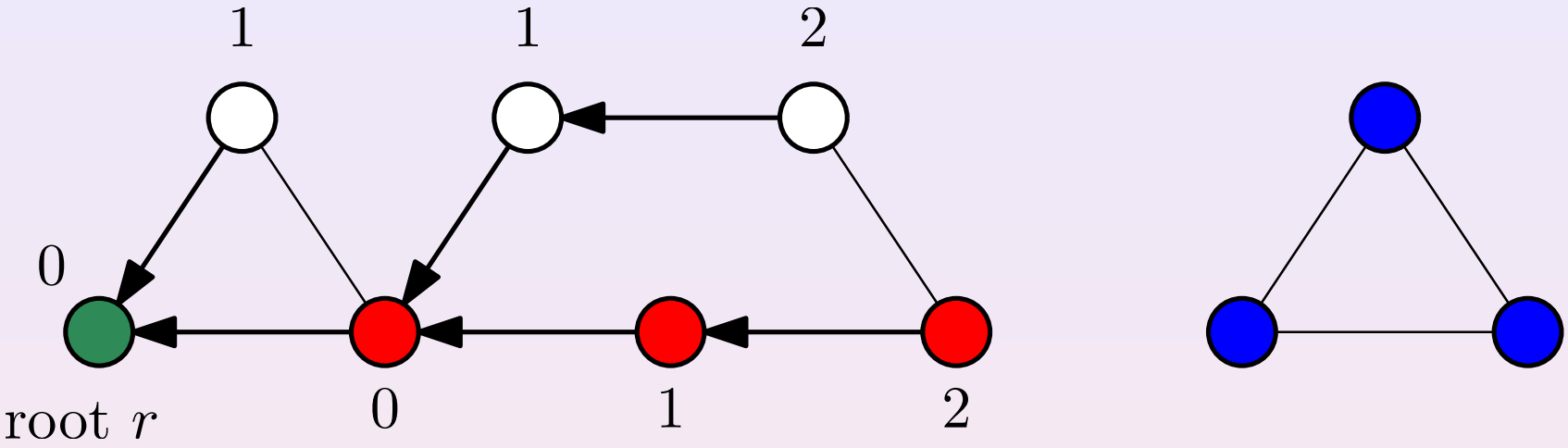
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm [Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps. [Devismes, I., Johnen, 2016]

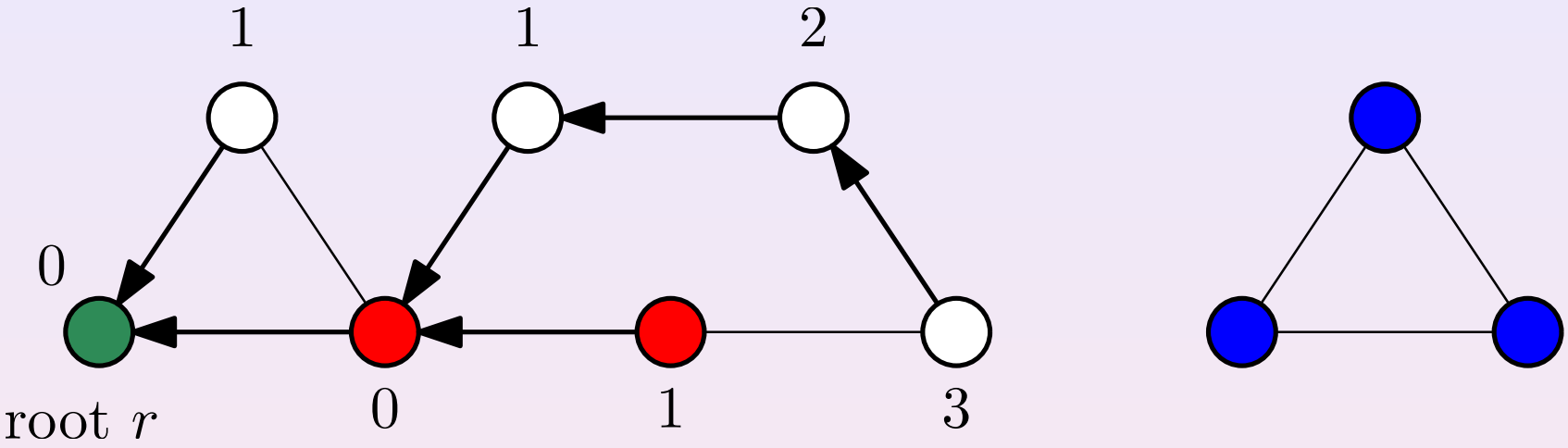
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm [Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps. [Devismes, I., Johnen, 2016]

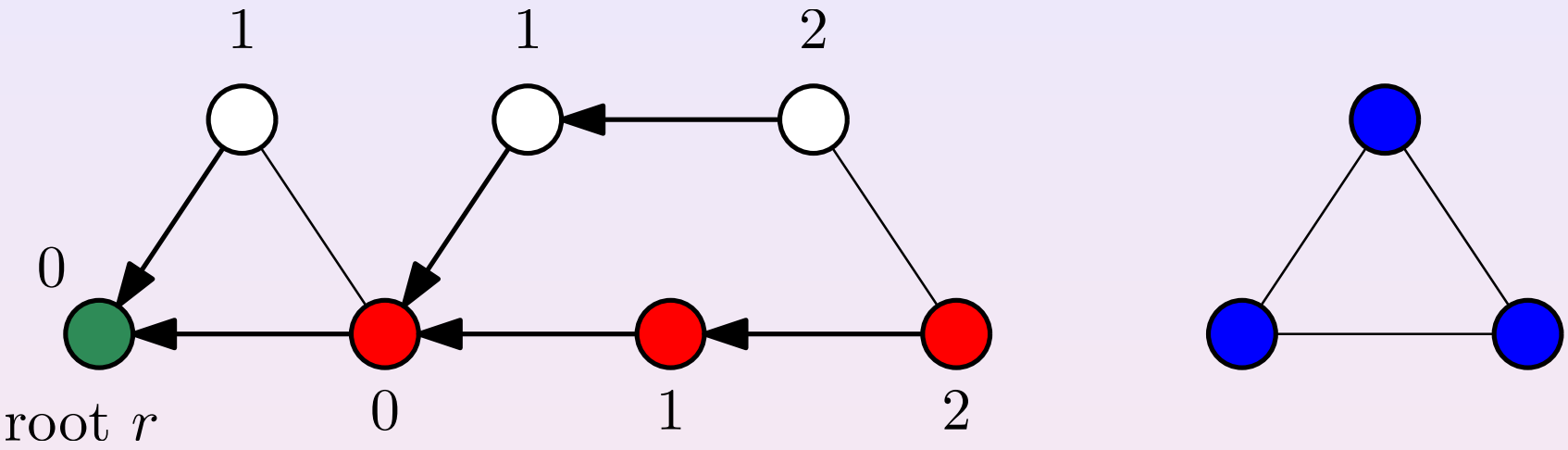
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm [Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps. [Devismes, I., Johnen, 2016]

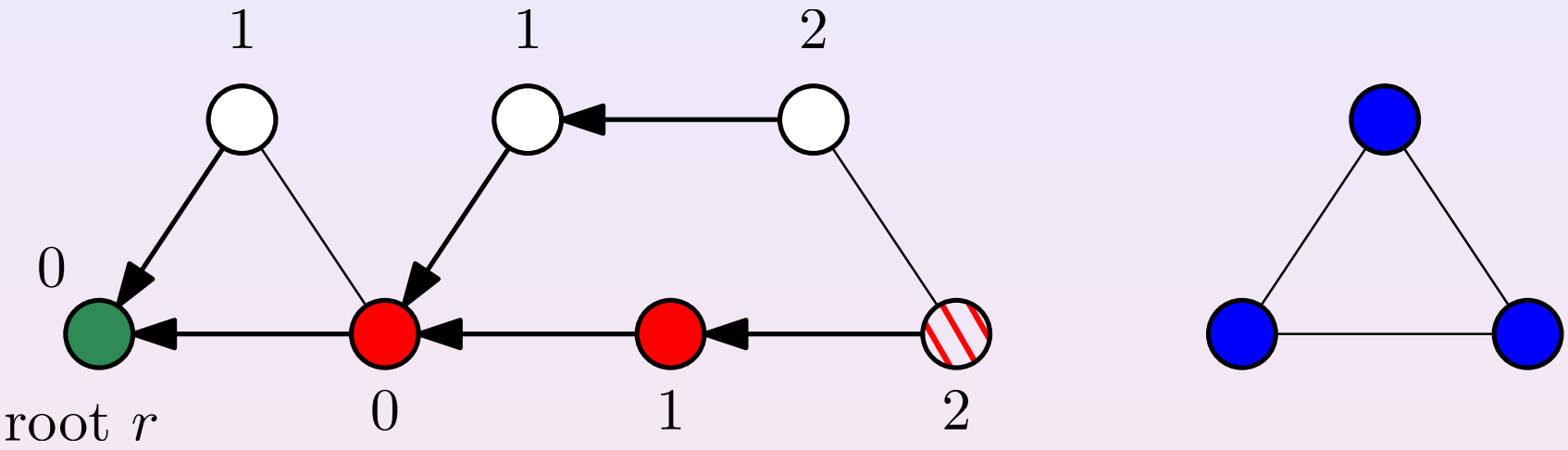
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

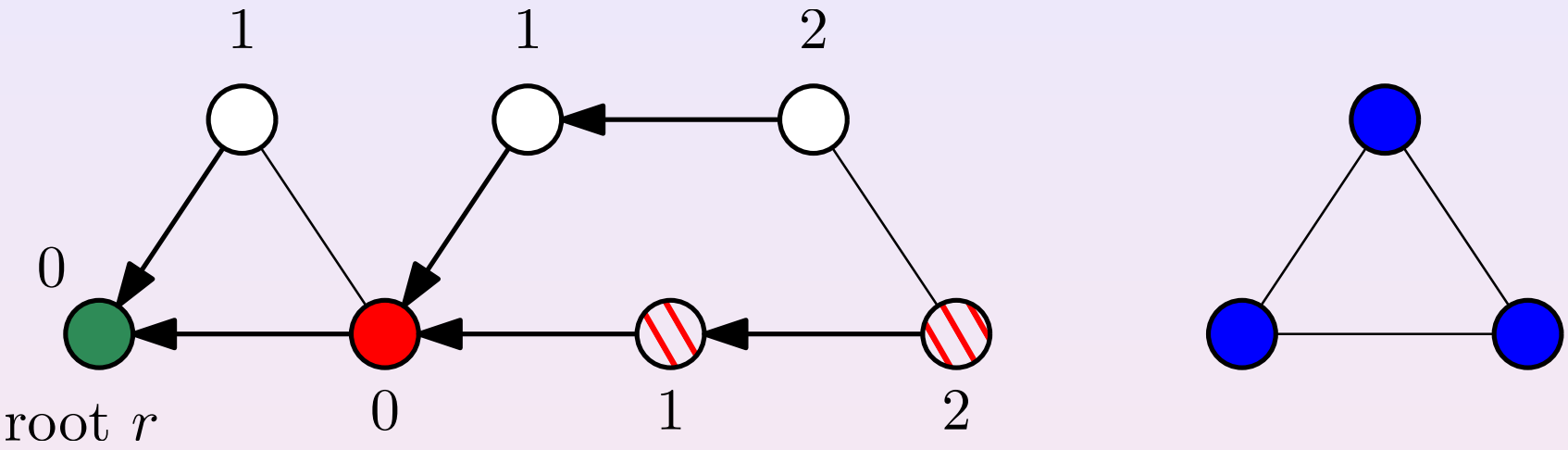
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

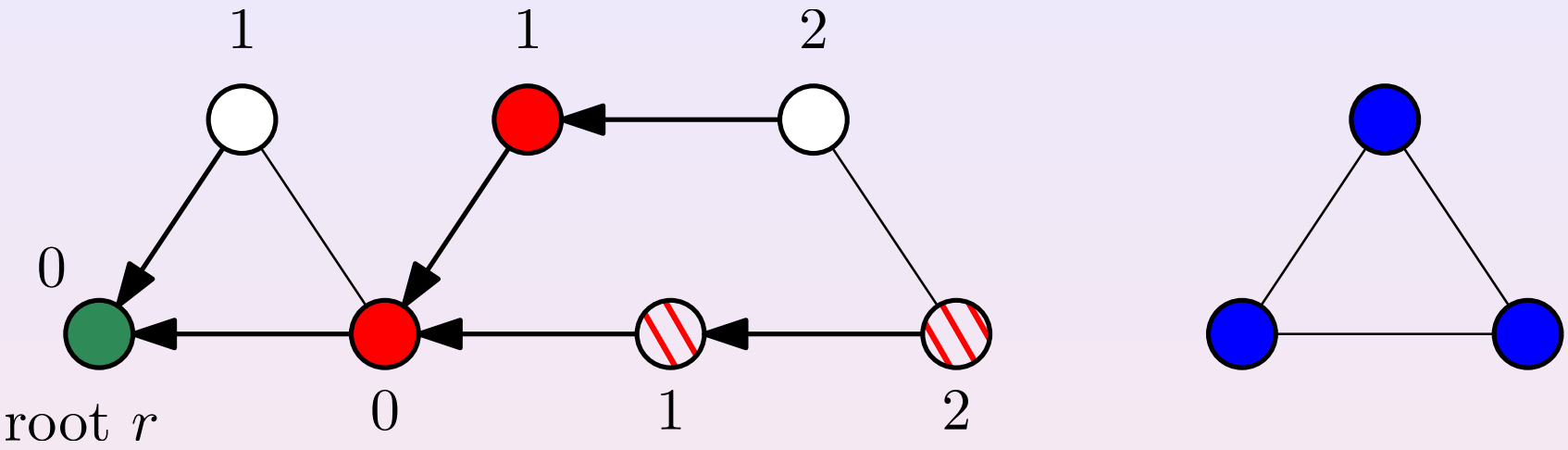
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

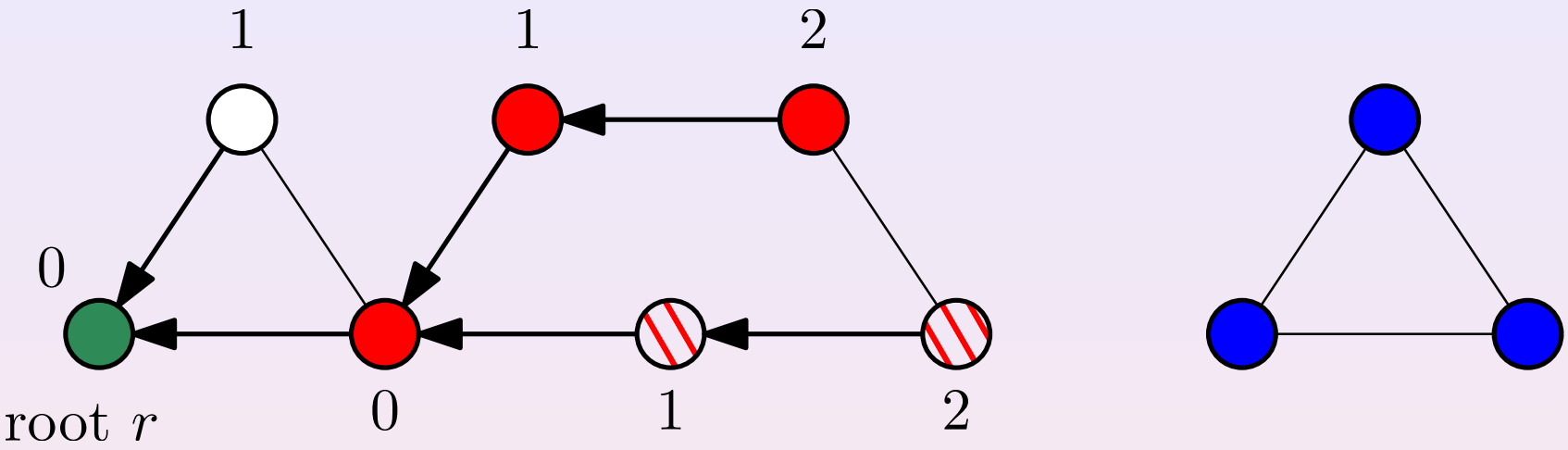
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

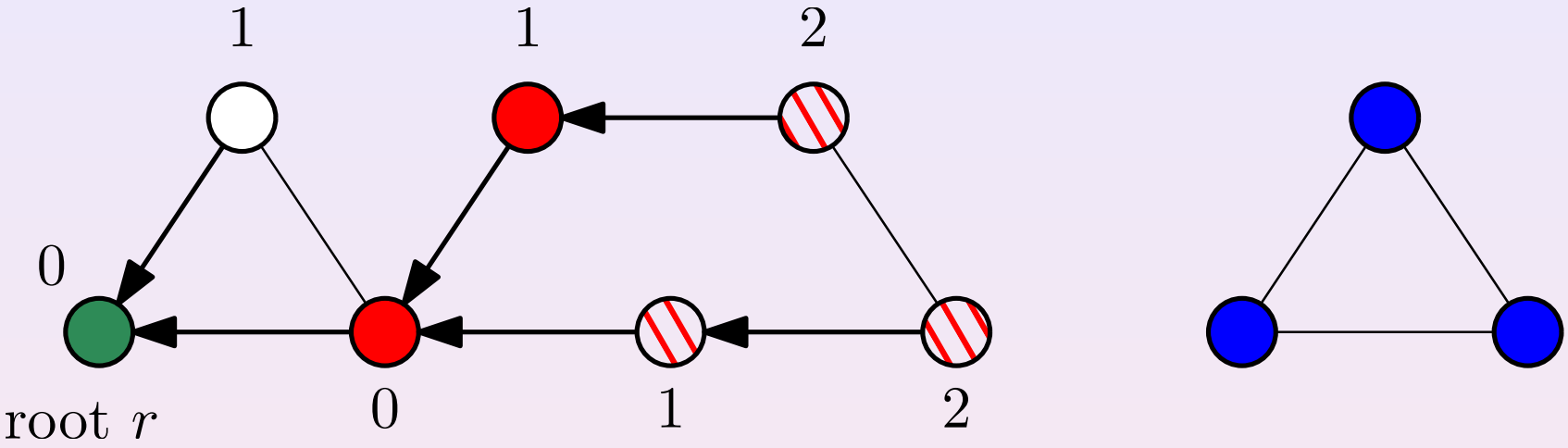
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

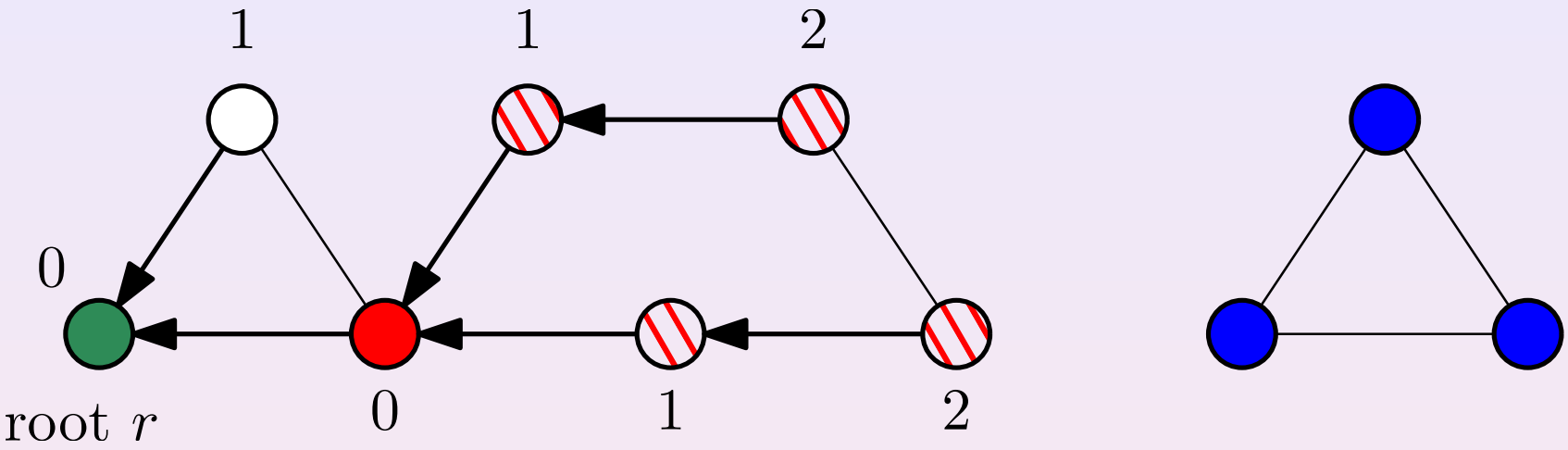
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

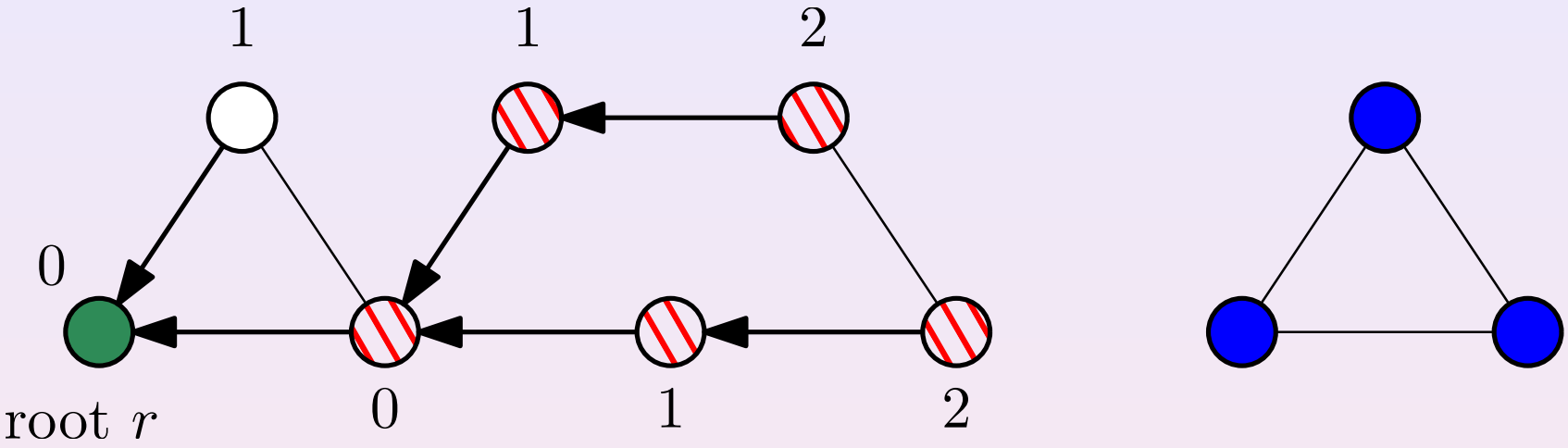
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

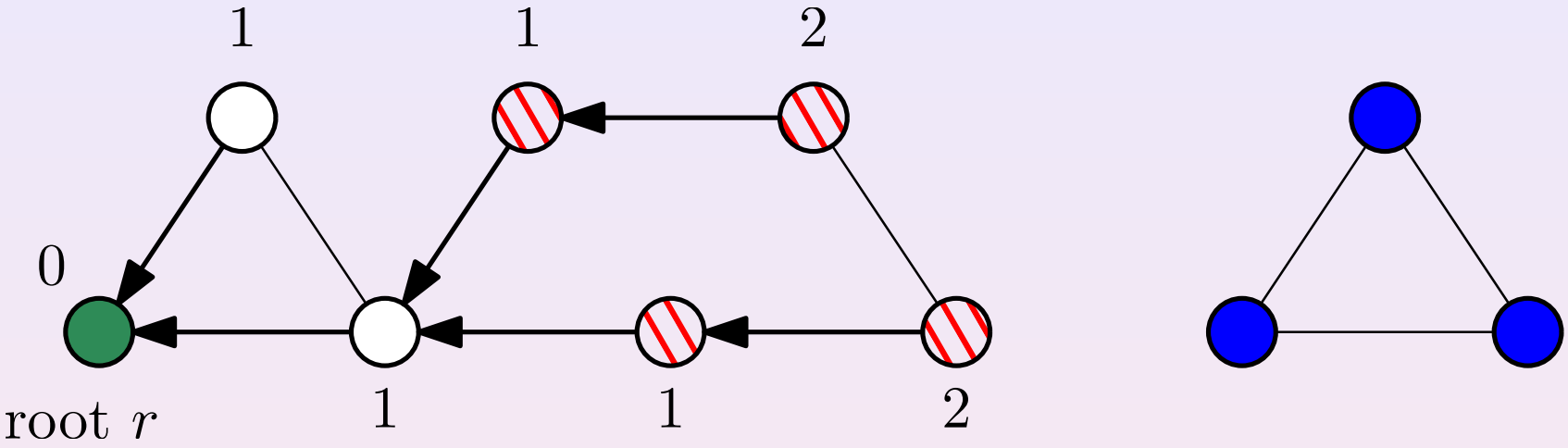
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

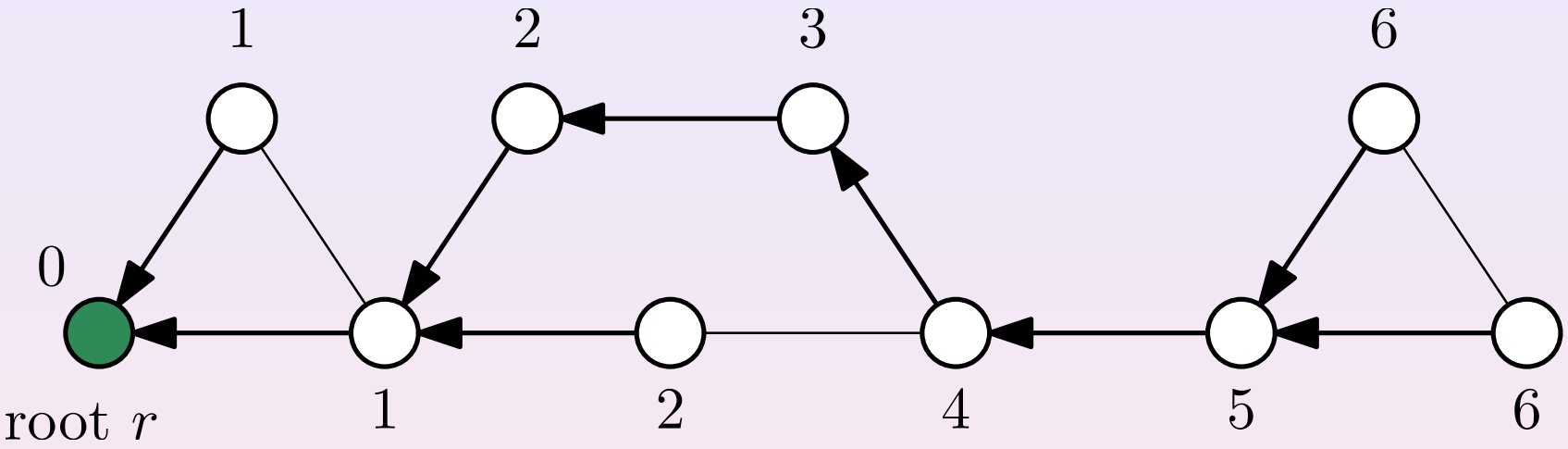
Pretty cautious. . .



1 wave (1 error state) gives a correct but exponential algorithm
[Glacet, Hanusse, I., Johnen, 2014]

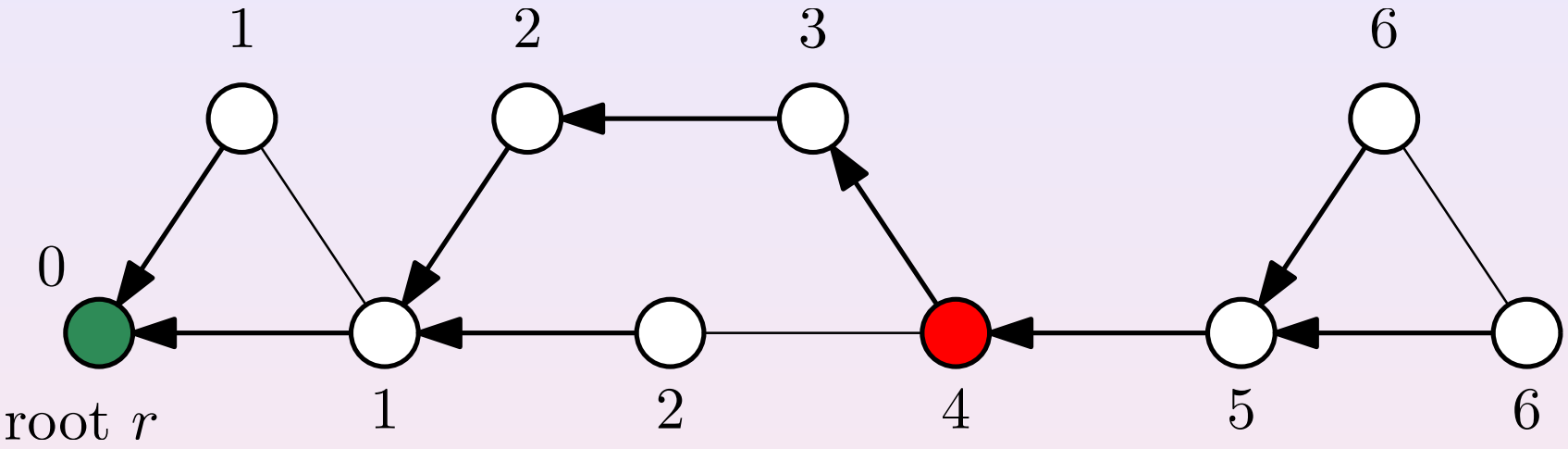
Two waves allow a polynomial number of steps.
[Devismes, I., Johnen, 2016]

. . . but not too much



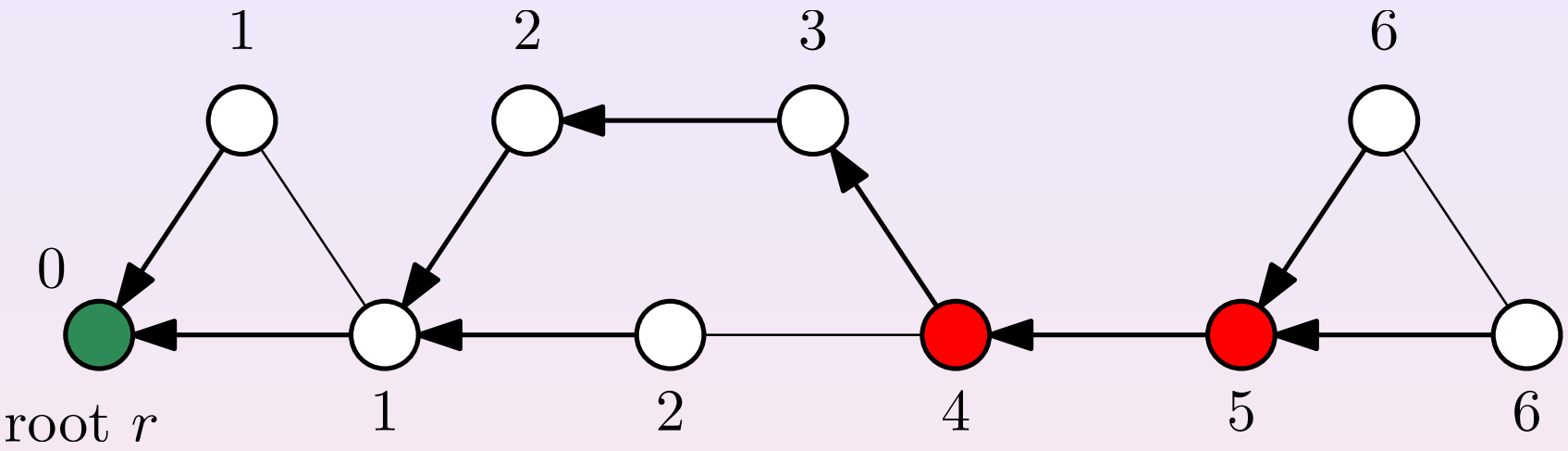
Not directly improving distances leads to a quadratic number of rounds, instead of linear for the previous two algorithms.

. . . but not too much



Not directly improving distances leads to a quadratic number of rounds, instead of linear for the previous two algorithms.

. . . but not too much



Not directly improving distances leads to a quadratic number of rounds, instead of linear for the previous two algorithms.

Formal algorithm

Algorithm 2: Code of RSP for any process $u \neq r$

Variables:

$st_u \in \{I, C, EB, EF\}$
$par_u \in Lbl$
$d_u \in \mathbb{N}^*$

Predicates:

$P\_reset(u) \equiv st_u = EF \wedge abRoot(u)$
$P\_correction(u) \equiv (\exists v \in \Gamma(u) \mid st_v = C \wedge d_v + \omega(u,v) < d_u)$

Macro:

$computePath(u) : par_u := \text{argmin}_{(v \in \Gamma(u) \wedge st_v = C)}(d_v + \omega(u,v));$
$d_u := d_{par_u} + \omega(u, par_u);$
$st_u := C$

Rules

$\mathbf{R_C}(u) : st_u = C \wedge P\_correction(u) \rightarrow computePath(u)$
$\mathbf{R_{EB}}(u) : st_u = C \wedge \neg P\_correction(u) \wedge (abRoot(u) \vee st_{par_u} = EB) \rightarrow st_u := EB$
$\mathbf{R_{EF}}(u) : st_u = EB \wedge (\forall v \in children(u) \mid st_v = EF) \rightarrow st_u := EF$
$\mathbf{R_I}(u) : P\_reset(u) \wedge (\forall v \in \Gamma(u) \mid st_v \neq C) \rightarrow st_u := I$
$\mathbf{R_R}(u) : (P\_reset(u) \vee st_u = I) \wedge (\exists v \in \Gamma(u) \mid st_v = C) \rightarrow computePath(u)$

Length of the proofs: approximately 12 pages

Formal algorithm

Algorithm 2: Code of RSP for any process $u \neq r$

Variables:

$st_u \in \{I, C, EB, EF\}$
$par_u \in Lbl$
$d_u \in \mathbb{N}^*$

Predicates:

$P\_reset(u) \equiv st_u = EF \wedge abRoot(u)$
$P\_correction(u) \equiv (\exists v \in \Gamma(u) \mid st_v = C \wedge d_v + \omega(u,v) < d_u)$

Macro:

$computePath(u) : par_u := \text{argmin}_{(v \in \Gamma(u) \wedge st_v = C)}(d_v + \omega(u,v));$
$d_u := d_{par_u} + \omega(u, par_u);$
$st_u := C$

Rules

$\mathbf{R_C}(u) : st_u = C \wedge P\_correction(u) \rightarrow computePath(u)$
$\mathbf{R_{EB}}(u) : st_u = C \wedge \neg P\_correction(u) \wedge (abRoot(u) \vee st_{par_u} = EB) \rightarrow st_u := EB$
$\mathbf{R_{EF}}(u) : st_u = EB \wedge (\forall v \in children(u) \mid st_v = EF) \rightarrow st_u := EF$
$\mathbf{R_I}(u) : P\_reset(u) \wedge (\forall v \in \Gamma(u) \mid st_v \neq C) \rightarrow st_u := I$
$\mathbf{R_R}(u) : (P\_reset(u) \vee st_u = I) \wedge (\exists v \in \Gamma(u) \mid st_v = C) \rightarrow computePath(u)$

Length of the proofs: approximately 12 pages

More generally in distributed computing

Usually

- Rather small algorithms
- Importance of details
- Rather long proofs

Specificities in distributed computing

- Local algorithm vs. global behavior
- Non-determinism
  - Asynchrony
  - Dynamicity
  - Crashes
  - Byzantines
- Computing entities only have partial knowledge

More generally in distributed computing

Usually

- Rather small algorithms
- Importance of details
- Rather long proofs

Specificities in distributed computing

- Local algorithm vs. global behavior
- Non-determinism
  - Asynchrony
  - Dynamicity
  - Crashes
  - Byzantines
- Computing entities only have partial knowledge

Confidence becomes an issue

- More and more complex and/or tedious proofs
- Full version of DISC 2019 best paper: **221 pages**
- Conferences became more prestigious than journals, leading to less journal versions:

Year	2000	2005	2010	2015
Number of PODC papers	32	36	39	45
% having a journal version	66%	47%	36%	31%

(97% of the journal versions appear within 5 years)

	None	(0, 10]	(10, 25]	(25, 50]	(50, 75]	>75
4–10 years	4	6	4	6	3	1
> 10 years	0	4	7	12	14	5

(% of journal versions w.r.t. activity in the field)

(thanks to Christoph Lenzen and Yuval Emek for the data)

Confidence becomes an issue

- More and more complex and/or tedious proofs
- Full version of DISC 2019 best paper: **221 pages**
- Conferences became more prestigious than journals, leading to less journal versions:

Year	2000	2005	2010	2015
Number of PODC papers	32	36	39	45
% having a journal version	66%	47%	36%	31%

(97% of the journal versions appear within 5 years)

	None	(0, 10]	(10, 25]	(25, 50]	(50, 75]	>75
4–10 years	4	6	4	6	3	1
> 10 years	0	4	7	12	14	5

(% of journal versions w.r.t. activity in the field)

(thanks to Christoph Lenzen and Yuval Emek for the data)

Errare humanum est

Classical

- Holes/shortcuts in the reasoning
- Missing case(s)
- Multivariate Omega (to match $O(n \cdot m)$, use of a sparse graph to show an $\Omega(n \cdot m)$ lower bound)

More specific to D.C.

- Missing case(s) again!
  (because of the non-determinism)
- Imprecise model/algorithm (details really matter)
- Reasoning about knowledge ("need to know n")

Errare humanum est

Classical

- Holes/shortcuts in the reasoning
- Missing case(s)
- Multivariate Omega (to match $O(n \cdot m)$, use of a sparse graph to show an $\Omega(n \cdot m)$ lower bound)

More specific to D.C.

- Missing case(s) again!
  (because of the non-determinism)
- Imprecise model/algorithm (details really matter)
- Reasoning about knowledge ("need to know n")

Detecting and preventing issues

Possible research directions to improve confidence

- Distributed decision / Distributed verification
- Model-checking
- Certification via a proof assistant

Distributed decision/verification

Distributed local checking of the system at runtime:
a process can raise an alarm if needed

Distributed decision

- Based on local knowledge
- Typical example: node-coloring

Distributed verification

- Enhancing the local knowledge with a certificate
- Still local checking
- Typical example: rooted spanning tree

A whole research field behind these concepts:
See **surveys by L. Feuilloley and P. Fraigniaud**

Distributed decision/verification

Distributed local checking of the system at runtime:
a process can raise an alarm if needed

Distributed decision

- Based on local knowledge
- Typical example: node-coloring

Distributed verification

- Enhancing the local knowledge with a certificate
- Still local checking
- Typical example: rooted spanning tree

A whole research field behind these concepts:
See **surveys by L. Feuilloley and P. Fraigniaud**

Distributed decision/verification

Distributed local checking of the system at runtime:
a process can raise an alarm if needed

Distributed decision

- Based on local knowledge
- Typical example: node-coloring

Distributed verification

- Enhancing the local knowledge with a certificate
- Still local checking
- Typical example: rooted spanning tree

A whole research field behind these concepts:
See **surveys by L. Feuilloley and P. Fraigniaud**

Formal verification / model-checking

Model-checking

Automated verification based on a formalization of

- the system model
- the algorithm
- the properties to be satisfied

Pros:

- Automatic verification
- If verification fails, generally gives a counter-example
- May sometimes be able to synthesize algorithms

Cons:

- Subject to decidability and tractability issues

Formal verification / model-checking

Model-checking

Automated verification based on a formalization of

- the system model
- the algorithm
- the properties to be satisfied

Pros:

- Automatic verification
- If verification fails, generally gives a counter-example
- May sometimes be able to synthesize algorithms

Cons:

- Subject to decidability and tractability issues

Formal verification / model-checking

Model-checking

Automated verification based on a formalization of

- the system model
- the algorithm
- the properties to be satisfied

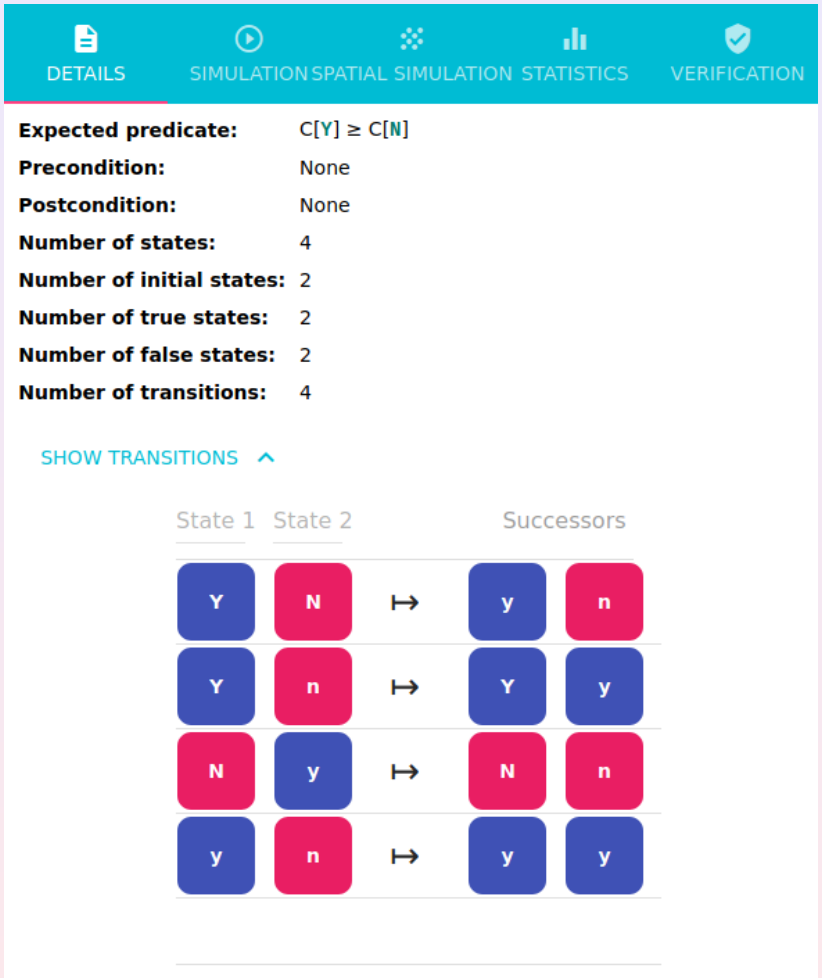
Pros:

- Automatic verification
- If verification fails, generally gives a counter-example
- May sometimes be able to synthesize algorithms

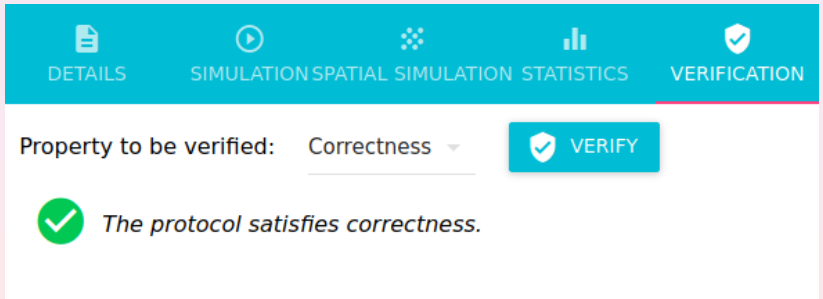
Cons:

- Subject to decidability and tractability issues

Peregrine – A tool for population protocols



- Population protocols: model in which anonymous agents interact stochastically
- Peregrine: Automated verification of protocols, and much more...



J. Esparza,'s team, TU Munich (Germany)

Certification

Proof assistant

- Also called Interactive theorem prover
- Checks whether a given proof of a given statement is correct
- Provides an interactive proof editor

Pros:

- More or less as powerful as paper proofs
- Provides some automation

Cons:

- Mainly manual
- Sometimes tedious even for simple proofs

Certification

Proof assistant

- Also called Interactive theorem prover
- Checks whether a given proof of a given statement is correct
- Provides an interactive proof editor

Pros:

- More or less as powerful as paper proofs
- Provides some automation

Cons:

- Mainly manual
- Sometimes tedious even for simple proofs

Certification

Proof assistant

- Also called Interactive theorem prover
- Checks whether a given proof of a given statement is correct
- Provides an interactive proof editor

Pros:

- More or less as powerful as paper proofs
- Provides some automation

Cons:

- Mainly manual
- Sometimes tedious even for simple proofs

Examples of French active projects in Coq

PACTOLE

- Framework for distributed computing by mobile robots
- Large variety of models and settings
- Both **positive** and **negative** results
- T. Balabonski, P. Courtieu, L. Rieg, S. Tixeuil, X. Urbain, ...

PADEC

- Framework for self-stabilizing algorithms
- Complex algorithms
- Complete analysis up to time complexities
- K. Altisen, P. Corbineau, S. Devismes

Examples of French active projects in Coq

PACTOLE

- Framework for distributed computing by mobile robots
- Large variety of models and settings
- Both **positive** and **negative** results
- T. Balabonski, P. Courtieu, L. Rieg, S. Tixeuil, X. Urbain, ...

PADEC

- Framework for self-stabilizing algorithms
- Complex algorithms
- Complete analysis up to time complexities
- K. Altisen, P. Corbineau, S. Devismes

Conclusion

Tending to more reliable research results

(Theoretical) research directions

- Write full version
- Submit full version (for reviewing)
- Specify the problem and the algorithm in a formal language
- Model-checking (at least particular cases)
- Prove in some proof assistant
- Develop new model-checker
- Develop libraries in proof assistant

A final note from L. Lamport

Quote about the Temporal Logic of Actions

TLA does have the following disadvantages:

- It can describe only a real algorithm, not a vague, incomplete sketch of an algorithm.
- You can specify an algorithm's correctness condition in TLA only if you understand what the algorithm is supposed to do.
- TLA makes it harder to cover gaps in a proof with handwaving.

Some researchers may find these drawbacks insurmountable.

By the way, Amazon started to use formal specification and model checking in 2011, year of a major disruption in their systems...

Thank you

for your attention