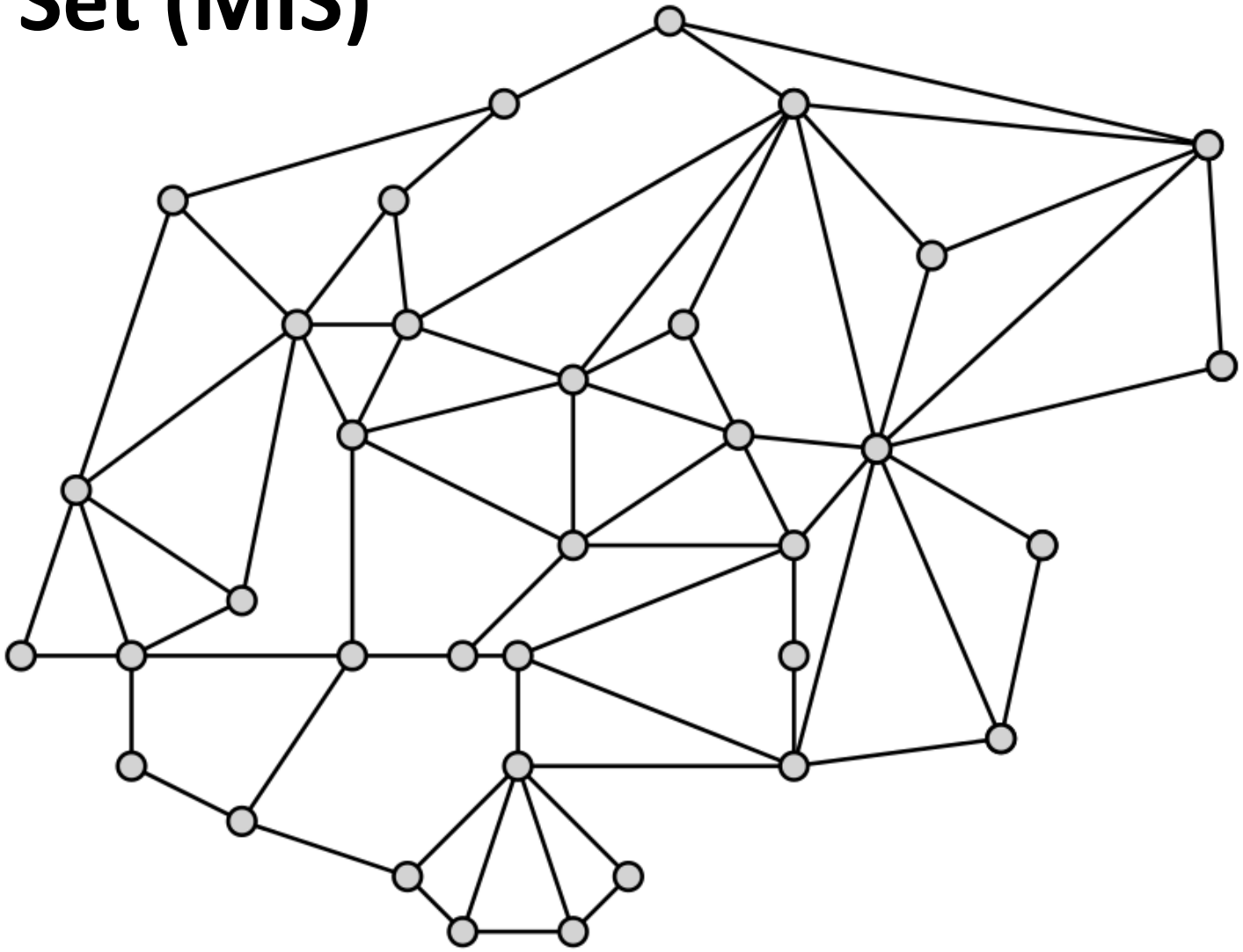
Tight Analysis of Parallel Randomized Greedy MIS

Manuela Fischer and **Andreas Noever**
ETH Zurich

Maximal Independent Set (MIS)

Maximal Independent Set (MIS)

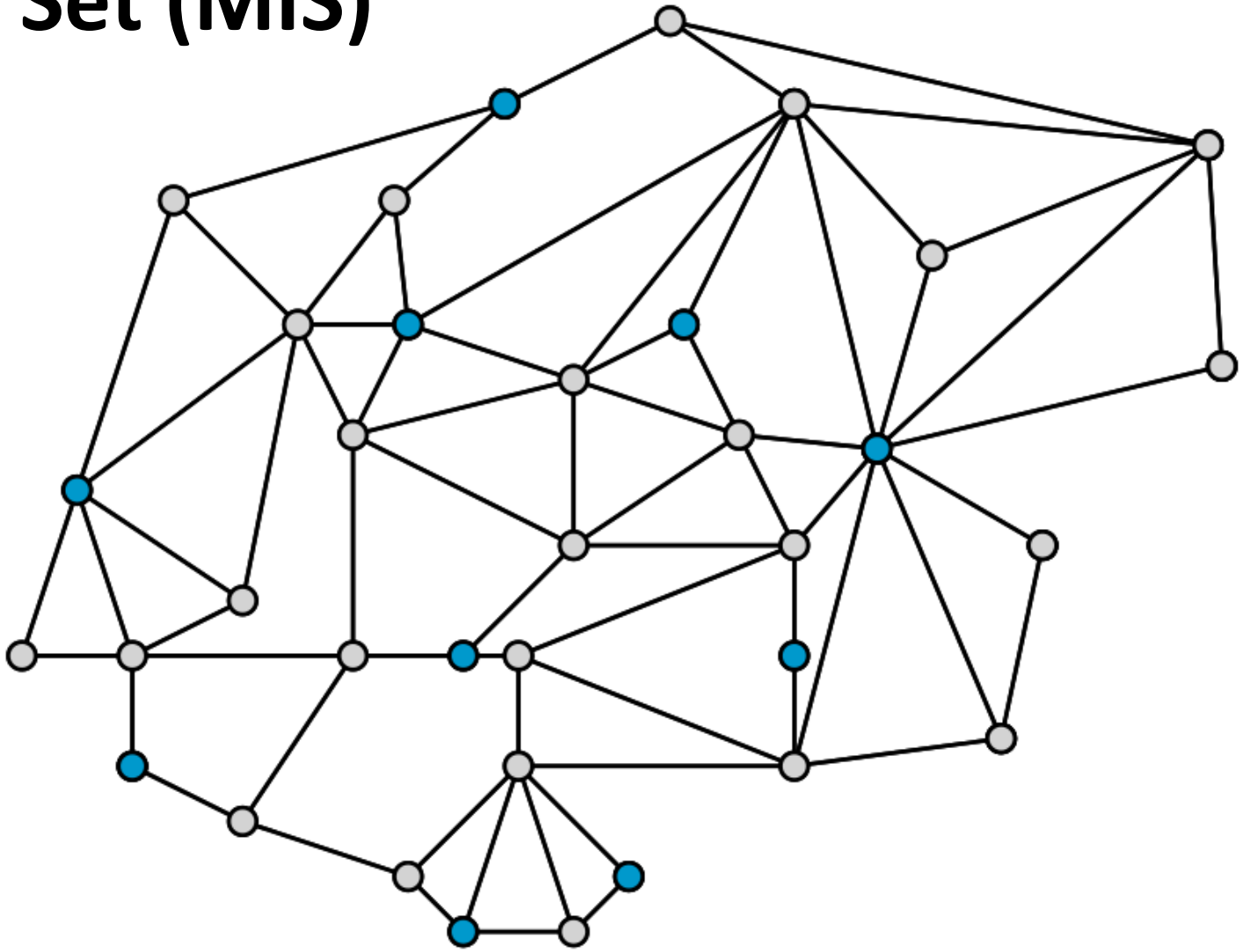
arbitrary graph G on n nodes



Maximal Independent Set (MIS)

arbitrary graph G on n nodes

<u>Independent:</u>
set of non-adjacent nodes

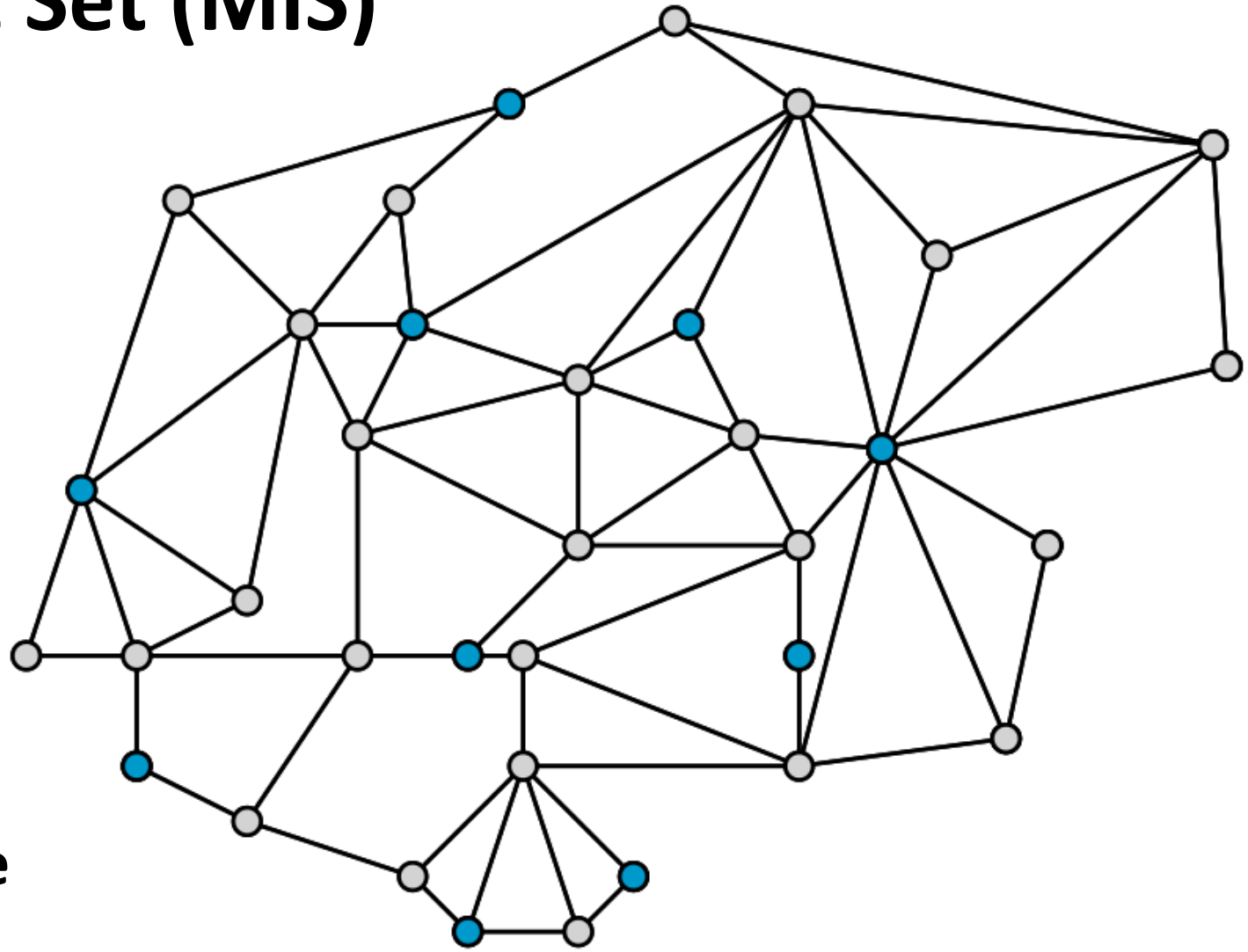


Maximal Independent Set (MIS)

arbitrary graph G on n nodes

<u>Independent:</u>
set of non-adjacent nodes

<u>Maximal:</u>
no node can be added
without violating independence

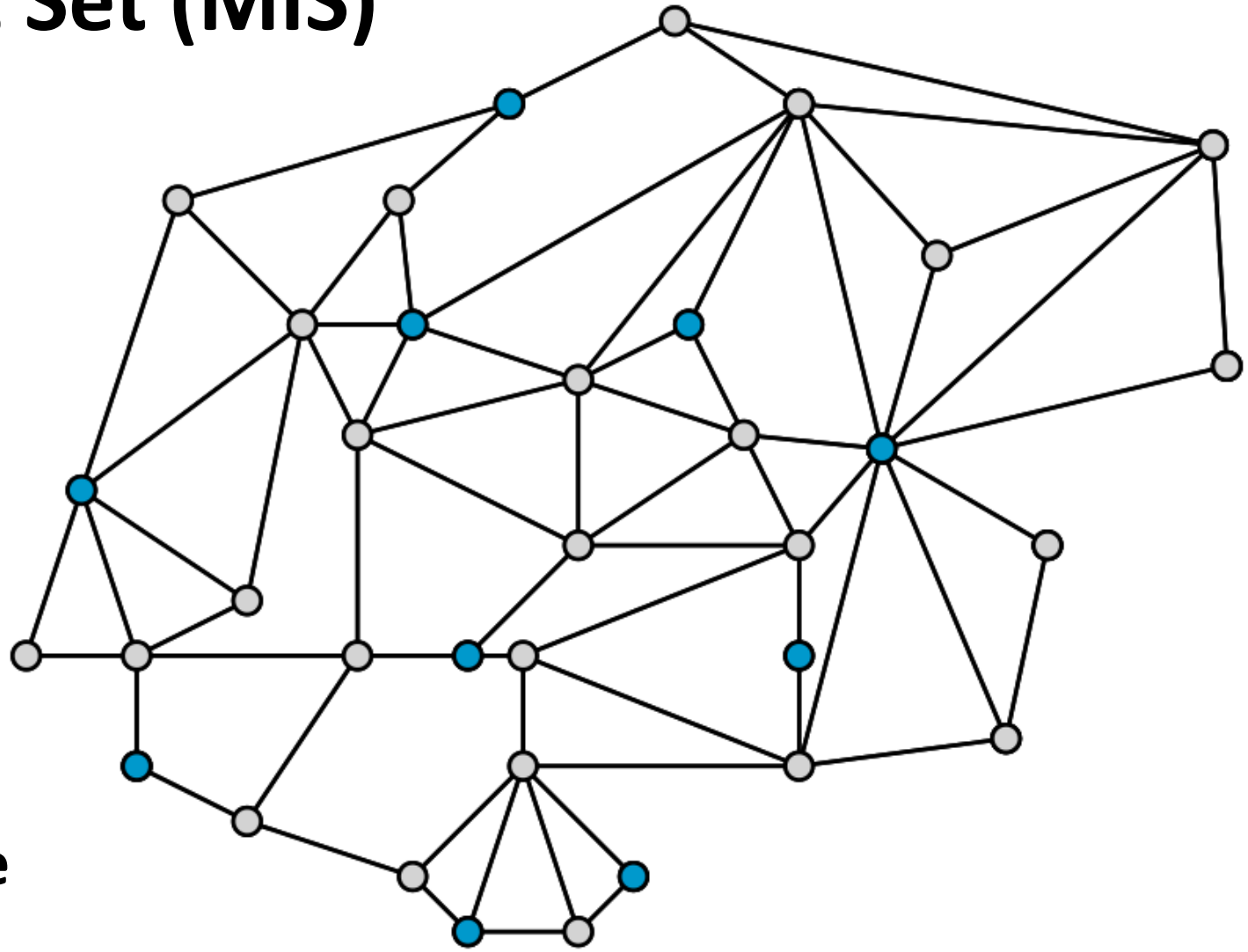


Maximal Independent Set (MIS)

arbitrary graph G on n nodes

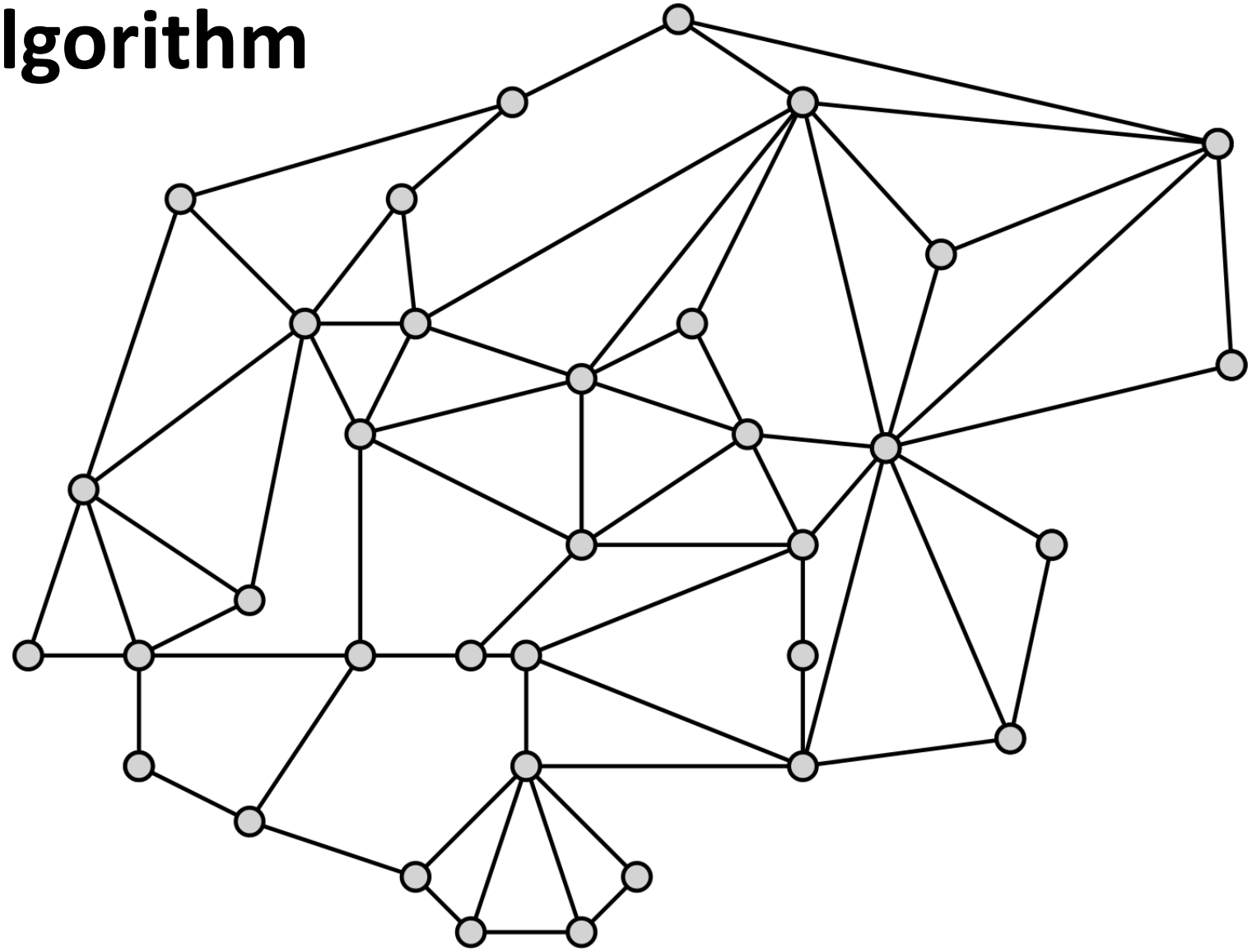
<u>Independent:</u>
set of non-adjacent nodes

<u>Maximal:</u> not maximum!
no node can be added
without violating independence



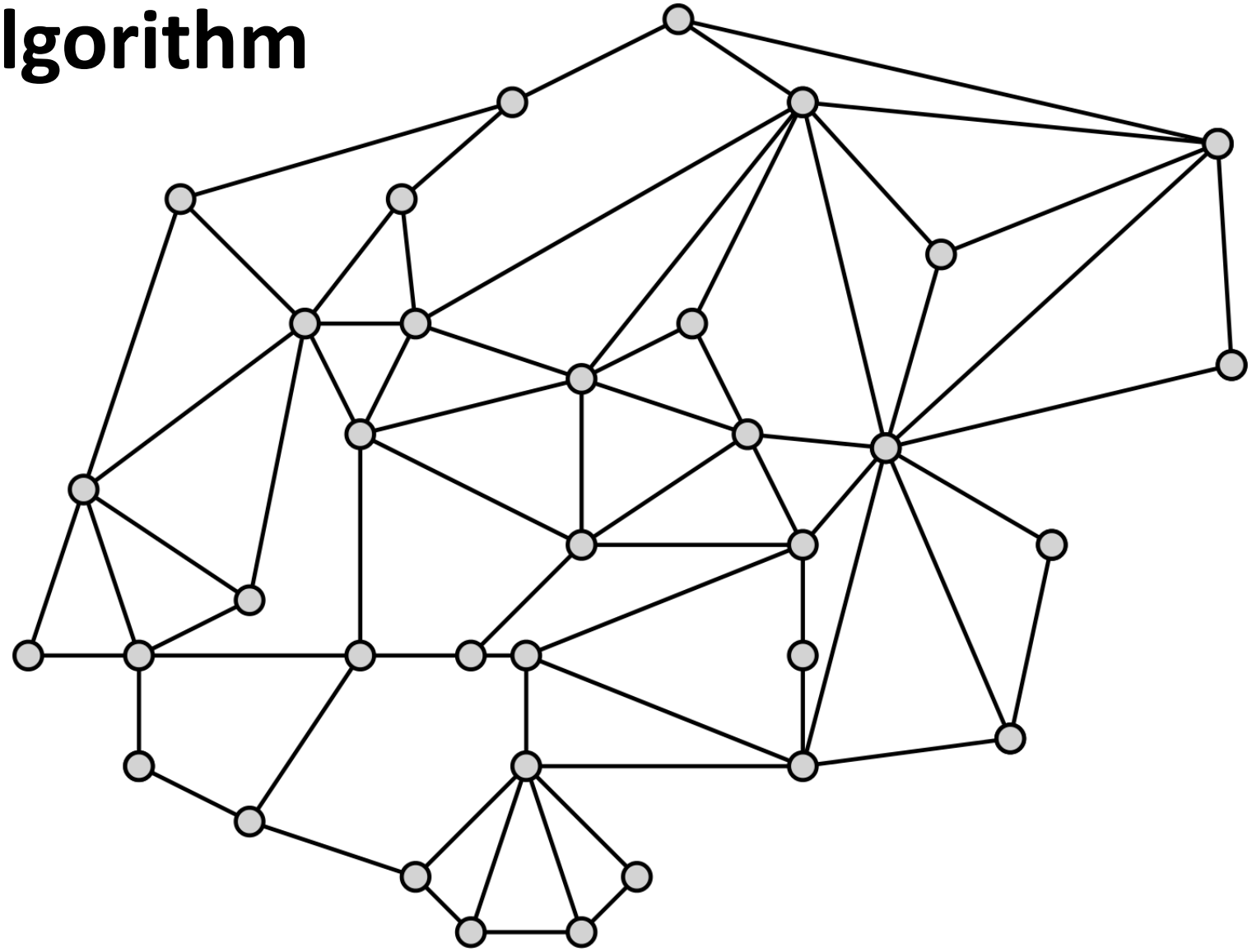
Sequential Greedy Algorithm

Sequential Greedy Algorithm



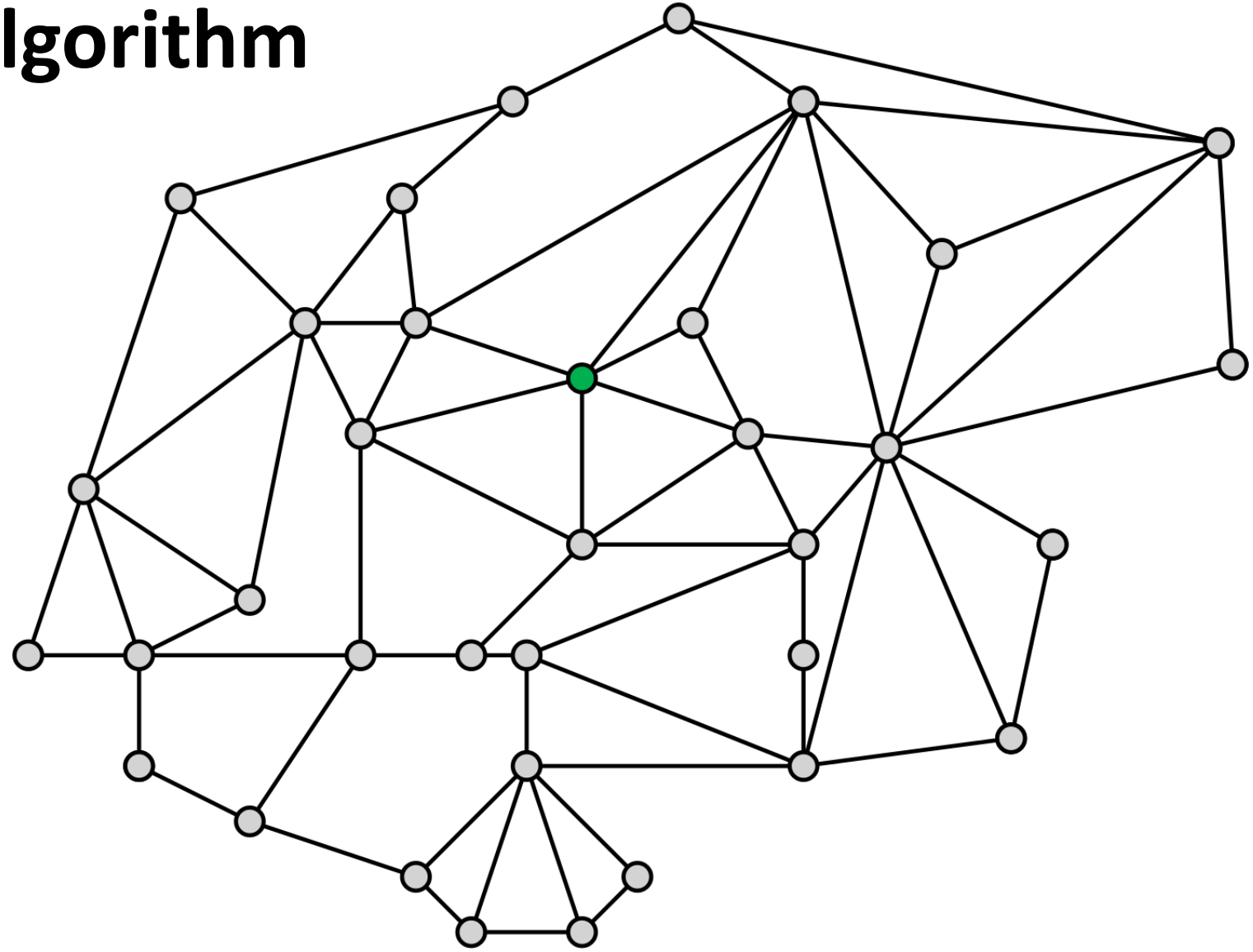
Sequential Greedy Algorithm

For an arbitrary order of nodes,



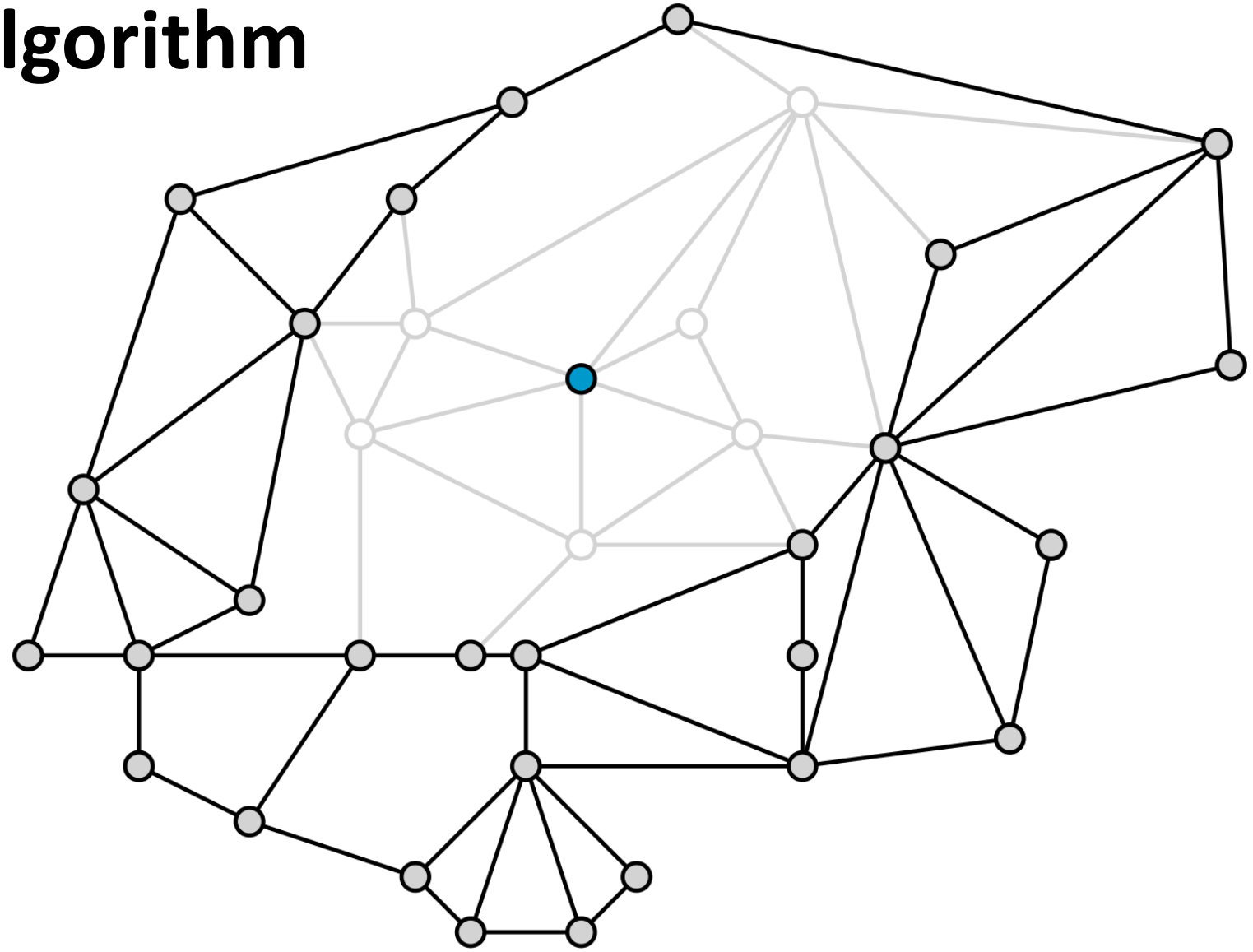
Sequential Greedy Algorithm

For an arbitrary order of nodes,
if current node is still alive,



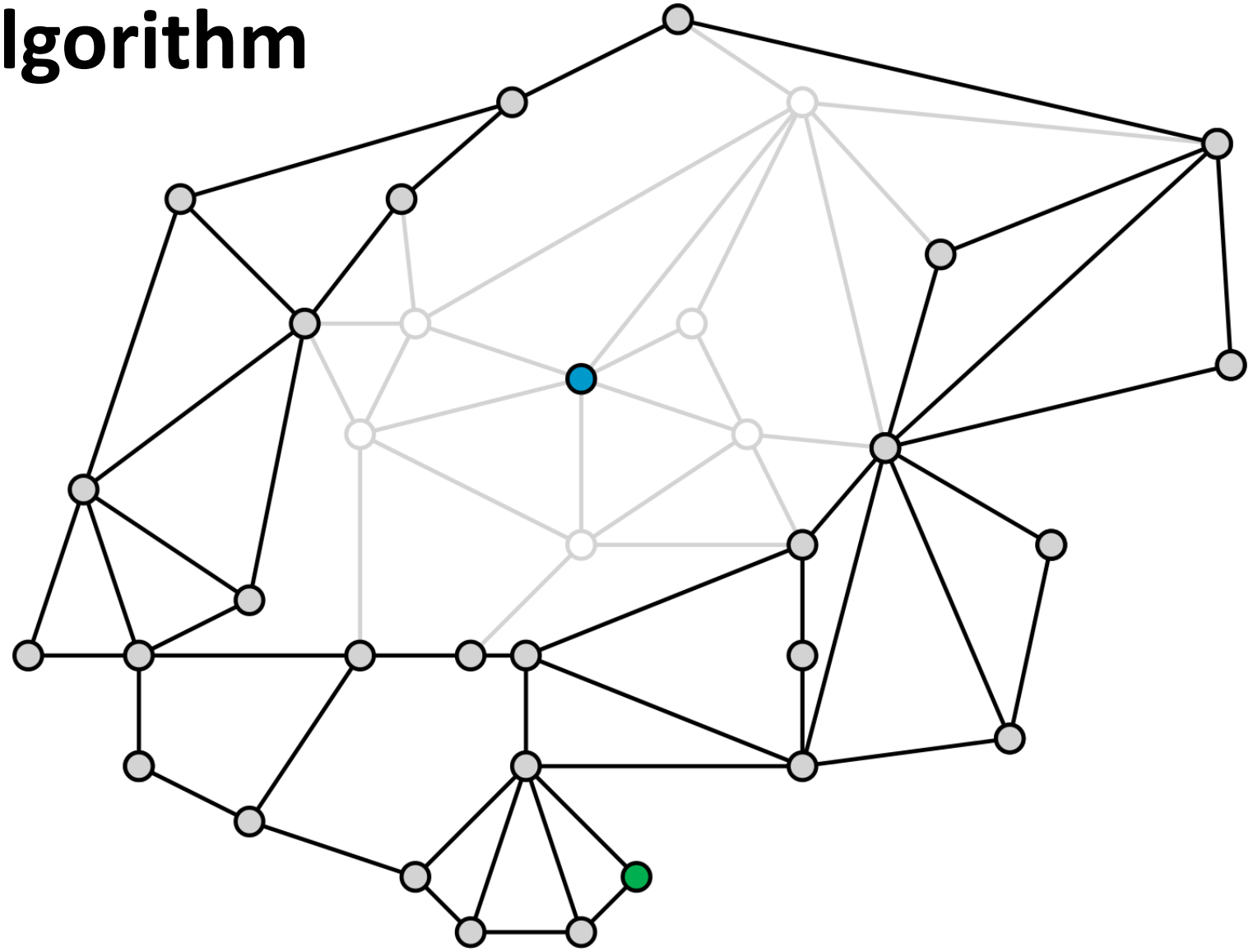
Sequential Greedy Algorithm

For an arbitrary order of nodes,
if current node is still alive,
add it to MIS,
and remove it
along with its neighbors.



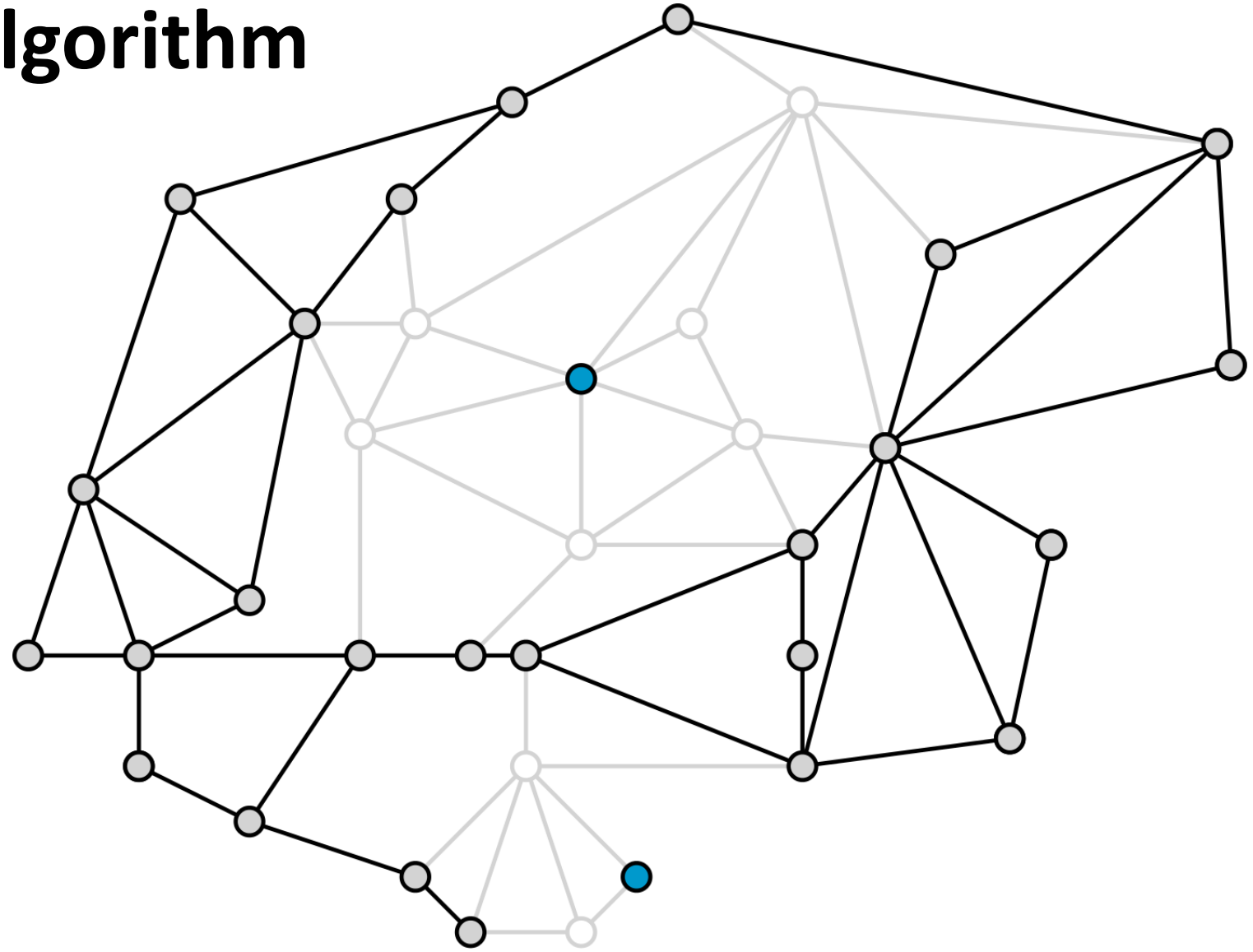
Sequential Greedy Algorithm

For an arbitrary order of nodes,
if current node is still alive,
add it to MIS,
and remove it
along with its neighbors.



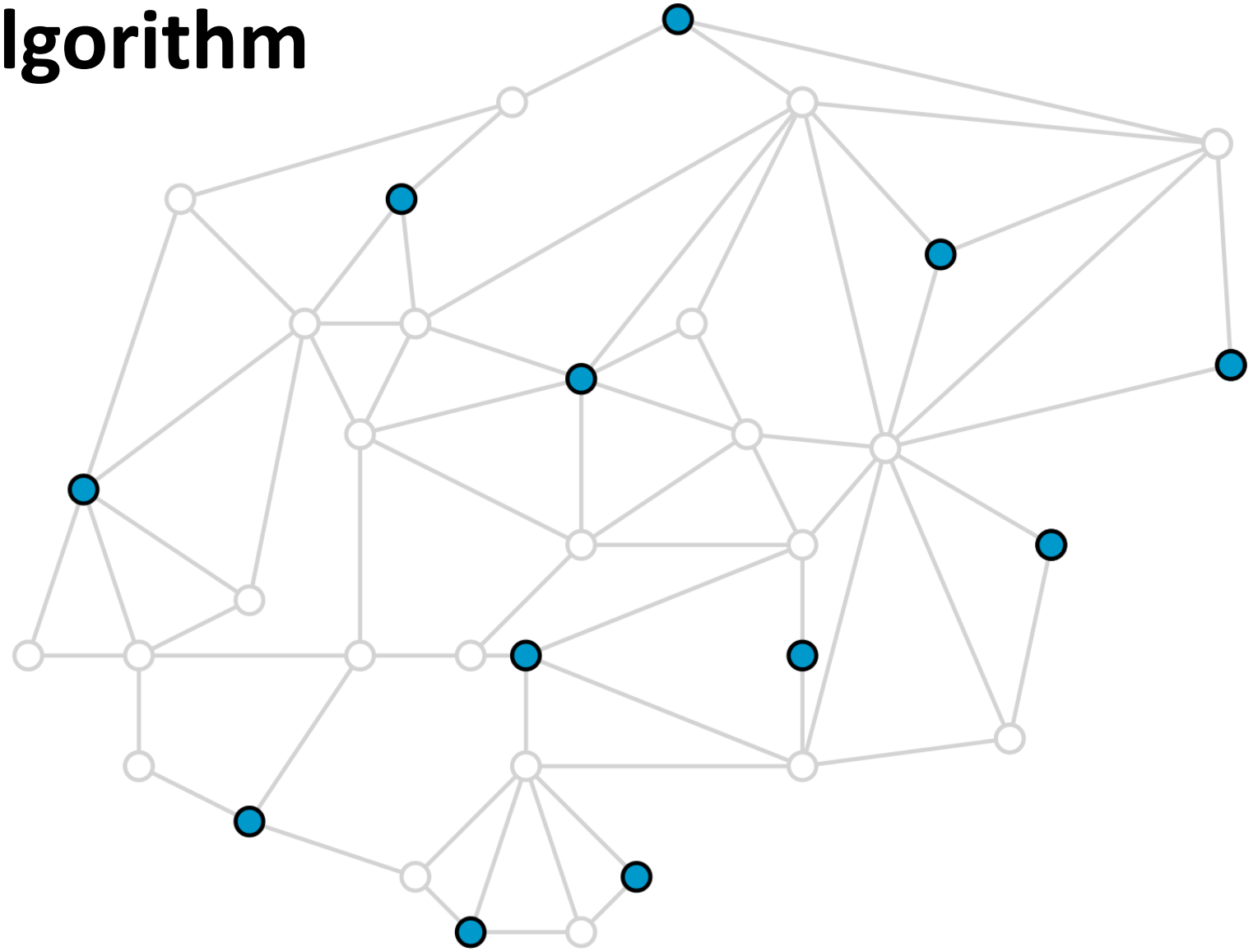
Sequential Greedy Algorithm

For an arbitrary order of nodes,
if current node is still alive,
add it to MIS,
and remove it
along with its neighbors.



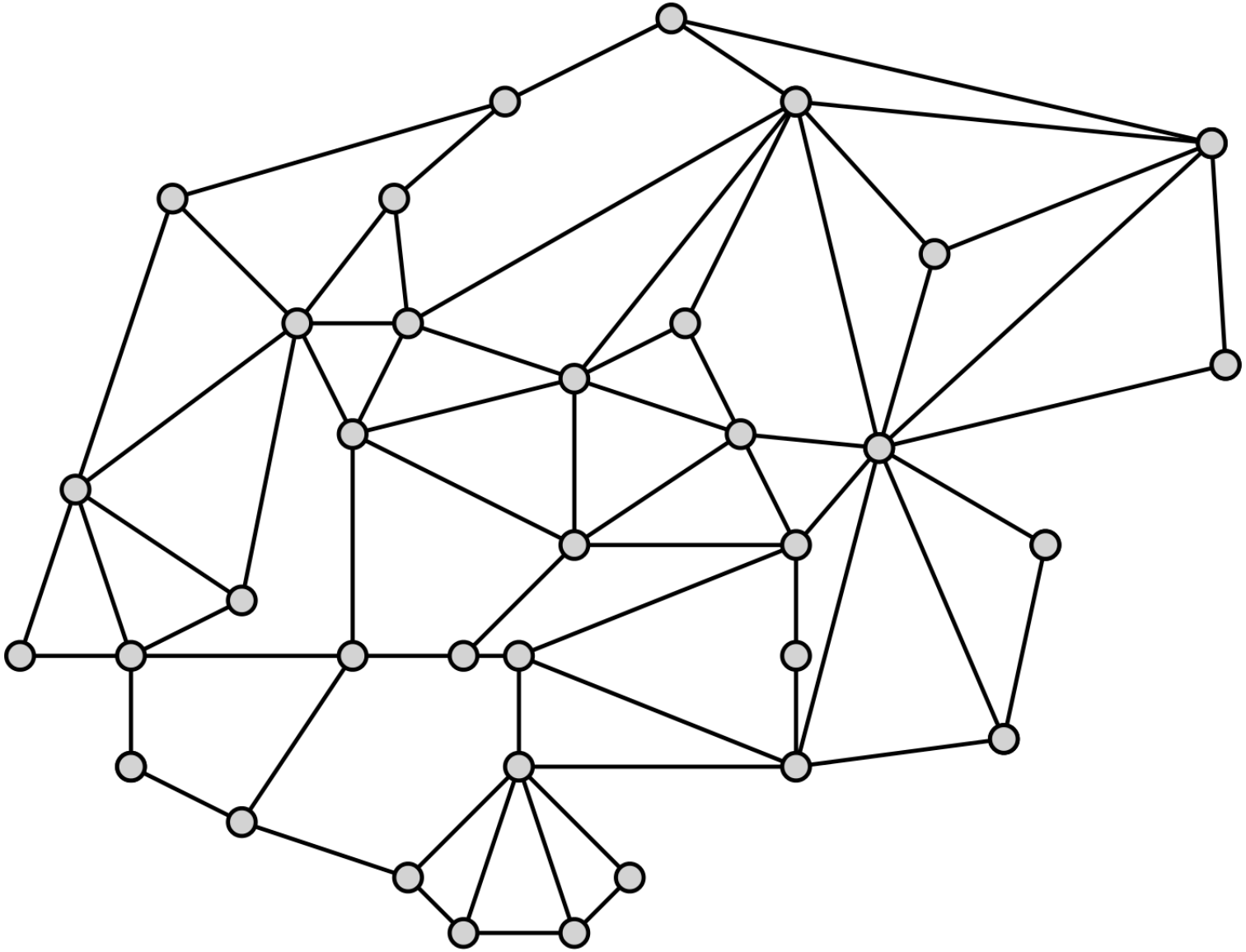
Sequential Greedy Algorithm

For an arbitrary order of nodes,
if current node is still alive,
add it to MIS,
and remove it
along with its neighbors.



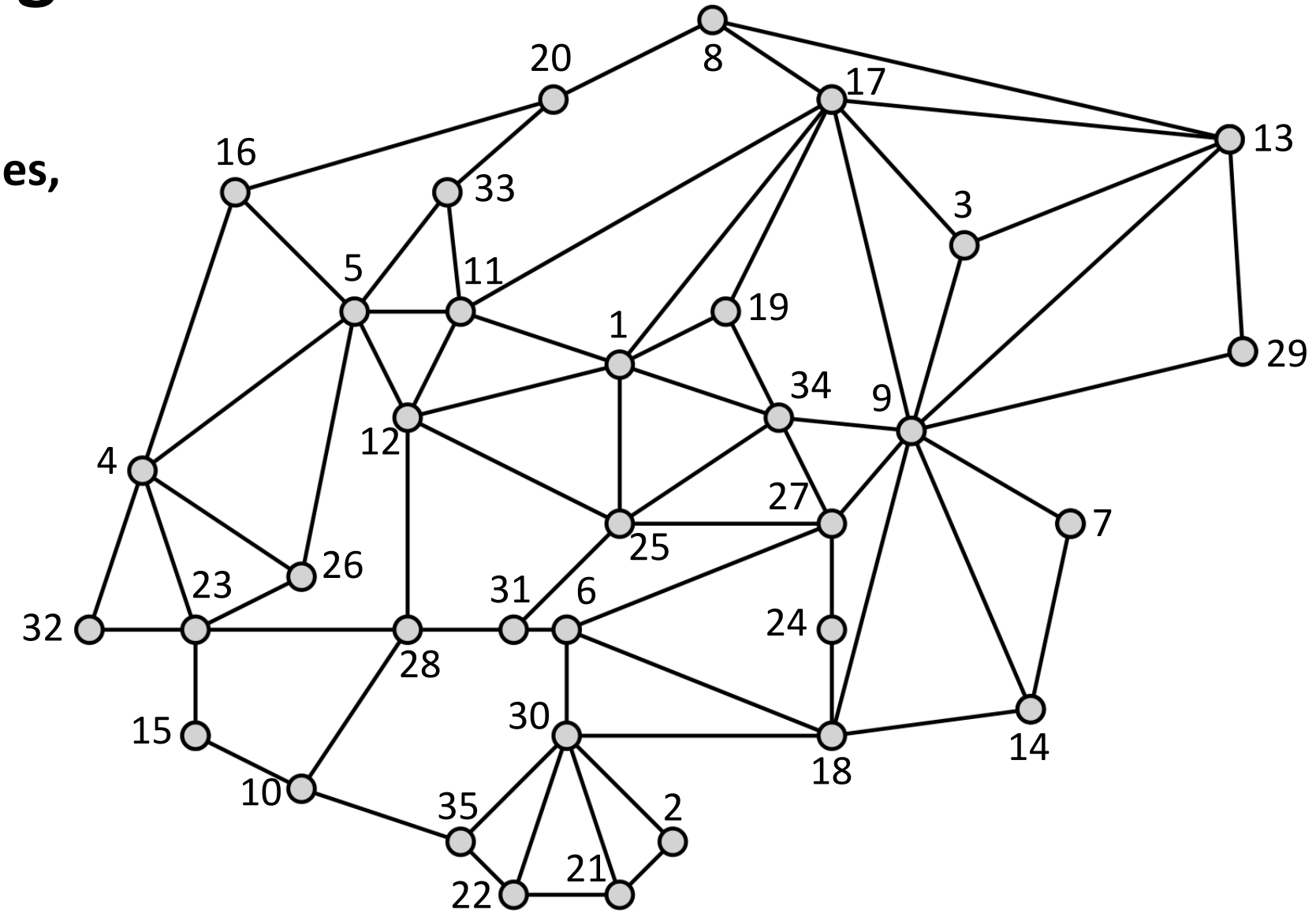
Parallel Greedy Algorithm

Parallel Greedy Algorithm



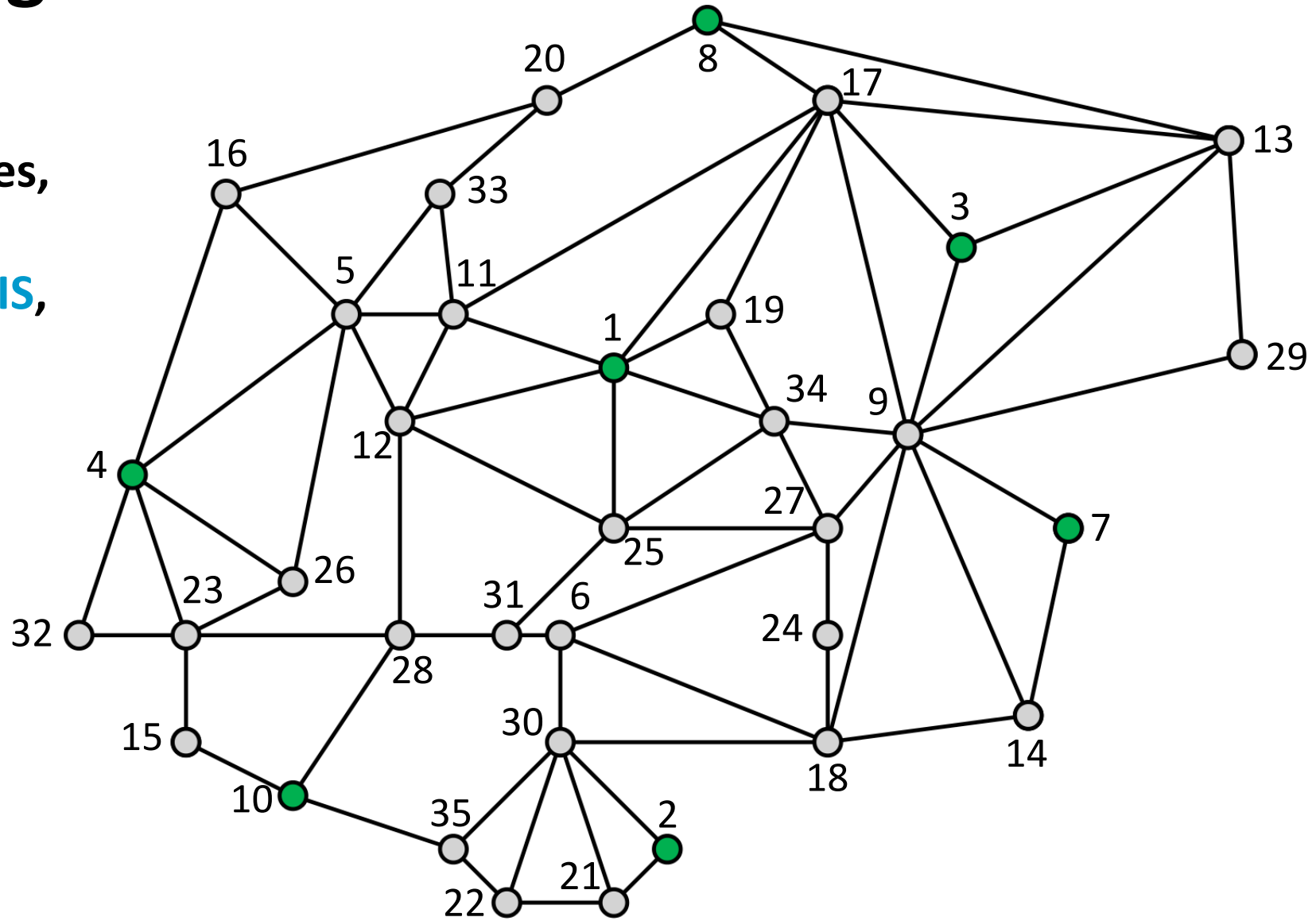
Parallel Greedy Algorithm

For an arbitrary order of nodes,



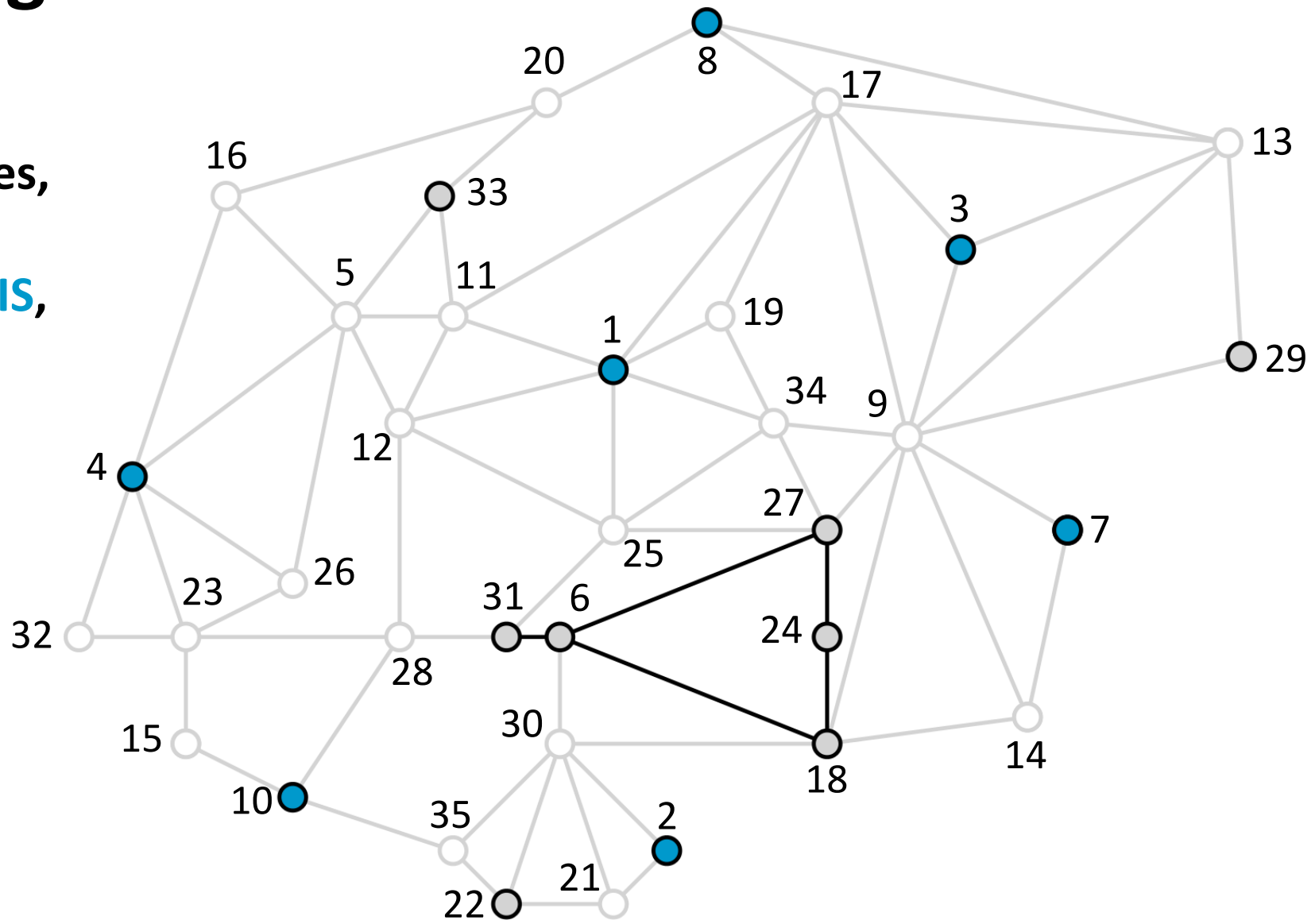
Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.



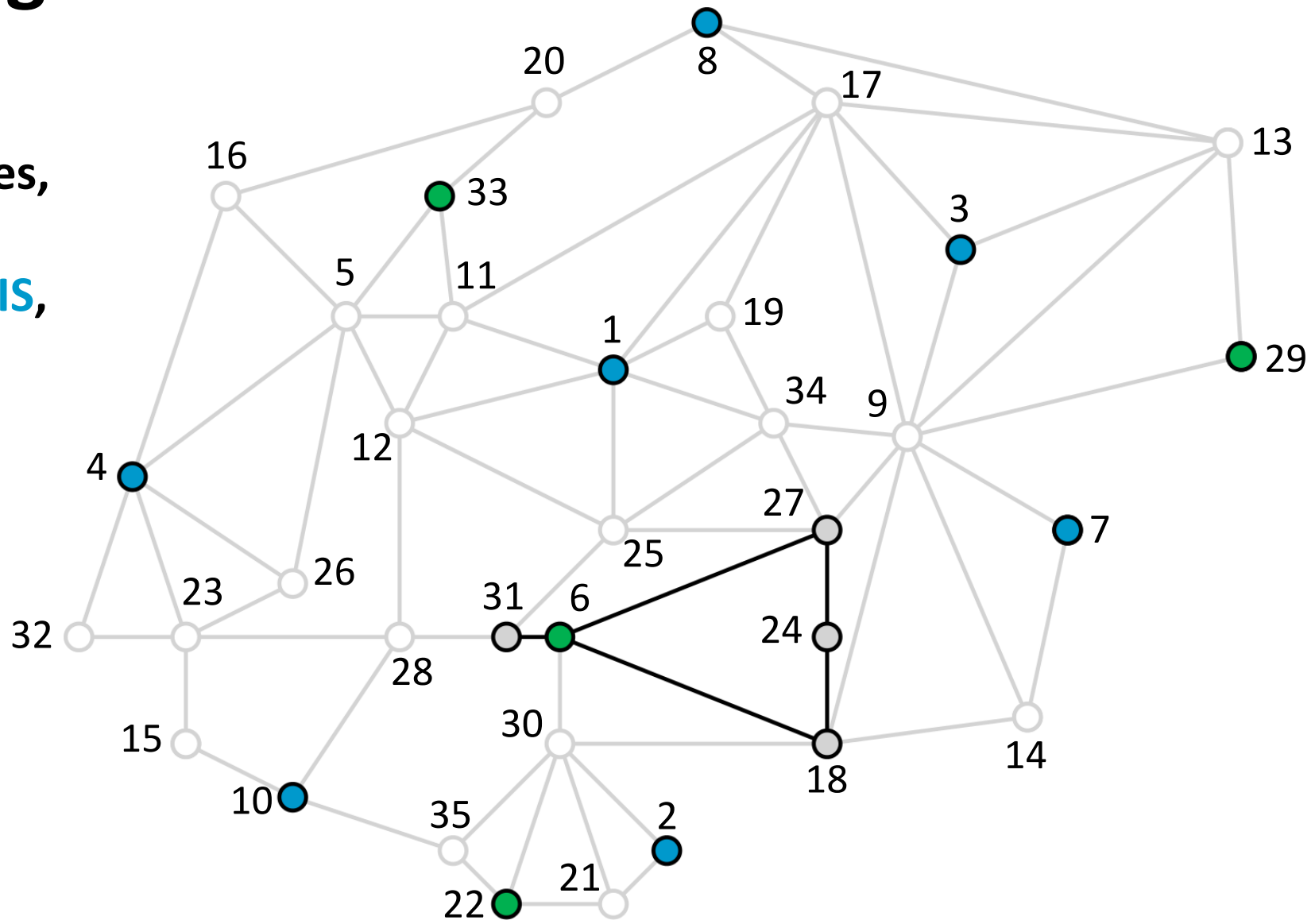
Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.



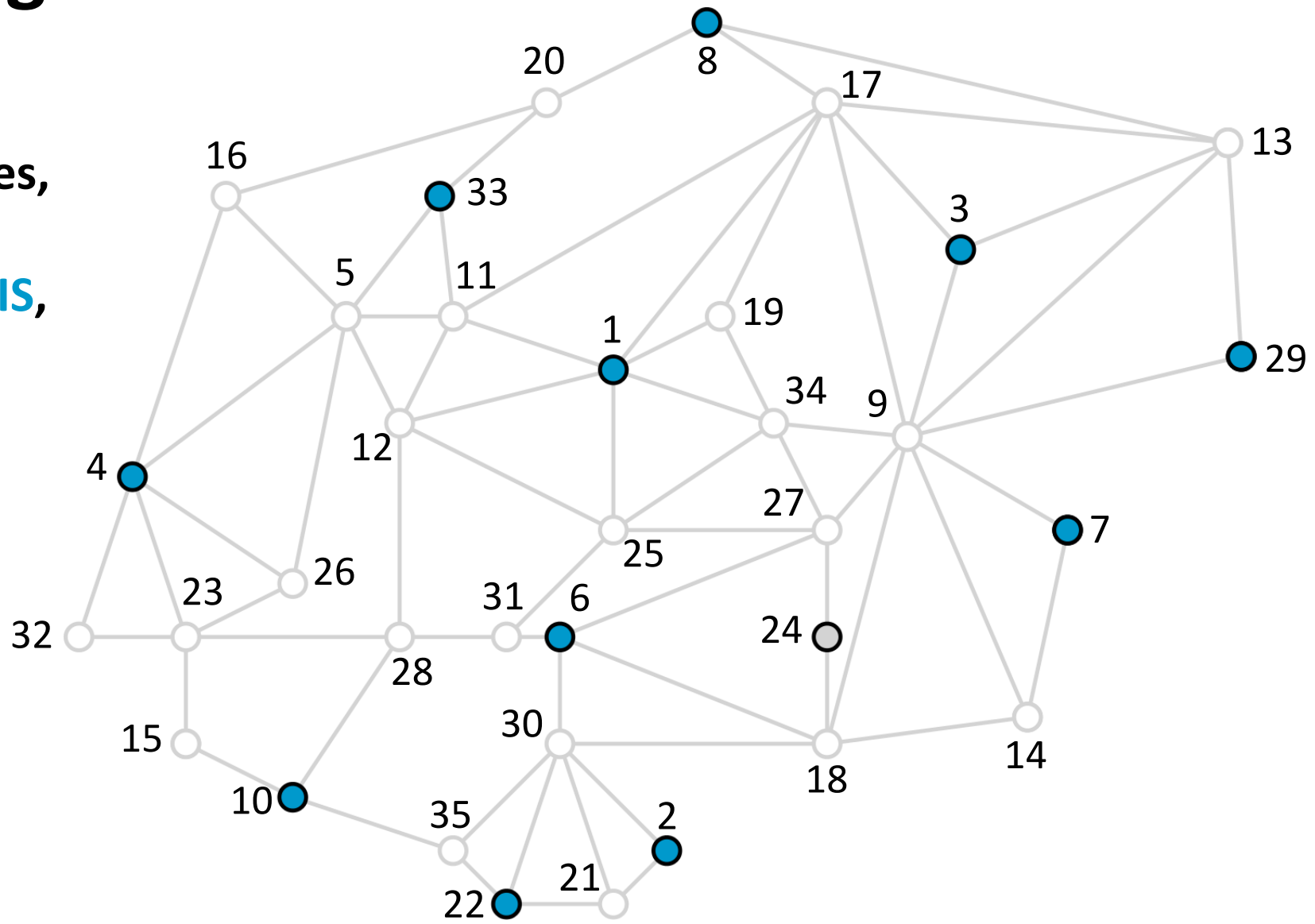
Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.



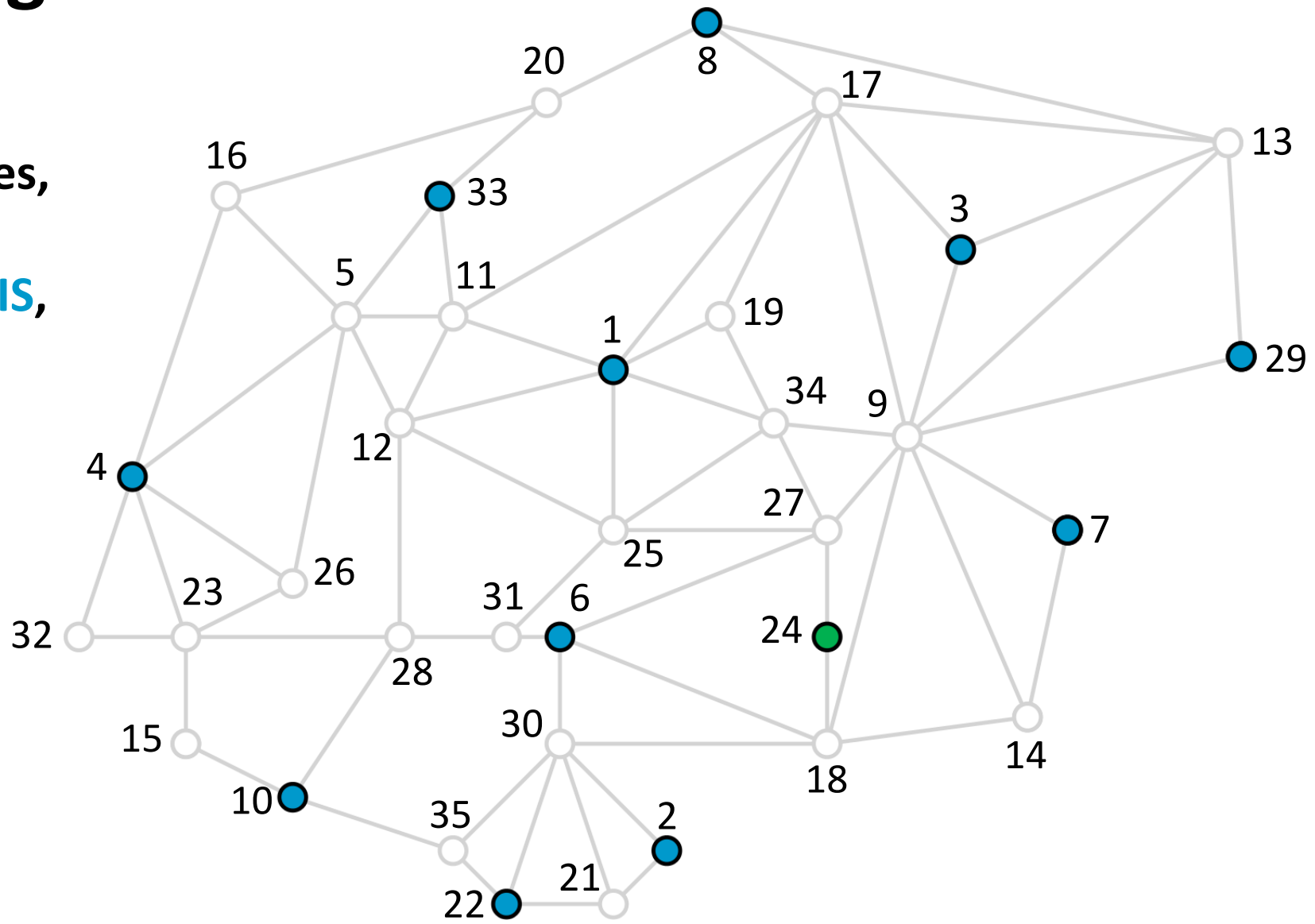
Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.



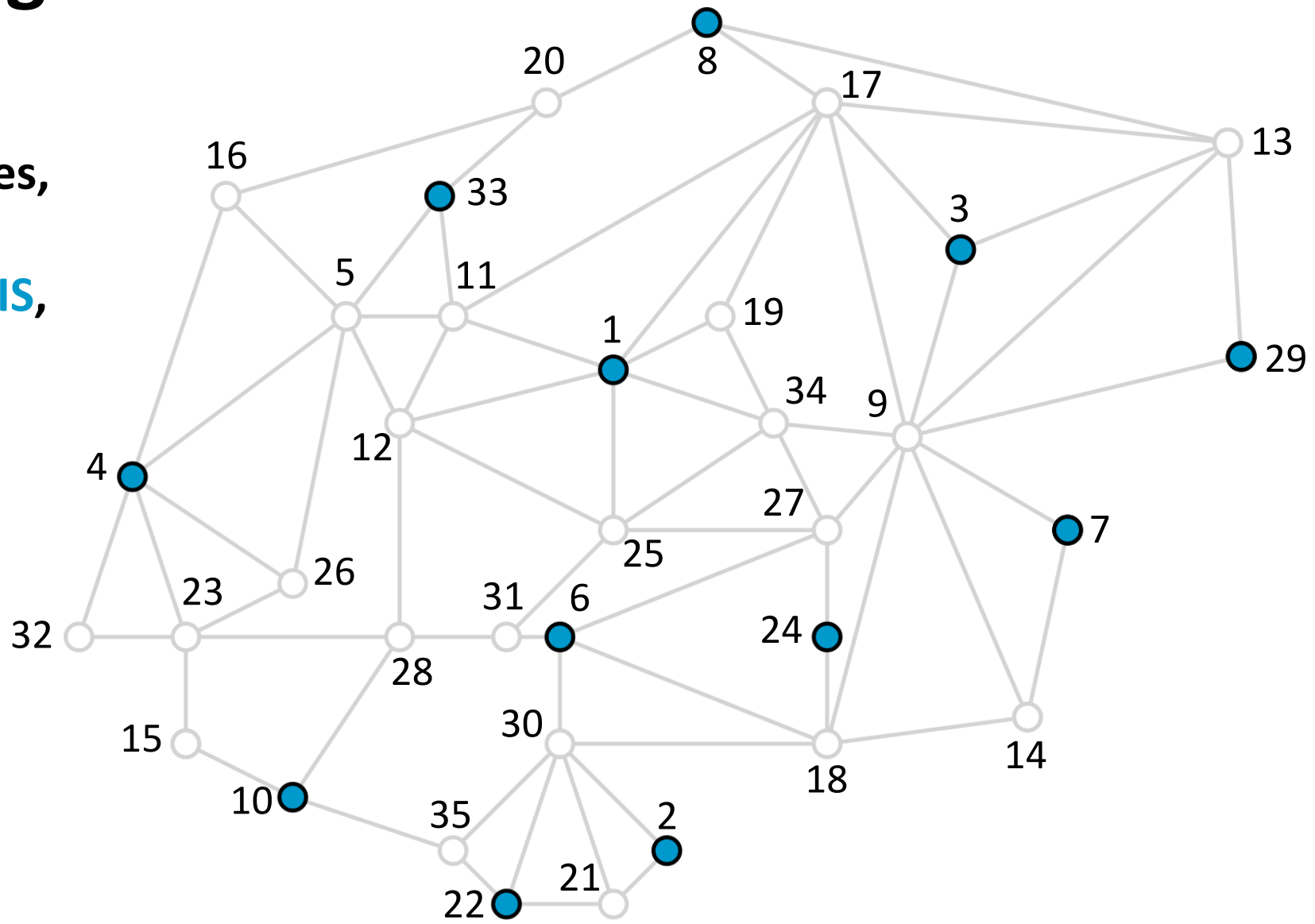
Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.



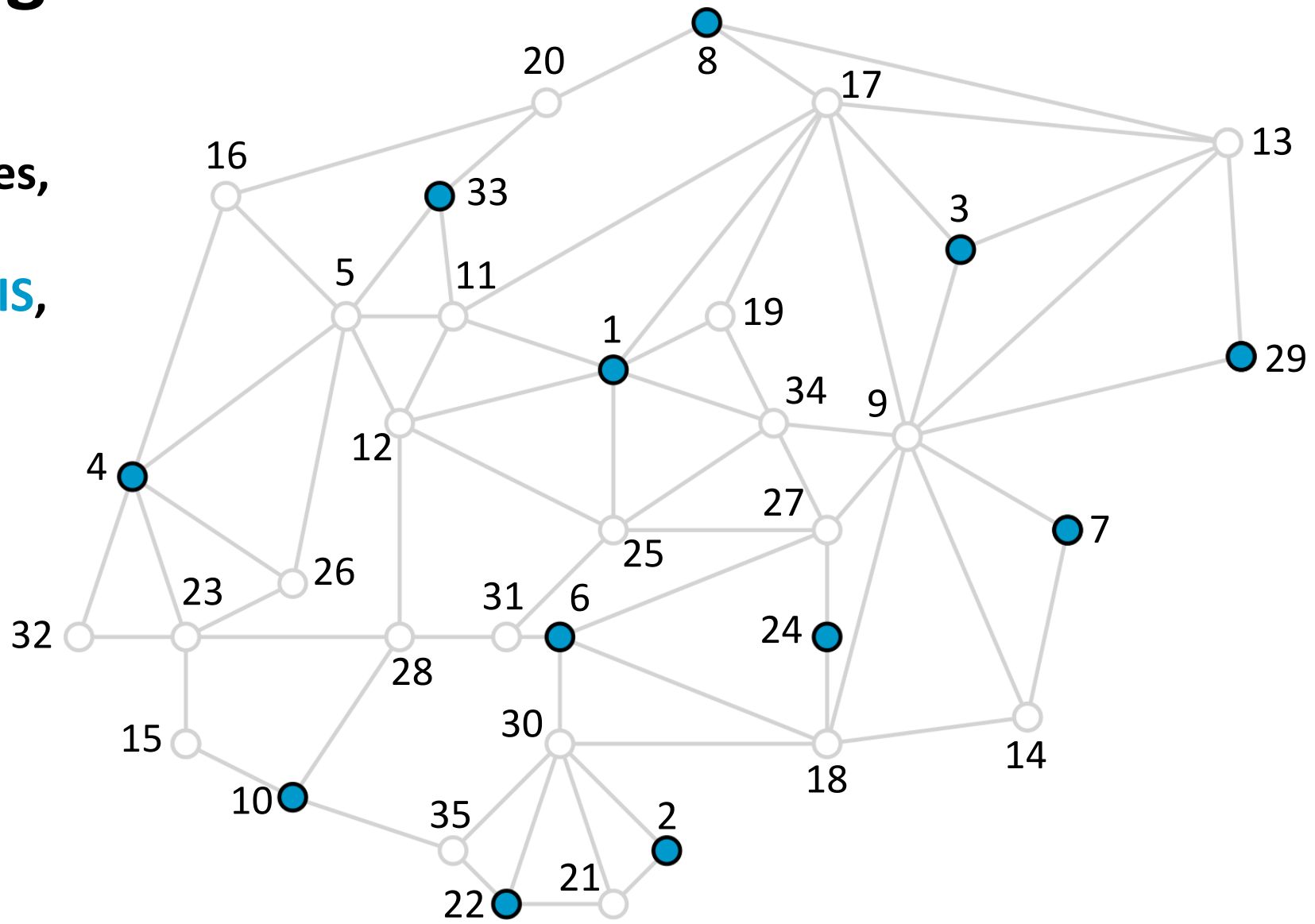
Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.



Parallel Greedy Algorithm

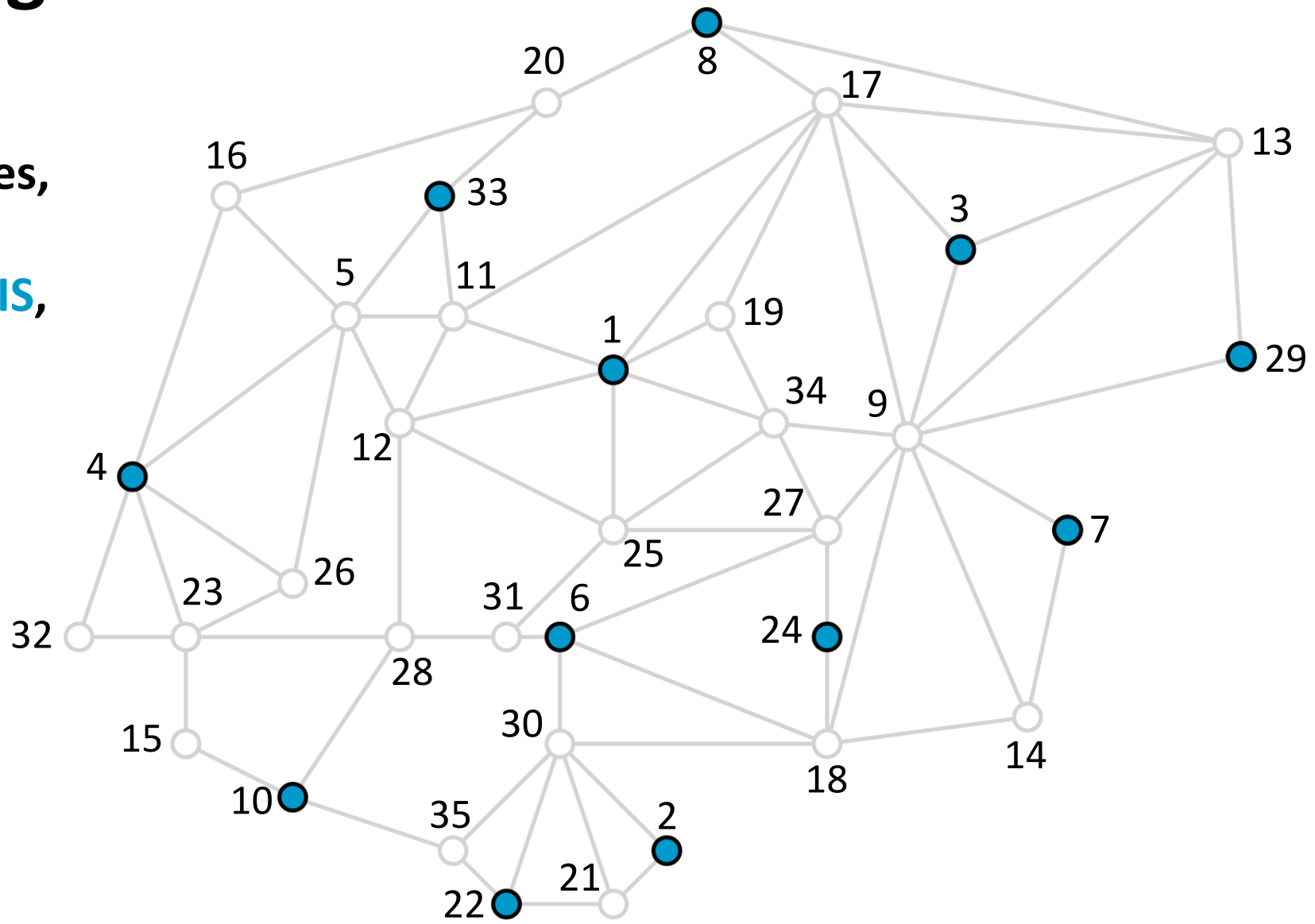
For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.



<u>Complexity</u>:
number of rounds

Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.

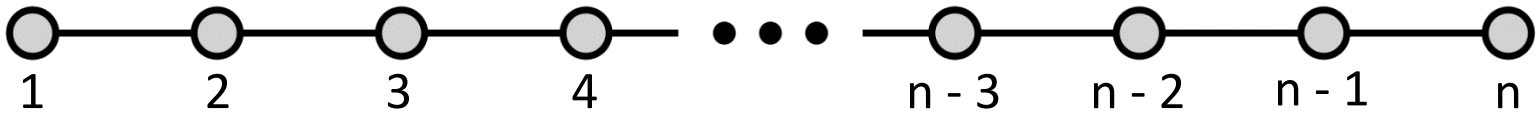


<u>Complexity</u>:
number of rounds

can take $\Omega(n)$ rounds!

Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.

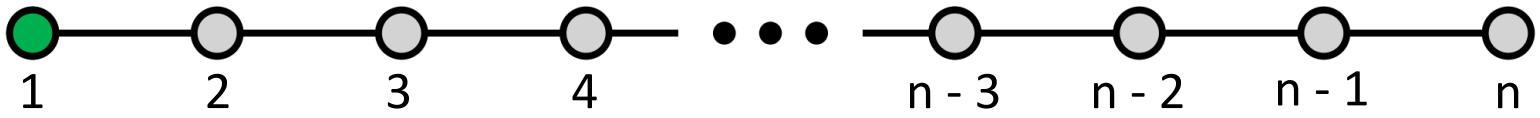


<u>Complexity</u>:
number of rounds

can take $\Omega(n)$ rounds!

Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.

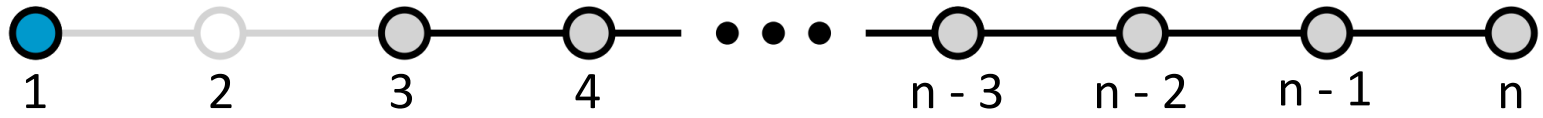


<u>Complexity</u>:
number of rounds

can take $\Omega(n)$ rounds!

Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.

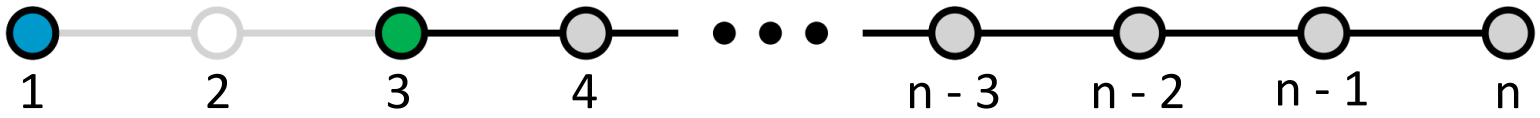


<u>Complexity</u>:
number of rounds

can take $\Omega(n)$ rounds!

Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.

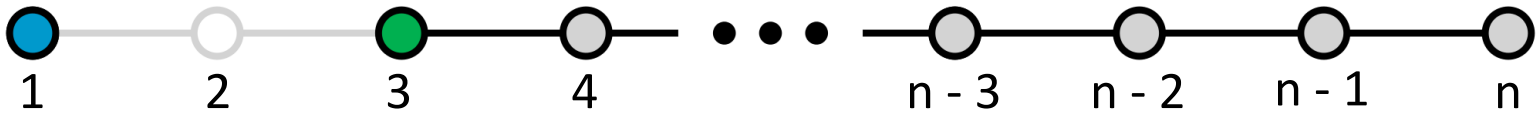


<u>Complexity</u>:
number of rounds

can take $\Omega(n)$ rounds!

Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.

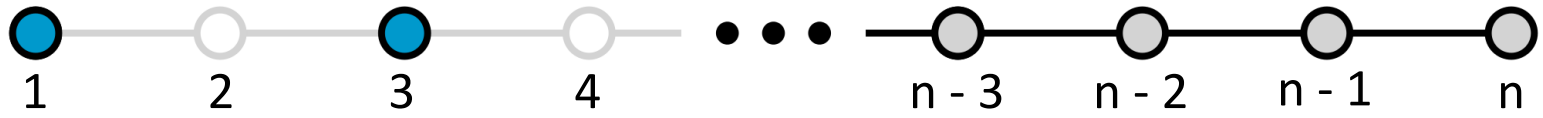


<u>Complexity</u>:
number of rounds

can take $\Omega(n)$ rounds!

Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.

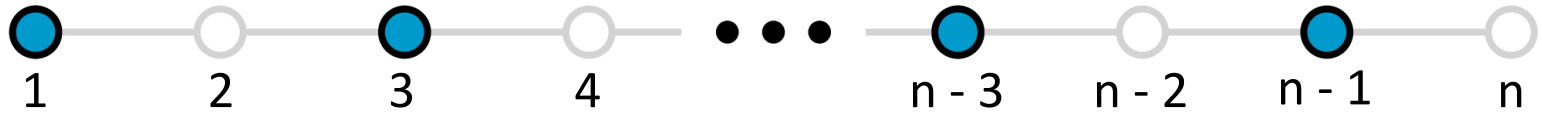


<u>Complexity</u>:
number of rounds

can take $\Omega(n)$ rounds!

Parallel Greedy Algorithm

For an arbitrary order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.

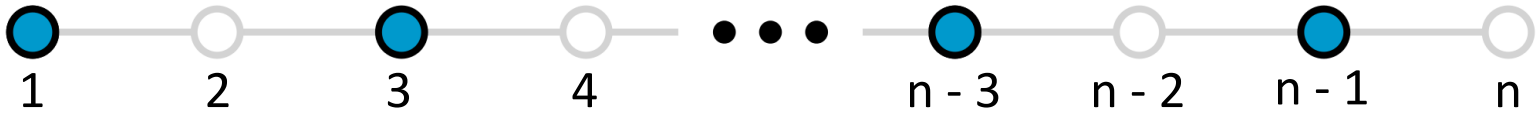


<u>Complexity</u>:
number of rounds

can take $\Omega(n)$ rounds!

Randomized
Parallel Greedy Algorithm

random
For an ~~arbitrary~~ order of nodes,
in each round,
every local minimum joins MIS,
and removes itself
along with its neighbors.



Complexity:
number of rounds

Randomized Parallel Greedy Algorithm

Randomized Parallel Greedy Algorithm

Random Graphs

Randomized Parallel Greedy Algorithm

Random Graphs $\mathbf{O}\left(\frac{\log^2 n}{\log \log n}\right)$ *Coppersmith, Raghavan, Tompa* [FOCS'87]

Randomized Parallel Greedy Algorithm

Random Graphs

$$\mathbf{O}\left(\frac{\log^2 n}{\log\log n}\right)$$ *Coppersmith, Raghavan, Tompa* [FOCS'87]

$$\mathbf{O}(\log n)$$ *Calkin, Frieze* [RSA'90]

Randomized Parallel Greedy Algorithm

Random Graphs

$$O\left(\frac{\log^2 n}{\log \log n}\right)$$ *Coppersmith, Raghavan, Tompa* [FOCS'87]

$$O(\log n)$$ *Calkin, Frieze* [RSA'90]

$$\Omega\left(\frac{\log n}{\log \log n}\right)$$ *Calkin, Frieze* [RSA'90]

Randomized Parallel Greedy Algorithm

Random Graphs

$$O\left(\frac{\log^2 n}{\log \log n}\right)$$ *Coppersmith, Raghavan, Tompa* [FOCS'87]

$$O(\log n)$$ *Calkin, Frieze* [RSA'90]

$$\Omega\left(\frac{\log n}{\log \log n}\right)$$ *Calkin, Frieze* [RSA'90]

General Graphs

Randomized Parallel Greedy Algorithm

Random Graphs	$\mathbf{O}\left(\frac{\log^2 n}{\log\log n}\right)$	*Coppersmith, Raghavan, Tompa* [FOCS'87]
	$\mathbf{O}(\log n)$	*Calkin, Frieze* [RSA'90]
	$\mathbf{\Omega}\left(\frac{\log n}{\log\log n}\right)$	*Calkin, Frieze* [RSA'90]
General Graphs	$\mathbf{O}(\log^2 n)$	*Blelloch, Fineman, Shun* [SPAA'12]

Randomized Parallel Greedy Algorithm

Random Graphs

$$O\left(\frac{\log^2 n}{\log\log n}\right)$$ *Coppersmith, Raghavan, Tompa* [FOCS'87]

$$O(\log n)$$ *Calkin, Frieze* [RSA'90]

$$\Omega\left(\frac{\log n}{\log\log n}\right)$$ *Calkin, Frieze* [RSA'90]

General Graphs

$$O(\log^2 n)$$ *Blelloch, Fineman, Shun* [SPAA'12]

$$\Omega\left(\frac{\log n}{\log\log n}\right)$$ *Calkin, Frieze* [RSA'90]

Our Results

Our Results

The **Randomized Parallel Greedy MIS** algorithm w.h.p. takes $\mathbf{O}(\mathbf{log}\, \boldsymbol{n})$ rounds on any n-node graph.

Our Results

The **Randomized Parallel Greedy MIS** algorithm w.h.p. takes $\mathbf{O}(\log n)$ rounds on any n-node graph.

This is **tight**: It takes $\mathbf{\Omega}(\log n)$ rounds.

Our Results

The **Randomized Parallel Greedy MIS** algorithm w.h.p. takes $\mathbf{O}(\log n)$ rounds on any n-node graph.

As fast as **Luby's** algorithm, but simpler.

This is **tight**: It takes $\mathbf{\Omega}(\log n)$ rounds.

Our Results

The **Randomized Parallel Greedy MIS** algorithm w.h.p. takes $\mathbf{O}(\log n)$ rounds on any n-node graph.

On **average**, Deterministic Parallel Greedy MIS is fast.

As fast as **Luby's** algorithm, but simpler.

This is **tight**: It takes $\mathbf{\Omega}(\log n)$ rounds.

Our Results

The **Randomized Parallel Greedy MIS** algorithm w.h.p. takes $\mathbf{O(\log n)}$ rounds on any n-node graph.

On **average**, Deterministic Parallel Greedy MIS is fast.

As fast as **Luby's** algorithm, but simpler.

This is **tight**: It takes $\mathbf{\Omega(\log n)}$ rounds.

Also applies to **$(\Delta + 1)$-Vertex-Coloring, Maximal Matching,** and **$(2\Delta - 1)$-Edge-Coloring**

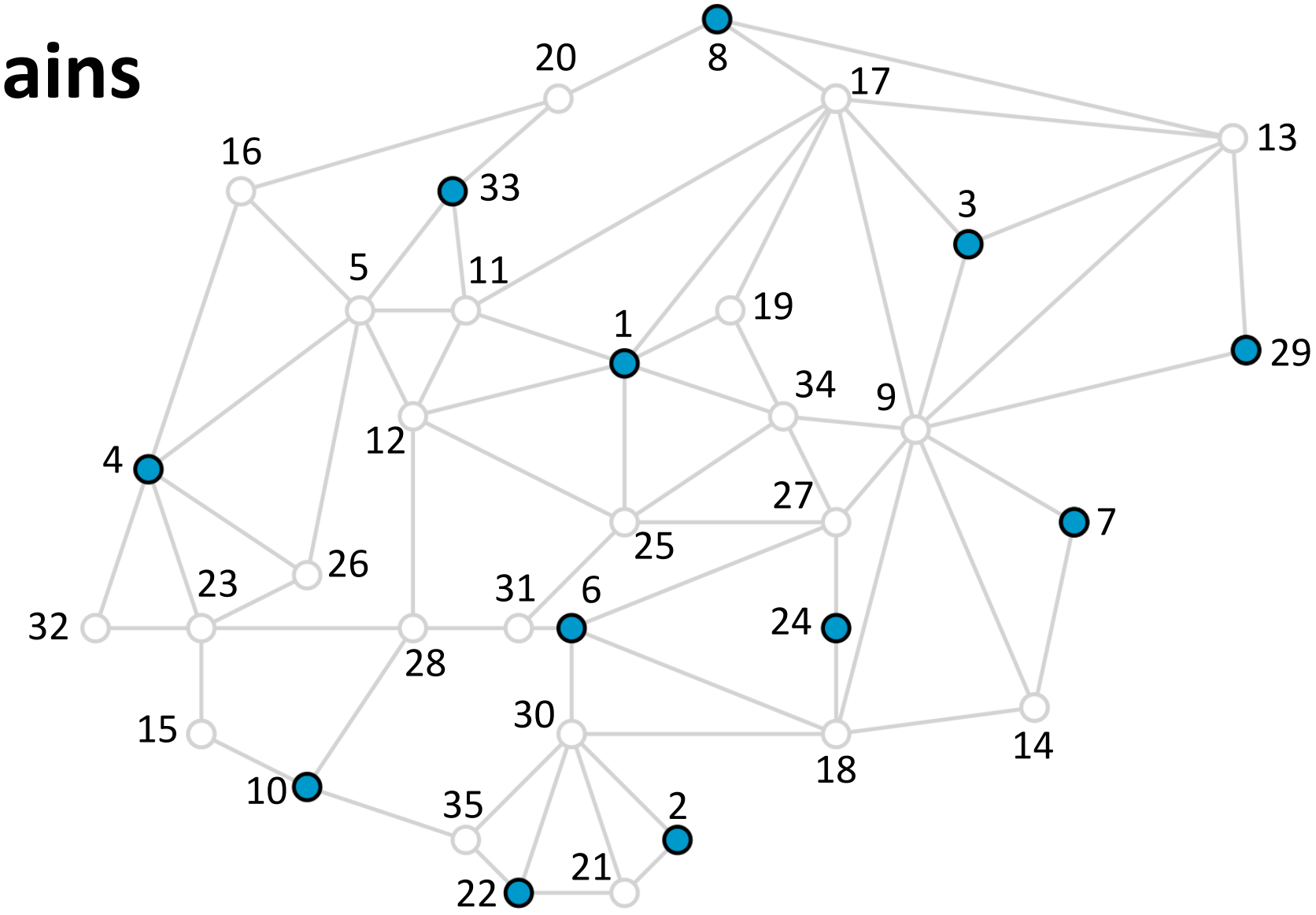
Proof Sketch

Proof Sketch

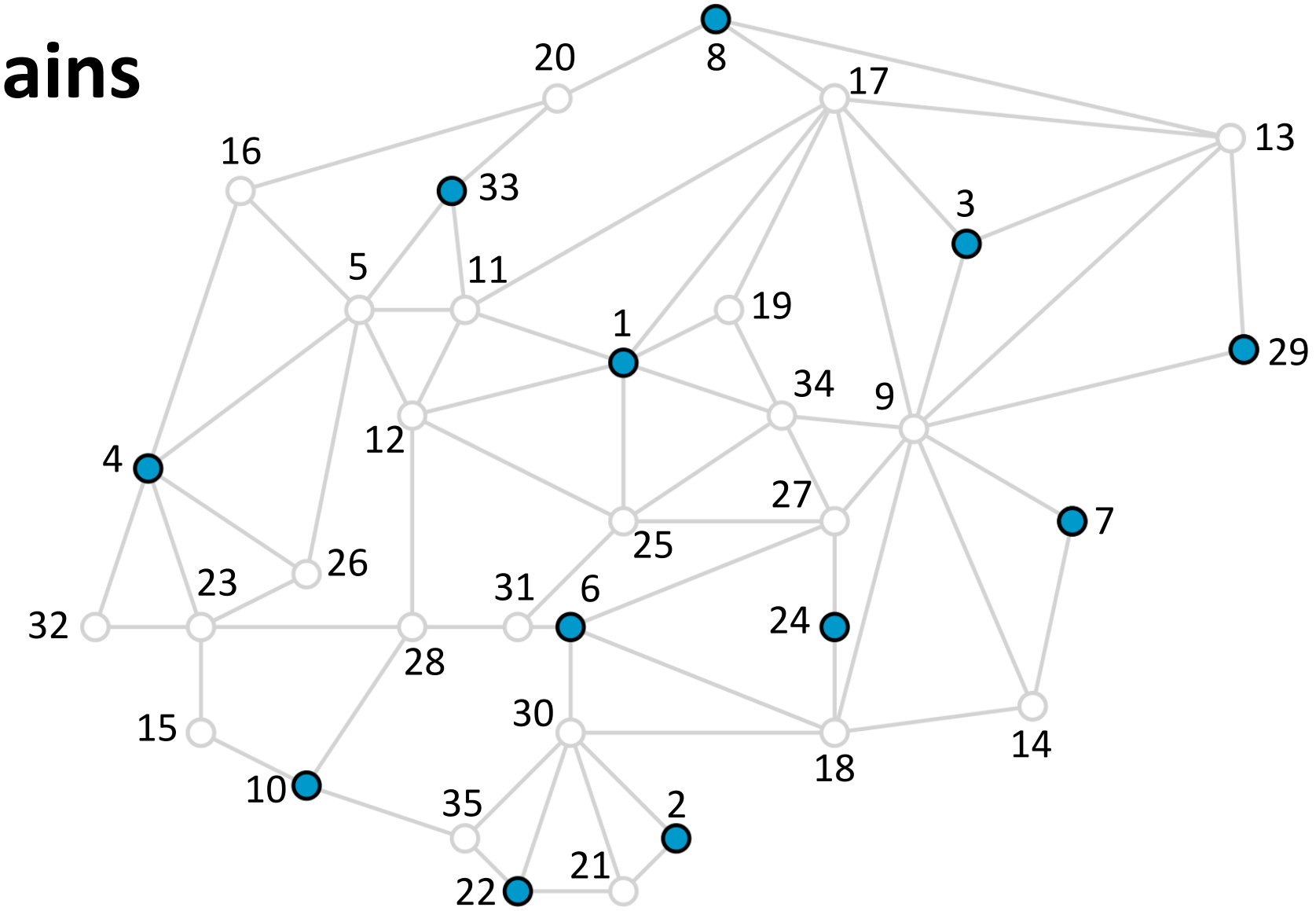
 # rounds $\leq$ dependency chain length $= \mathbf{O}(\log n)$

Dependency Chains

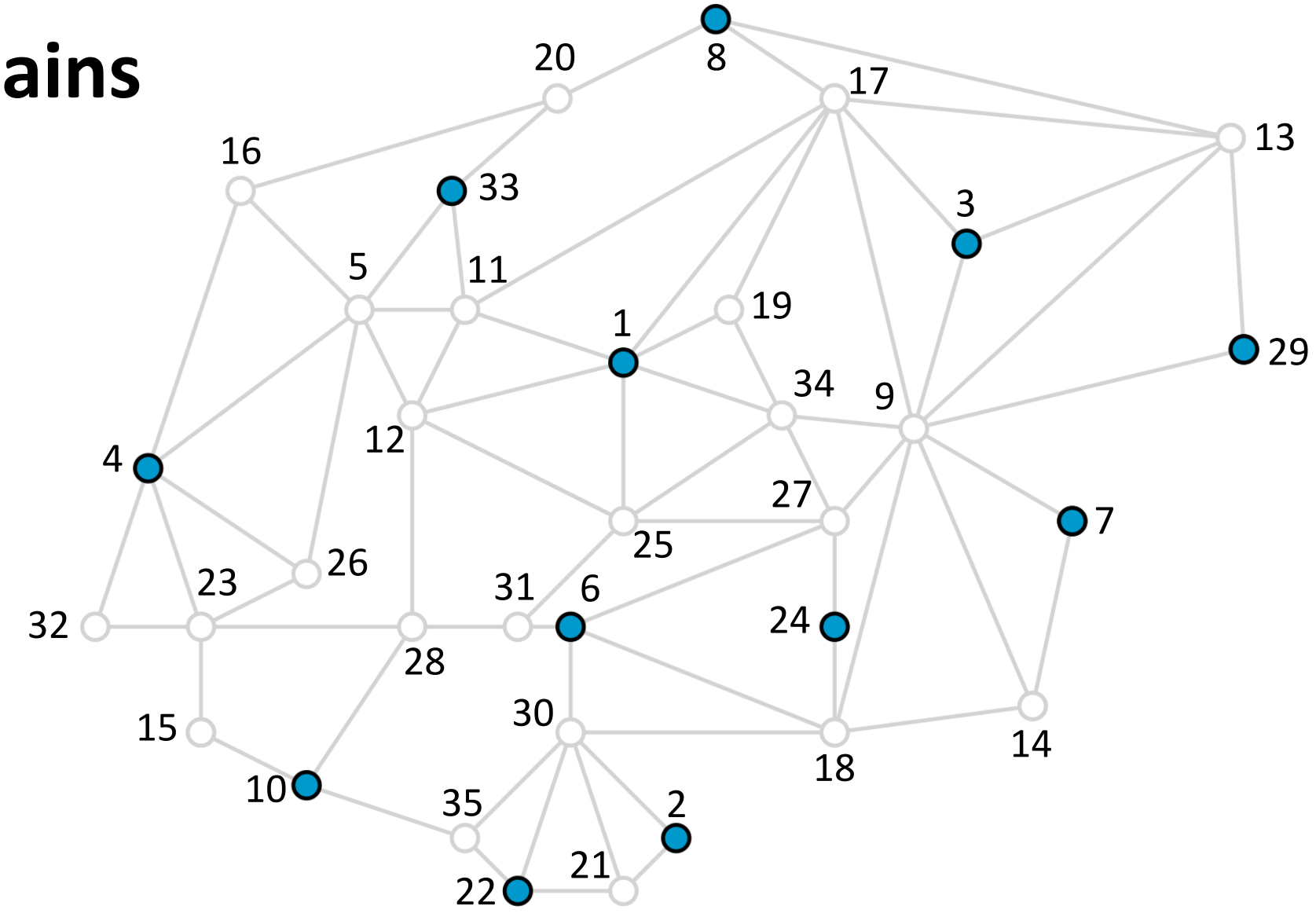
Dependency Chains



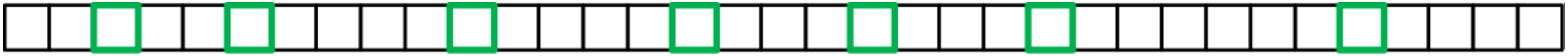
Dependency Chains



Dependency Chains



Dependency Chains



positions form dependency chain if

Dependency Chains

positions form dependency chain if

assigned nodes form a (monotonically increasing) path

alternating between MIS nodes and non-MIS nodes

MIS node is inhibitor of next non-MIS node

Dependency Chains

positions form dependency chain if

assigned nodes form a (monotonically increasing) path

alternating between MIS nodes and non-MIS nodes

MIS node is inhibitor of next non-MIS node

MIS node is inhibitor
of non-MIS node if
it is its
lowest-rank MIS neighbor

Dependency Chains

positions form dependency chain if

assigned nodes form a (monotonically increasing) path

alternating between MIS nodes and non-MIS nodes

MIS node is inhibitor of next non-MIS node

MIS node is inhibitor of non-MIS node if it is its lowest-rank MIS neighbor

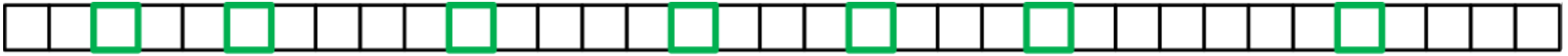
rounds $\leq$ dependency chain length

Proof Sketch

 # rounds $\leq$ dependency chain length $= \mathbf{O}(\log n)$

Dependency Chain Length

Dependency Chain Length



bound probability that $\Omega(\log n)$ positions form dependency chain

Dependency Chain Length

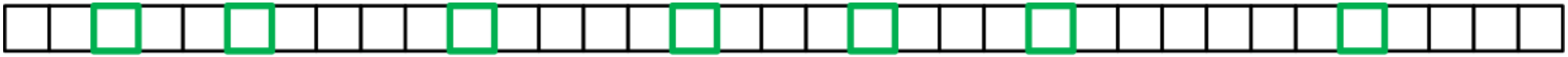
bound probability that $\Omega(\log n)$ positions form dependency chain

Dependency Chain Length

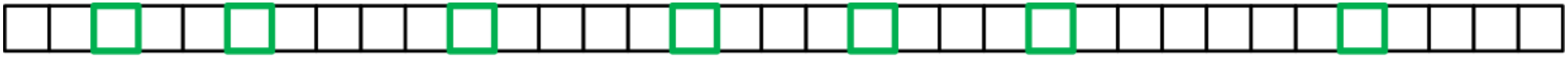
bound probability that $\Omega(\log n)$ positions form dependency chain

union bound over all choices of positions

Probability of Positions Forming Dependency Chain

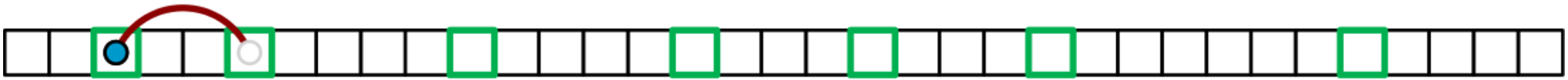


Probability of Positions Forming Dependency Chain



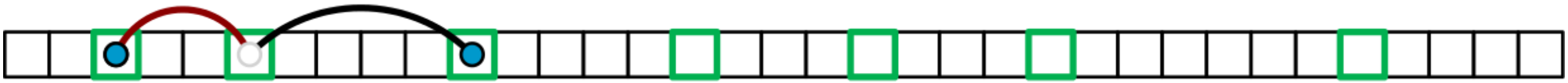
product of probabilities that a position continues dependency chain

Probability of Positions Forming Dependency Chain



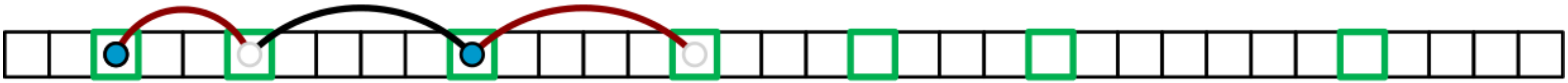
product of probabilities that a position continues dependency chain

Probability of Positions Forming Dependency Chain



product of probabilities that a position continues dependency chain

Probability of Positions Forming Dependency Chain



product of probabilities that a position continues dependency chain

Probability of Positions Forming Dependency Chain

product of probabilities that a position continues dependency chain

Probability of Positions Forming Dependency Chain

product of probabilities that a position continues dependency chain

Probability of Positions Forming Dependency Chain

product of probabilities that a position continues dependency chain

Probability of Continuing Dependency Chain

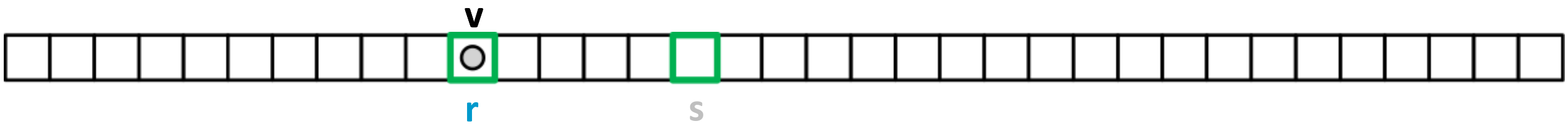
Probability of Continuing Dependency Chain

positions form dependency chain if

assigned nodes form a path

alternating MIS and non-MIS nodes

MIS is inhibitor of next non-MIS node



analyze probability dependency chain continues through positions r and s

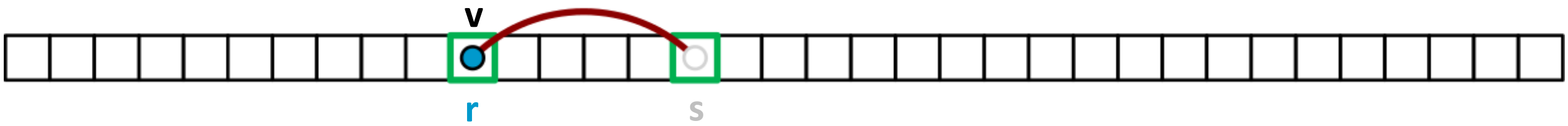
Probability of Continuing Dependency Chain

positions form dependency chain if

assigned nodes form a path

alternating MIS and non-MIS nodes

MIS is inhibitor of next non-MIS node



analyze probability dependency chain continues through positions r and s

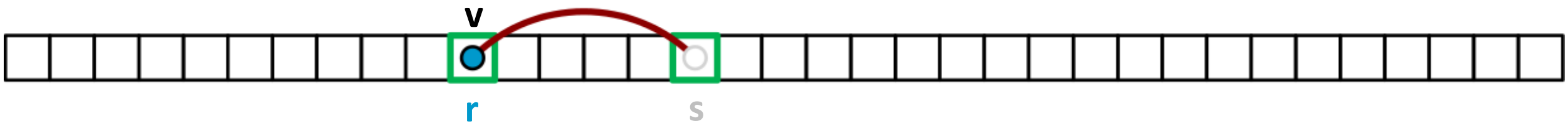
Probability of Continuing Dependency Chain

positions form dependency chain if

assigned nodes form a path

alternating MIS and non-MIS nodes

MIS is inhibitor of next non-MIS node



analyze probability dependency chain continues through positions r and s

exposing the positions one by one

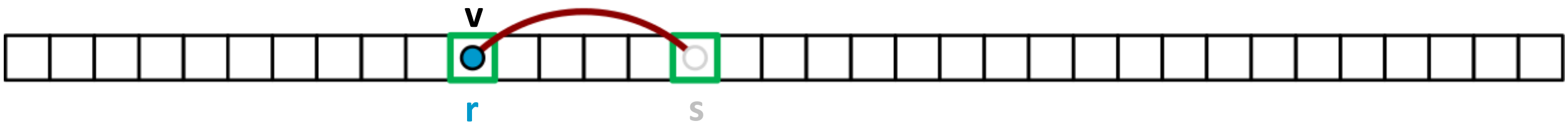
Probability of Continuing Dependency Chain

positions form dependency chain if

assigned nodes form a path

alternating MIS and non-MIS nodes

MIS is inhibitor of next non-MIS node



analyze probability dependency chain continues through positions r and s

exposing the positions one by one

at step step $t \leq r$,

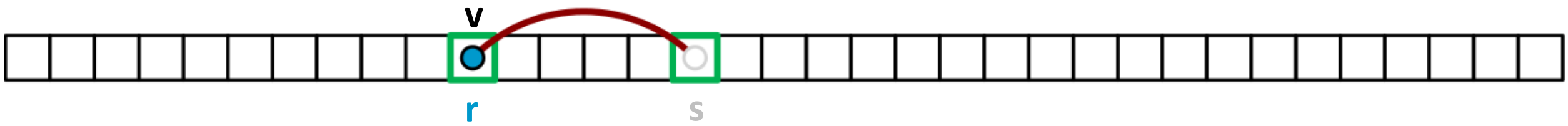
Probability of Continuing Dependency Chain

positions form dependency chain if

assigned nodes form a path

alternating MIS and non-MIS nodes

MIS is inhibitor of next non-MIS node



analyze probability dependency chain continues through positions r and s

exposing the positions one by one

at step step $t \leq r$,

A) v must be alive

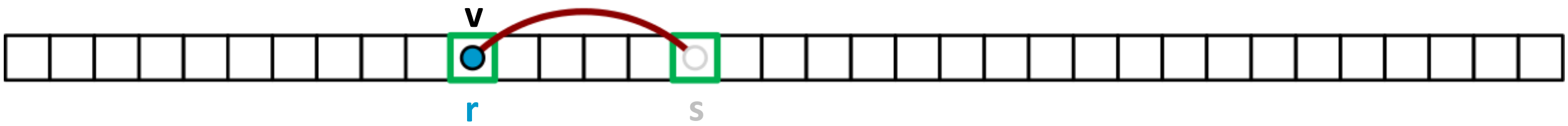
Probability of Continuing Dependency Chain

positions form dependency chain if

assigned nodes form a path

alternating MIS and non-MIS nodes

MIS is inhibitor of next non-MIS node



analyze probability dependency chain continues through positions r and s

exposing the positions one by one

at step step $t \leq r$,

A) v must be alive

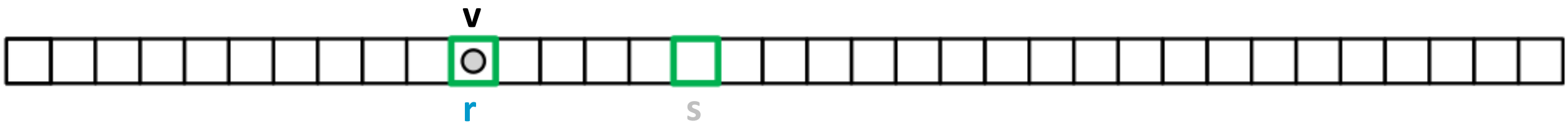
B) an alive neighbor of v must be at position s

Probability of Continuing Dependency Chain

at step $t \leq r$,

A) v must be alive

B) an alive neighbor of v must be at position s



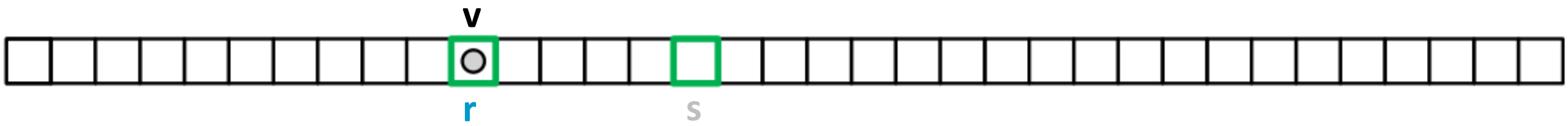
Probability of Continuing Dependency Chain

at step $t \leq r$,

A) v must be alive

B) an alive neighbor of v must be at position s

degree sequence of v: $d_0, d_1, \ldots, d_t$

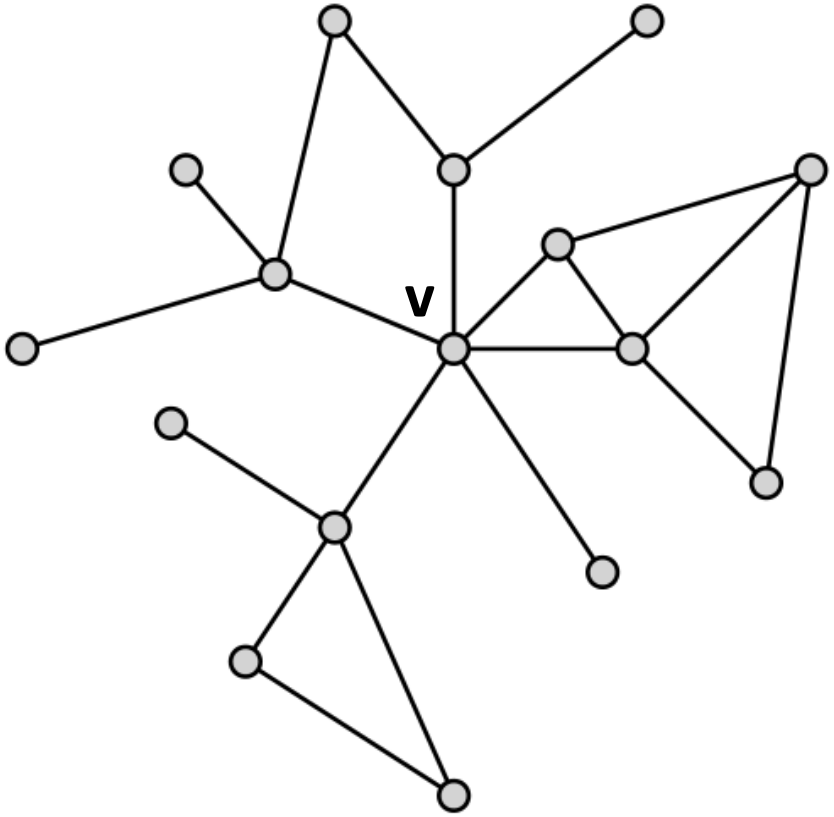


Probability of Continuing Dependency Chain

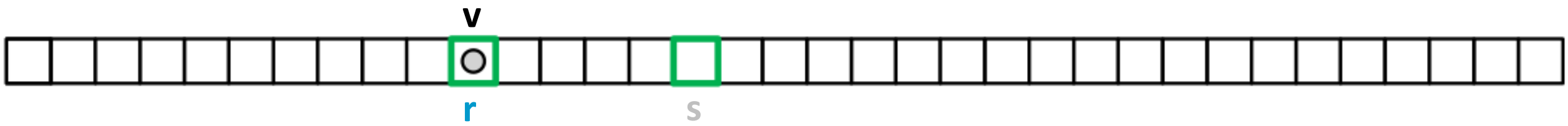
at step $t \leq r$,

A) v must be alive

B) an alive neighbor of v must be at position s



degree sequence of v: $d_0, d_1, \ldots, d_t$

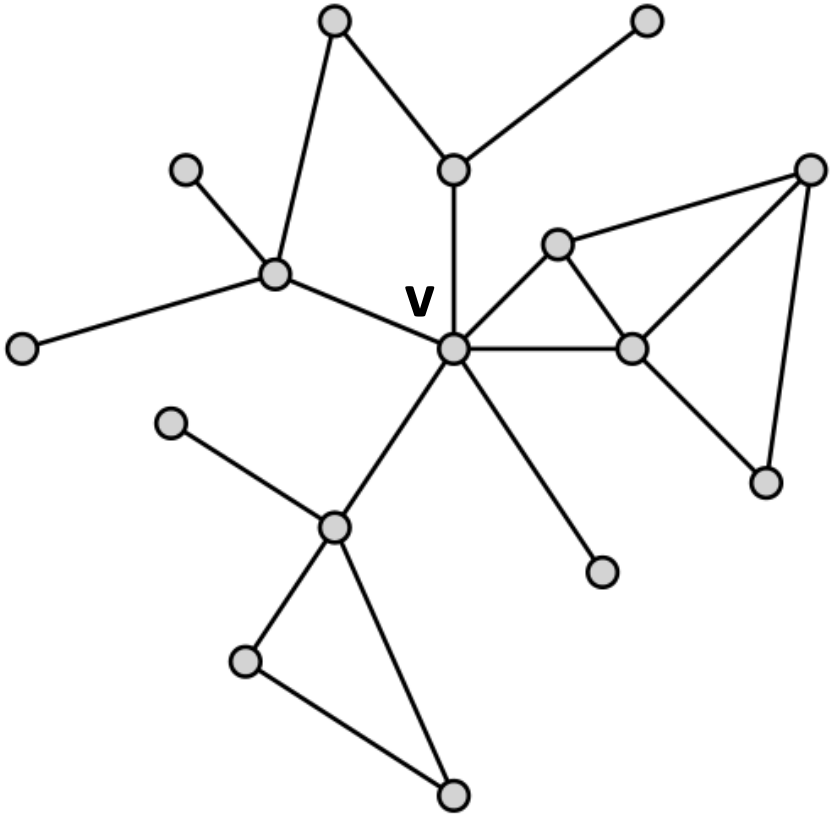


Probability of Continuing Dependency Chain

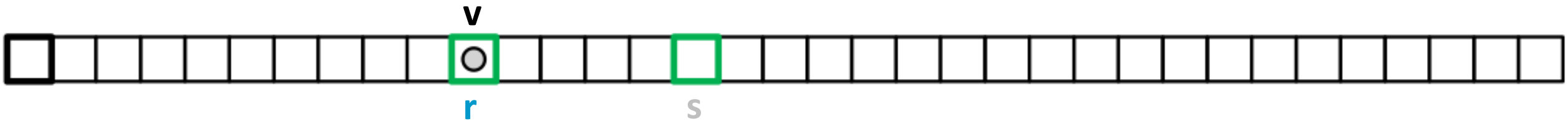
at step $t \leq r$,

A) v must be alive

B) an alive neighbor of v must be at position s



degree sequence of v: $d_0, d_1, \ldots, d_t$

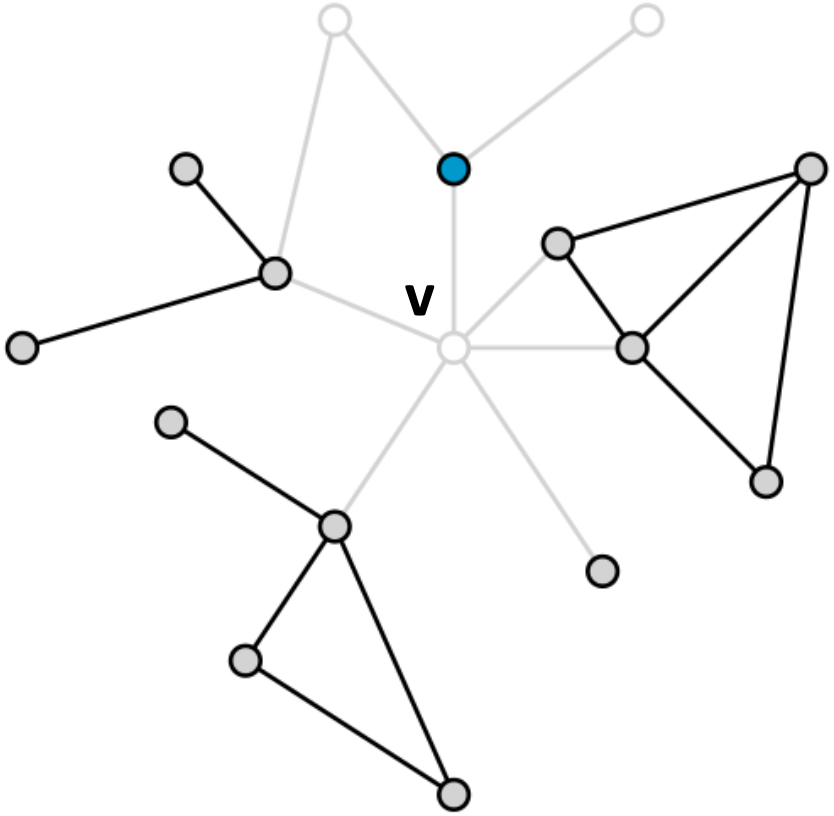


Probability of Continuing Dependency Chain

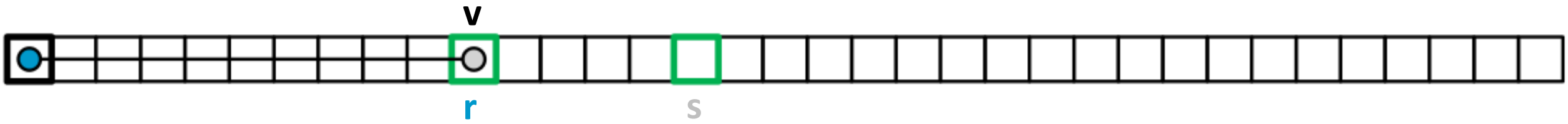
at step $t \leq r$,

A) v must be alive

B) an alive neighbor of v must be at position s



degree sequence of v: $d_0, d_1, \ldots, d_t$



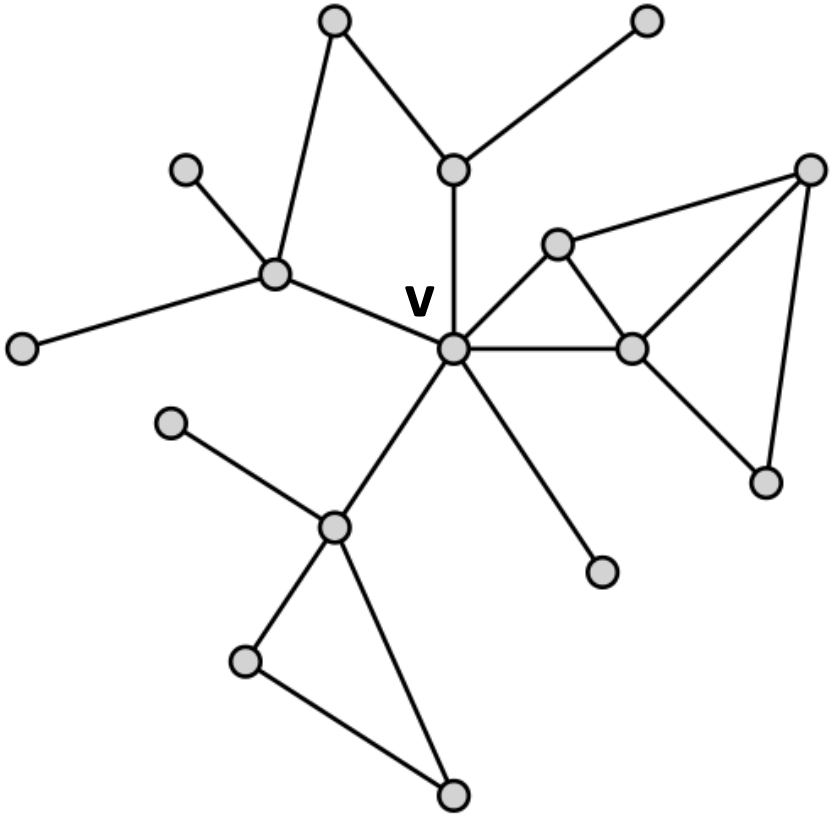
Probability of Continuing Dependency Chain

at step $t \leq r$,

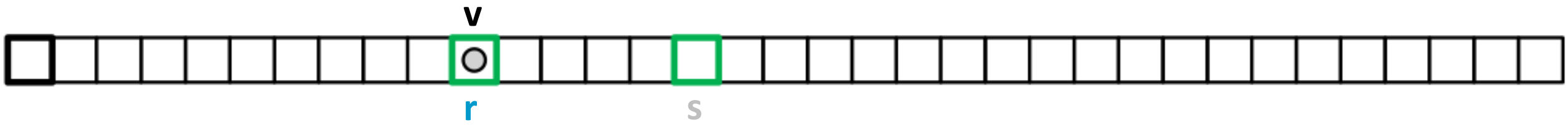
A) v must be alive

B) an alive neighbor of v must be at position s

$$\mathbf{P}[\,\mathbf{A}\,] \approx \left(1 - \frac{d_0}{n}\right)$$



degree sequence of v: $d_0, d_1, \ldots, d_t$



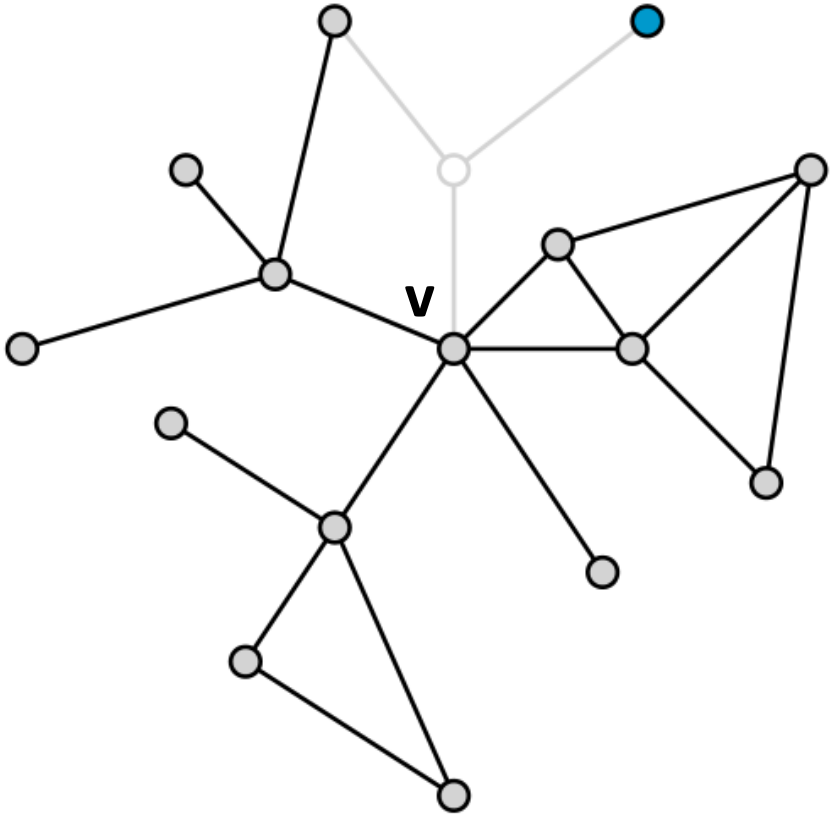
Probability of Continuing Dependency Chain

at step $t \leq r$,

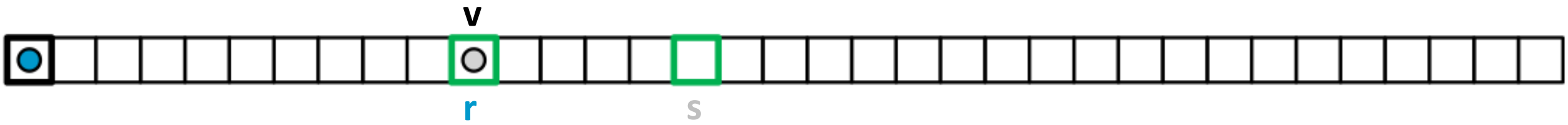
A) v must be alive

B) an alive neighbor of v must be at position s

$$\mathbf{P}[\,\mathbf{A}\,] \approx \left(1 - \frac{d_0}{n}\right)$$



degree sequence of v: $d_0, d_1, \ldots, d_t$



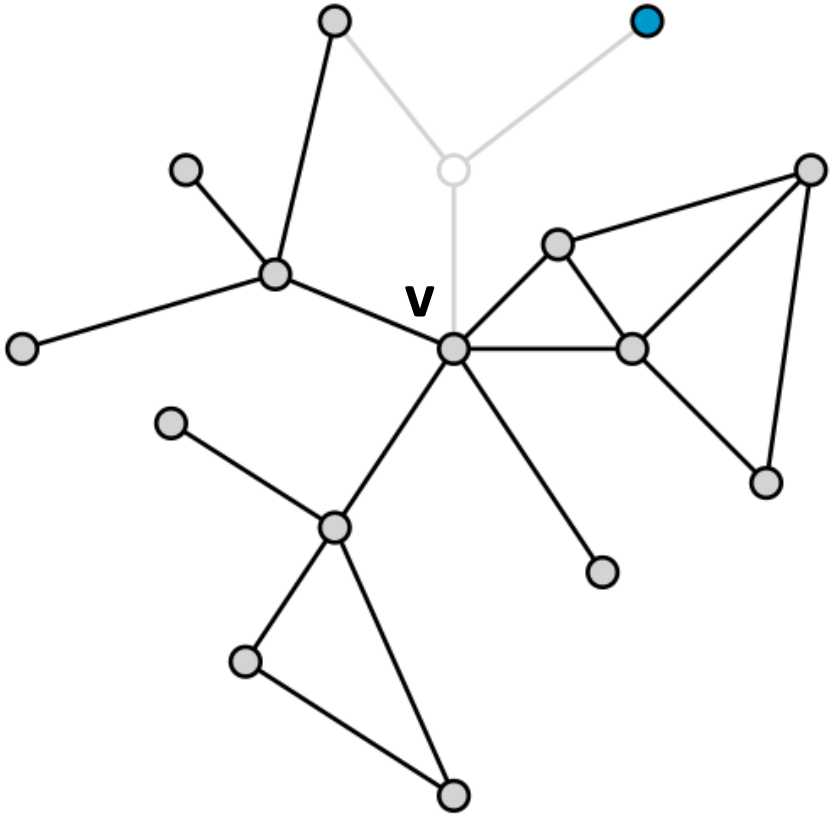
Probability of Continuing Dependency Chain

at step $t \leq r$,

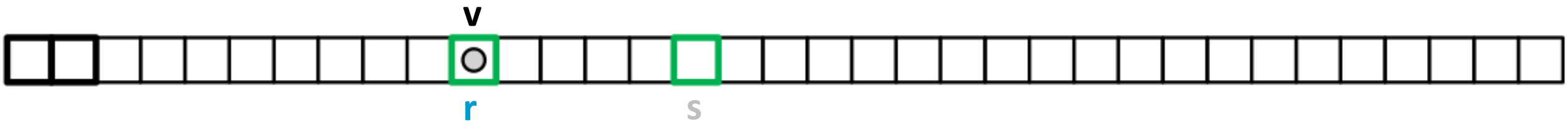
A) v must be alive

B) an alive neighbor of v must be at position s

$$\mathbf{P}[\,\mathbf{A}\,] \approx \left(1 - \frac{d_0}{n}\right)$$



degree sequence of v: $d_0, d_1, \ldots, d_t$



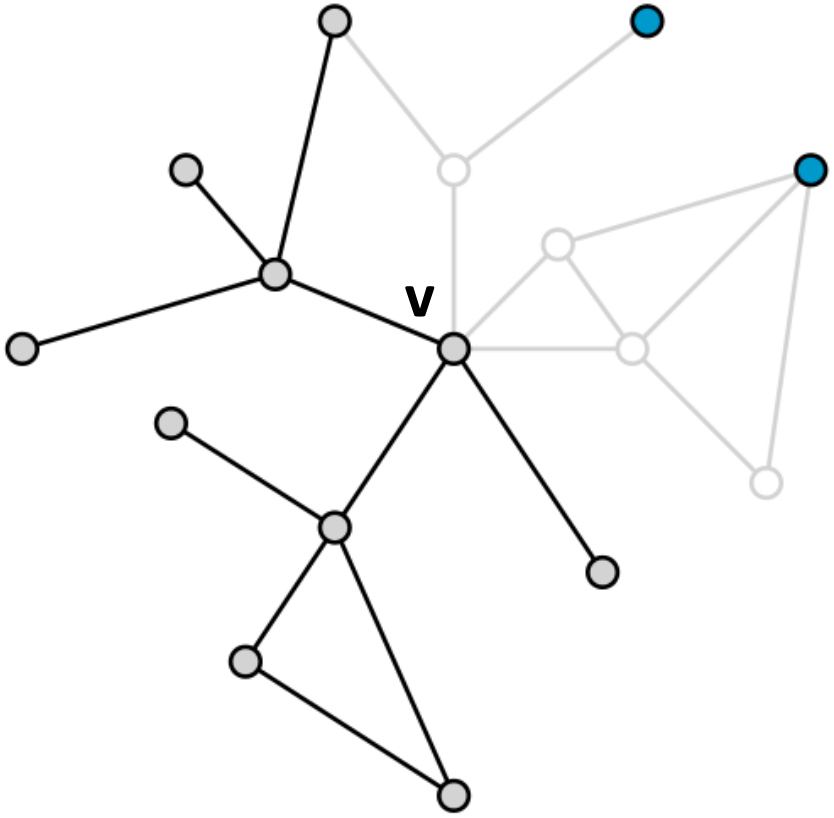
Probability of Continuing Dependency Chain

at step $t \leq r$,

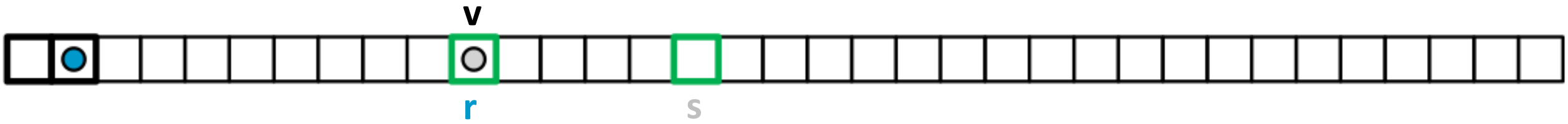
A) v must be alive

B) an alive neighbor of v must be at position s

$$\mathbf{P}[\,\mathbf{A}\,] \approx \left(1 - \frac{d_0}{n}\right) \cdot \left(1 - \frac{d_1}{n}\right)$$



degree sequence of v: $d_0, d_1, \ldots, d_t$



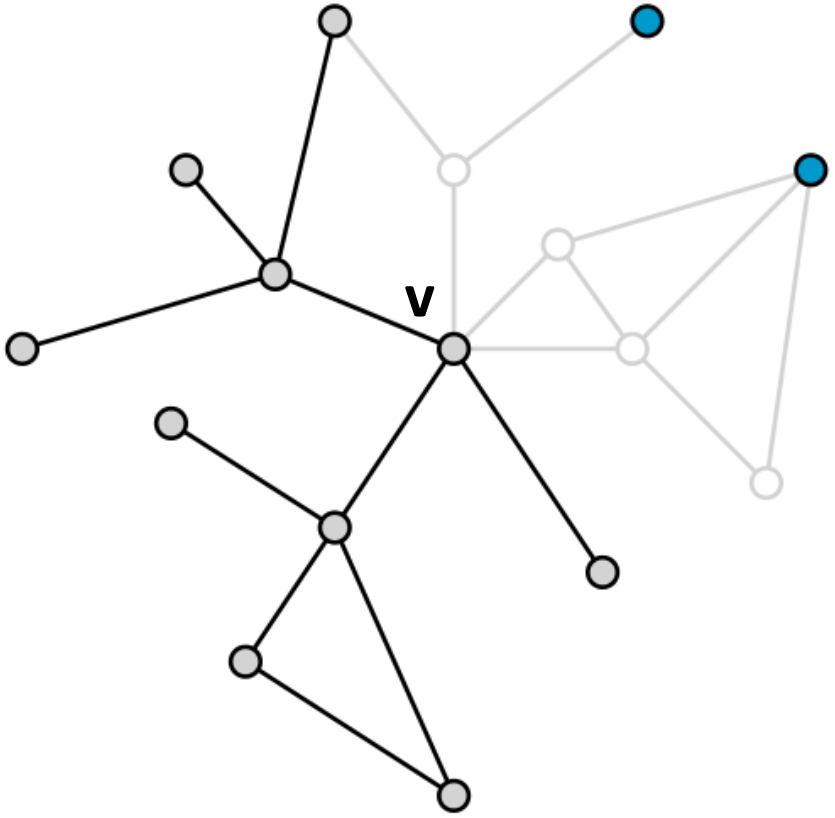
Probability of Continuing Dependency Chain

at step $t \leq r$,

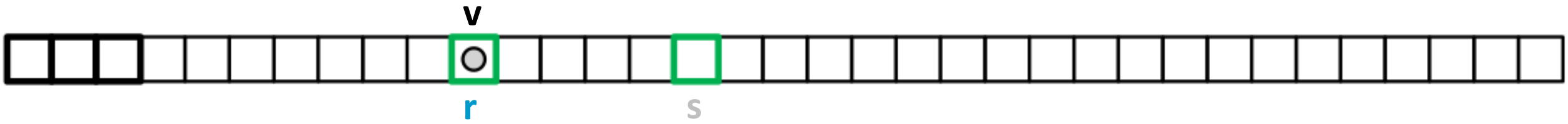
A) v must be alive

B) an alive neighbor of v must be at position s

$$\mathbf{P}[\,\mathbf{A}\,] \approx \left(1 - \frac{d_0}{n}\right) \cdot \left(1 - \frac{d_1}{n}\right) \cdot \left(1 - \frac{d_2}{n}\right)$$



degree sequence of v: $d_0, d_1, \ldots, d_t$



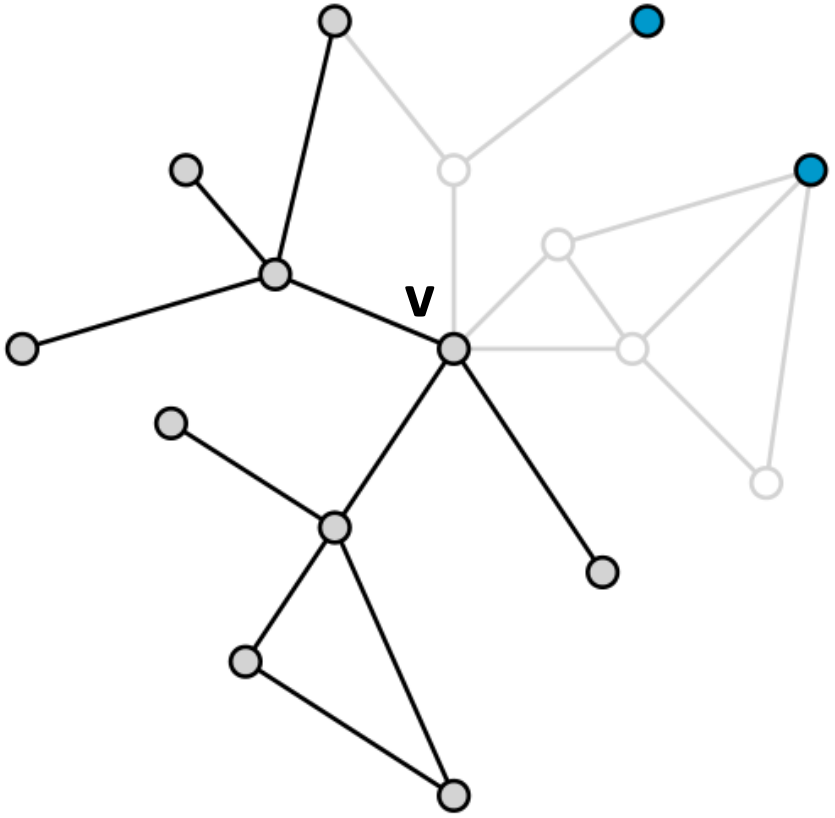
Probability of Continuing Dependency Chain

at step $t \leq r$,

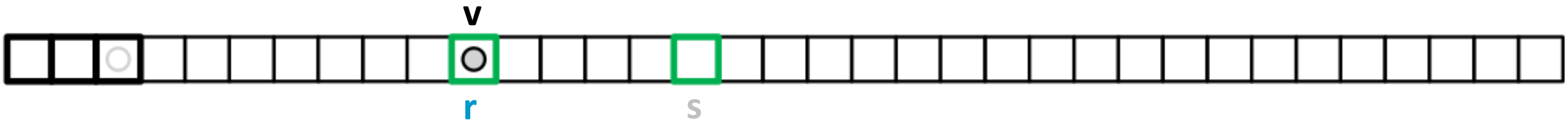
A) v must be alive

B) an alive neighbor of v must be at position s

$$\mathbf{P}[\,\mathbf{A}\,] \approx \left(1 - \frac{d_0}{n}\right) \cdot \left(1 - \frac{d_1}{n}\right) \cdot \left(1 - \frac{d_2}{n}\right)$$



degree sequence of v: $d_0, d_1, \ldots, d_t$



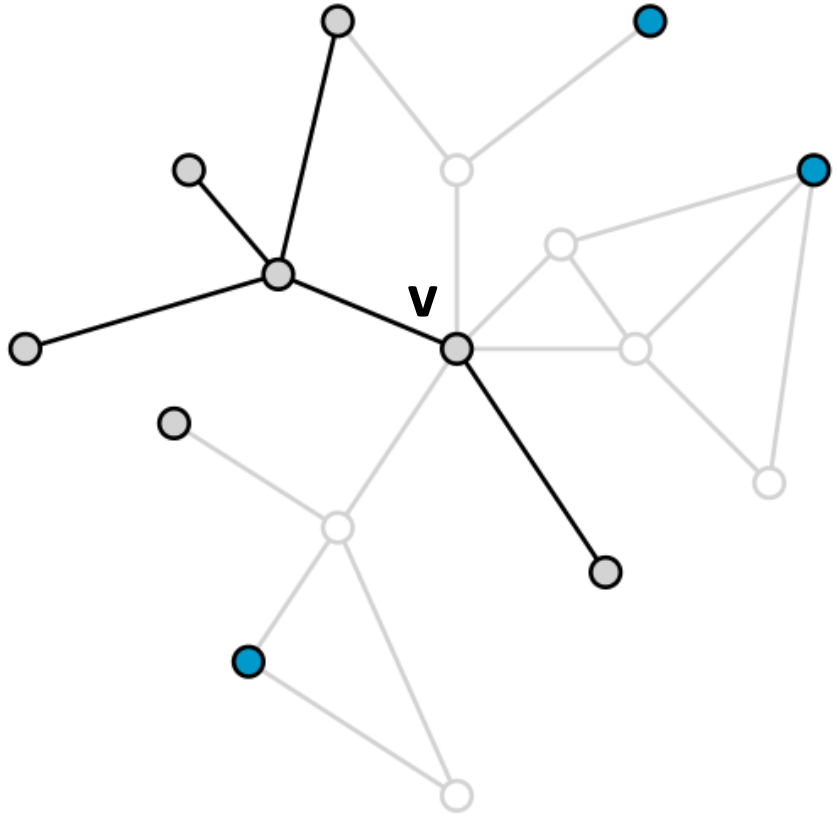
Probability of Continuing Dependency Chain

at step $t \leq r$,

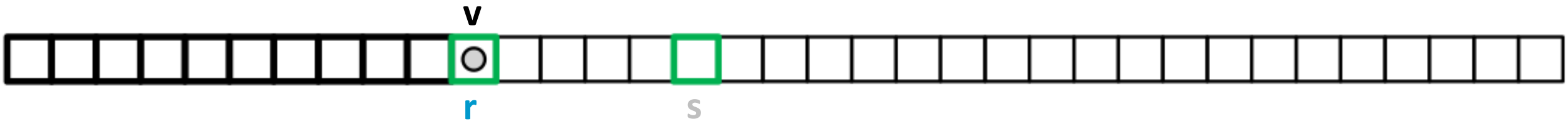
A) v must be alive

B) an alive neighbor of v must be at position s

$$\mathbf{P}[\,\mathbf{A}\,] \approx \left(1 - \frac{d_0}{n}\right) \cdot \ldots \cdot \left(1 - \frac{d_{t-1}}{n}\right)$$



degree sequence of v: $d_0, d_1, \ldots, d_t$



Probability of Continuing Dependency Chain

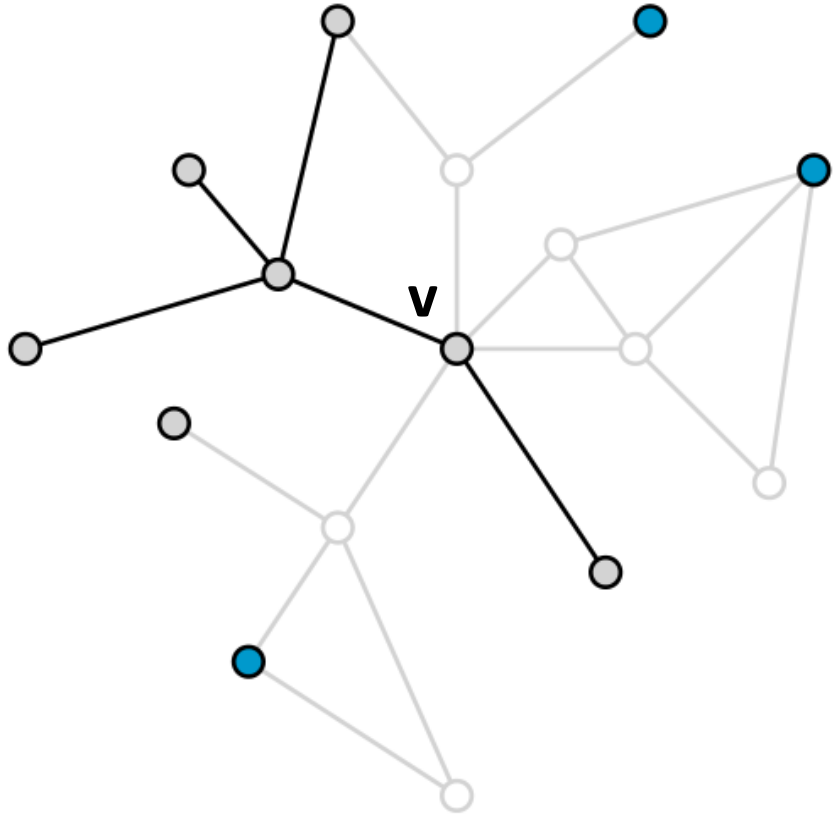
at step $t \leq r$,

A) v must be alive

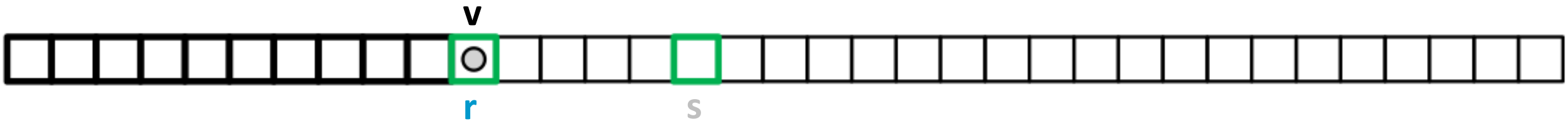
B) an alive neighbor of v must be at position s

$$\mathbf{P}[\,\mathbf{A}\,] \approx \left(1 - \frac{d_0}{n}\right) \cdot \ldots \cdot \left(1 - \frac{d_{t-1}}{n}\right)$$

$$\mathbf{P}[\,\mathbf{B}\,] \approx \frac{d_t}{n}$$



degree sequence of v: $d_0, d_1, \ldots, d_t$



Probability of Continuing Dependency Chain

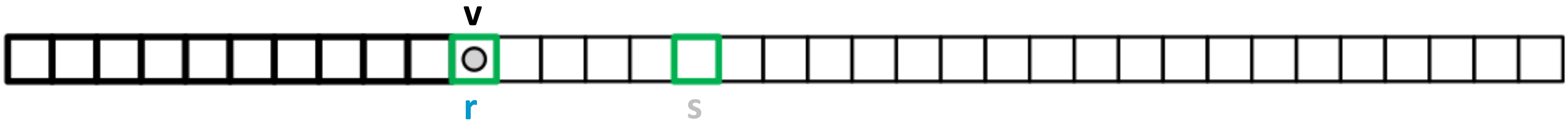
at step $t \leq r$,

A) v must be alive

B) an alive neighbor of v must be at position s

$$\mathbf{P}[\ \mathbf{continue}\ ] \approx \prod_{i=0}^{t-1} \left(1 - \frac{d_i}{n}\right) \cdot \frac{d_t}{n}$$

degree sequence of v: $d_0, d_1, \ldots, d_t$



Probability of Continuing Dependency Chain

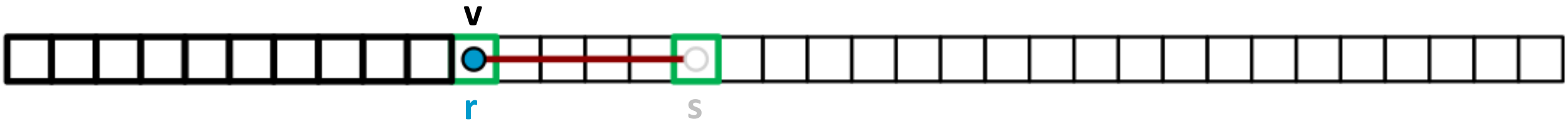
at step $t \leq r$,

A) v must be alive

B) an alive neighbor of v must be at position s

$$\mathbf{P}[\ \mathbf{continue}\ ] \approx \prod_{i=0}^{t-1} \left(1 - \frac{d_i}{n}\right) \cdot \frac{d_t}{n}$$

degree sequence of v: $d_0, d_1, \ldots, d_t$



Probability of Continuing Dependency Chain

at step $t \leq r$,

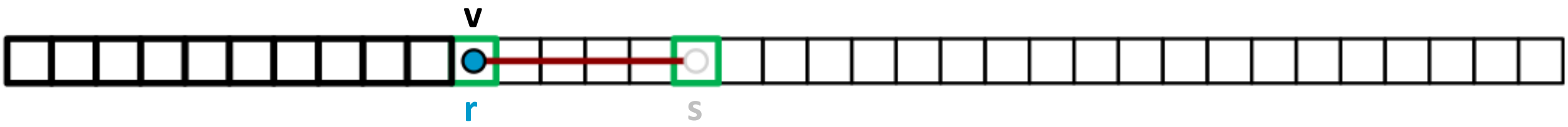
A) v must be alive

B) an alive neighbor of v must be at position s

$$\mathbf{P}[\ \mathbf{continue}\ ] \approx \prod_{i=0}^{t-1} \left(1 - \frac{d_i}{n}\right) \cdot \frac{d_t}{n}$$

degree sequence of v: $d_0, d_1, \ldots, d_t$

small enough for any degree sequence



Probability of Continuing Dependency Chain

at step $t \leq r$,

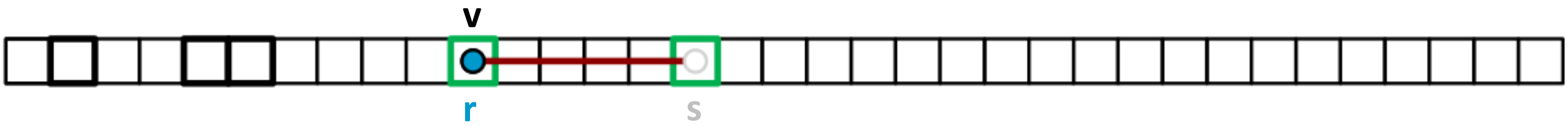
A) v must be alive

B) an alive neighbor of v must be at position s

$$\mathbf{P}[\ \mathbf{continue}\ ] \approx \prod_{i=0}^{t-1} \left(1 - \frac{d_i}{n}\right) \cdot \frac{d_t}{n}$$

degree sequence of v: $d_0, d_1, \ldots, d_t$

small enough for any degree sequence, even if we only look at (any) t positions



Recap of Proof

Recap of Proof

 # rounds $\leq$ dependency chain length $= \mathbf{O}(\log n)$

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain

reserve <u>disjoint set of positions</u> for each position

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain

reserve <u>disjoint set of positions</u> for each position

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain

reserve disjoint set of positions for each position

use these positions to bound continuation probability of position

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain

reserve disjoint set of positions for each position

use these positions to bound continuation probability of position

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain

reserve disjoint set of positions for each position

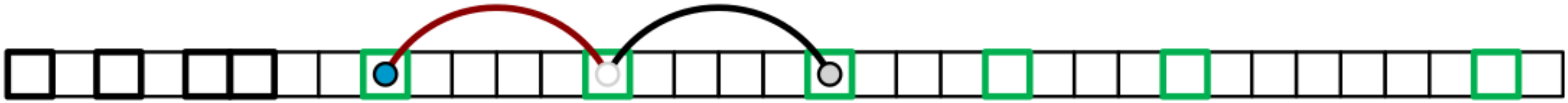
use these positions to bound continuation probability of position

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain



reserve disjoint set of positions for each position

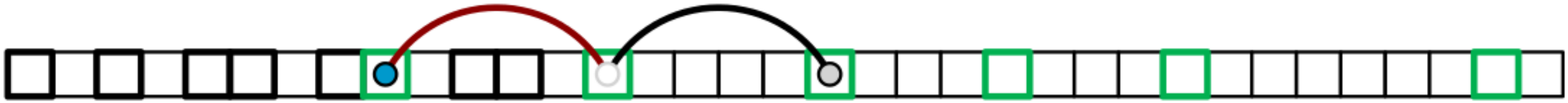
use these positions to bound continuation probability of position

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= \mathbf{O}(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain



reserve <u>disjoint set of positions</u> for each position

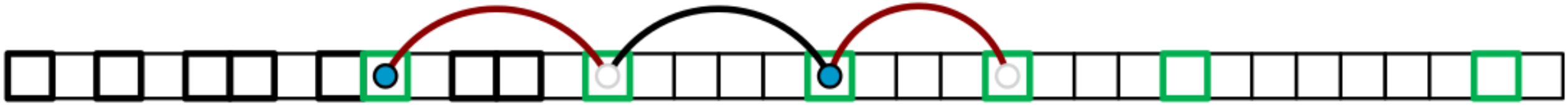
use these positions to bound continuation probability of position

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain



reserve disjoint set of positions for each position

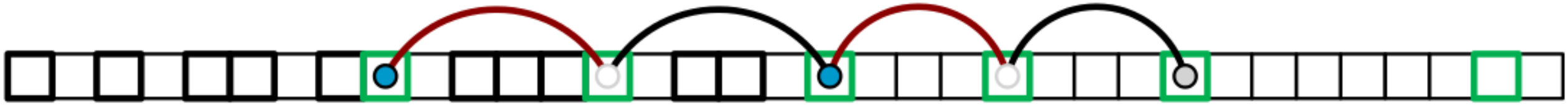
use these positions to bound continuation probability of position

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain



reserve <u>disjoint set of positions</u> for each position

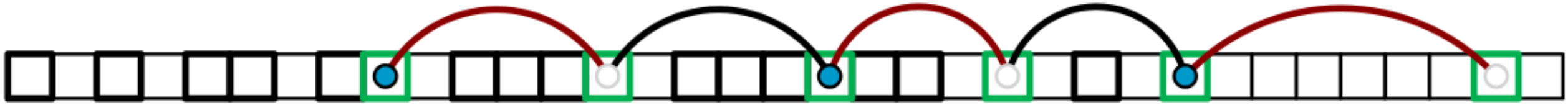
use these positions to bound continuation probability of position

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain



reserve <u>disjoint set of positions</u> for each position

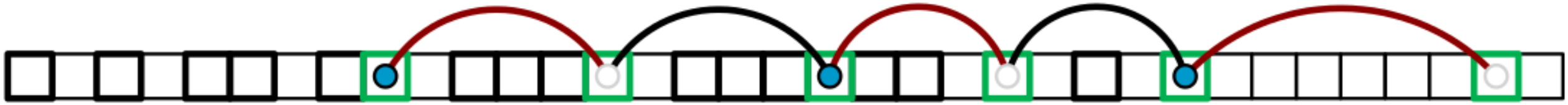
use these positions to bound continuation probability of position

union bound over all choices of positions

Recap of Proof

 # rounds $\leq$ dependency chain length $= O(\log n)$

bound probability that $\Omega(\log n)$ positions form dependency chain



reserve disjoint set of positions for each position

use these positions to bound continuation probability of position

probability of dependency chain is product of continuation probabilities

union bound over all choices of positions

Summary and Open Questions

Summary and Open Questions

The **Parallel Randomized Greedy MIS** algorithm w.h.p. takes $\boldsymbol{\Theta(\log n)}$ rounds on n-node graphs.

Summary and Open Questions

The **Parallel Randomized Greedy MIS** algorithm w.h.p. takes $\boldsymbol{\Theta(\log n)}$ rounds on n-node graphs.

As fast as **Luby's** algorithm, but simpler.

Summary and Open Questions

The **Parallel Randomized Greedy MIS** algorithm w.h.p. takes $\boldsymbol{\Theta(\log n)}$ rounds on n-node graphs.

As fast as **Luby's** algorithm, but simpler.

On **average,** Deterministic Parallel Greedy MIS algorithm is fast.

Summary and Open Questions

The **Parallel Randomized Greedy MIS** algorithm w.h.p. takes $\boldsymbol{\Theta(\log n)}$ rounds on n-node graphs.

As fast as **Luby's** algorithm, but simpler.

On **average,** Deterministic Parallel Greedy MIS algorithm is fast.

Which algorithms can be replaced by simpler ones, on average?

Summary and Open Questions

The **Parallel Randomized Greedy MIS** algorithm w.h.p. takes $\boldsymbol{\Theta(\log n)}$ rounds on n-node graphs.

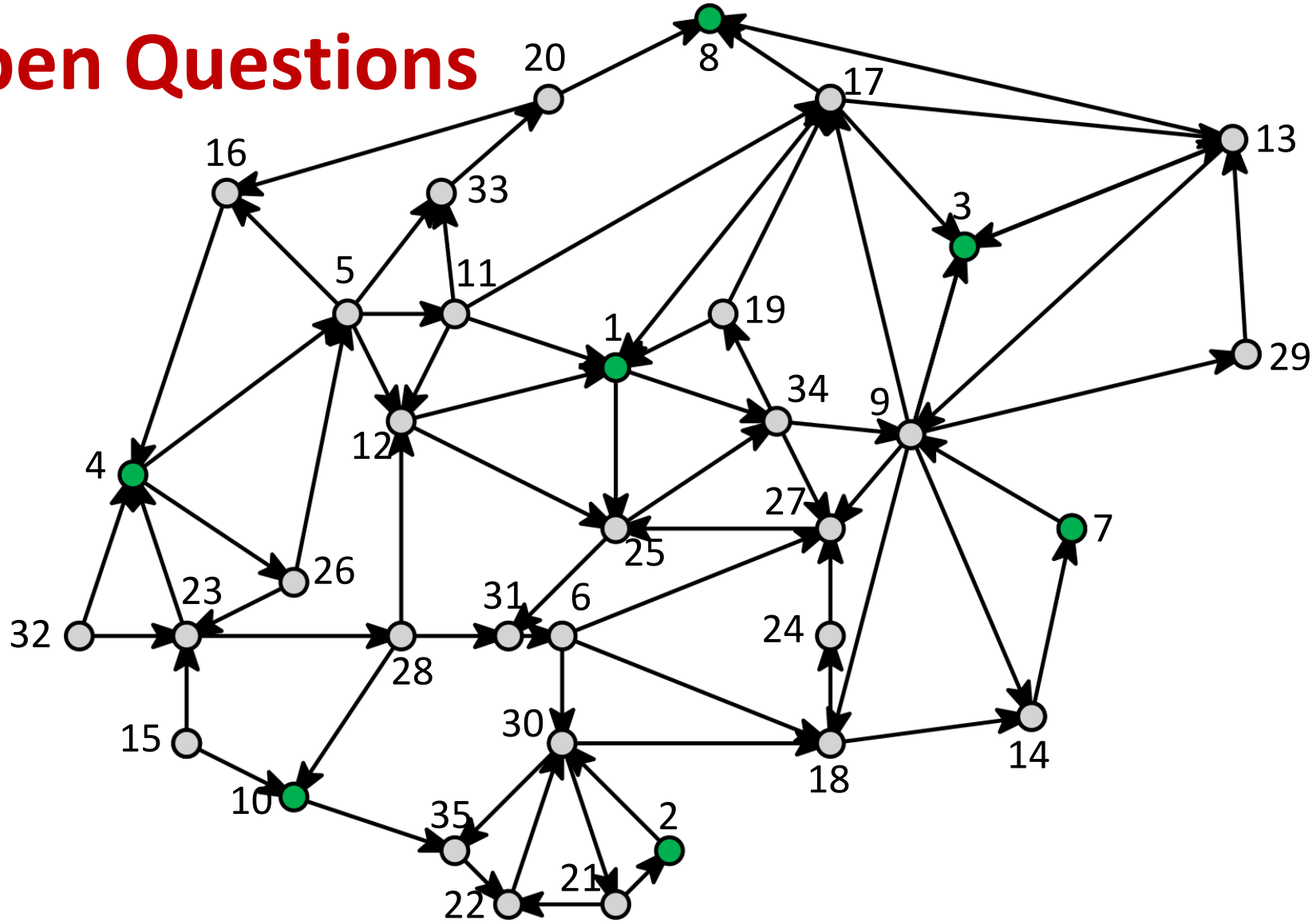
As fast as **Luby's** algorithm, but simpler.

On **average,** Deterministic Parallel Greedy MIS algorithm is fast.

Which algorithms can be replaced by simpler ones, on average?

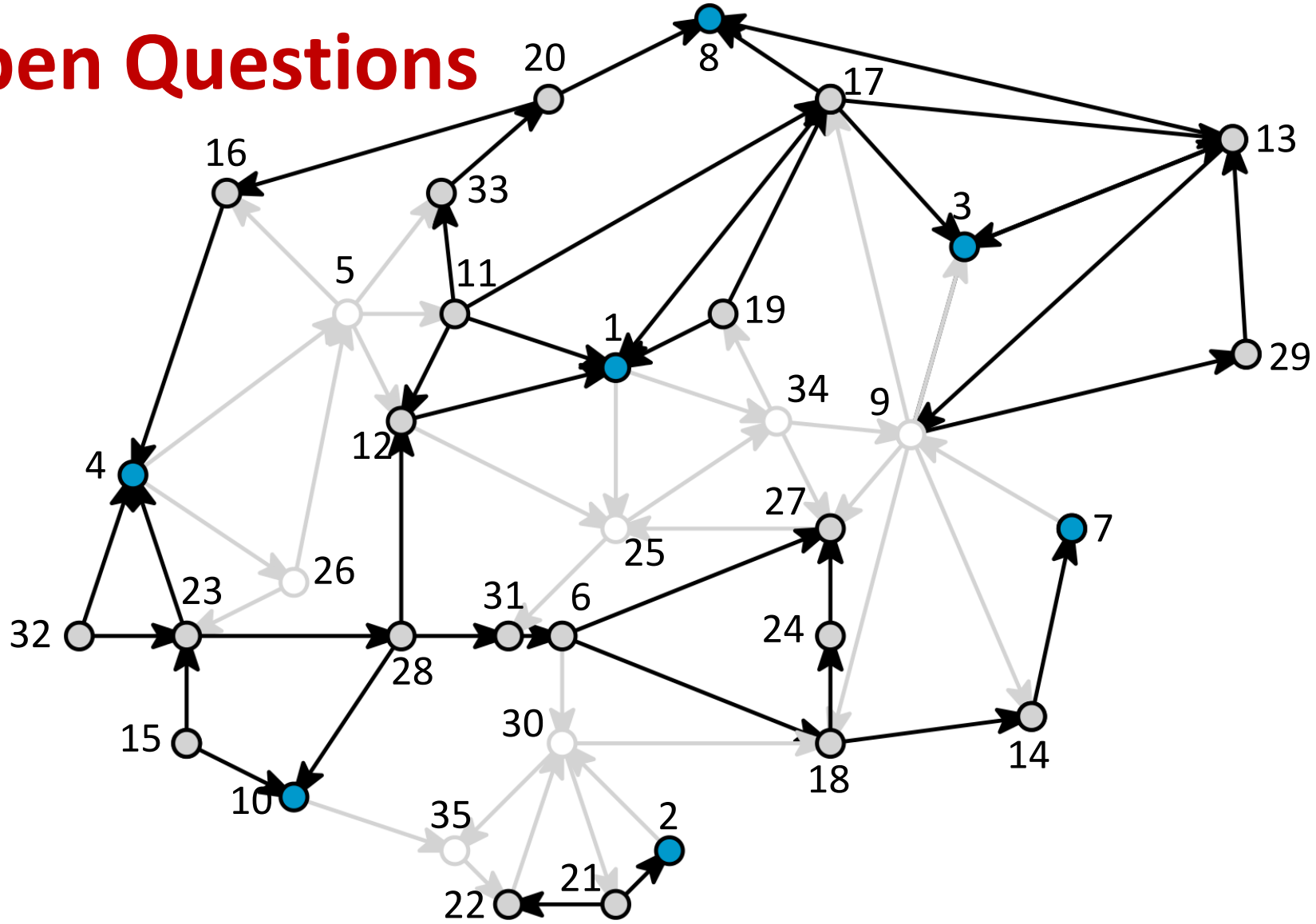
What about same process on directed graphs?

Summary and Open Questions



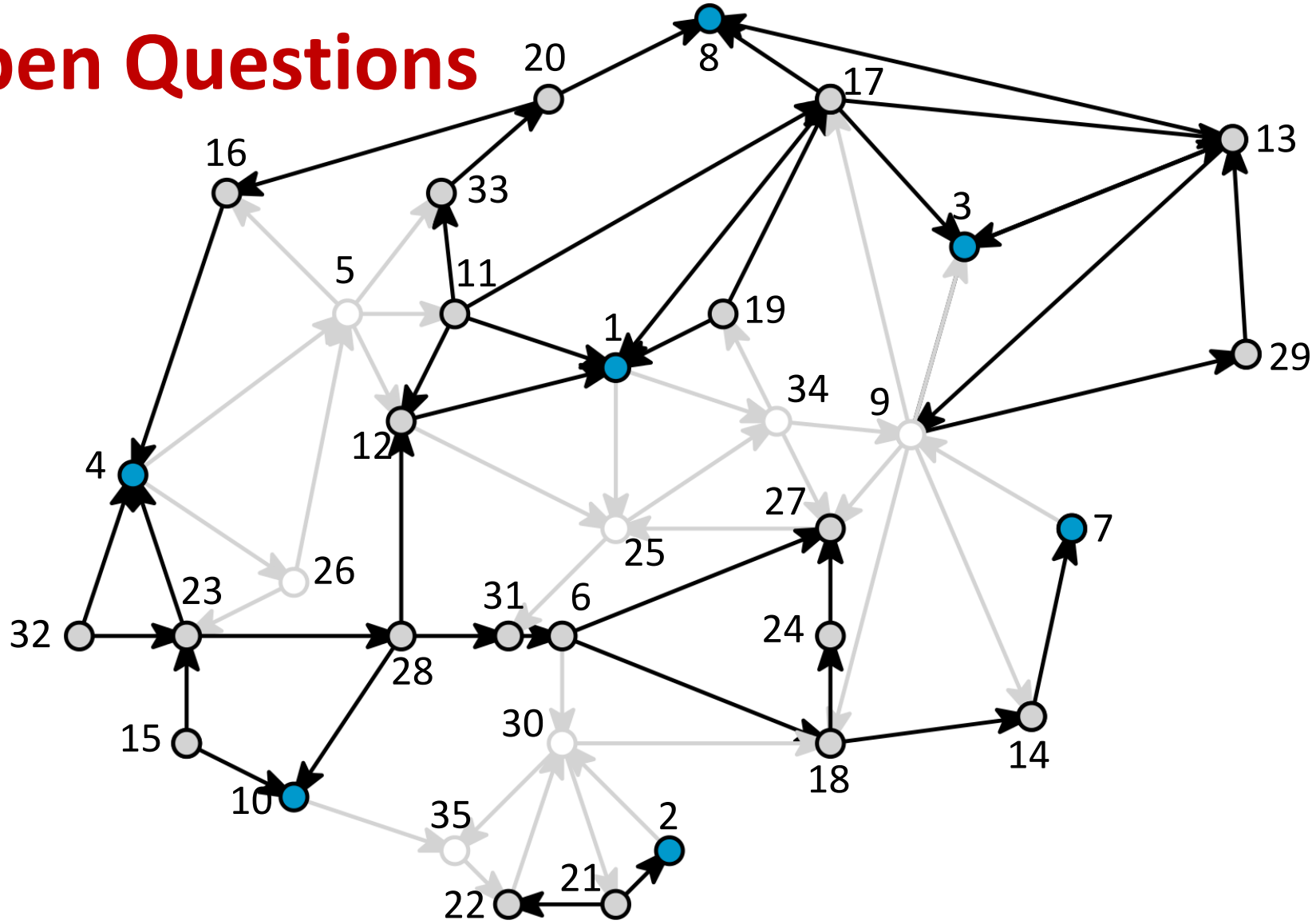
What about same process on directed graphs?

Summary and Open Questions



What about same process on directed graphs?

Summary and Open Questions



Thank you!

What about same process on directed graphs?