- **Weeks 1–2: informal introduction**
  - network = path

- **Week 3: graph theory**

- **Weeks 4–7: models of computing**
  - what can be computed (efficiently)?

- **Weeks 8–11: lower bounds**
  - what cannot be computed (efficiently)?

- **Week 12: recap**

Mid-term exams

- **Mid-term exams:**
  - Thursday, 27 October 2016, 9:00am
  - Thursday, 15 December 2016, 9:00am

- **Register in time (*one week before*) in Oodi**

Week 5

– LOCAL model: unique identifiers

LOCAL model

- **Idea: nodes have unique names**

- **Names arbitrary but fairly short**

- **IPv4 addresses, IPv6 addresses, MAC addresses, IMEI numbers…**

LOCAL model

- **LOCAL model =
  PN model + unique identifiers**

- **Assumption: unique identifiers
  are given as local inputs**

LOCAL model

- **Algorithm has to work correctly for any port numbering and for any unique identifiers**

- **Adversarial setting:**
  - you design algorithms
  - adversary picks graph, port numbering, IDs

LOCAL model

- **Fixed constant c**

- **In a network with n nodes, identifiers are a subset of $\{1, 2, \ldots, n^c\}$**

- **Equivalently: unique identifiers can be encoded with $O(\log n)$ bits**

PN vs. LOCAL

- **PN: few problems can be solved**

- **LOCAL: all problems can be solved**
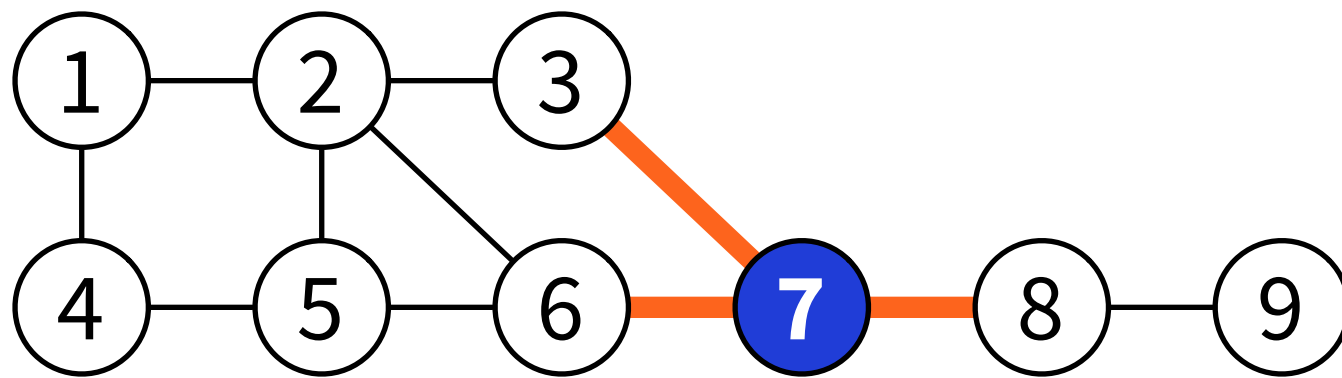 (on connected graphs)

PN vs. LOCAL

- **PN: "*what can be computed?*"**

- **LOCAL: "*what can be computed efficiently?*"**

Solving everything

- **All nodes learn everything about the graph**
  - $O(\text{diam}(G))$ rounds

- **All nodes solve the problem locally (e.g., by brute force)**
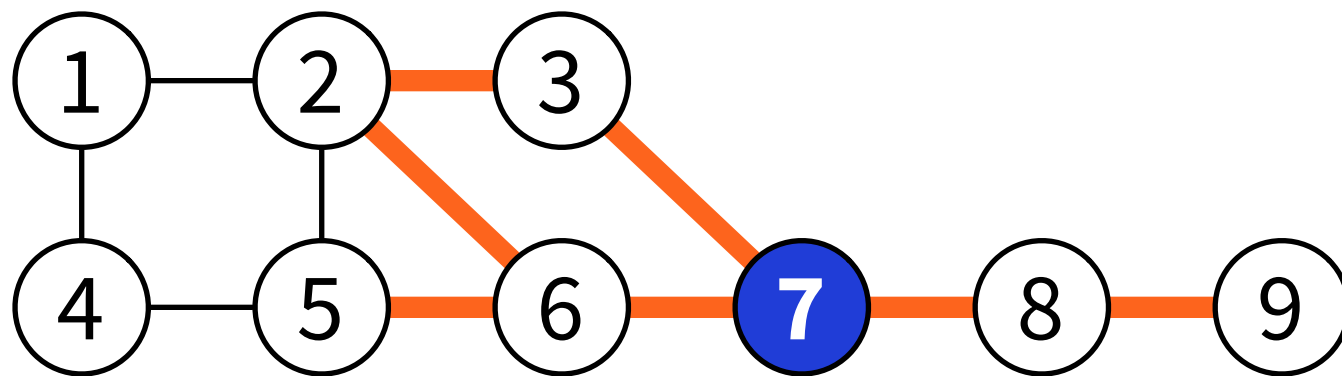 - 0 rounds

Gathering everything

- *E*(*v*, *r*) = "edges within distance *r* from *v*"
  = one endpoint at distance at most *r* − 1 from *v*
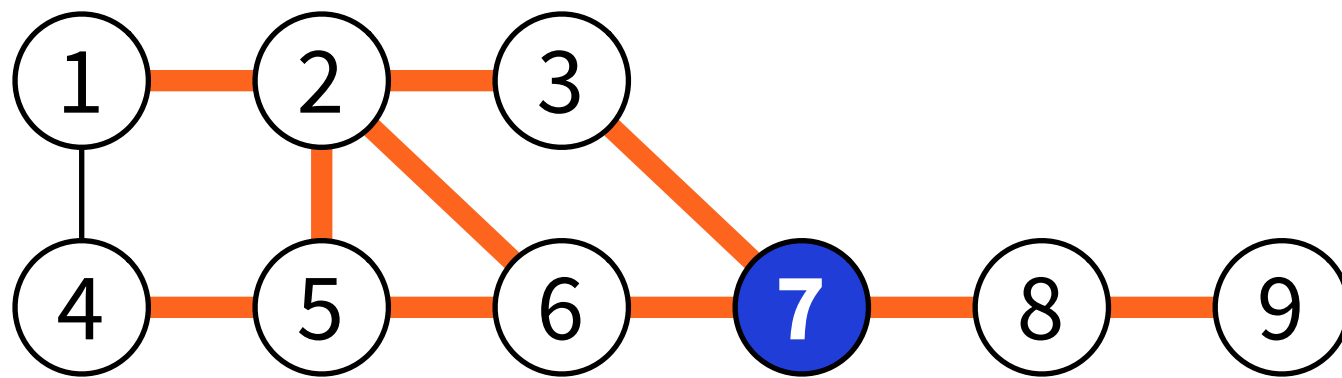

E(7, 1)

Gathering everything

- *E*(*v*, *r*) = "edges within distance *r* from *v*"
  = one endpoint at distance at most *r* − 1 from *v*
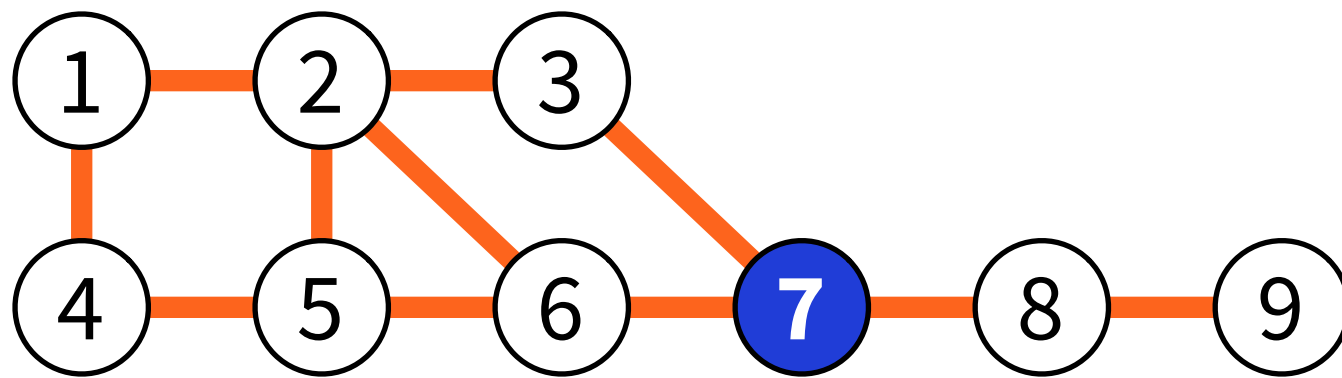

E(7, 2)

Gathering everything

- *E*(*v*, *r*) = "edges within distance *r* from *v*"
  = one endpoint at distance at most *r* − 1 from *v*

E(7, 3)
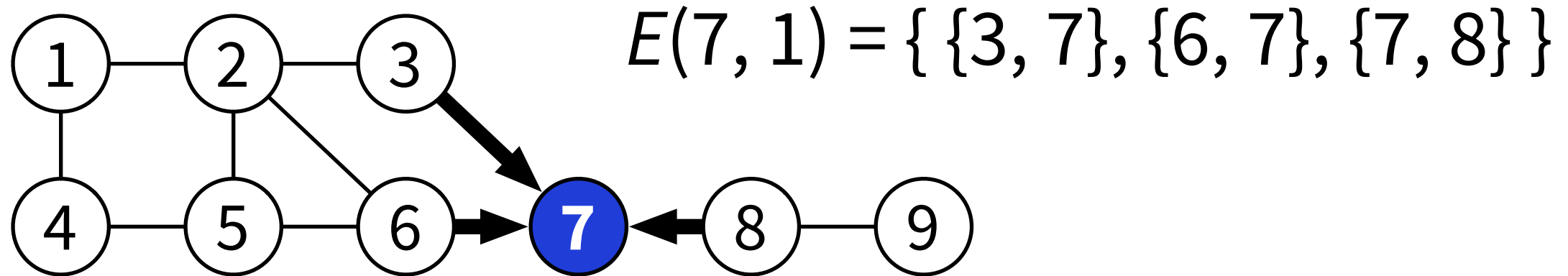
Gathering everything

- $E(v, r)$ = "edges within distance r from v"
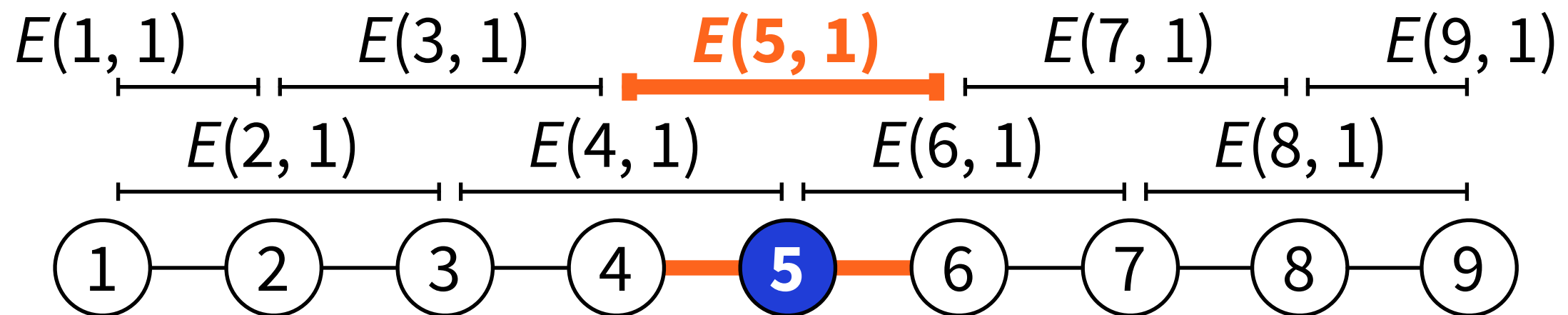 = one endpoint at distance at most $r - 1$ from v

$E(7, 4)$

Gathering everything

- **Each node *v* can learn *E*(*v*, 1) in 1 round**
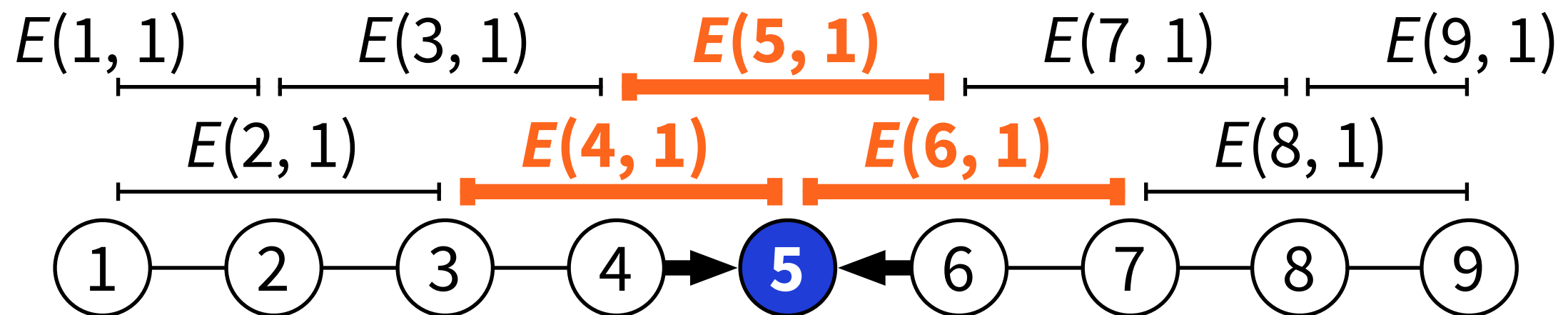 - send own ID to all neighbours

$E(7, 1) = \{ \{3, 7\}, \{6, 7\}, \{7, 8\} \}$

Gathering everything

- **Each node *v* can learn $E(v, 1)$ in 1 round**
  - send own ID to all neighbours

$E(1, 1)$ $E(3, 1)$ **$E(5, 1)$** $E(7, 1)$ $E(9, 1)$

$E(2, 1)$ $E(4, 1)$ $E(6, 1)$ $E(8, 1)$

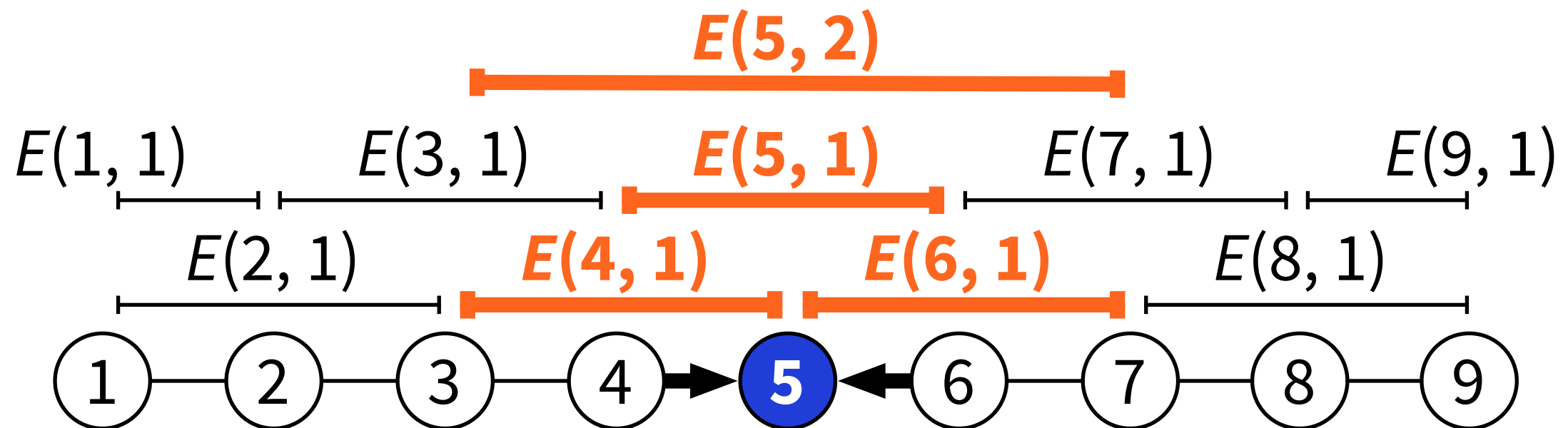① — ② — ③ — ④ — **⑤** — ⑥ — ⑦ — ⑧ — ⑨

Gathering everything

- **Given $E(v, r)$, we can learn $E(v, r + 1)$ in 1 round**
  - send $E(v, r)$ to all neighbours, take union

Gathering everything

- **Given $E(v, r)$, we can learn $E(v, r + 1)$ in 1 round**
  - send $E(v, r)$ to all neighbours, take union

Gathering everything

- **One of the following holds:**

  - $E(v, r) \neq E(v, r + 1)$: **learn something new**
  - $E(v, r) = E(v, r + 1) = E$: **we can stop**

- **Proof idea:**

  - if $E(v, r) \neq E$, there are unseen edges adjacent to $E(v, r)$, and they will be in $E(v, r + 1)$

Example:
Graph colouring

- **We can solve everything in $O(\text{diam}(G))$ time**

- **What can be solved much faster?**

- **Example: graph colouring with $\Delta + 1$ colours**
  - can be solved in $O(\Delta^{0.51} + \log^* n)$ rounds
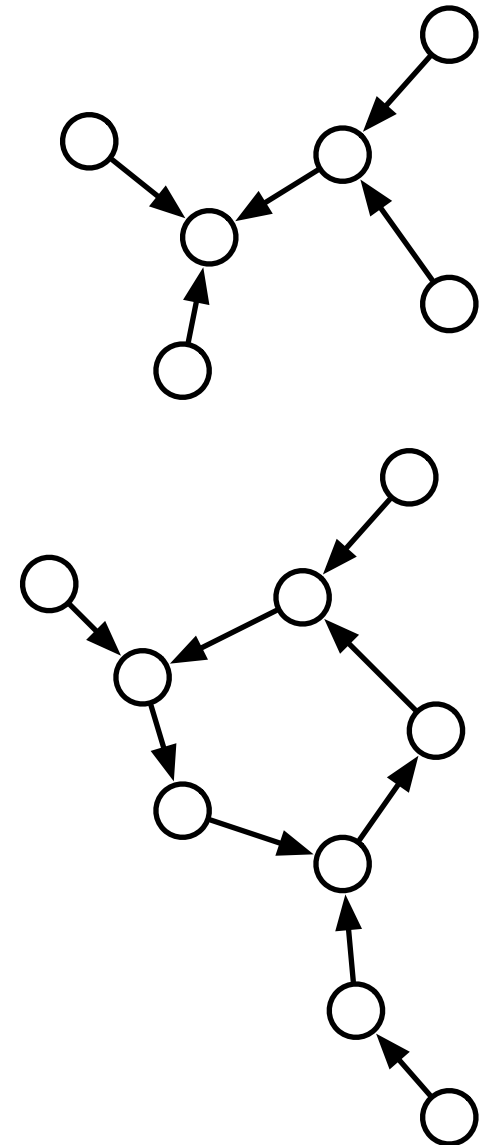 - today: how to do it in $O(\Delta^2 + \log^* n)$ rounds?

Example:
Graph colouring

- **Setting: LOCAL model, n nodes, any graph of maximum degree Δ**

- **We assume that n and Δ are known**

  - if not known: guess some n and Δ, colour what you can, increase n and Δ, …

Directed pseudoforest

- **Directed graph, outdegree ≤ 1**

- **Each node has at most one "successor"**

- **Easy to 3-colour in time $O(\log^* n)$, we will use this as subroutine**

Directed pseudoforest

- **Colouring directed pseudoforests almost as easy as colouring directed paths**

- **Recall path-colouring algorithm P3CBit…**
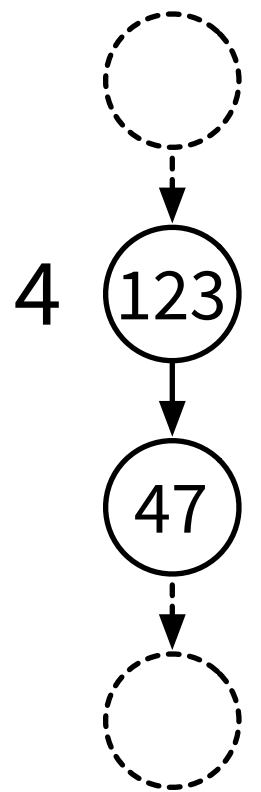
Algorithm P3CBit:
Fast colour reduction

$c_0 = 123 = 01111011_2$ (my colour)

$c_1 = 47 = 00101111_2$ (successor's colour)

$i = 2$ (bits numbered 0, 1, 2, … from right)

$b = 0$ (in my colour bit number i was 0)

$c = 2 \cdot 2 + 0 = 4$ (my new colour)

$k = 8$, reducing from $2^8 = 256$ to $2 \cdot 8 = 16$ colours

Directed pseudoforest

- **Colouring directed pseudoforests almost as easy as colouring directed paths**

- **Recall path-colouring algorithm P3CBit:**
  - nodes **only look at their successor**
  - my new colour ≠ successor's new colour
  - works equally well in directed pseudoforests!

Algorithm DPBit:
Fast colour reduction

c_0 = **123** = **01111011**$_2$ (my colour)

c_1 =   **47** = **00101111**$_2$ (successor's colour)

i = **2** (bits numbered 0, 1, 2, … from right)

b = **0** (in my colour bit number i was 0)

c = **2·2 + 0 = 4** (my new colour)
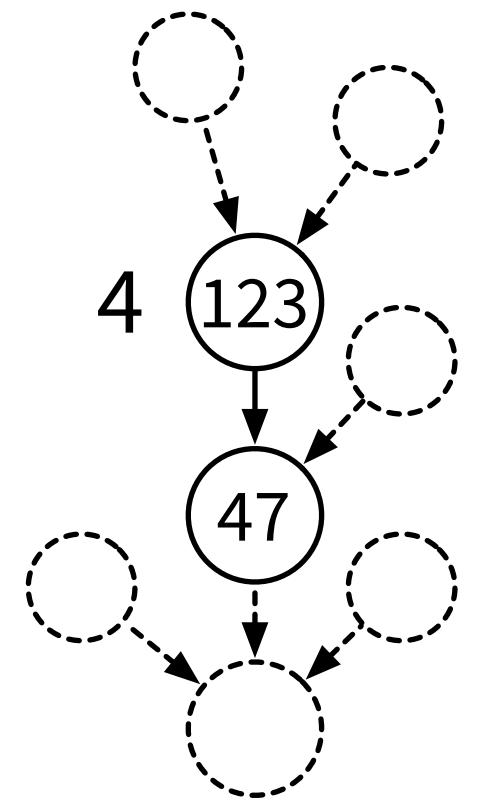
k = 8, reducing from 2^8 = 256 to 2·8 = 16 colours

Directed pseudoforests

- **Unique identifiers = $n^{O(1)}$ colours**

- **Iterate DPBit for $O(\log^* n)$ steps
  to reduce the number of colours to 6**

- **Iterate DPGreedy for 3 steps
 to reduce the number of colours to 3**

Algorithm DPGreedy:
Slow colour reduction

1. **Shift:** *predecessors have the same colour*
2. **Recolour local maxima**
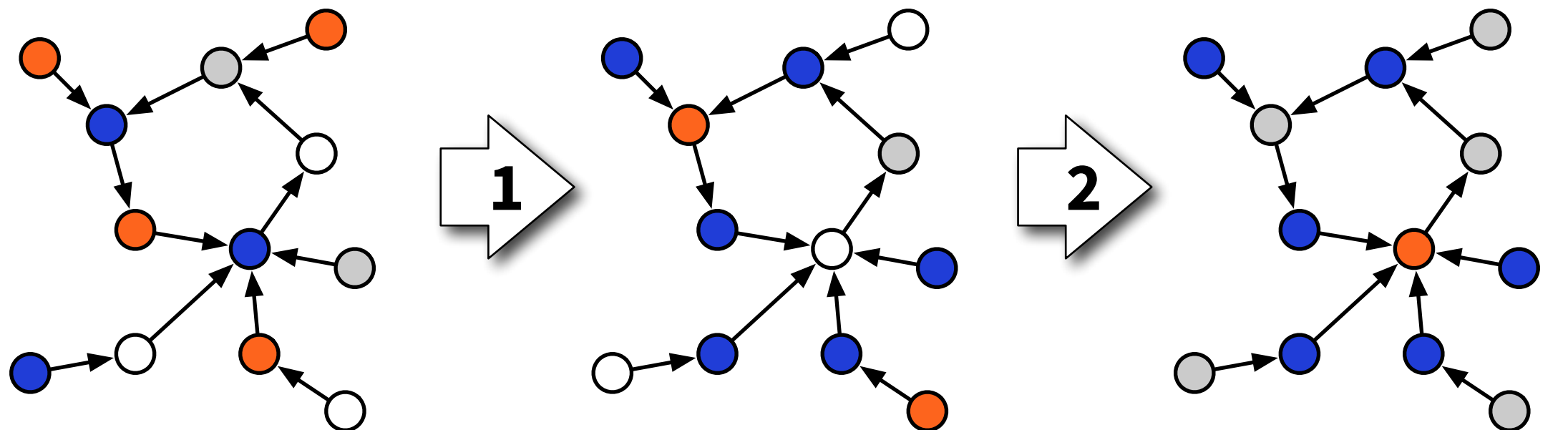
Directed pseudoforests

- **Unique identifiers = $n^{O(1)}$ colours**

- **Iterate DPBit for $O(\log^* n)$ steps to reduce the number of colours to 6**

- **Iterate DPGreedy for 3 steps to reduce the number of colours to 3**

Algorithm BDColour:
Fast graph colouring

- **Unique identifiers → orientation**

- **Port numbers → partition edges in Δ directed pseudoforests**

- **3-colour pseudoforests in time $O(\log^* n)$**

- **Merge pseudoforests in time $O(\Delta^2)$**

Algorithm BDColour:
Fast graph colouring

- **Unique identifiers → orientation**
  - edges directed from smaller to larger ID
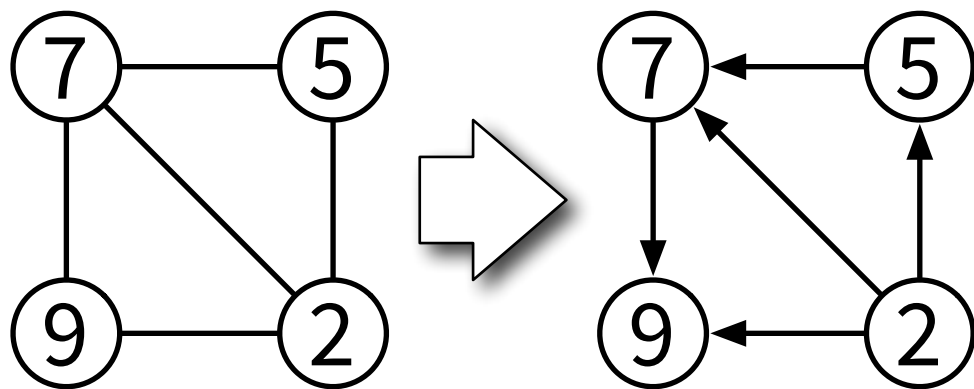
Algorithm BDColour:
Fast graph colouring

- **Port numbers → partition edges in Δ directed pseudoforests**
  - *k*th outgoing edge → *k*th pseudoforest
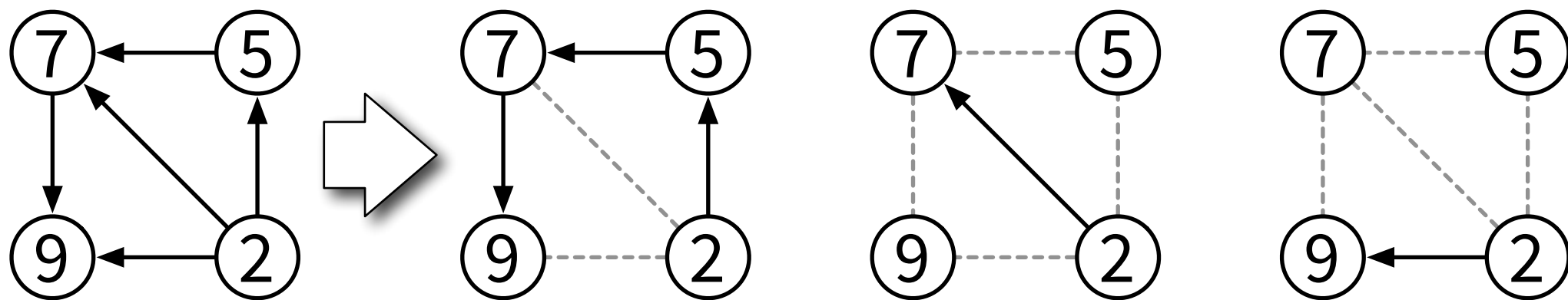
Algorithm BDColour:
Fast graph colouring

- **3-colour pseudoforests in time *O*(log* *n*)**
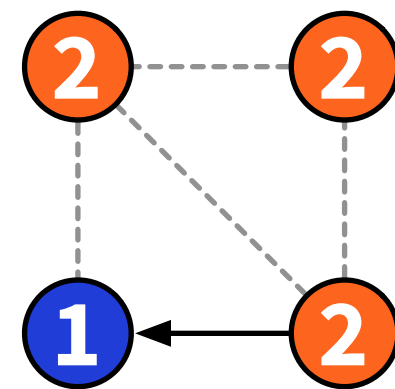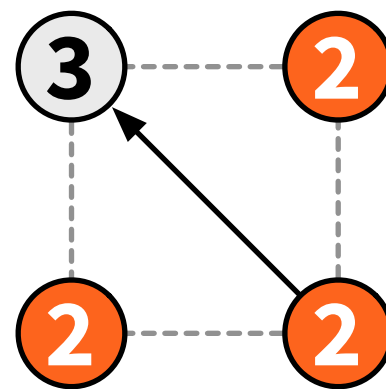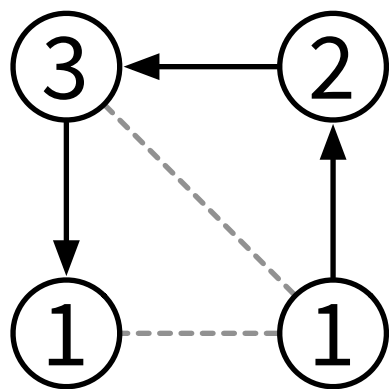 - all in parallel
  - each node has Δ roles
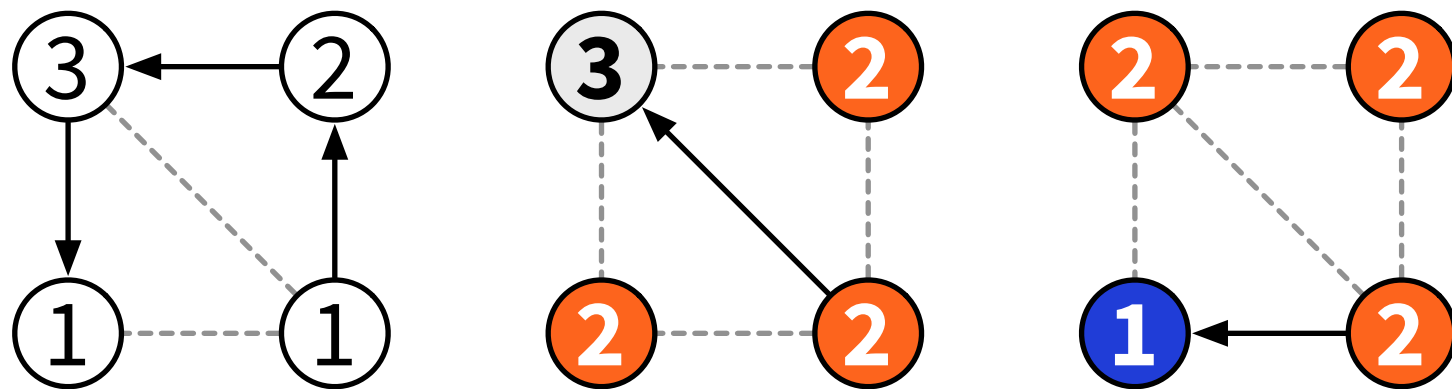
Algorithm BDColour:
Fast graph colouring

- **Merge pseudoforests in time $O(\Delta^2)$**
  - maintain colouring with $\Delta + 1$ colours
  - add first forest: trivial

Algorithm BDColour:
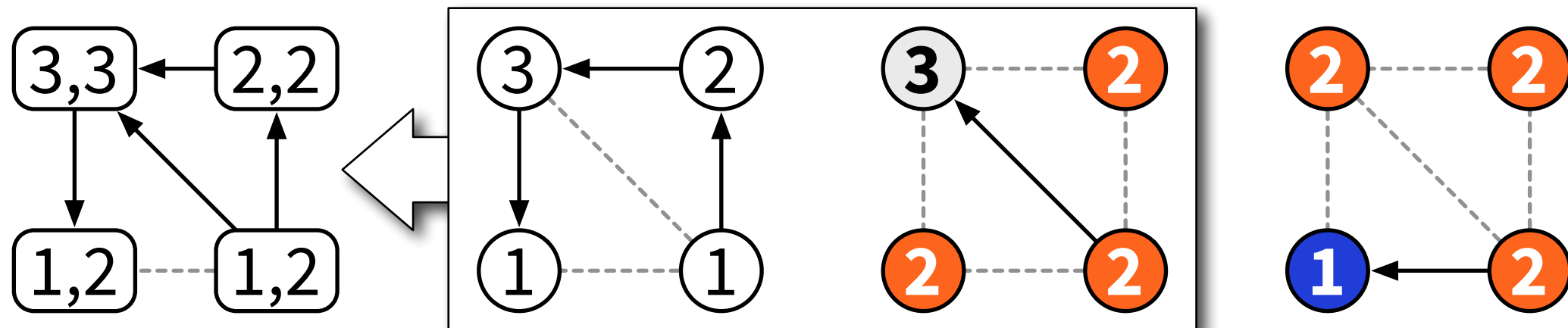# Fast graph colouring

- **Merge pseudoforests in time $O(\Delta^2)$**
  - maintain colouring with $\Delta + 1$ colours
  - add first forest: trivial
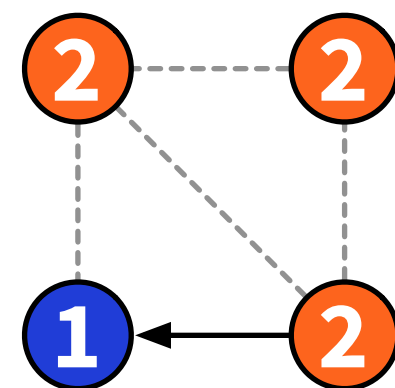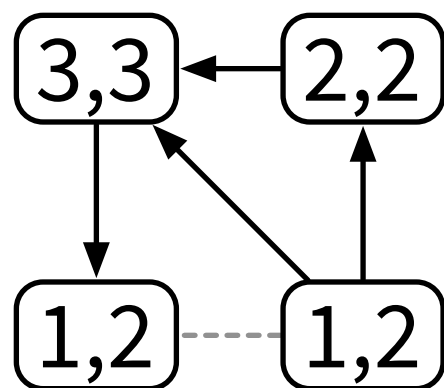
Algorithm BDColour:
Fast graph colouring

- **Merge pseudoforests in time $O(\Delta^2)$**
  - maintain colouring with $\Delta + 1$ colours
  - add first forest: trivial

Algorithm BDColour:
Fast graph colouring

- **Merge pseudoforests in time $O(\Delta^2)$**
  - maintain colouring with $\Delta + 1$ colours
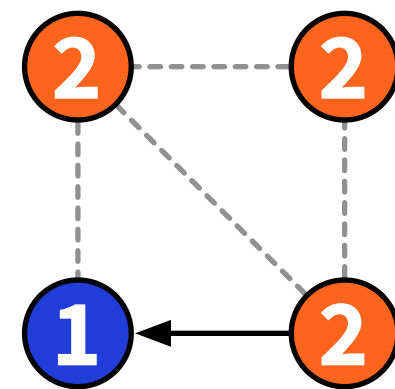 - add one forest $\rightarrow 3(\Delta + 1)$ colours

Algorithm BDColour:
Fast graph colouring

- **Merge pseudoforests in time $O(\Delta^2)$**
  - maintain colouring with $\Delta + 1$ colours
  - add one forest → $3(\Delta + 1)$ colours

Algorithm BDColour:
Fast graph colouring

- **Merge pseudoforests in time $O(\Delta^2)$**

  - maintain colouring with $\Delta + 1$ colours

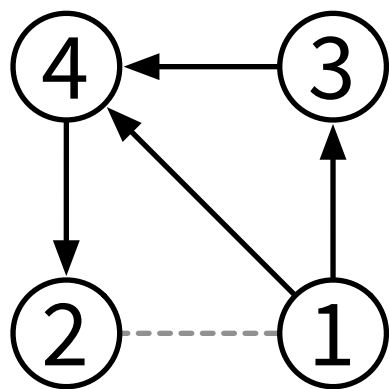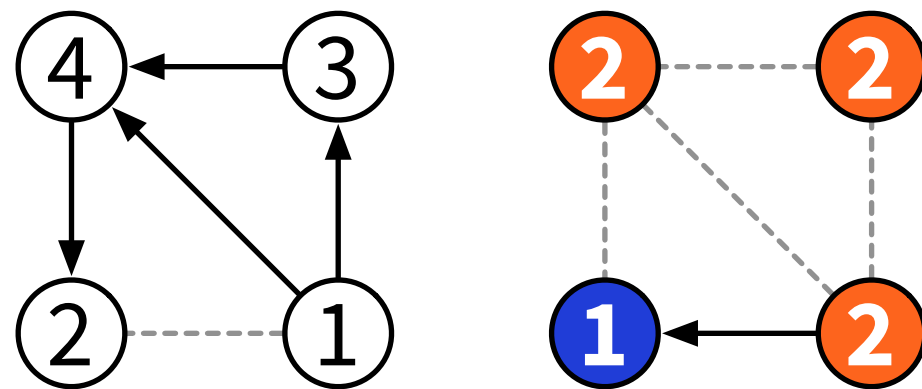 - add one forest $\rightarrow 3(\Delta + 1)$ colours $\rightarrow$ reduce
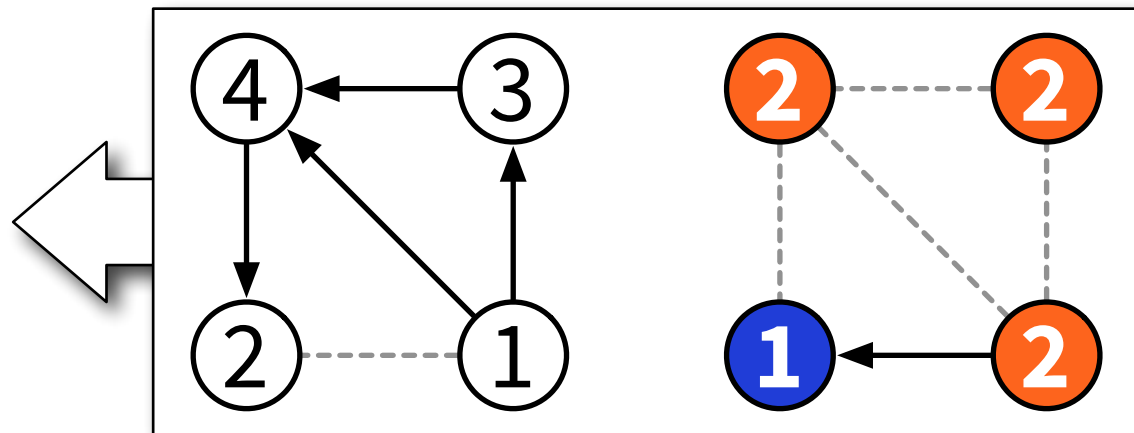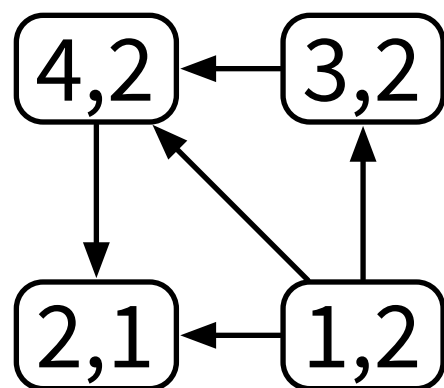
Algorithm BDColour:
Fast graph colouring

- **Merge pseudoforests in time $O(\Delta^2)$**
  - maintain colouring with $\Delta + 1$ colours
  - add one forest → $3(\Delta + 1)$ colours → reduce

Algorithm BDColour:
Fast graph colouring

- **Merge pseudoforests in time $O(\Delta^2)$**
  - maintain colouring with $\Delta + 1$ colours
  - add one forest $\to 3(\Delta + 1)$ colours

Algorithm BDColour:
Fast graph colouring

- **Merge pseudoforests in time $O(\Delta^2)$**
  - maintain colouring with $\Delta + 1$ colours
  - add one forest $\to 3(\Delta + 1)$ colours

Algorithm BDColour:
Fast graph colouring

- **Merge pseudoforests in time $O(\Delta^2)$**
  - maintain colouring with $\Delta + 1$ colours
  - add one forest $\to 3(\Delta + 1)$ colours $\to$ reduce

Algorithm BDColour:
Fast graph colouring

- **Merge pseudoforests in time $O(\Delta^2)$**
  - maintain colouring with $\Delta + 1$ colours
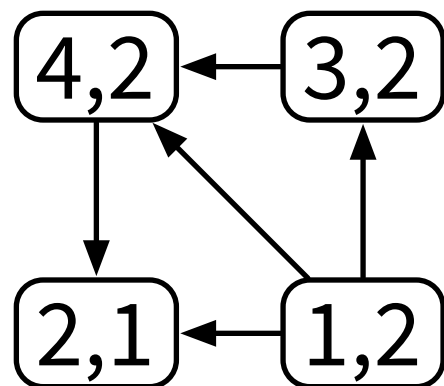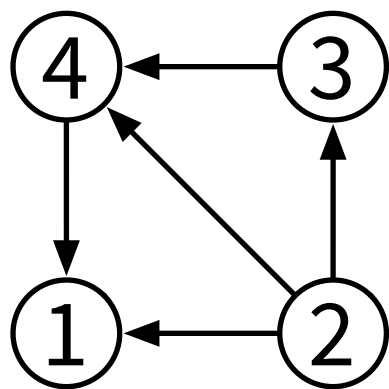 - add one forest $\rightarrow 3(\Delta + 1)$ colours $\rightarrow$ reduce

- **Each merge + reduce takes $O(\Delta)$ rounds**

- **There are $O(\Delta)$ such steps**

Algorithm BDColour:
Fast graph colouring

- **Unique identifiers → orientation**

- **Port numbers → partition edges
  in Δ directed pseudoforests**

- **3-colour pseudoforests in time $O(\log^* n)$**

- **Merge pseudoforests in time $O(\Delta^2)$**

Summary:
LOCAL model

- **Unique identifiers**

- **Everything can be computed**

- **What can be computed fast?**
  - example: graph colouring

Summary:
LOCAL model

- **Unique identifiers**

- **Everything can be computed**
  - cheating with large messages
  - what if we can only use small messages?
  - this is covered next week…

- **Weeks 1–2: informal introduction**

  - network = path

- **Week 3: graph theory**

- **Weeks 4–7: models of computing**

  - what can be computed (efficiently)?

- **Weeks 8–11: lower bounds**

  - what cannot be computed (efficiently)?

- **Week 12: recap**