Self-Stabilizing Labeling and Ranking in Ordered Trees

Ajoy K. Datta[1] Stéphane Devismes[2]
Lawrence L. Larmore[1] Yvan Rivierre[2]

[1]School of Computer Science, University of Nevada Las Vegas, USA

[2]VERIMAG Laboratory, Université Joseph Fourier, Grenoble, France

UNLV

Verimag

Introduction

Overview

Two self-stabilizing algorithms
in ordered directed tree networks:

- Labeling processes, so to ease routing.
- Application: Ranking processes by weight.

Assumptions

Model of Computation:

- Shared memory model
- Read/write atomicity
- Asynchronous setting
- Weakly fair scheduler

Assumptions

Model of Computation:

- Shared memory model
- Read/write atomicity
- Asynchronous setting
- Weakly fair scheduler

Assumptions

Model of Computation:

- Shared memory model
- Read/write atomicity
- Asynchronous setting
- Weakly fair scheduler

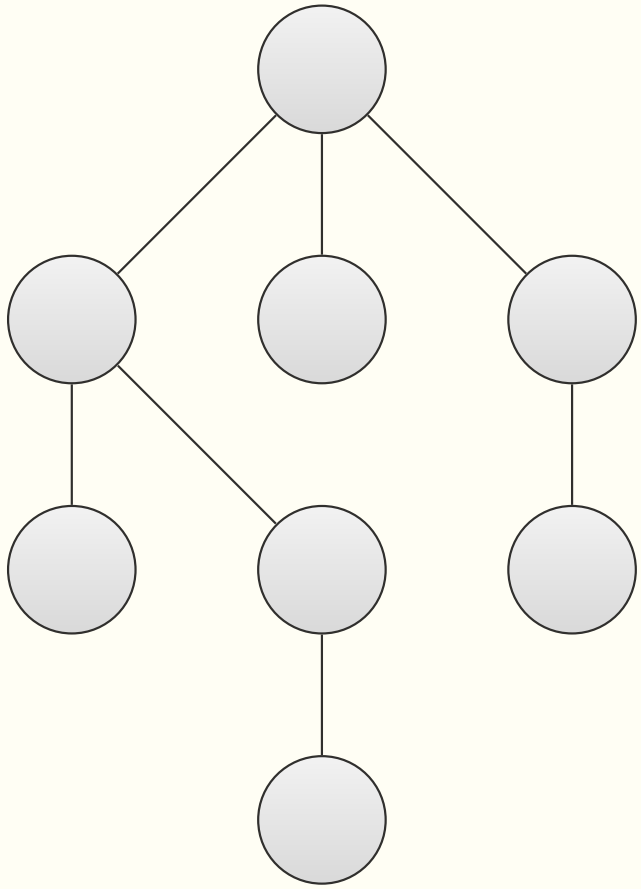
Topology:

- Ordered directed tree network

Topology

Ordered directed tree network:

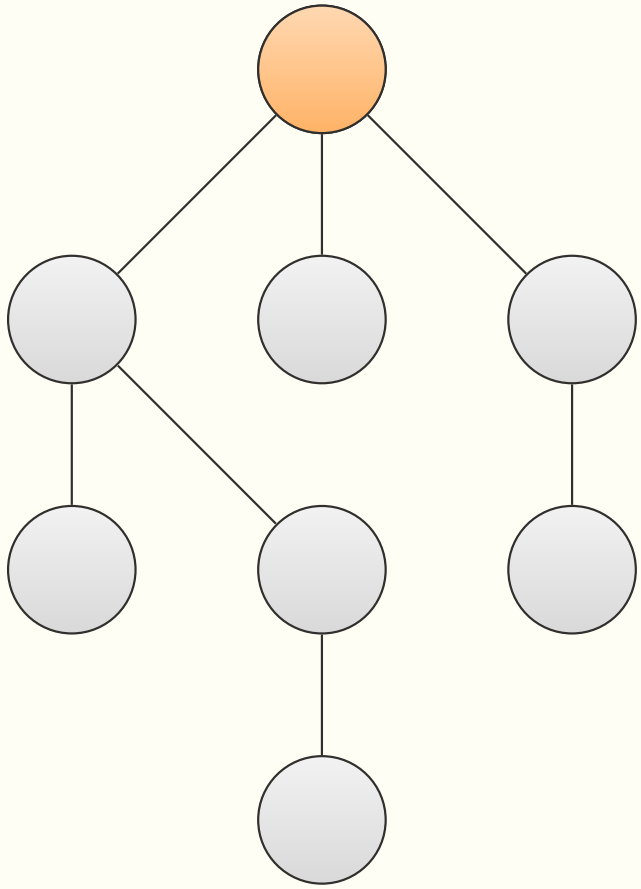
- tree
- rooted at some process
- directed toward the root
- left-to-right order on children



Topology

Ordered directed tree network:

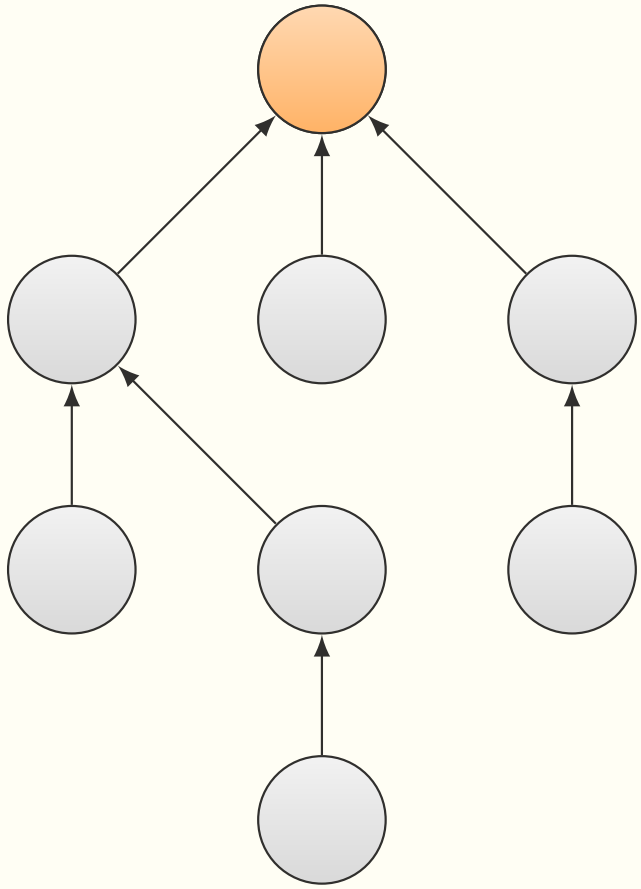
- tree
- rooted at some process
- directed toward the root
- left-to-right order on children



Topology

Ordered directed tree network:

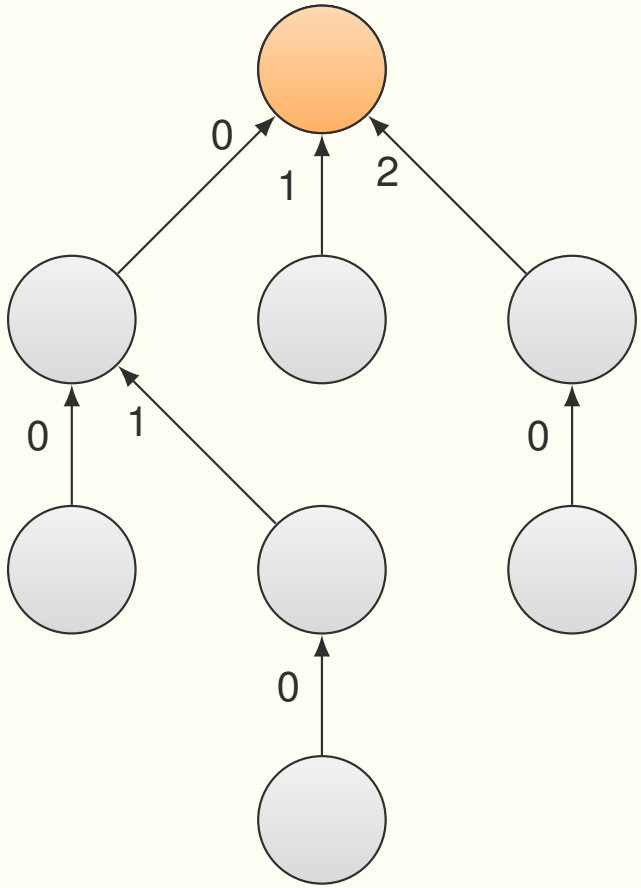
- tree
- rooted at some process
- directed toward the root
- left-to-right order on children



Topology

Ordered directed tree network:

- tree
- rooted at some process
- directed toward the root
- left-to-right order on children

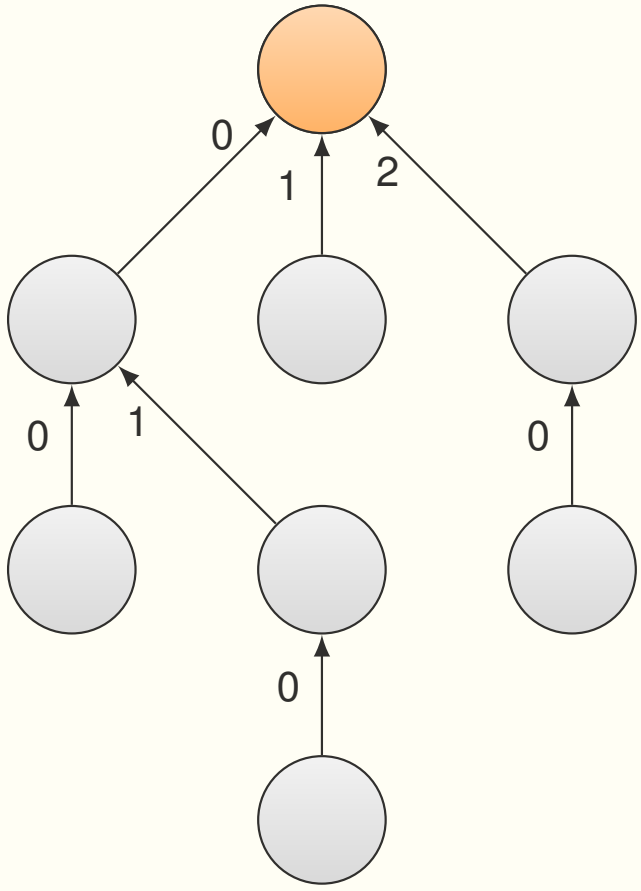


Topology

Ordered directed tree network:

- tree
- rooted at some process
- directed toward the root
- left-to-right order on children

Computable in arbitrary rooted network. *(e.g. Chen & al., 1991)*



Labeling with Guide Pairs

Guide Pair: Definition

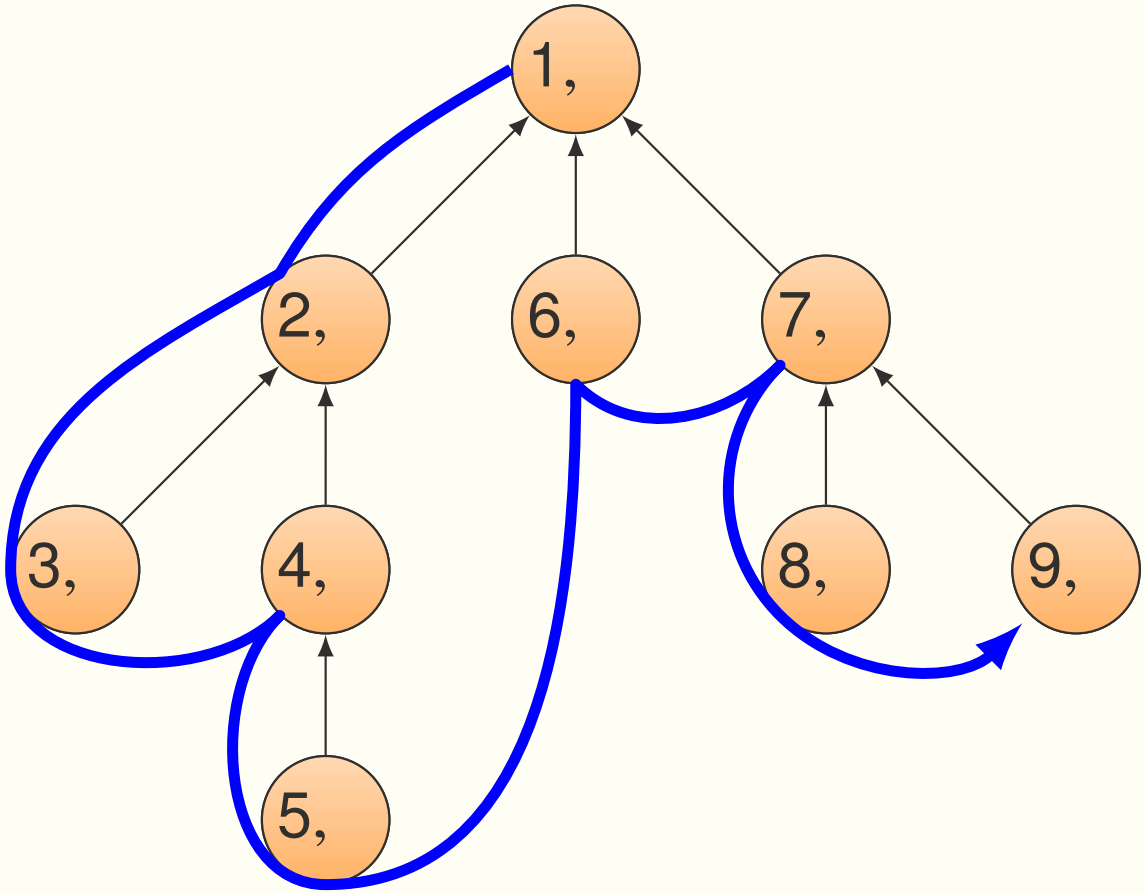
The guide pair of any process P is defined as $gp(P) = (i, j)$, where:

- i is the rank of P in the preorder traversal (left-to-right, top-down)
- j is the rank of P in the reverse postorder traversal (right-to-left, top-down)

Introduced by Flocchini & al., 2006.

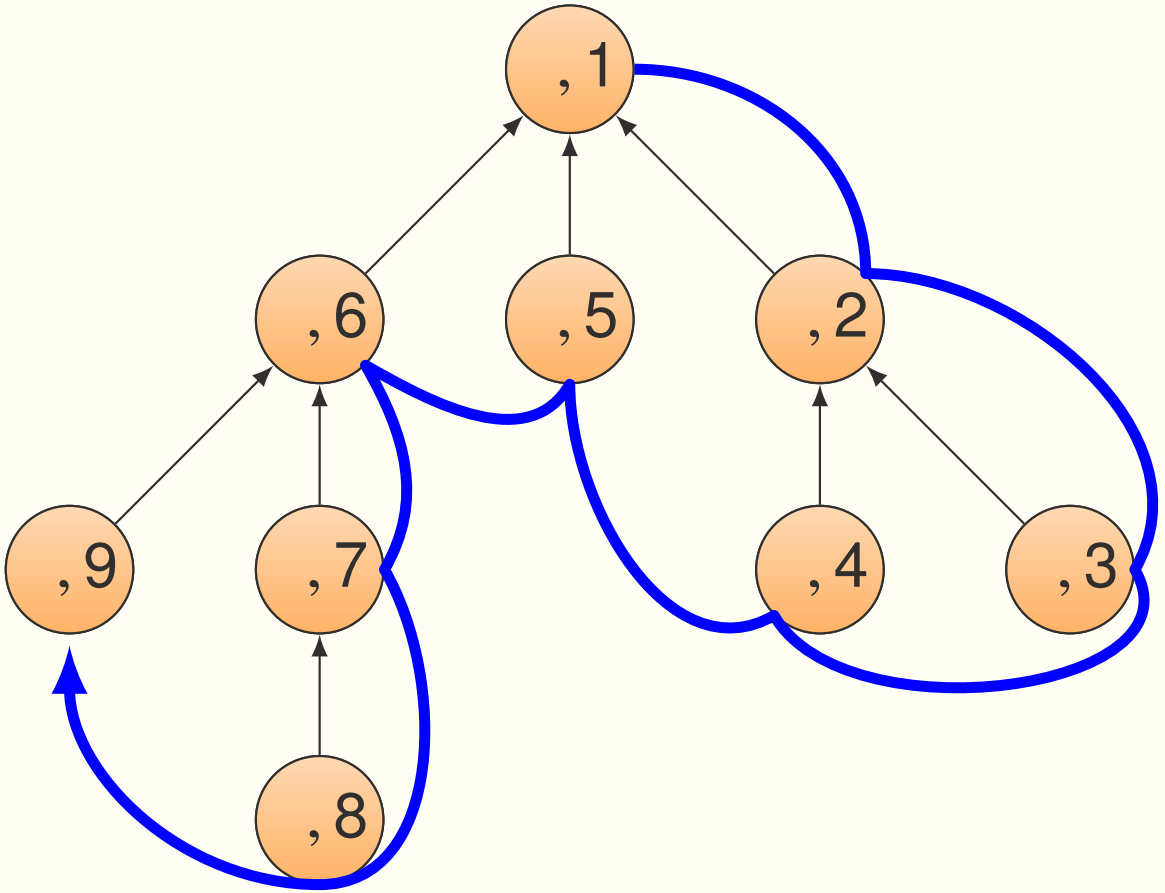
Guide Pair: First Term

$gp(P) = (i, j)$, where i is the preorder rank of P.



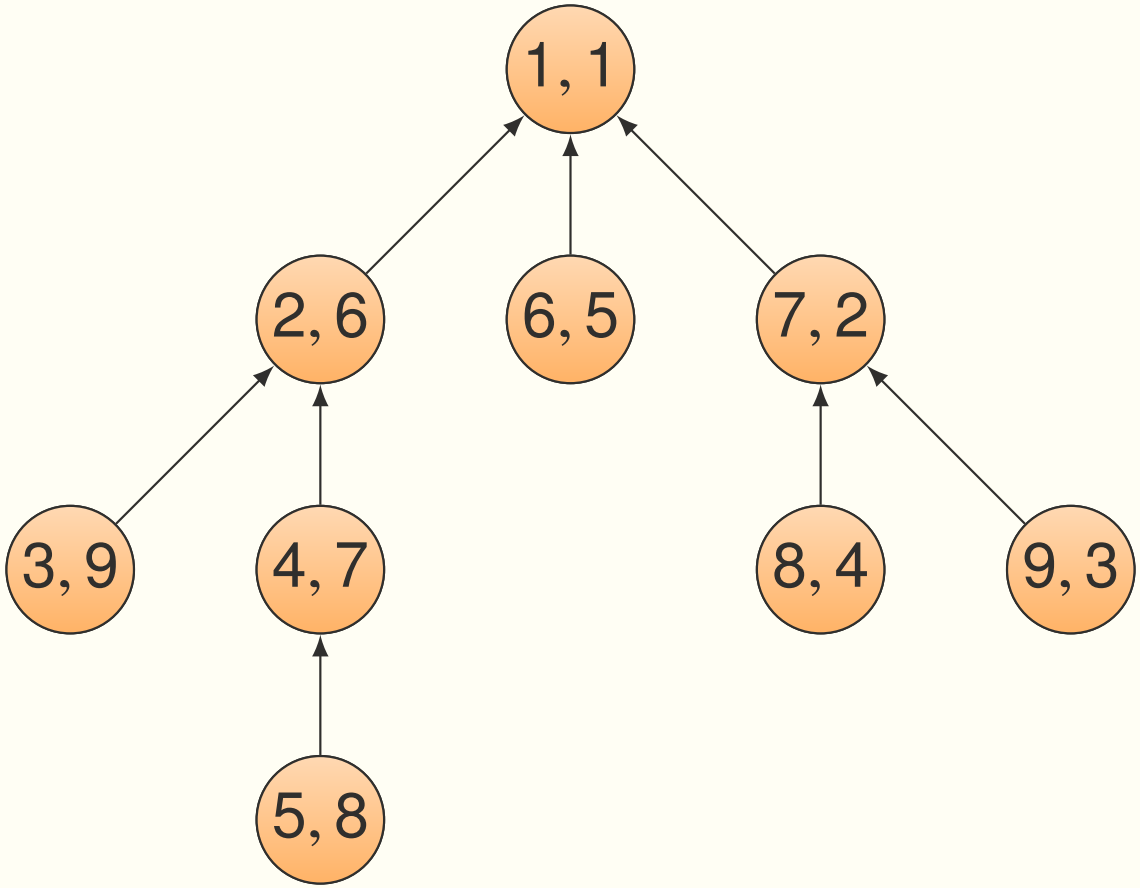
Guide Pair: Second Term

$gp(P) = (i, j)$, where j is the reverse postorder rank of P.



Guide Pair: Both Terms

$$gp(P) = (i = \text{preorder rank}, j = \text{reverse preorder rank})$$

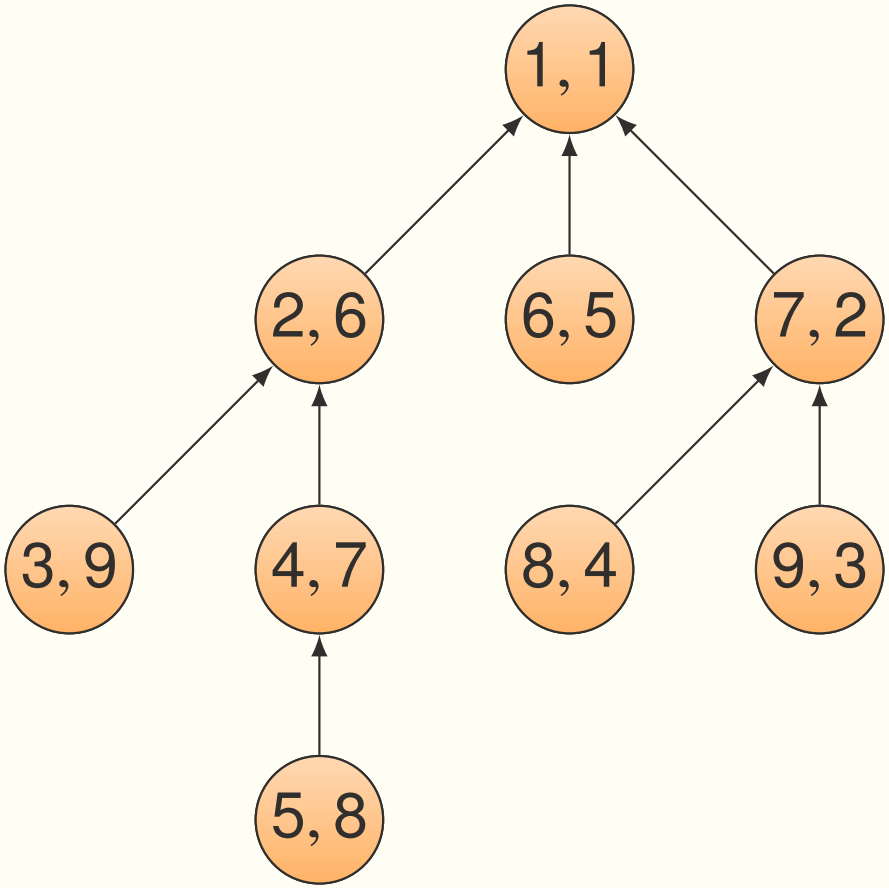


Guide Pair: Partial Order

We define a partial order on guide pairs as follows:

$$gp(P) \leq gp(Q) \equiv$$

$$i(P) \leq i(Q) \wedge j(P) \leq j(Q)$$

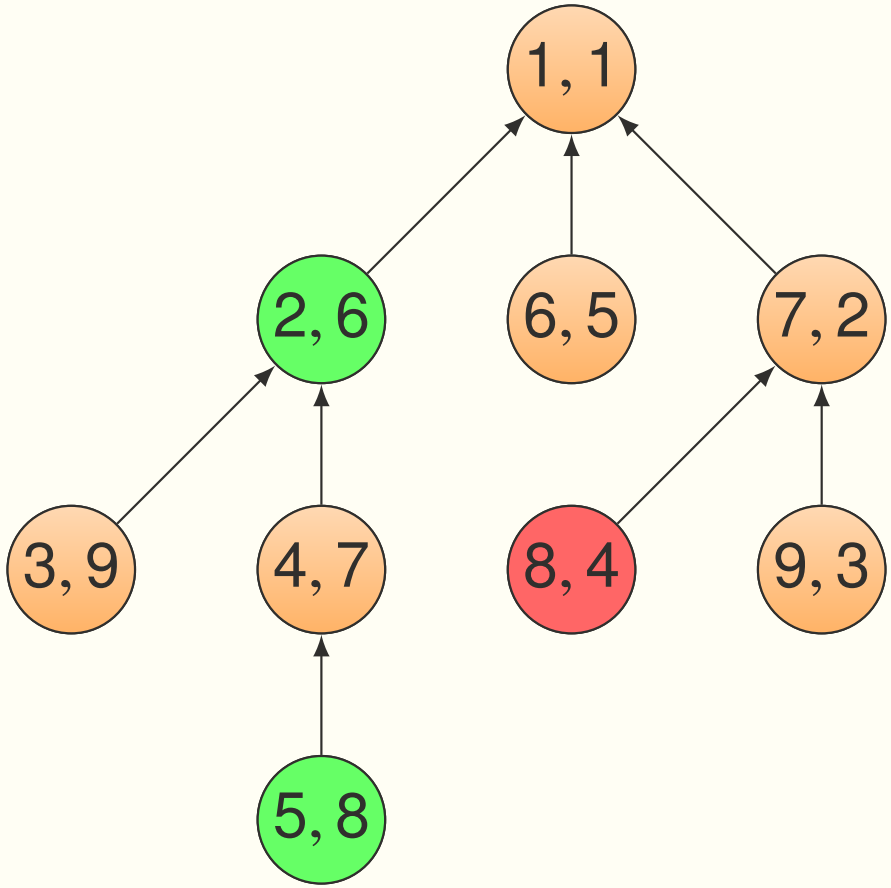


Guide Pair: Partial Order

We define a partial order on guide pairs as follows:

$$gp(P) \leq gp(Q) \equiv$$

$$i(P) \leq i(Q) \wedge j(P) \leq j(Q)$$



Guide Pair: Partial Order

We define a partial order on guide pairs as follows:

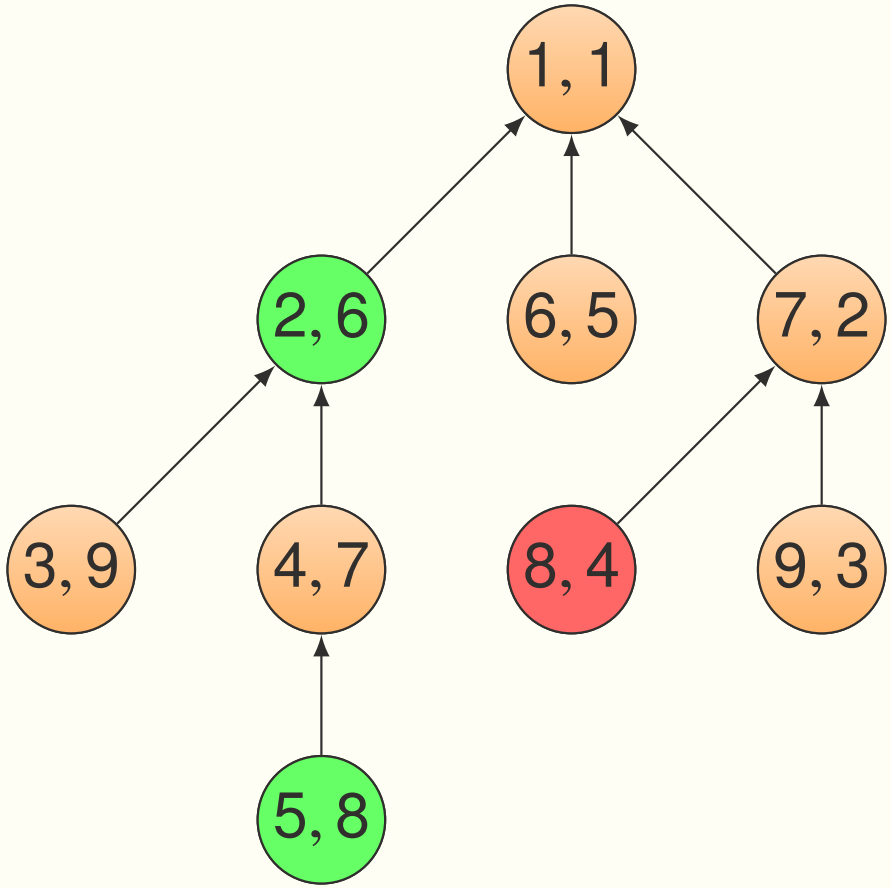
$$gp(P) \leq gp(Q) \equiv$$

$$i(P) \leq i(Q) \wedge j(P) \leq j(Q)$$

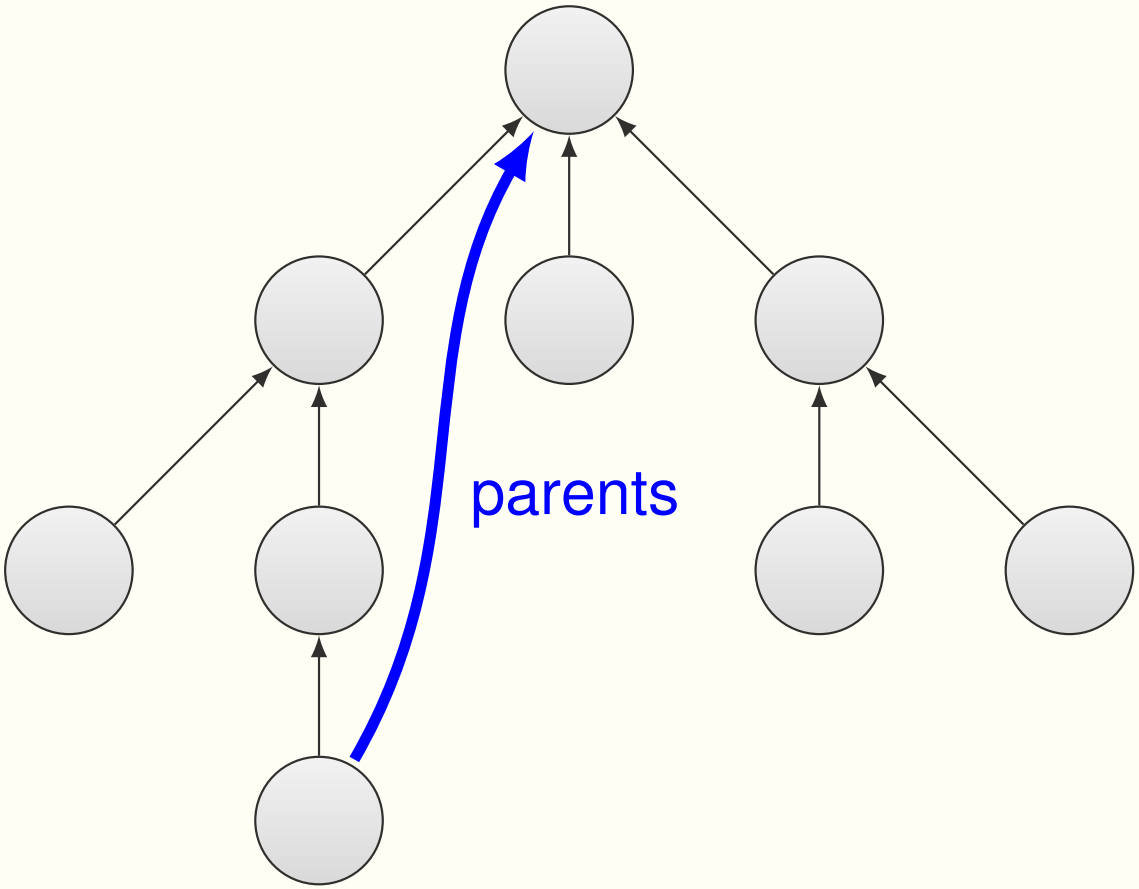
Note that:

$$gp(P) \leq gp(Q) \equiv$$

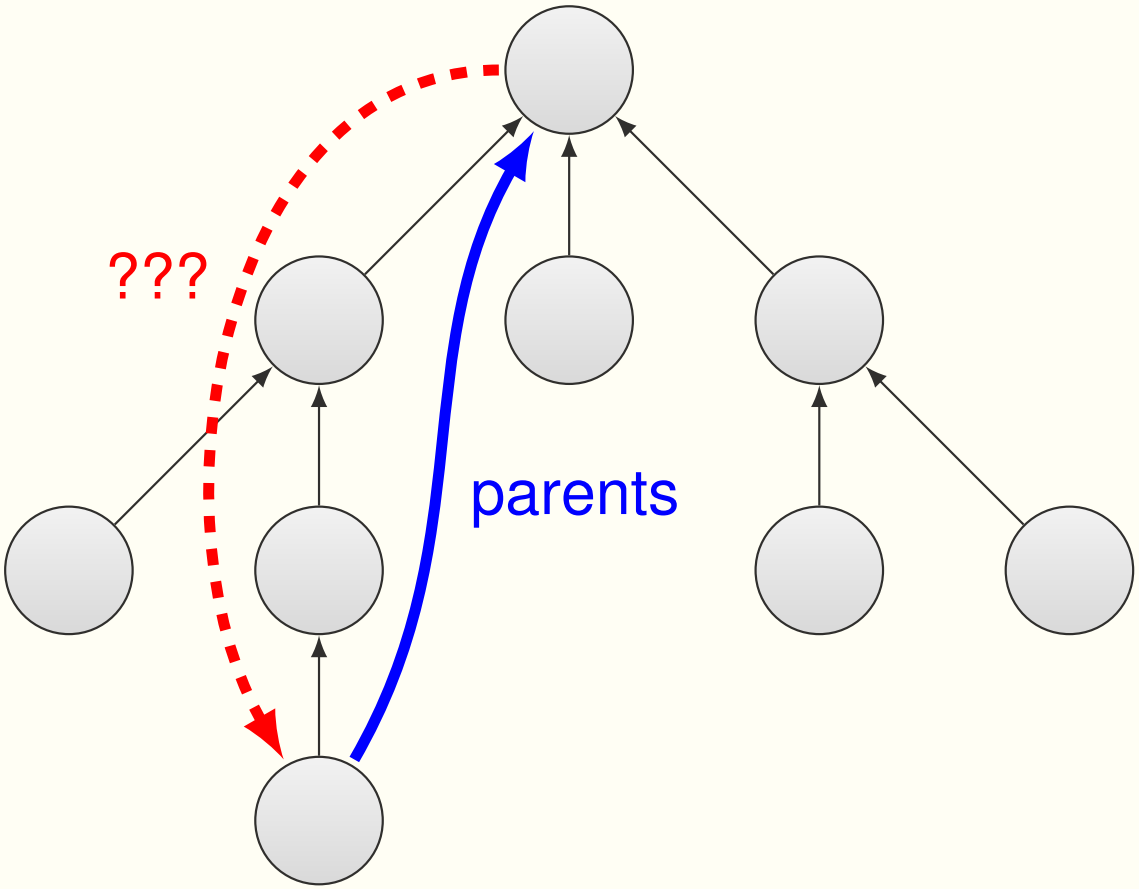
Q is a descendant of P



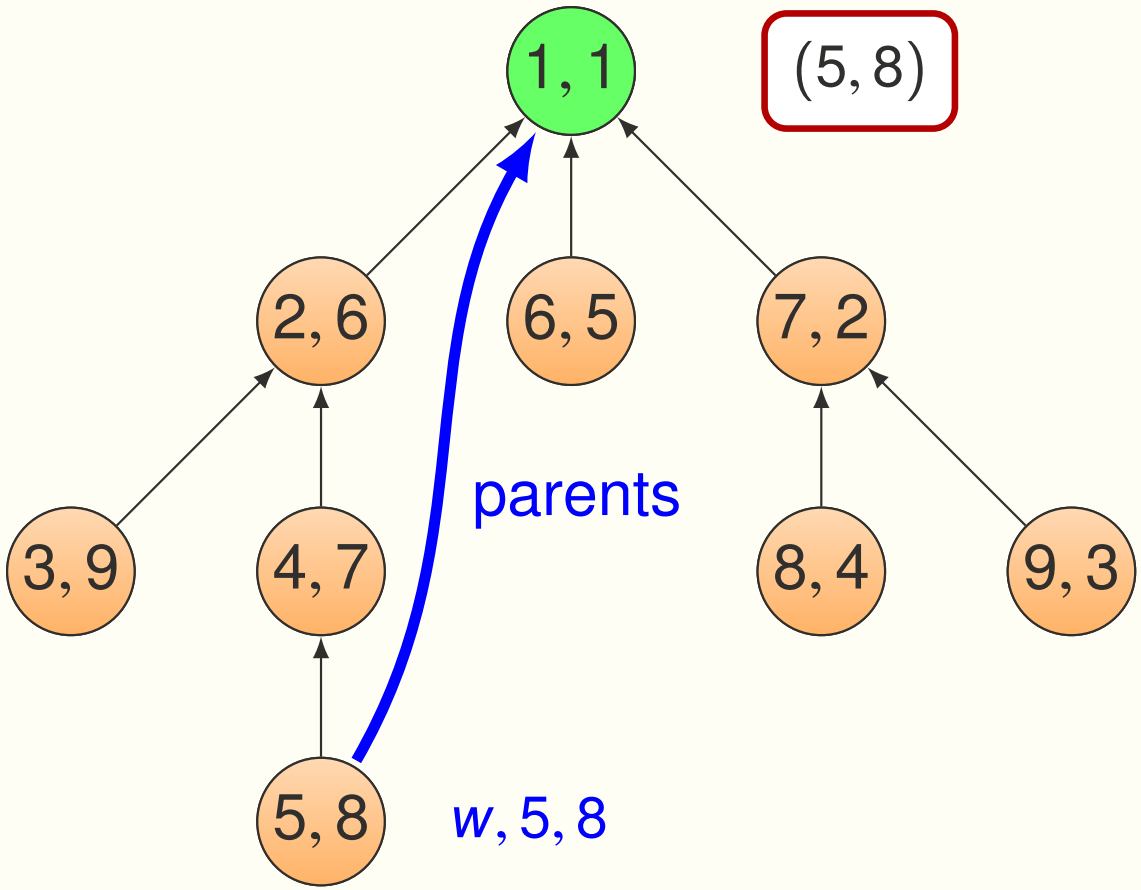
Guide Pair: Help for Communication



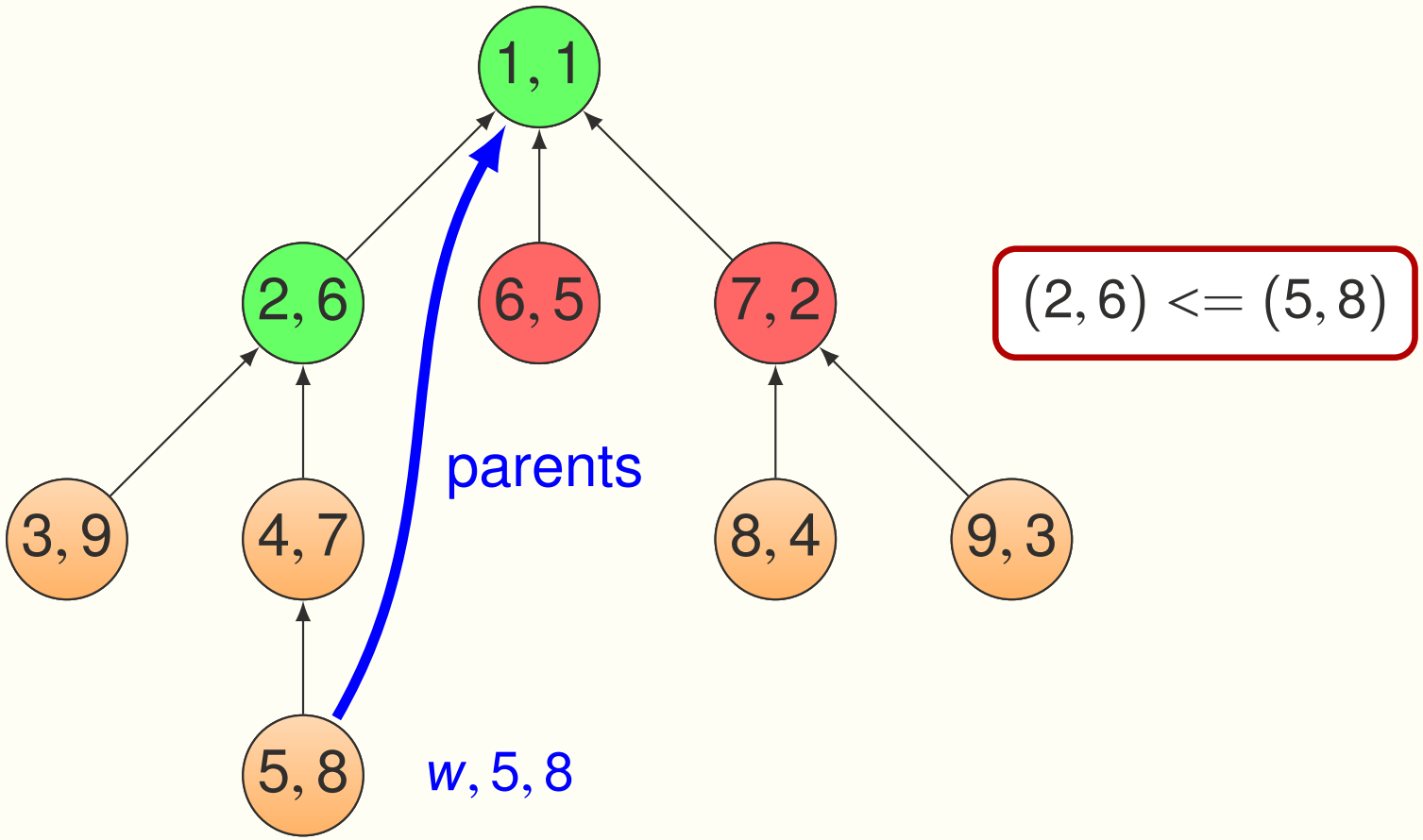
Guide Pair: Help for Communication



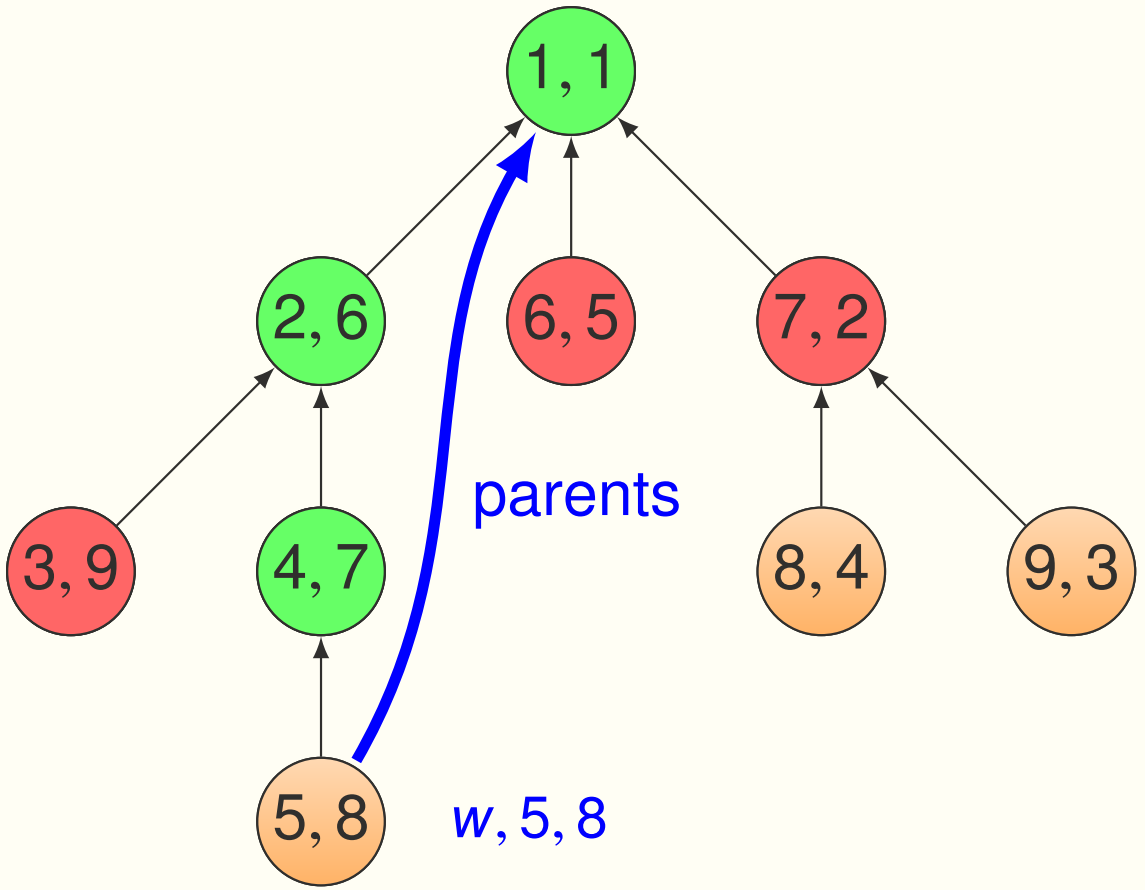
Guide Pair: Help for Communication



Guide Pair: Help for Communication

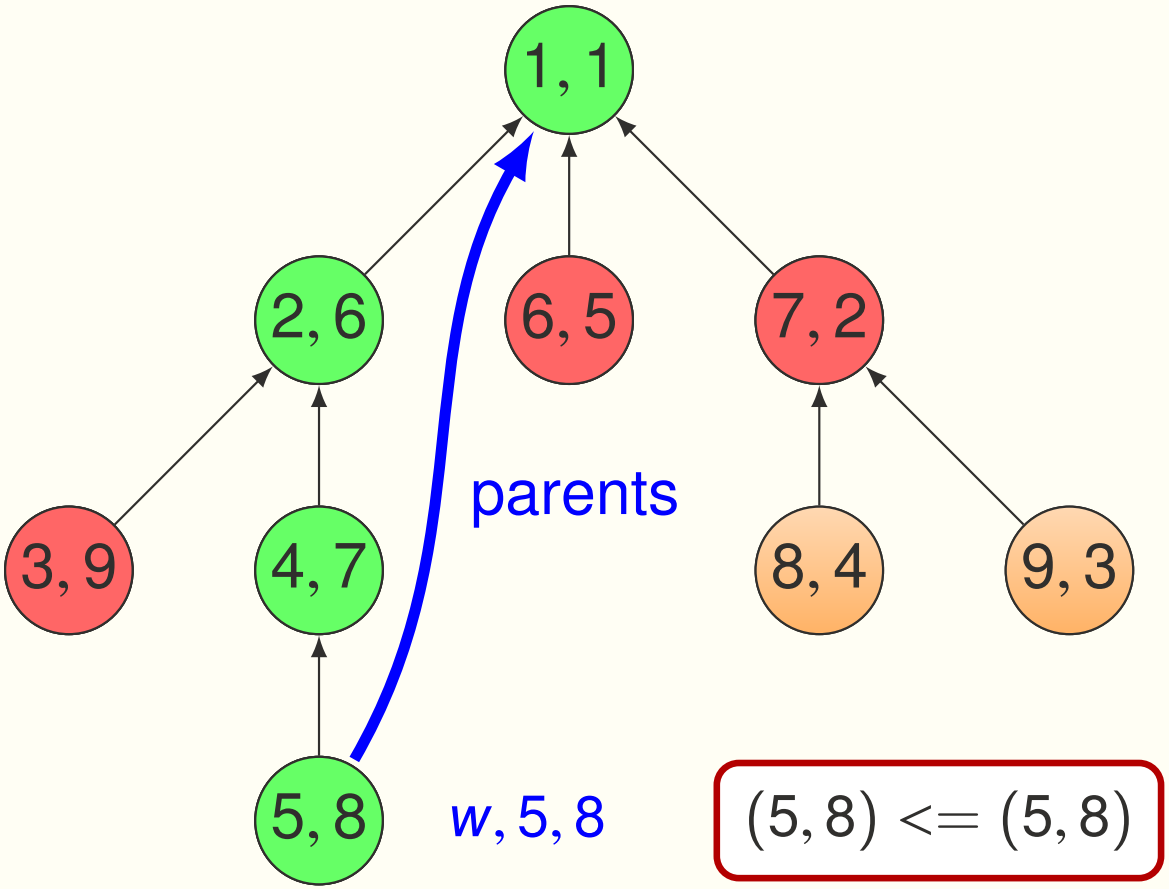


Guide Pair: Help for Communication

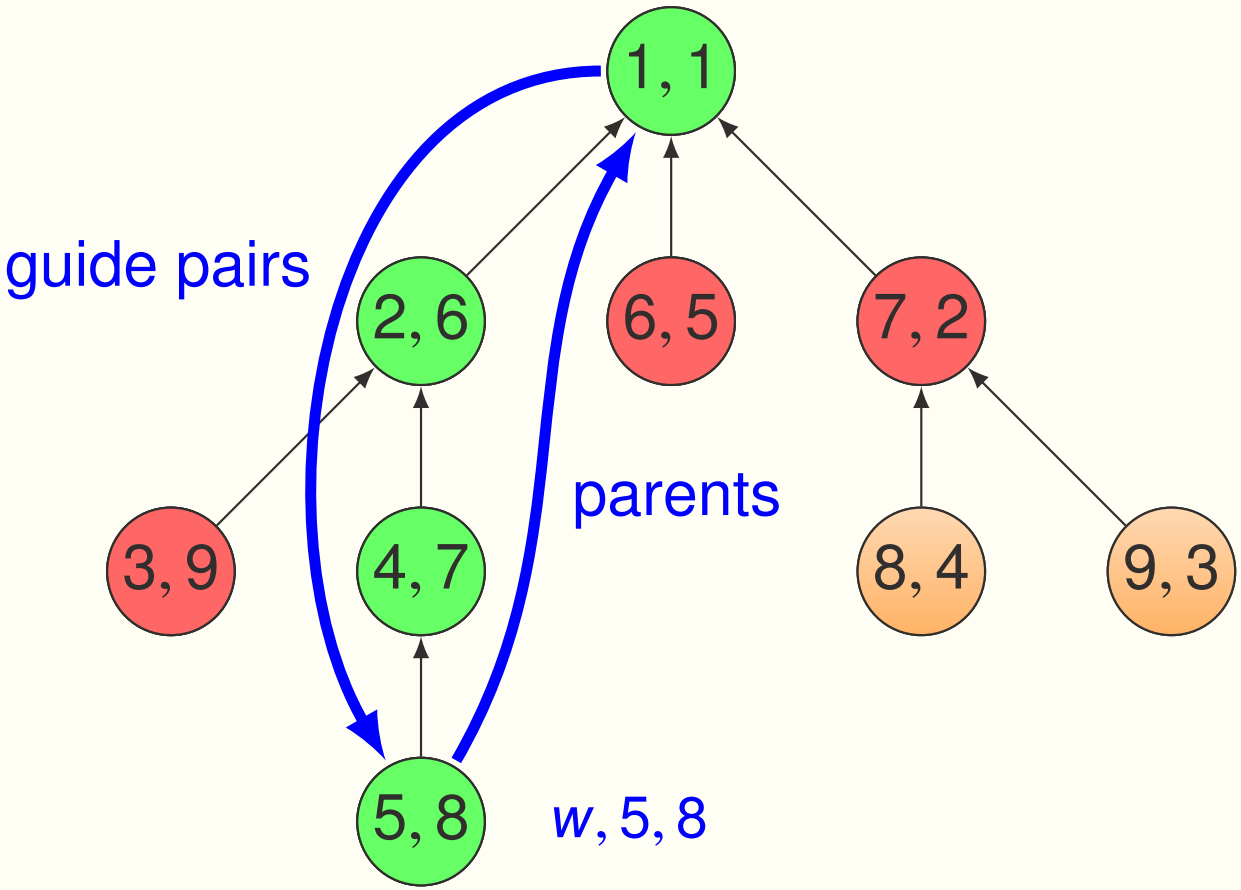


$(4, 7) <= (5, 8)$

Guide Pair: Help for Communication



Guide Pair: Help for Communication



Bidirectional communication between root and non-root.

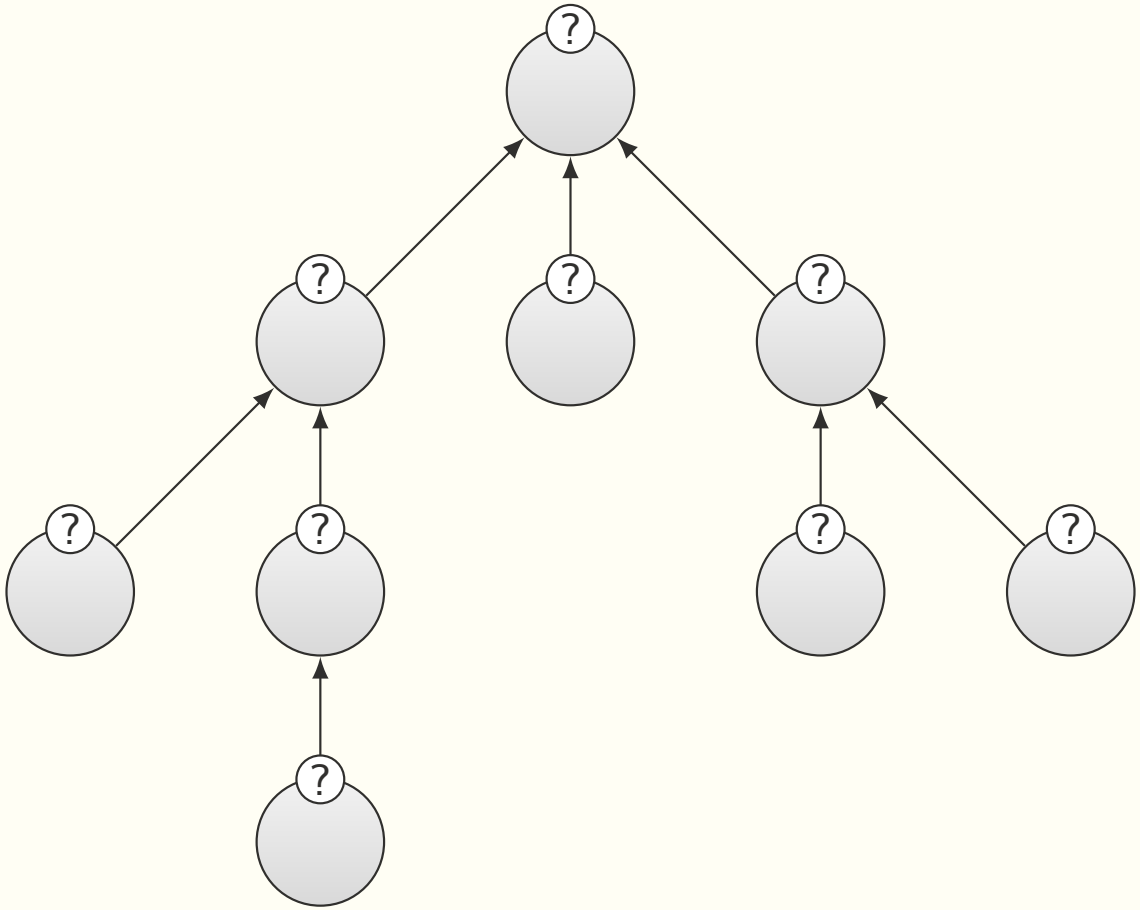
GUIDE Algorithm

Composition of two modules:

- Counting processes by subtrees (bottom-up).
- Labeling based on those counts (top-down).

GUIDE: Counting

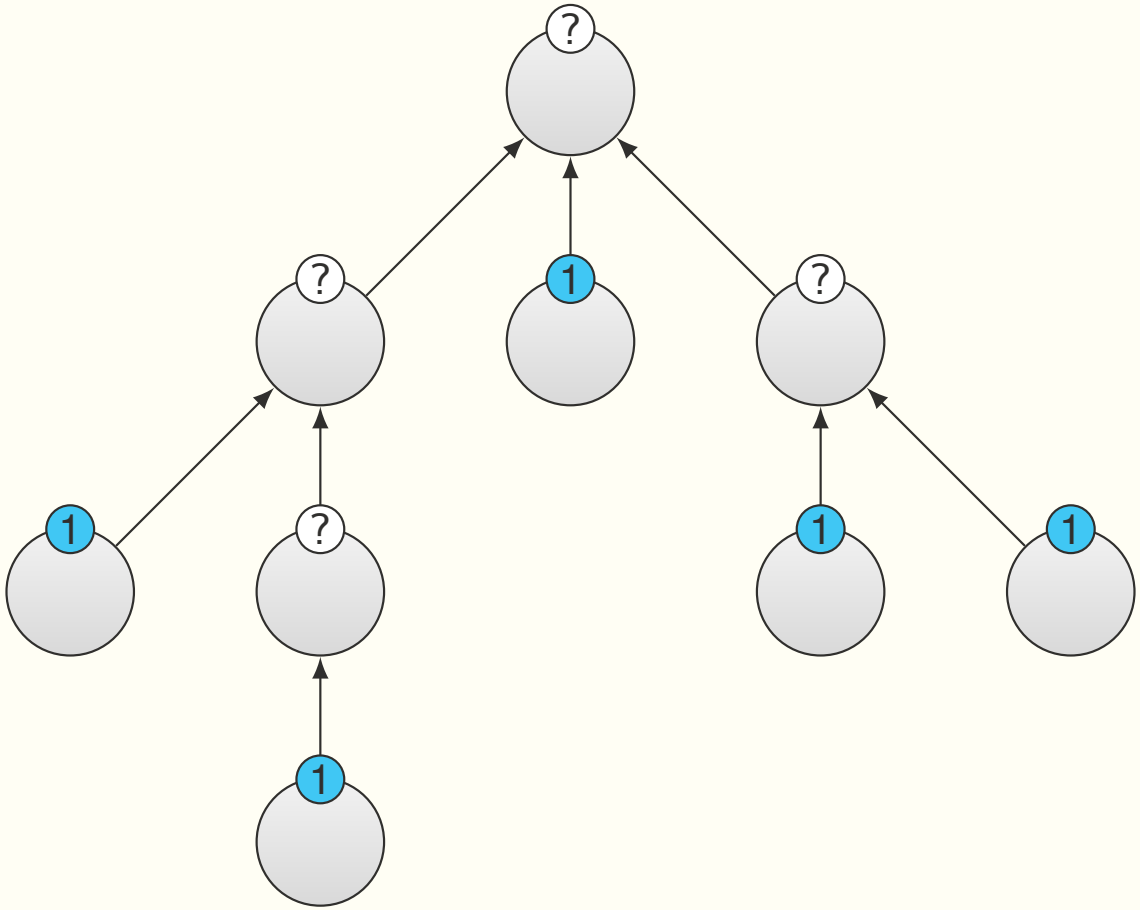
Count processes in each subtree, in $O(h)$ rounds.



$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Counting

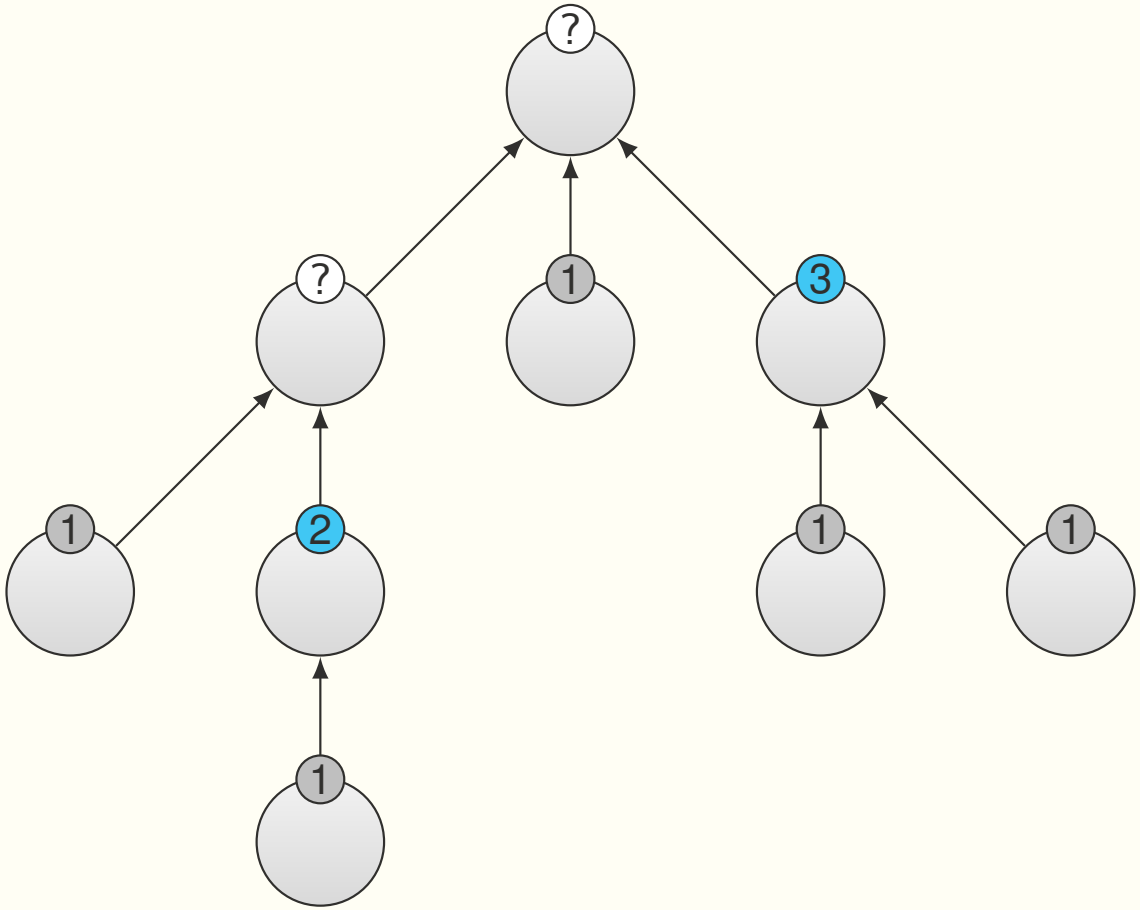
Count processes in each subtree, in $O(h)$ rounds.



$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Counting

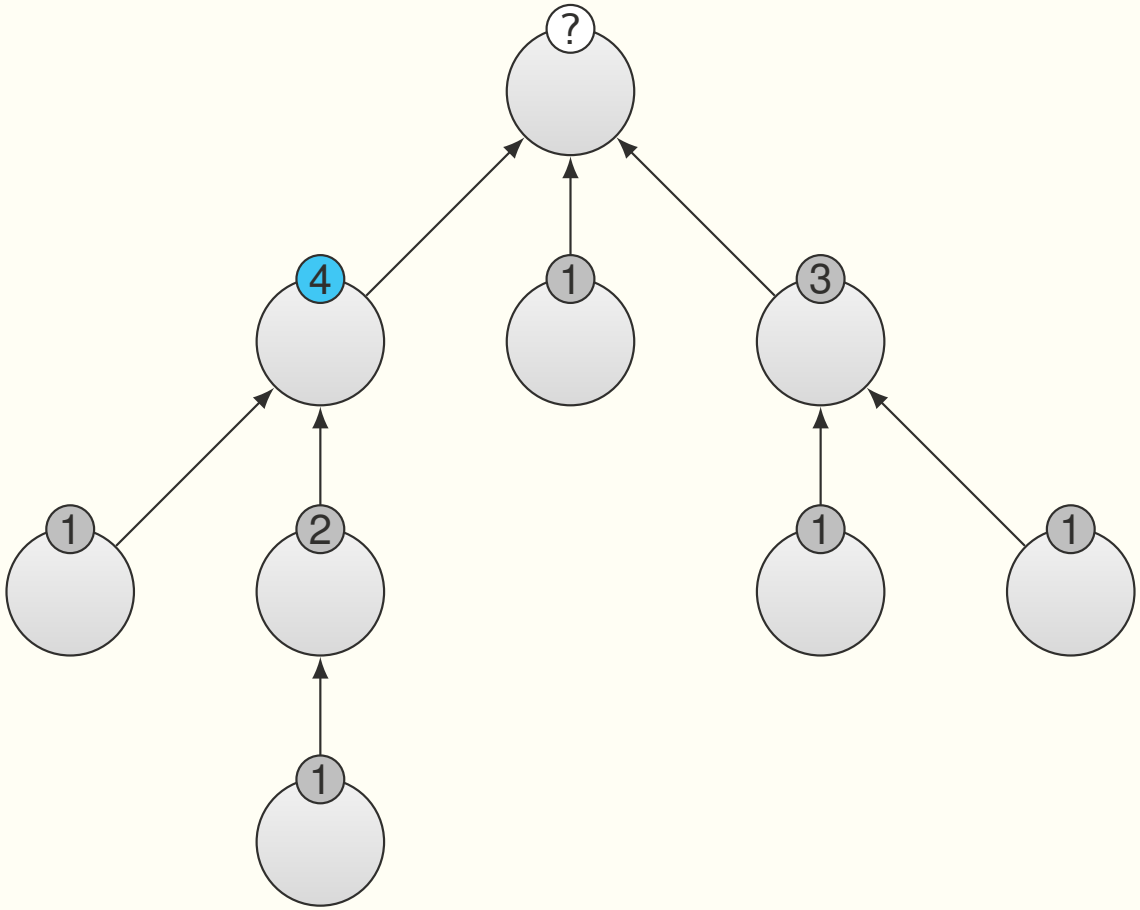
Count processes in each subtree, in $O(h)$ rounds.



$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Counting

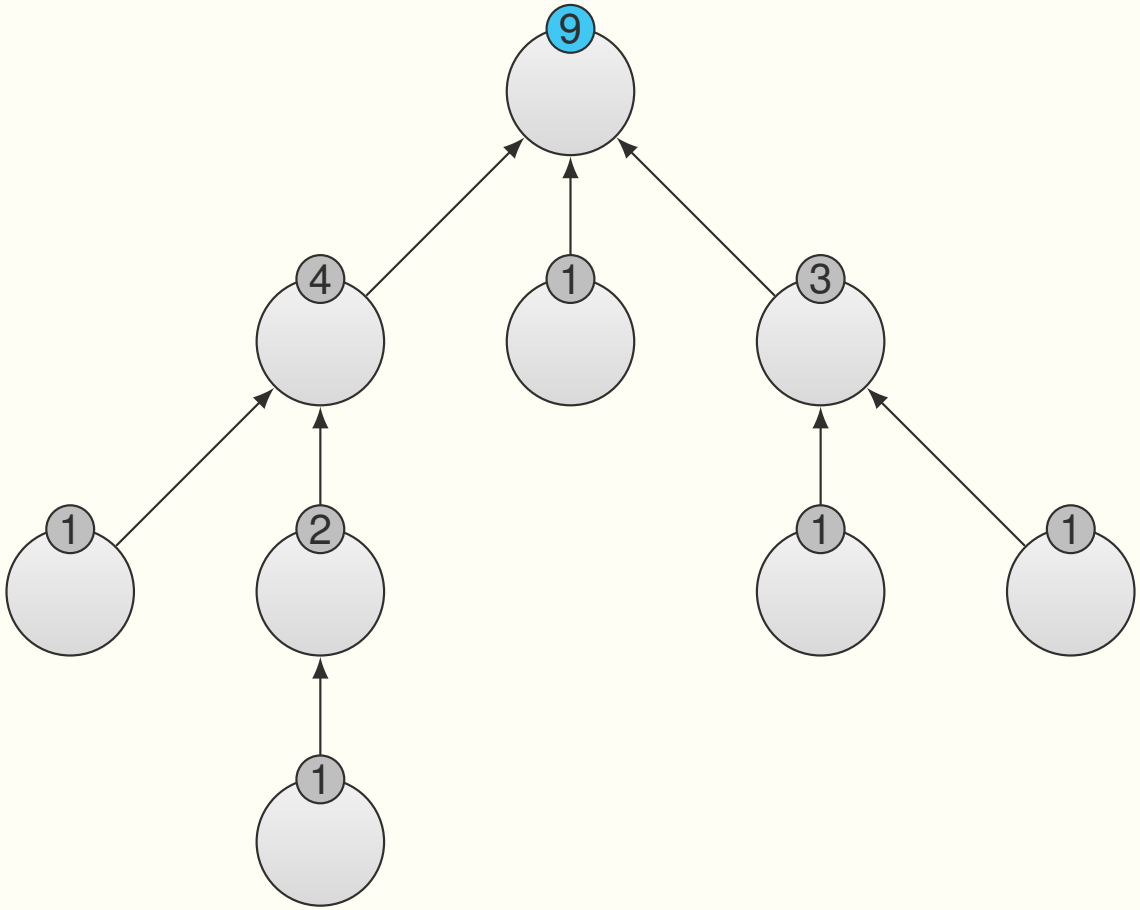
Count processes in each subtree, in $O(h)$ rounds.



$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Counting

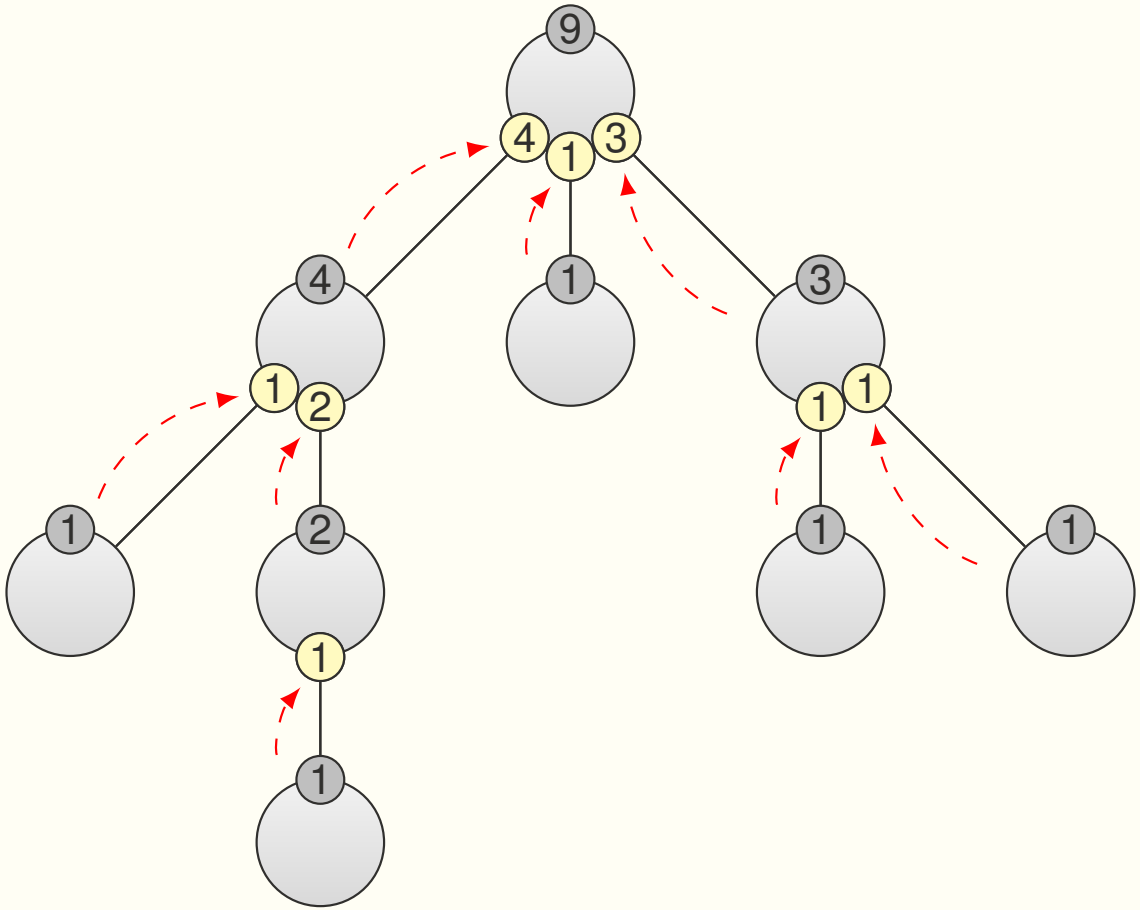
Count processes in each subtree, in $O(h)$ rounds.



$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Labeling

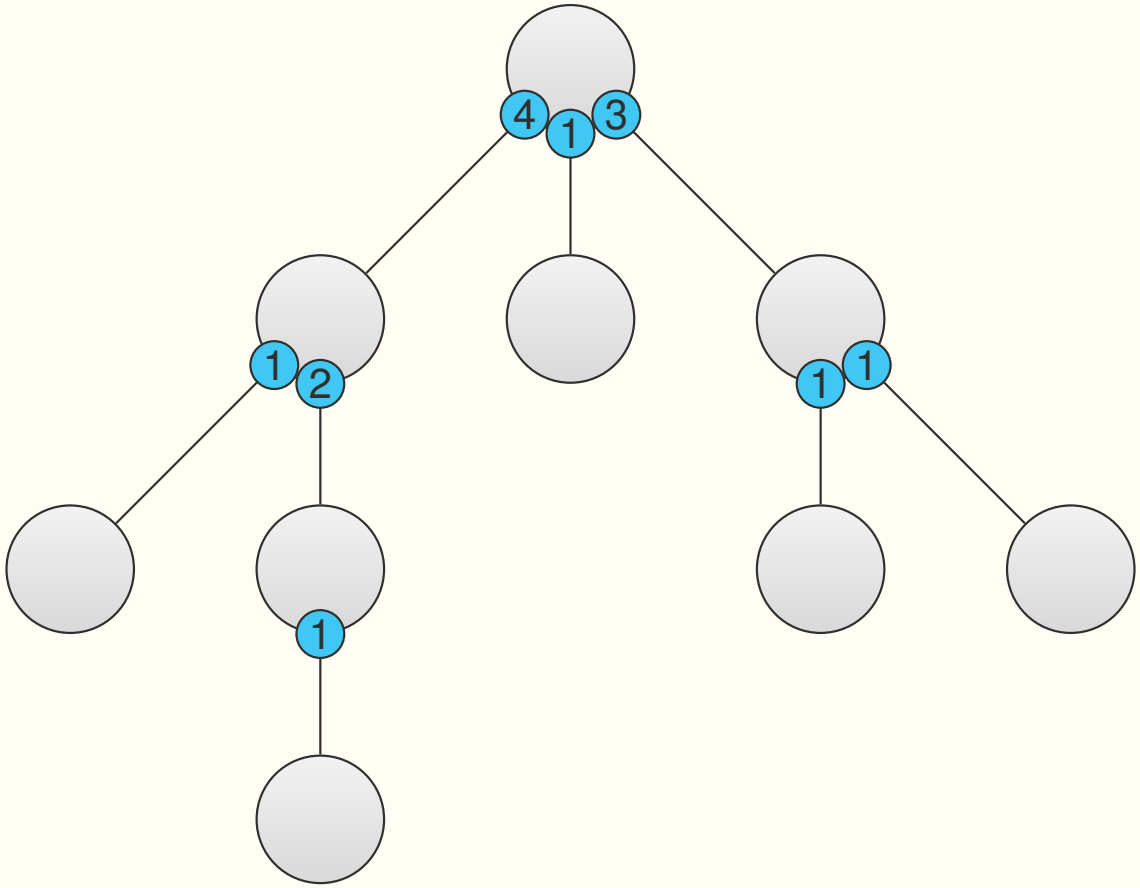
Preliminary: Copy counts from children.



$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Labeling

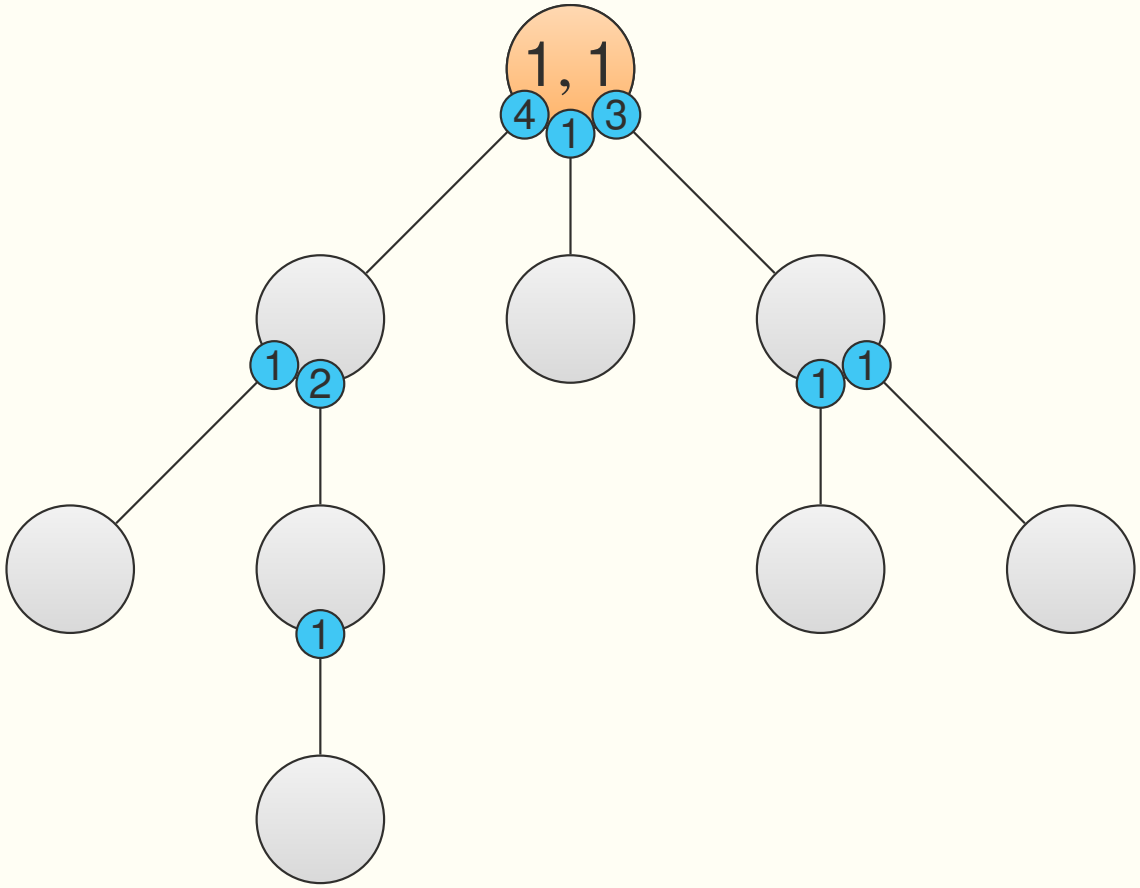
Compute guide pairs from counts, in $O(h)$ rounds.



$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Labeling

Compute guide pairs from counts, in $O(h)$ rounds.

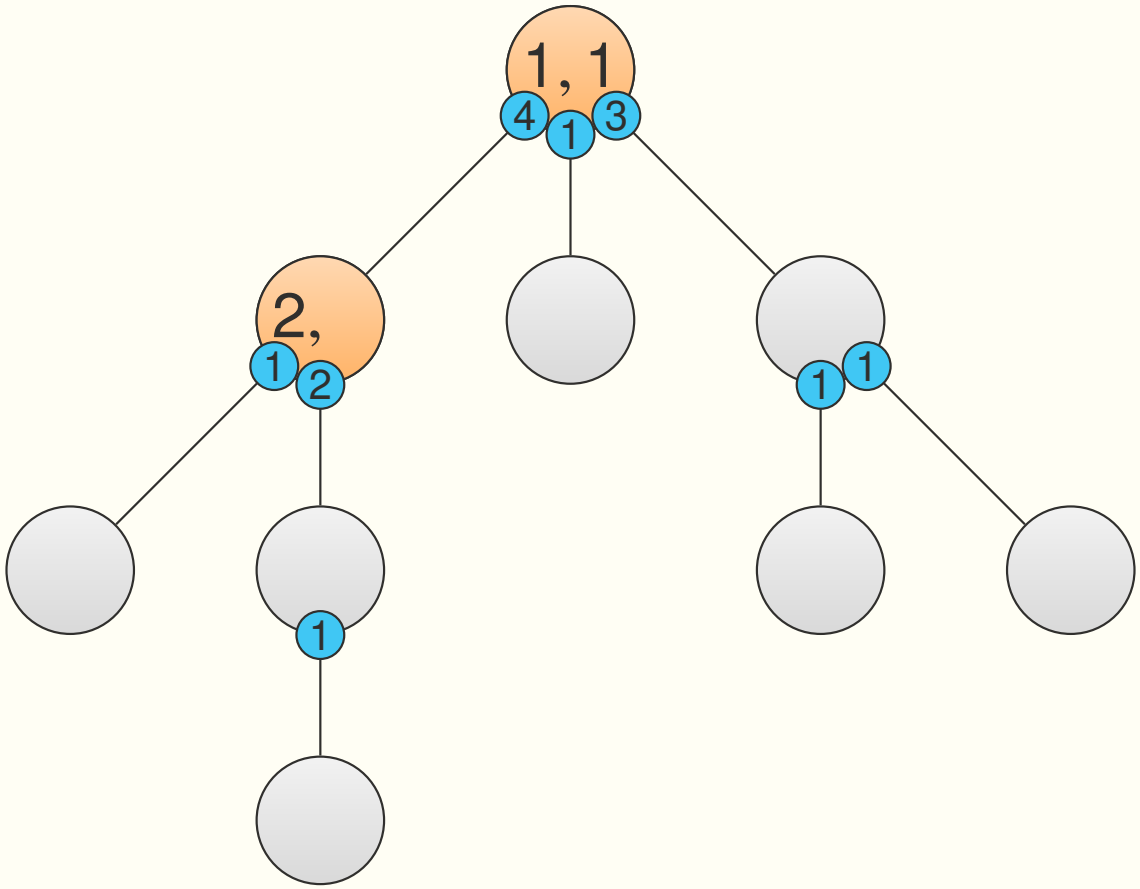


$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Labeling

Compute guide pairs from counts, in $O(h)$ rounds.

$1 + 1 = 2$

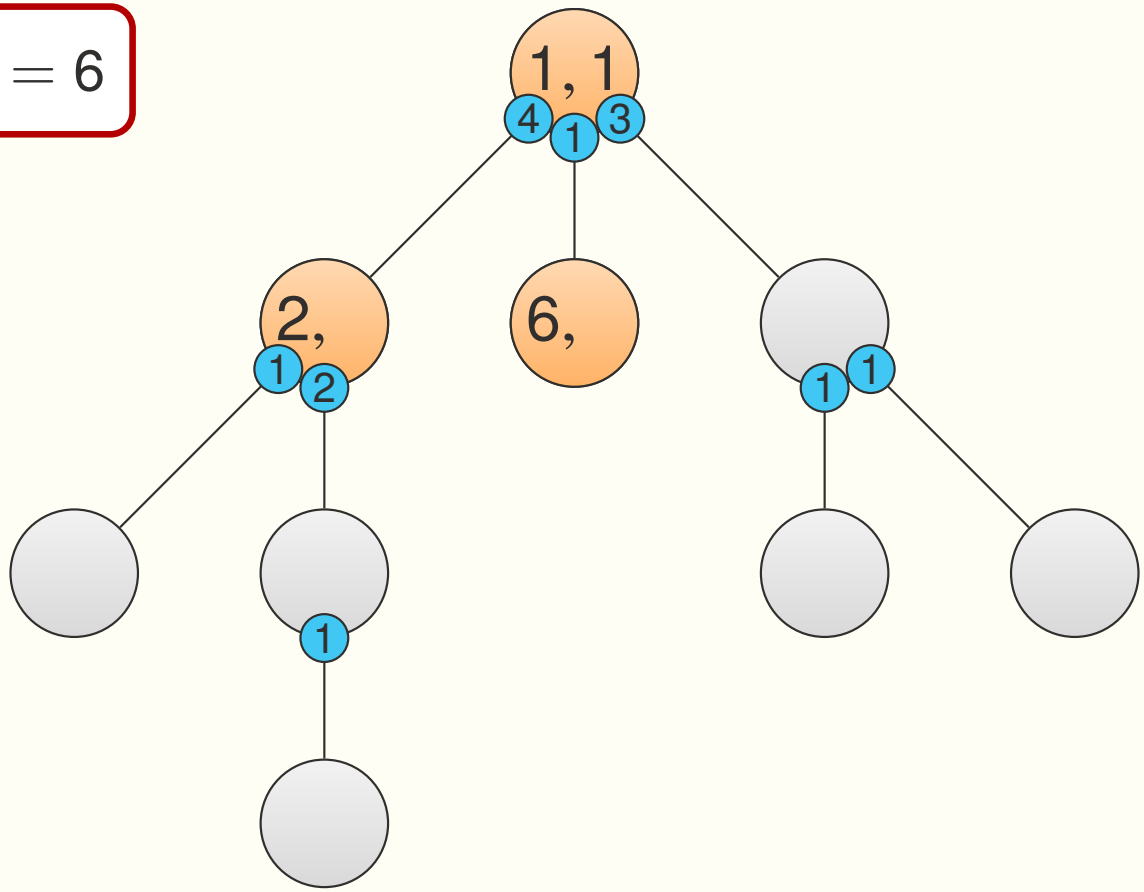


$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Labeling

Compute guide pairs from counts, in $O(h)$ rounds.

$1 + 4 + 1 = 6$

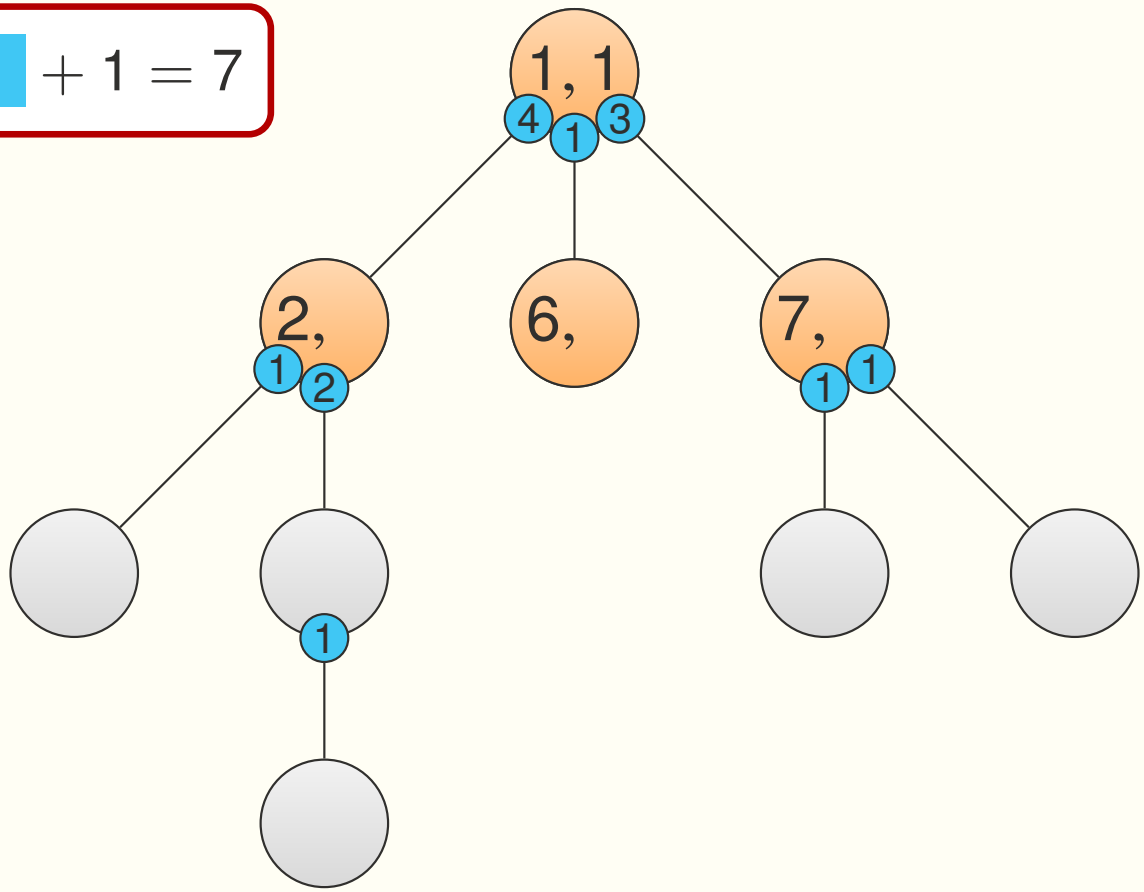


$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Labeling

Compute guide pairs from counts, in $O(h)$ rounds.

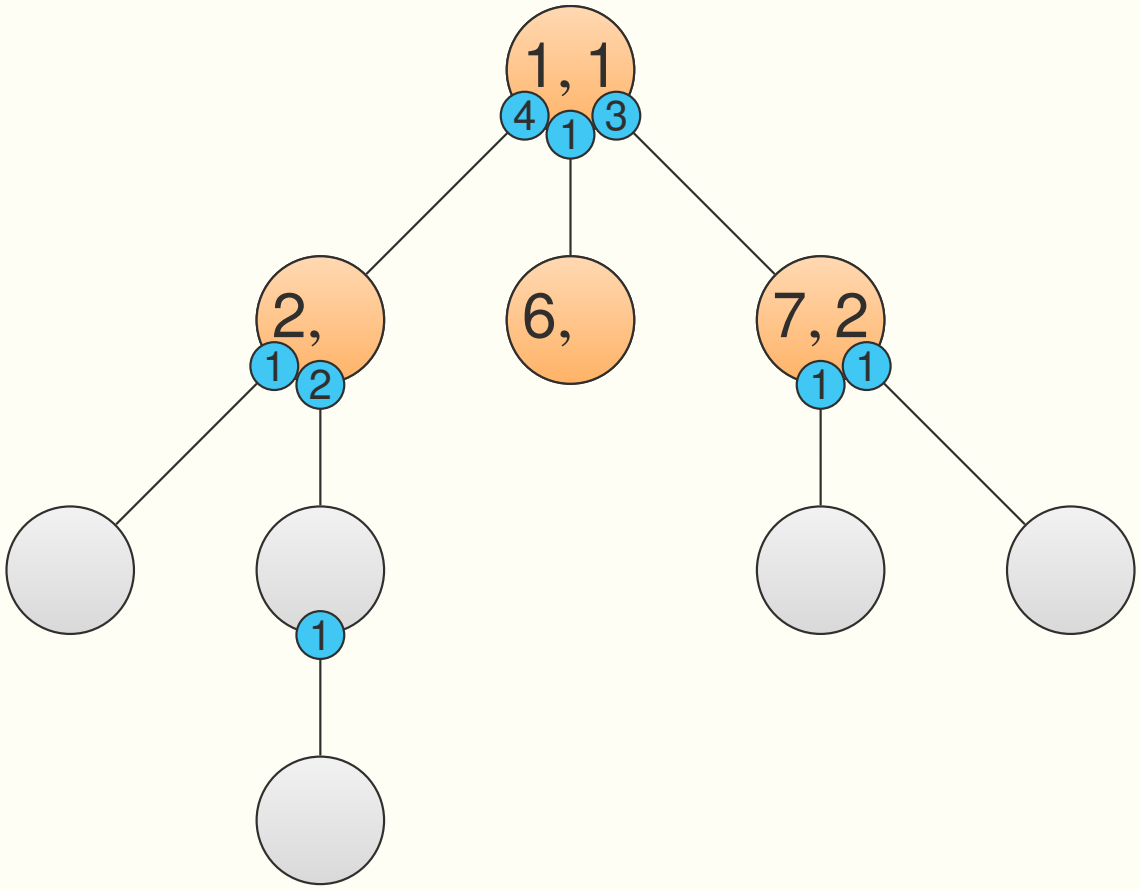
$1 + 4 + 1 + 1 = 7$



$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Labeling

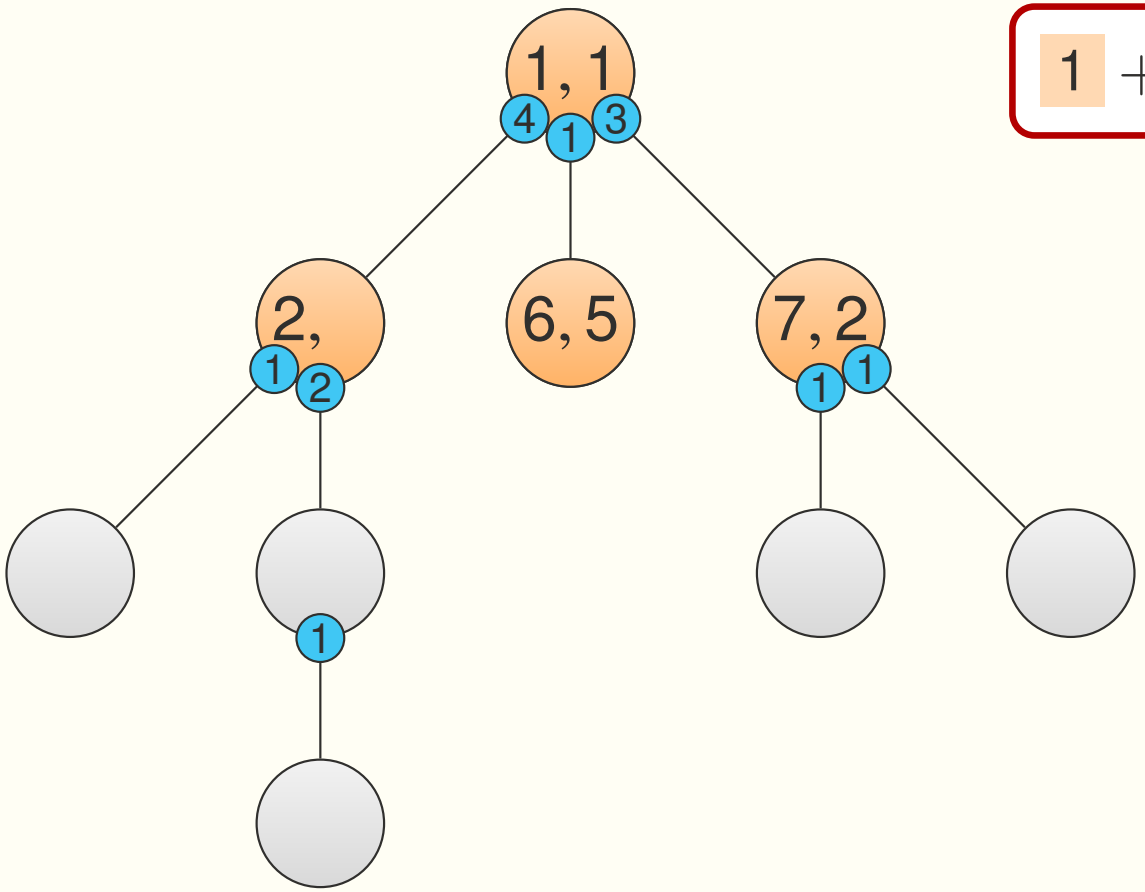
Compute guide pairs from counts, in $O(h)$ rounds.



h = height, δ = degree, n = number of processes

GUIDE: Labeling

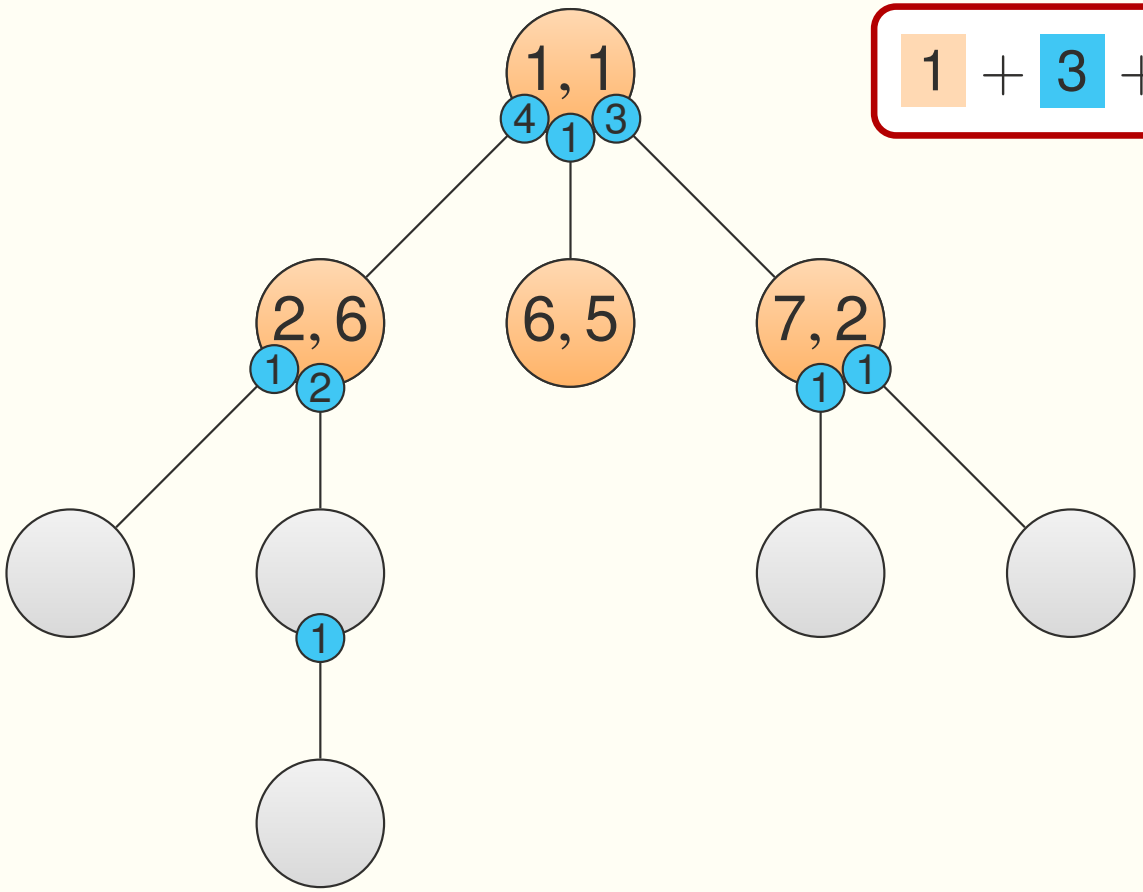
Compute guide pairs from counts, in $O(h)$ rounds.



h = height, δ = degree, n = number of processes

GUIDE: Labeling

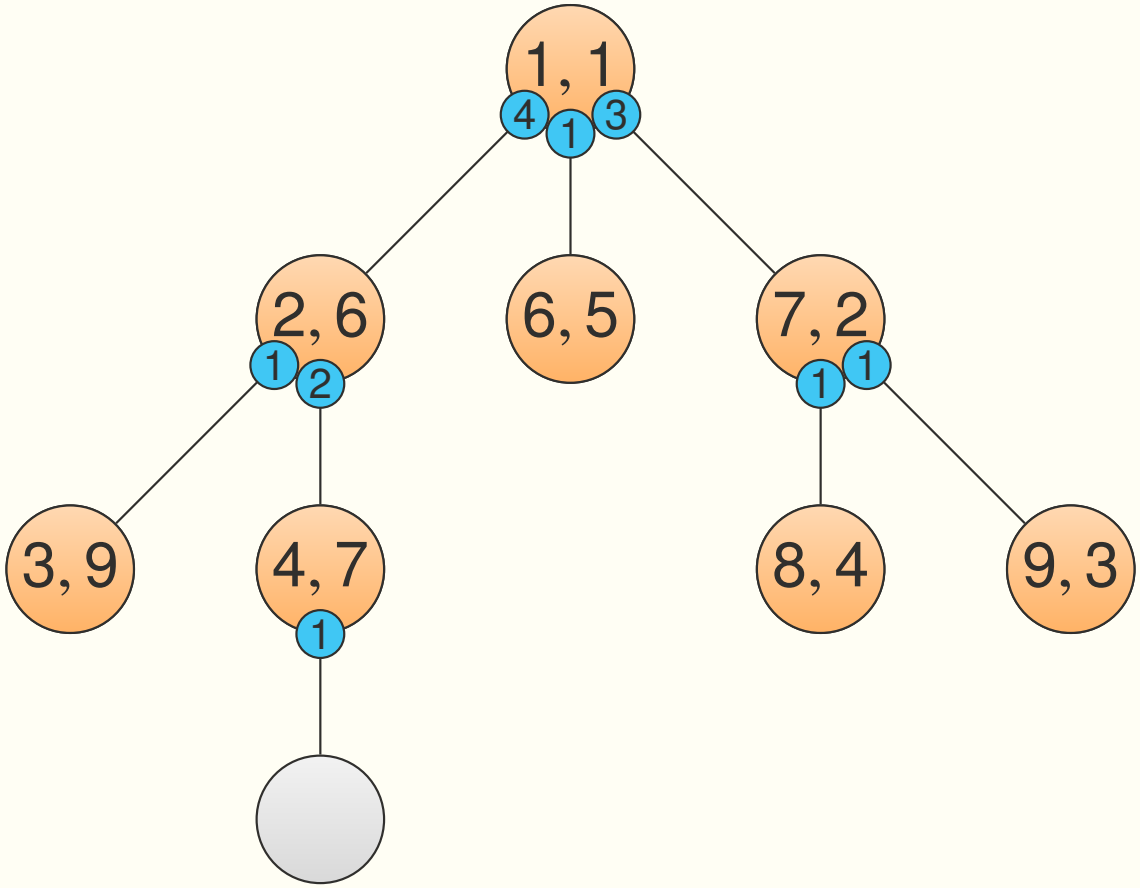
Compute guide pairs from counts, in $O(h)$ rounds.



$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Labeling

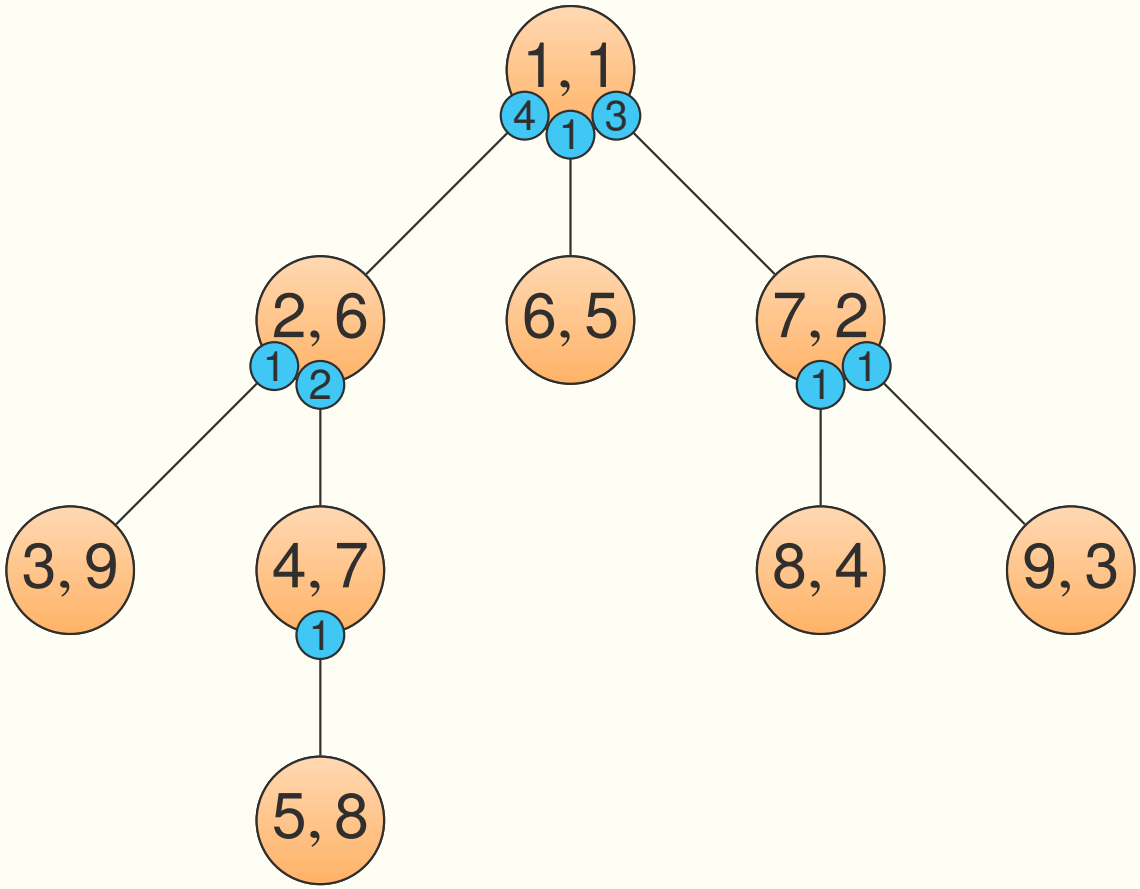
Compute guide pairs from counts, in $O(h)$ rounds.



$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Labeling

Compute guide pairs from counts, in $O(h)$ rounds.



$h =$ height, $\delta =$ degree, $n =$ number of processes

GUIDE: Properties

Algorithm that computes guide pairs,

- is self-stabilizing,
- is silent,
- stabilizes in $O(h)$ rounds,
- uses $O(\delta \log n)$ bits

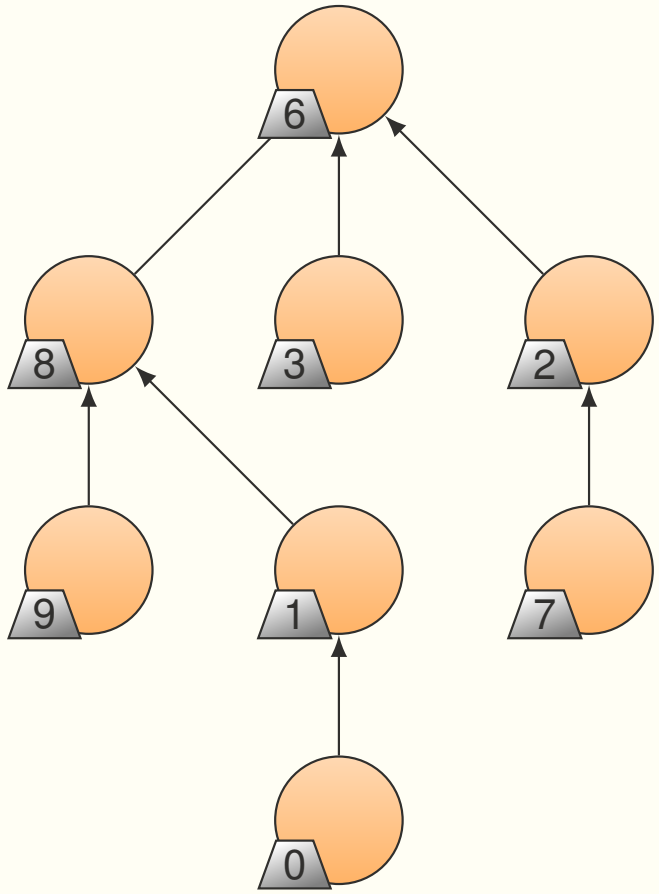
and allows for bidirectional communication.

$h =$ height, $\delta =$ degree, $n =$ number of processes

Application: Ranking

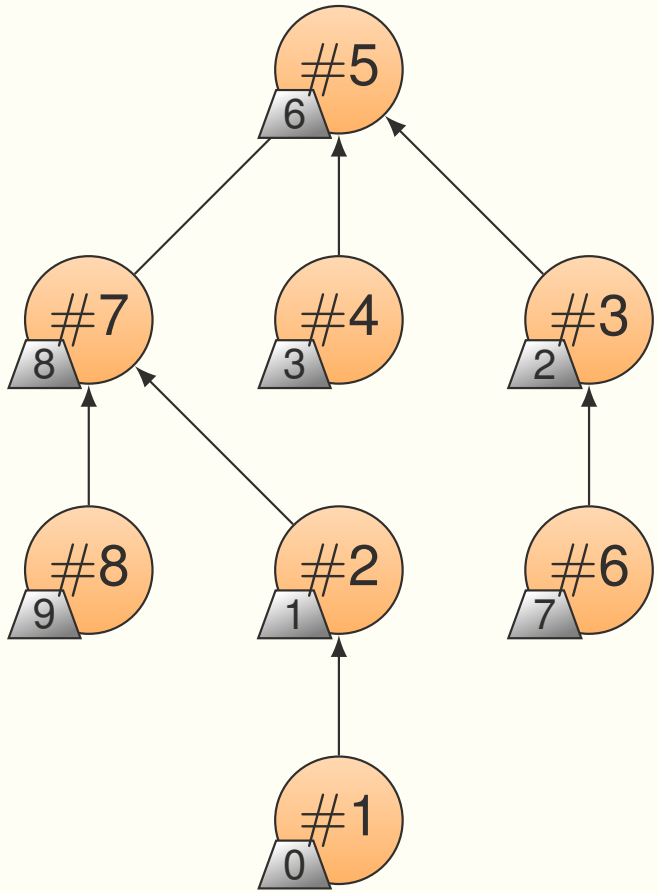
Ranking Problem

- Input: weight, of ordered type.
- Output: rank, by order of ascending weight.



Ranking Problem

- Input: weight, of ordered type.
- Output: rank, by order of ascending weight.



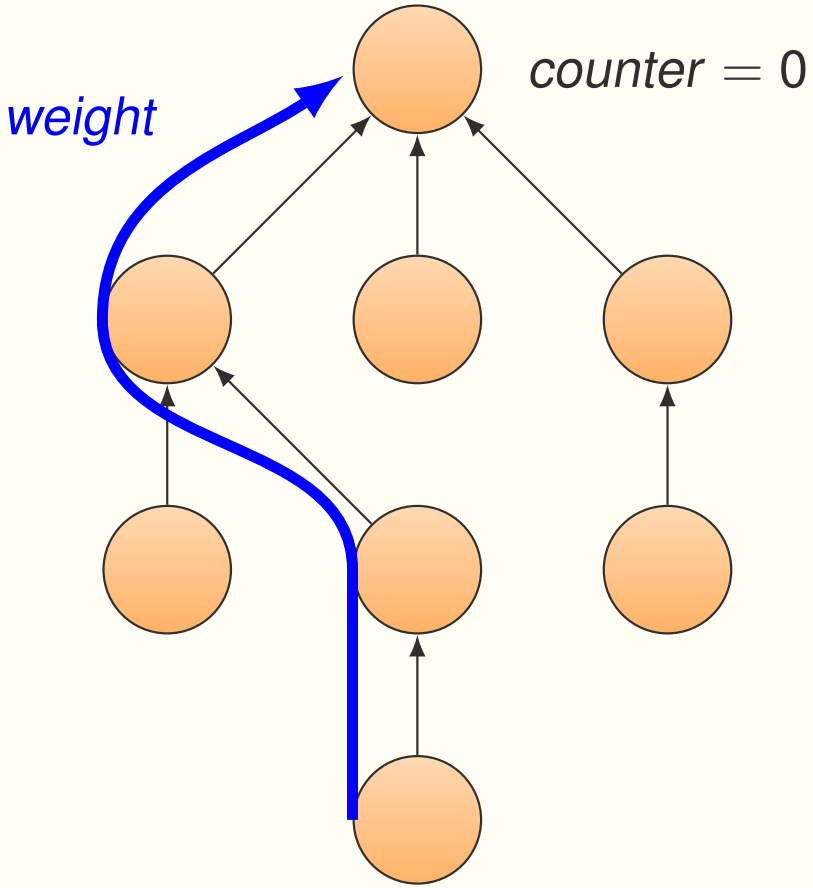
RANK: Algorithm Overview

Self-stabilizing, but *not* silent: Endlessly compute ranks.

- Compute ranks
- Repeat computation waves
- Handle errors

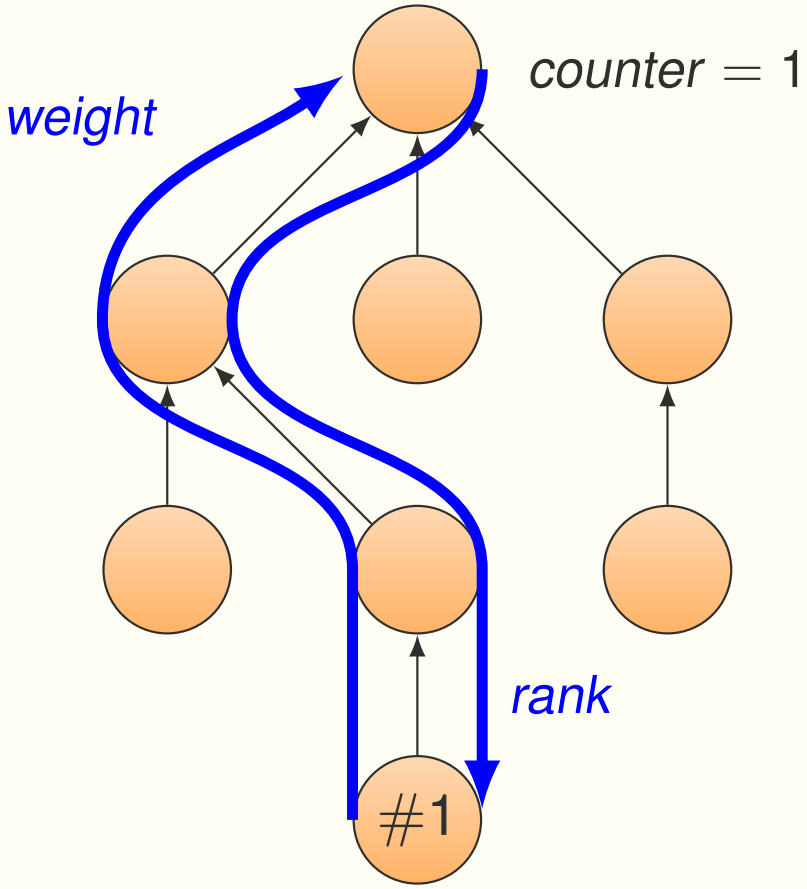
RANK: One Computation Wave

- Weight packages are (bottom-up) routed by priority of weight.
- Rank packages are (top-down) routed thanks to guide pairs.



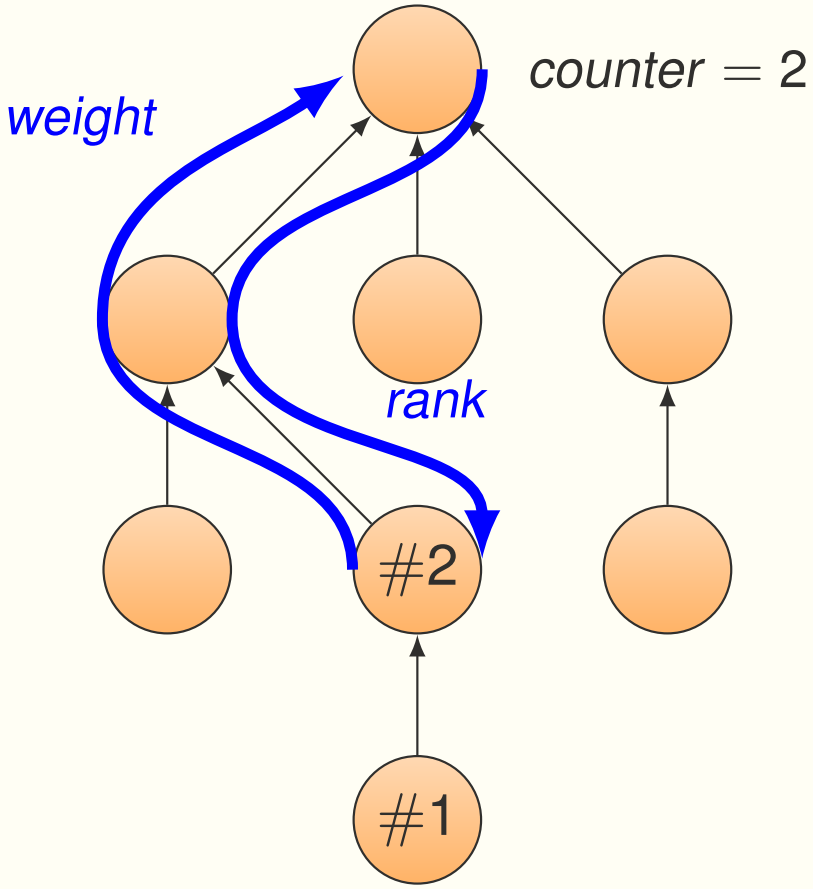
RANK: One Computation Wave

- Weight packages are (bottom-up) routed by priority of weight.
- Rank packages are (top-down) routed thanks to guide pairs.



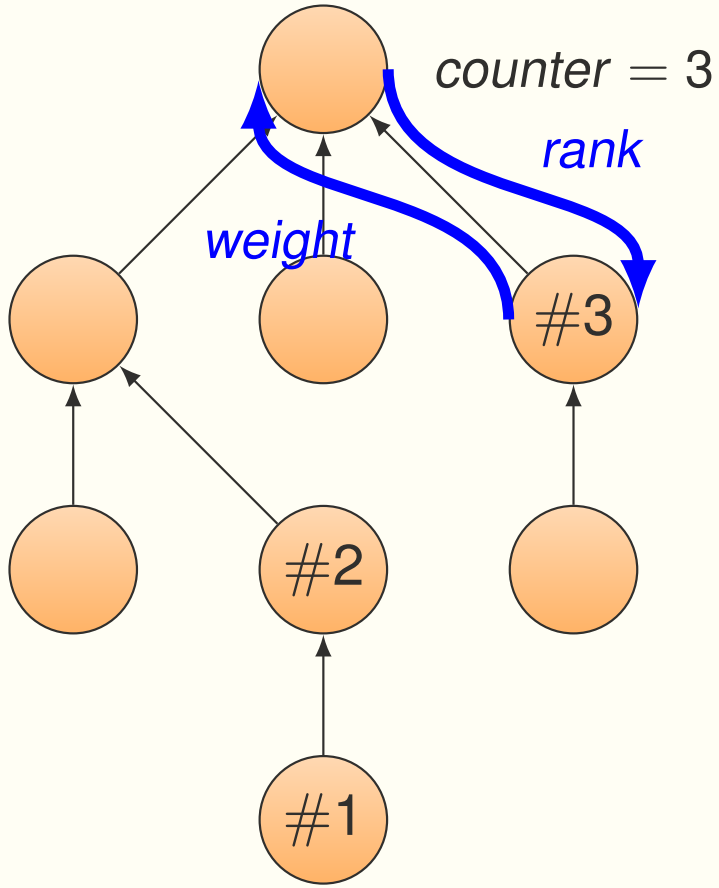
RANK: One Computation Wave

- Weight packages are (bottom-up) routed by priority of weight.
- Rank packages are (top-down) routed thanks to guide pairs.



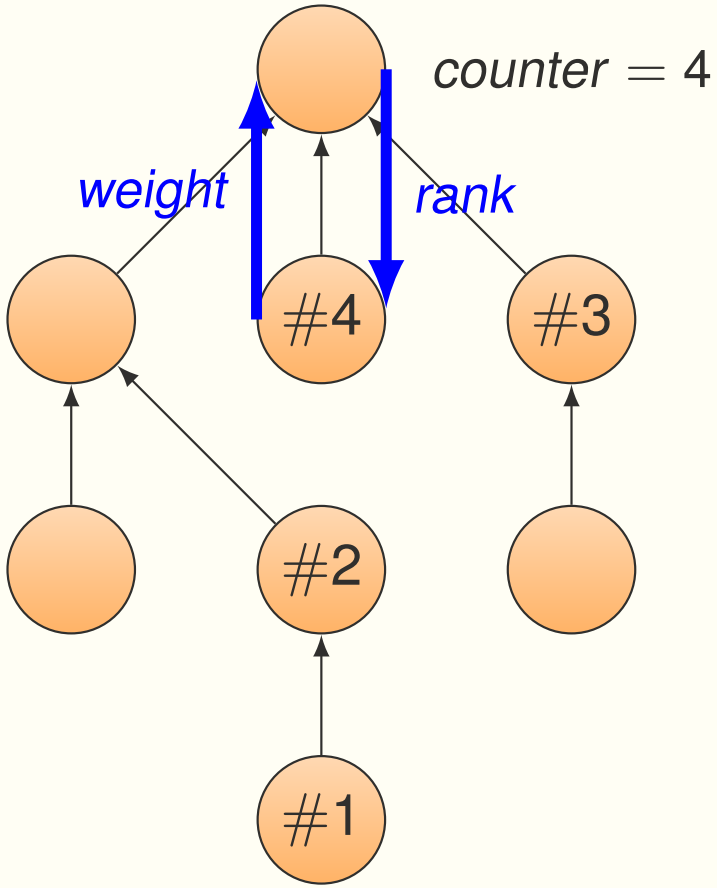
RANK: One Computation Wave

- Weight packages are (bottom-up) routed by priority of weight.
- Rank packages are (top-down) routed thanks to guide pairs.



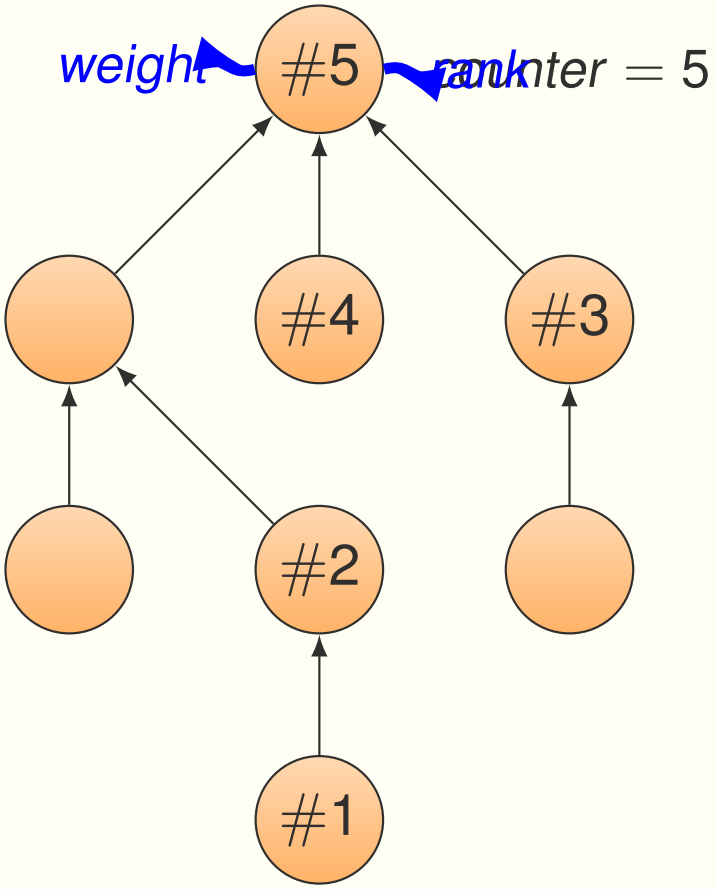
RANK: One Computation Wave

- Weight packages are (bottom-up) routed by priority of weight.
- Rank packages are (top-down) routed thanks to guide pairs.



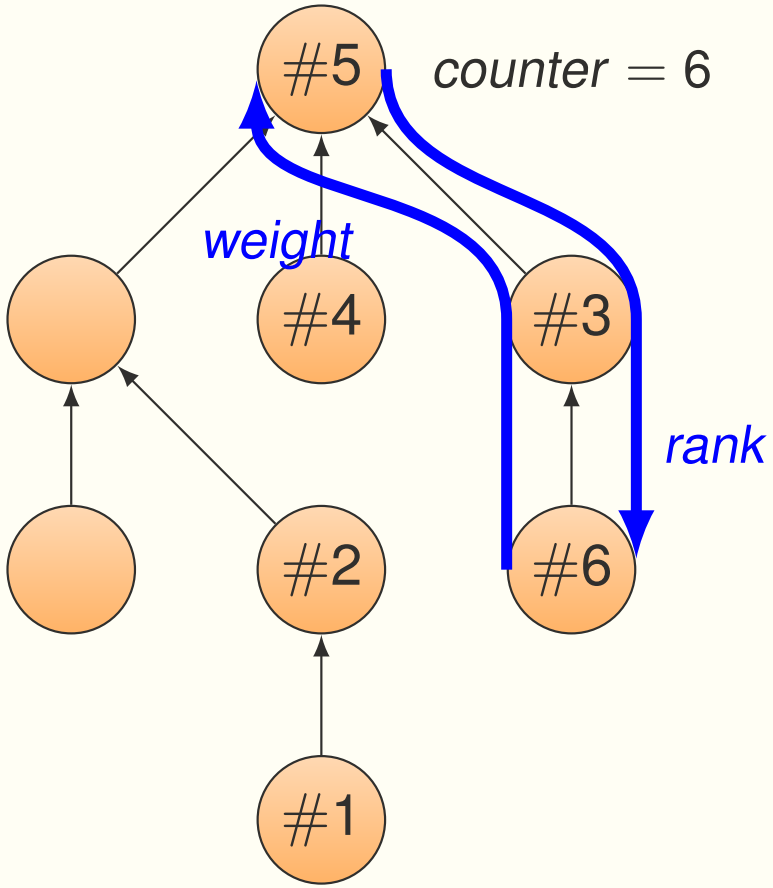
RANK: One Computation Wave

- Weight packages are (bottom-up) routed by priority of weight.
- Rank packages are (top-down) routed thanks to guide pairs.



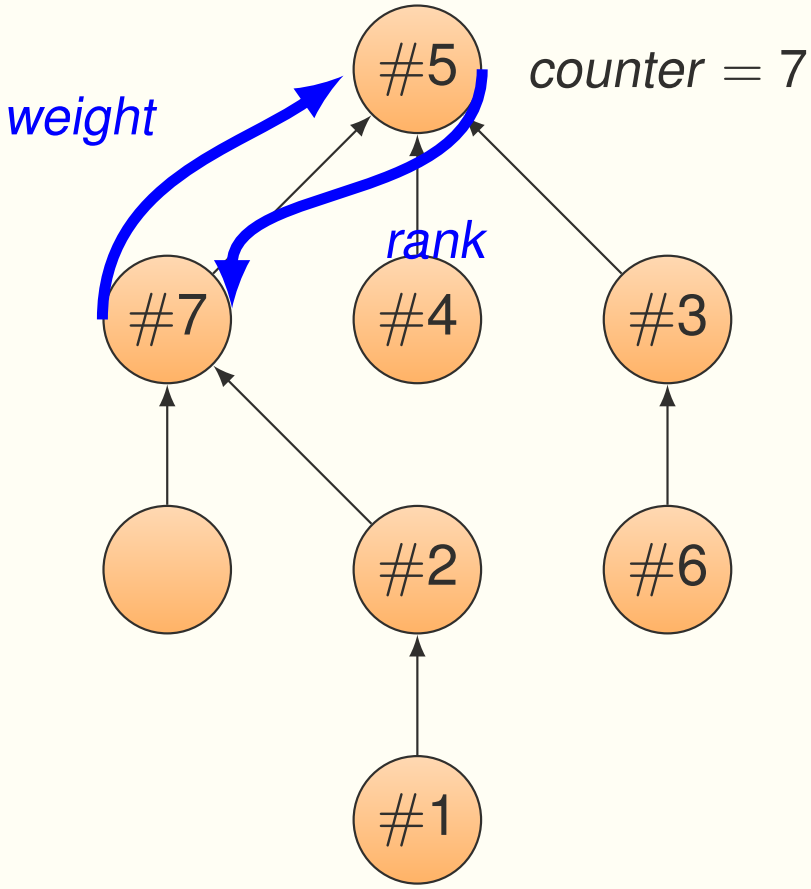
RANK: One Computation Wave

- Weight packages are (bottom-up) routed by priority of weight.
- Rank packages are (top-down) routed thanks to guide pairs.



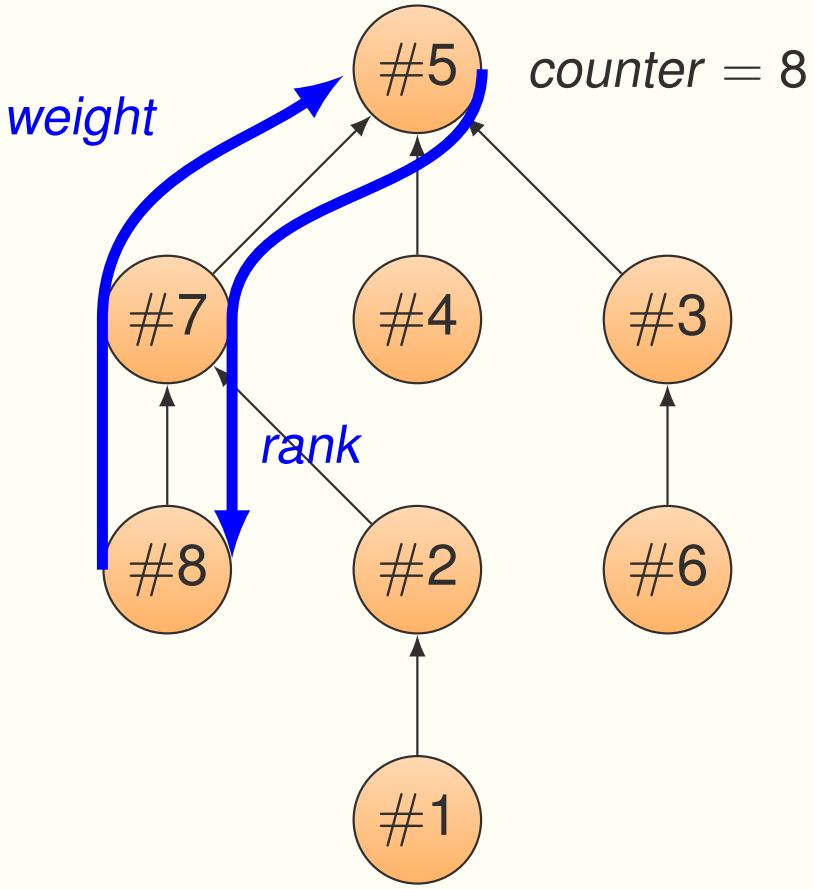
RANK: One Computation Wave

- Weight packages are (bottom-up) routed by priority of weight.
- Rank packages are (top-down) routed thanks to guide pairs.



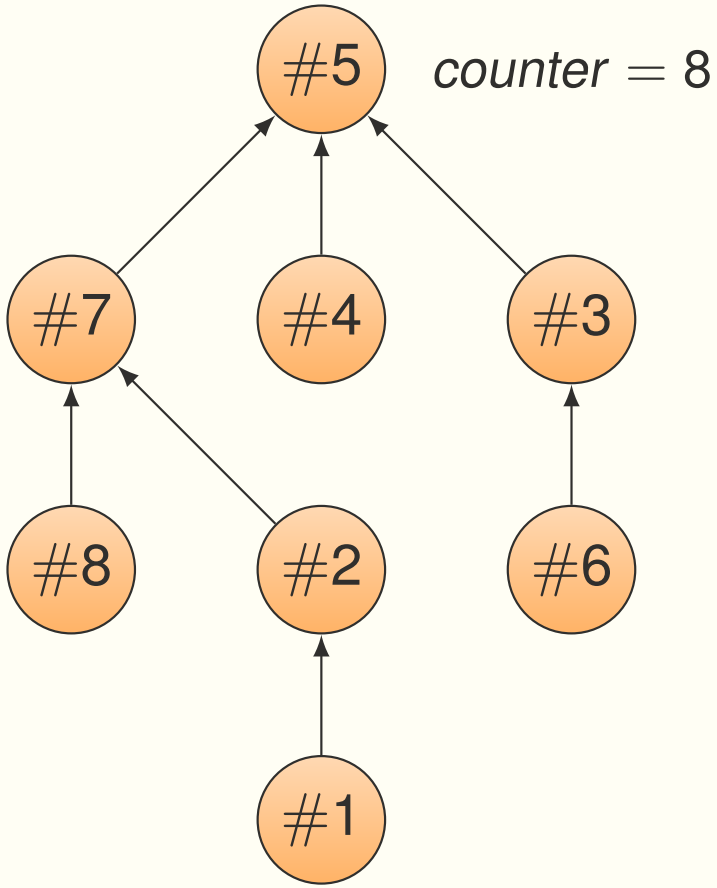
RANK: One Computation Wave

- Weight packages are (bottom-up) routed by priority of weight.
- Rank packages are (top-down) routed thanks to guide pairs.



RANK: One Computation Wave

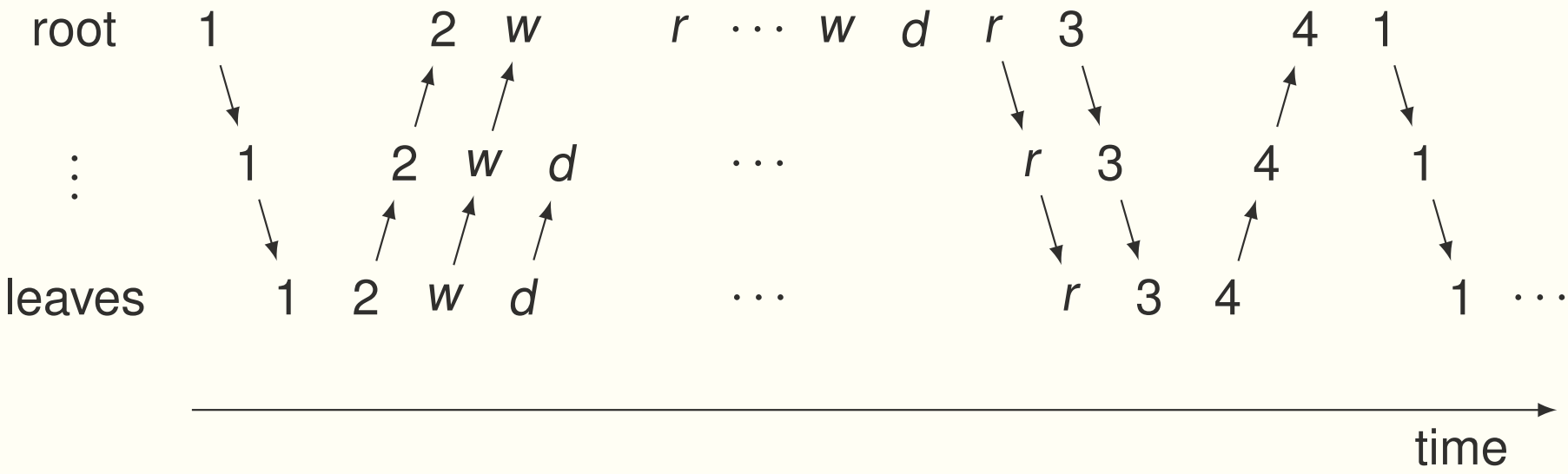
- Weight packages are (bottom-up) routed by priority of weight.
- Rank packages are (top-down) routed thanks to guide pairs.



RANK: Repeated Computation Waves

Status variable:

- 1: new epoch (reset)
- 2: new comp. wave
- 3: end of comp. wave
- 4: end of epoch

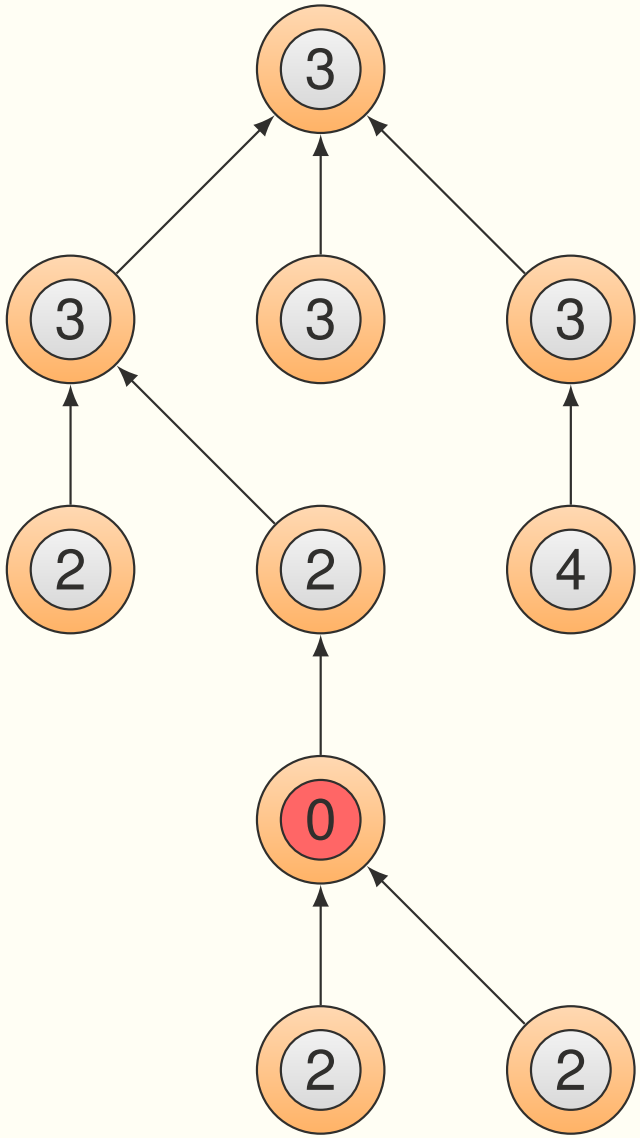


w = weight package, d = done package, r = rank package

RANK: Error Correction

- 0: error correction
- propagated up and down the tree, except below status 1 (reset).

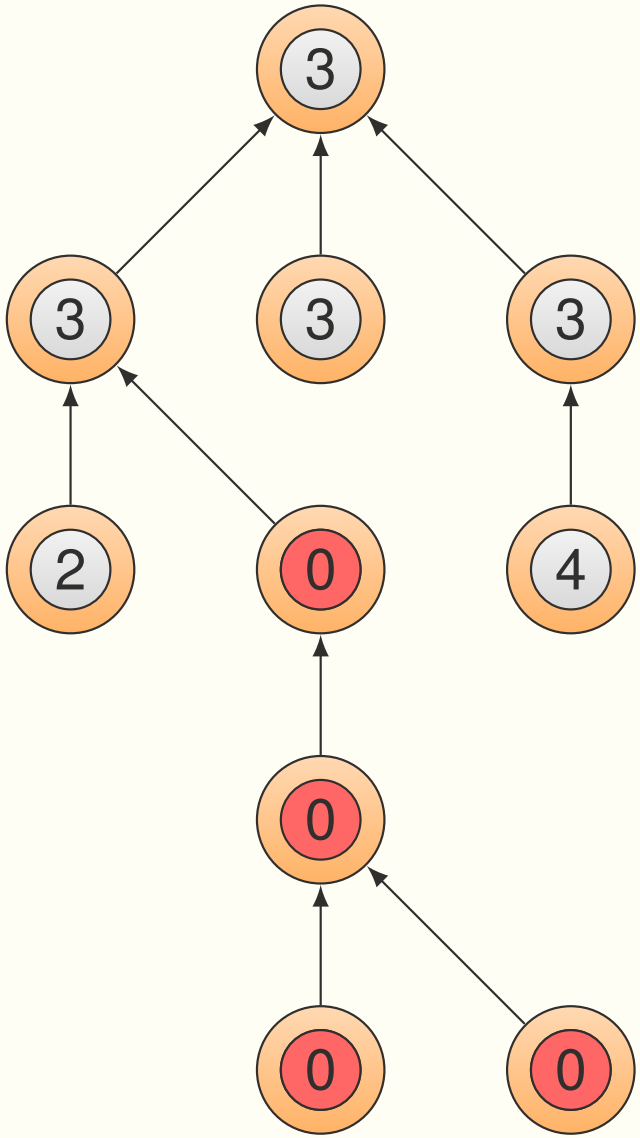
status 0 propagation



RANK: Error Correction

- 0: error correction
- propagated up and down the tree, except below status 1 (reset).

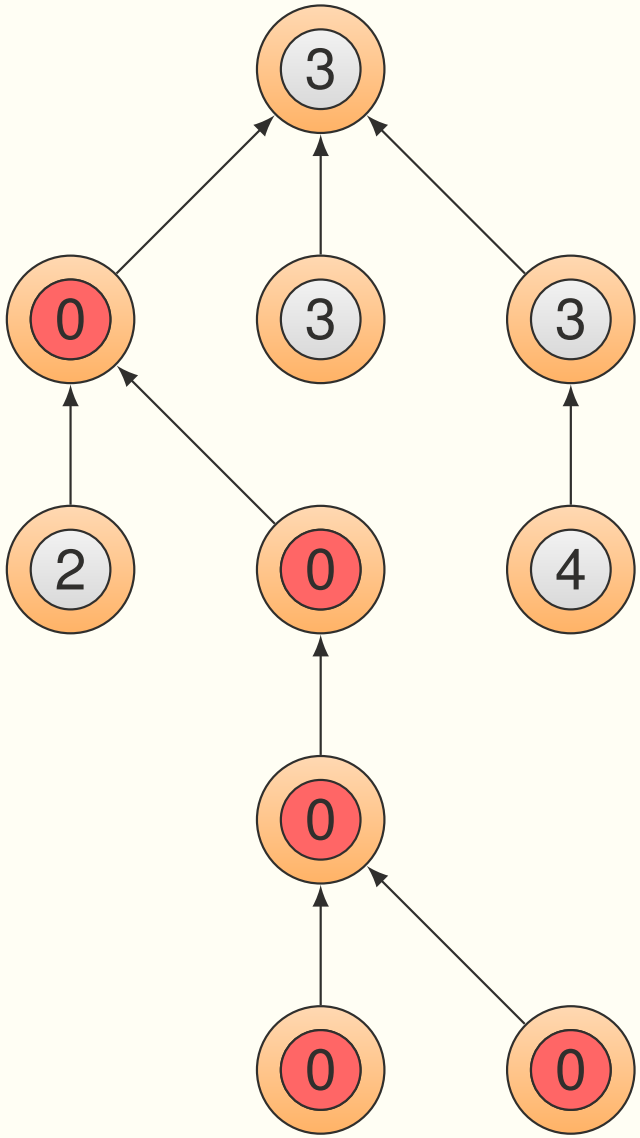
status 0 propagation



RANK: Error Correction

- 0: error correction
- propagated up and down the tree, except below status 1 (reset).

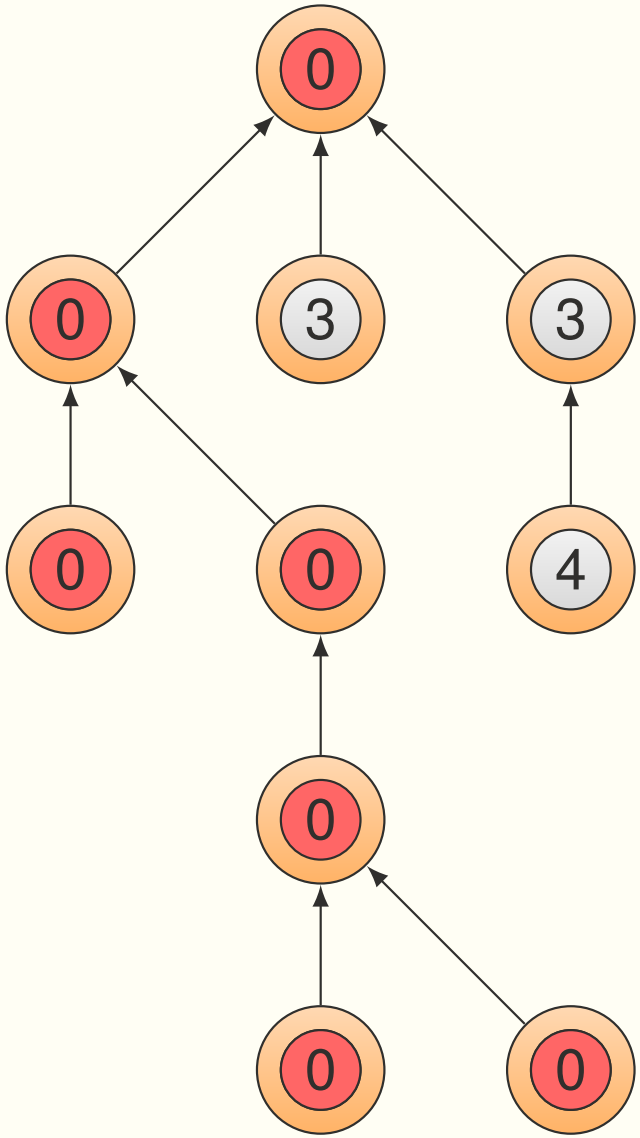
status 0 propagation



RANK: Error Correction

- 0: error correction
- propagated up and down the tree, except below status 1 (reset).

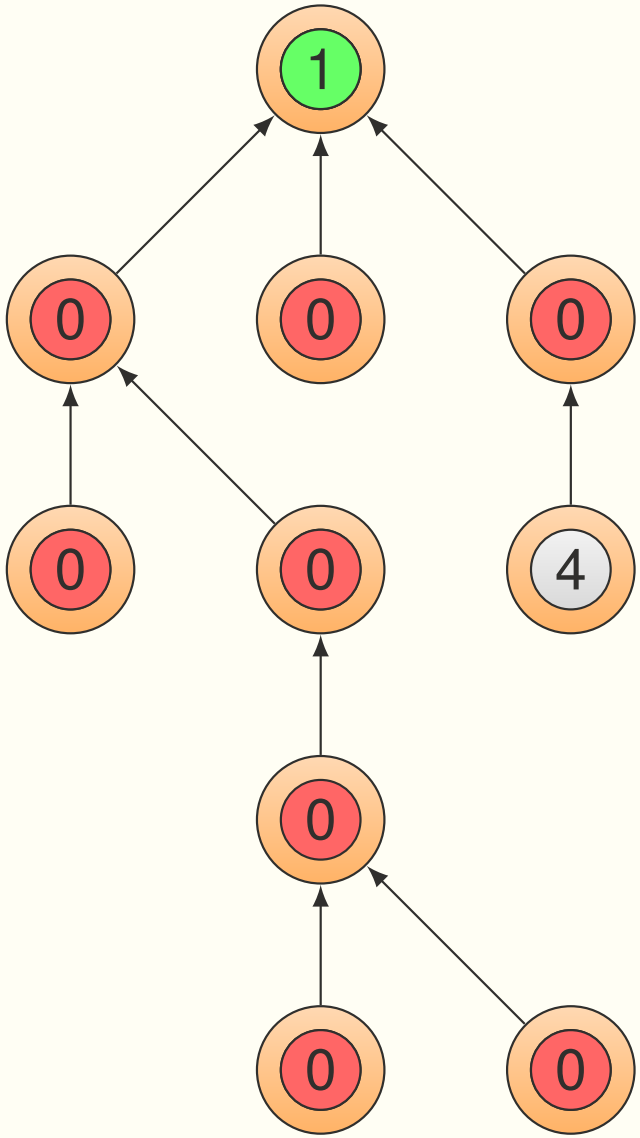
status 0 propagation



RANK: Error Correction

- 0: error correction
- propagated up and down the tree, except below status 1 (reset).

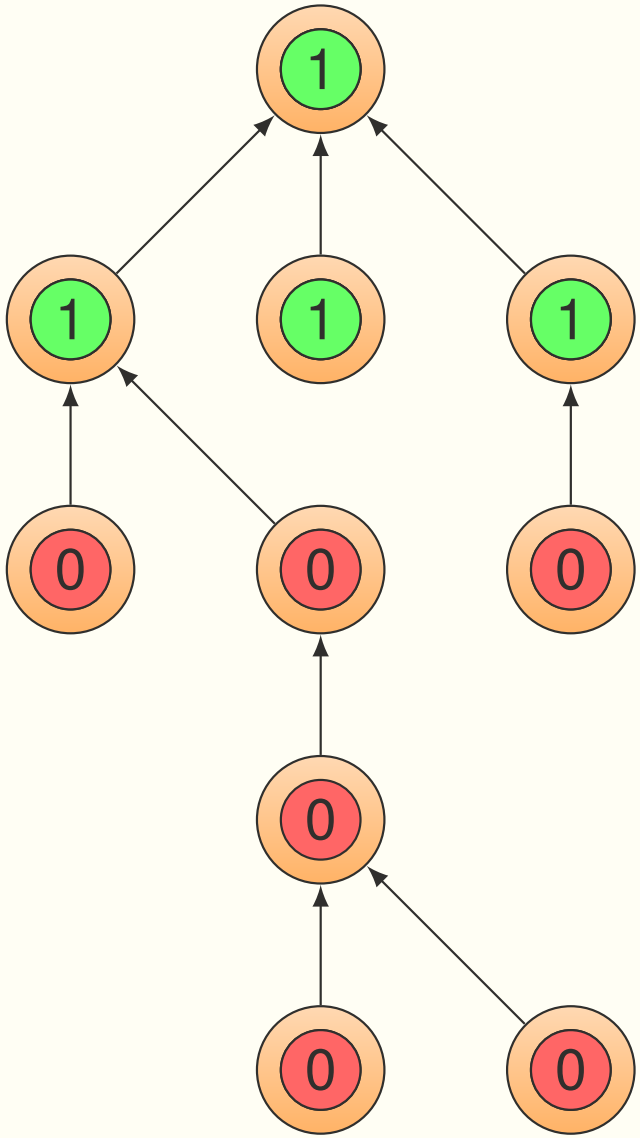
status 1 global reset



RANK: Error Correction

- 0: error correction
- propagated up and down the tree, except below status 1 (reset).

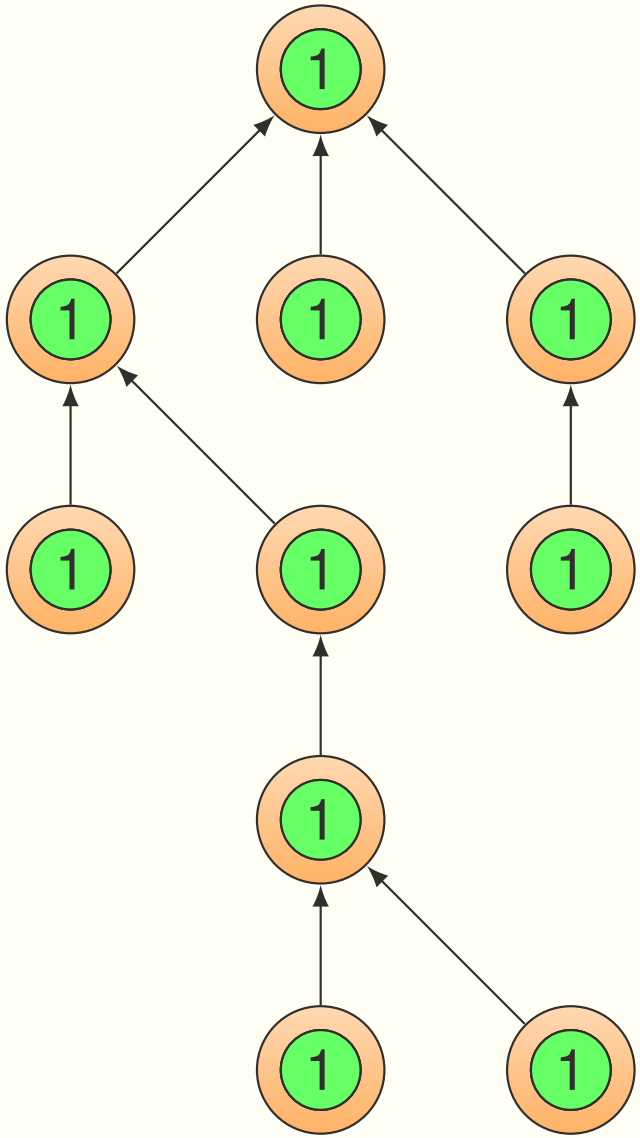
status 1 global reset



RANK: Error Correction

- 0: error correction
- propagated up and down the tree, except below status 1 (reset).

status 1 global reset



RANK: Properties

Algorithm that ranks processes by weight,

- is self-stabilizing,
- is *not* silent,
- stabilizes in $O(n)$ rounds and
  (previously $O(n^2)$ by Alari & al., 1998)
- uses $O(b + \delta \log n)$ bits.

$b =$ weight's bits, $\delta =$ degree, $n =$ number of processes

Conclusion

Contributions

- Labeling algorithm that computes guide pairs, thus allows for bidirectionnal communication.

  (first self-stabilizing solution)

- Application: Distributed ranking of processes.

  (better stab. time than Alari & al., 1998)

Perspectives

- What about an unfair scheduler?
  Which step complexity? in $O(n^2)$?

- Find other applications to guide pairs.

Thanks for your attention