Ring Exploration by Semi-Synchronous Oblivious Robots

Franck Petit

INRIA/LIP6-CNRS/UPMC

Context

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes

Context

- A team of k *"weak"* robots evolving into a ring of n nodes

Context

- A team of *k* *"weak"* robots evolving into a ring of *n* nodes
  - Autonomous : No central authority

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes
  - Autonomous
  - Anonymous : Undistinguishable

Context

- A team of *k* *"weak"* robots evolving into a ring of *n* nodes
  - Autonomous
  - Anonymous
  - Oblivious : No mean to know the past

Context

- A team of *k* *"weak"* robots evolving into a ring of *n* nodes
  - Autonomous
  - Anonymous
  - Oblivious
  - Disoriented : No mean to agree on a common direction or orientation

Context

- A team of *k* *"weak"* robots evolving into a ring of *n* nodes

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes
  - Atomicity : In every configuration, each robot is located at exactly one node

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes
  - Atomicity
  - Multiplicity : In every configuration, each node contains zero, one, or more than one robot

    (every robot is able to detect it)

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes
  - SSM : In every configuration, k' robots are activated ($0 < k' \leq k$)

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes
  - SSM : In every configuration, k' robots are activated ($0 < k' \leq k$)
  - The k' activated robots execute the cycle:

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes
  - SSM : In every configuration, k' robots are activated ($0 < k' \leq k$)
  - The k' activated robots execute the cycle:
    1. Look : Instantaneous snapshot with multiplicity detection

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes
  - SSM : In every configuration, k' robots are activated ($0 < k' \leq k$)
  - The k' activated robots execute the cycle:
    1. Look
    2. Compute: Based on this observation, decides to either stay idle or move to one of the neighboring nodes

Context

- A team of *k* "*weak*" robots evolving into a ring of *n* nodes
  - SSM : In every configuration, k' robots are activated ($0 < k' \leq k$)
  - The k' activated robots execute the cycle:
    1. Look
    2. Compute
    3. Move : Move toward its destination

Problem

Problem

Starting from a configuration where no two robots are located at the same node:

Problem

Starting from a configuration where no two robots are located at the same node:

- Exploration:
  Each node must be visited by at least one robot

Problem

Starting from a configuration where no two robots are located at the same node:

- Exploration:
  Each node must be visited by at least one robot
- Termination:
  Eventually, every robot stays idle

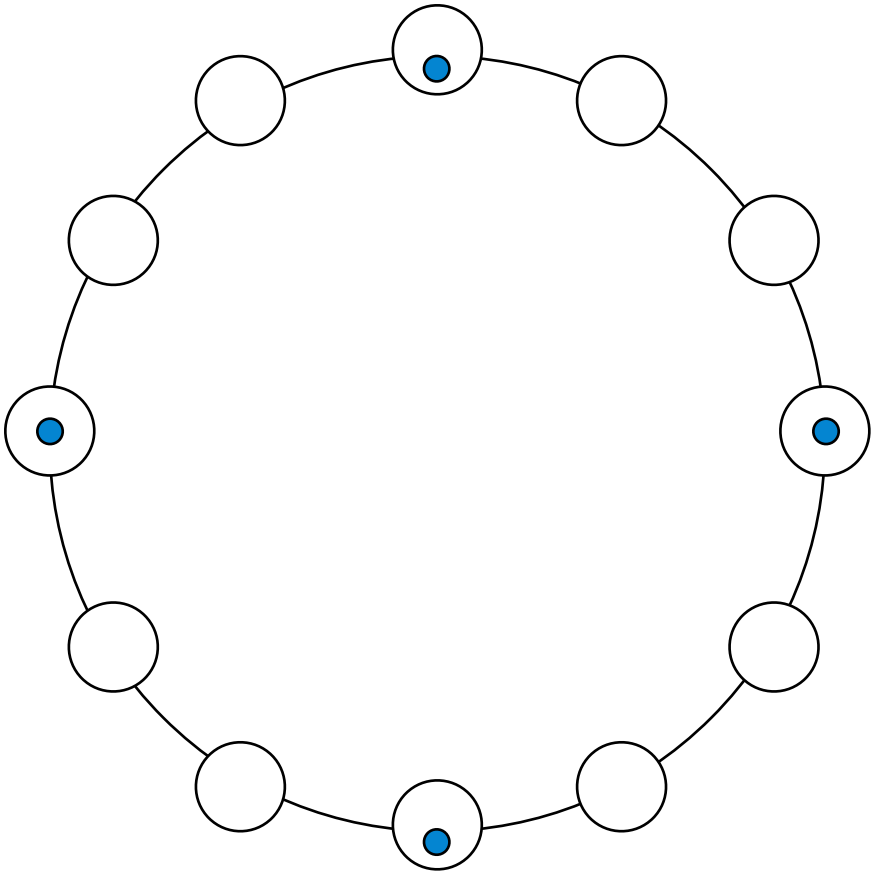
Problem

Starting from a configuration where no two robots are located at the same node:

- Exploration:
  Each node must be visited by at least one robot
- Termination:
  Eventually, every robot stays idle
- Performance: Number of robots ($k<n$)

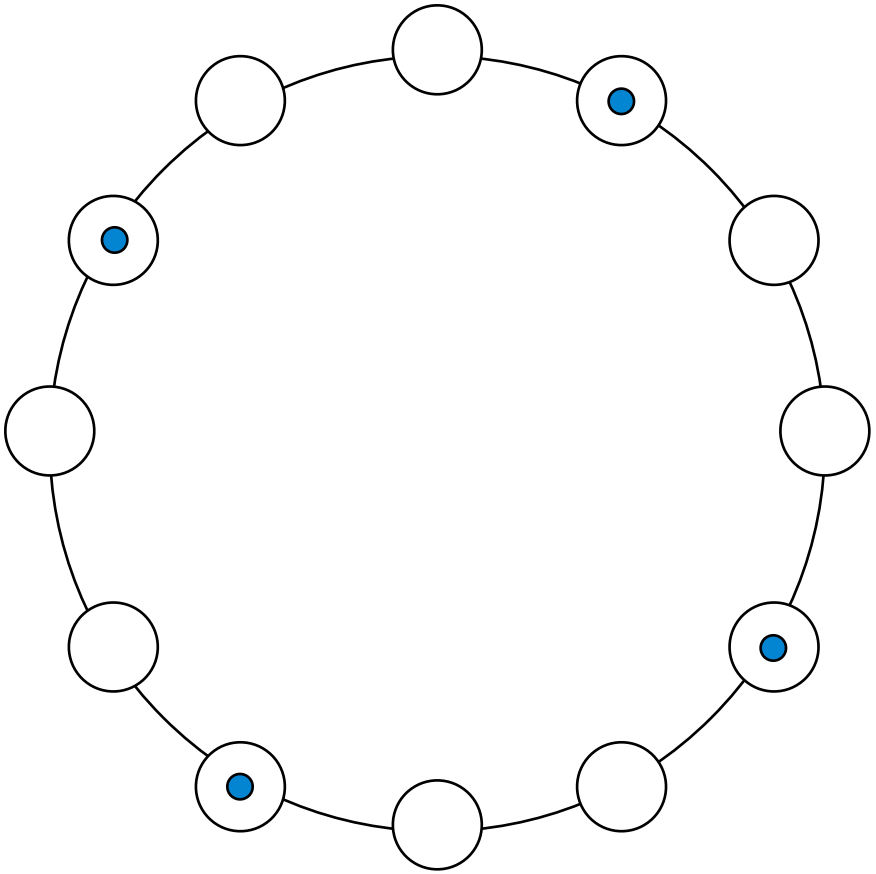
Lower Bound (1/2)

Deterministic Exploration impossible if $k \mid n$



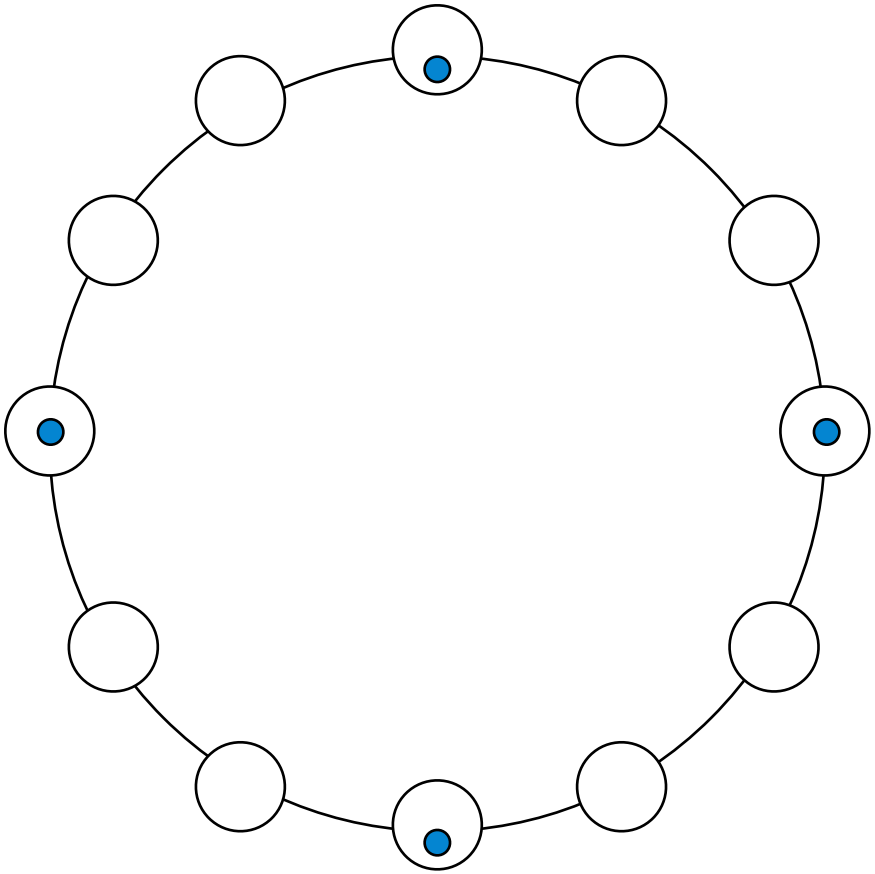
Lower Bound (1/2)

Deterministic Exploration impossible if $k \mid n$



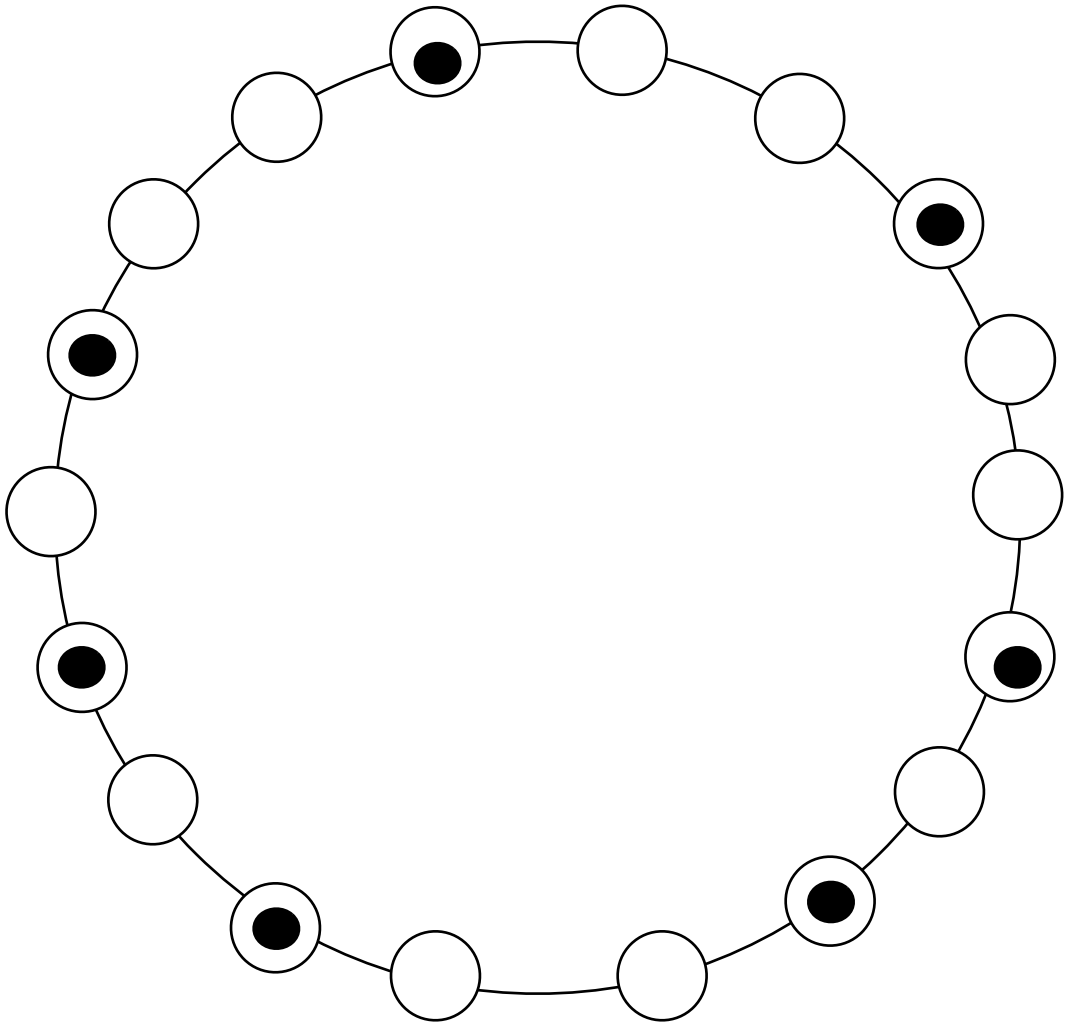
Lower Bound (1/2)

Deterministic Exploration impossible if $k \mid n$



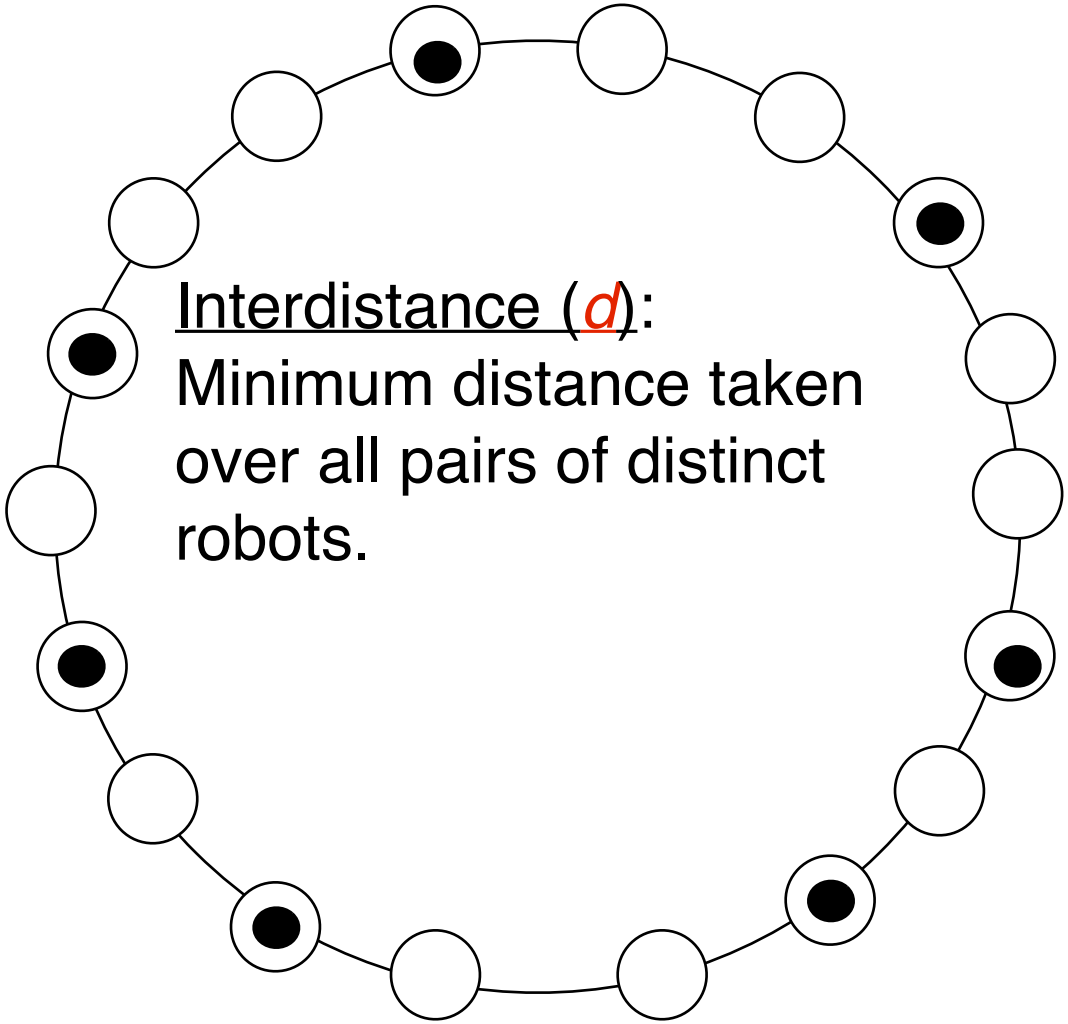
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



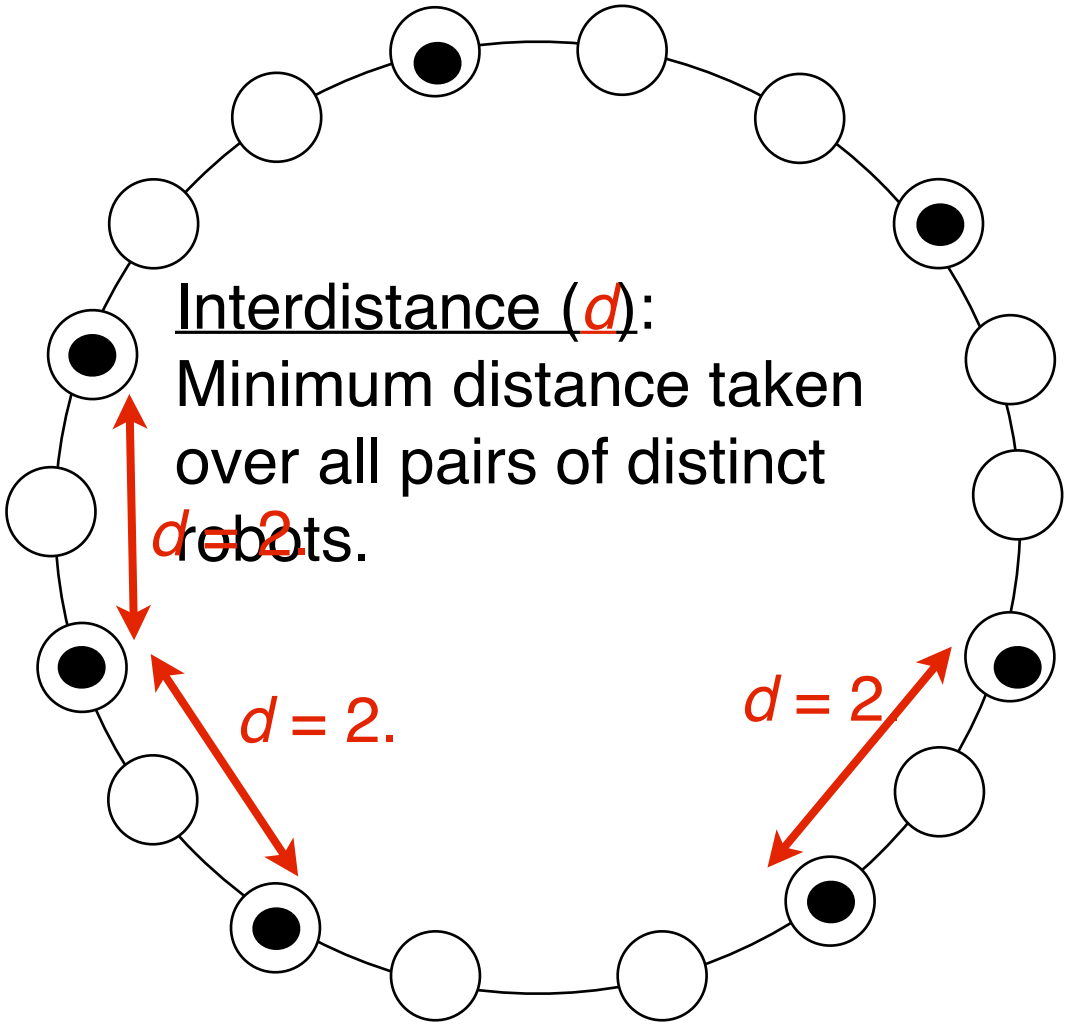
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



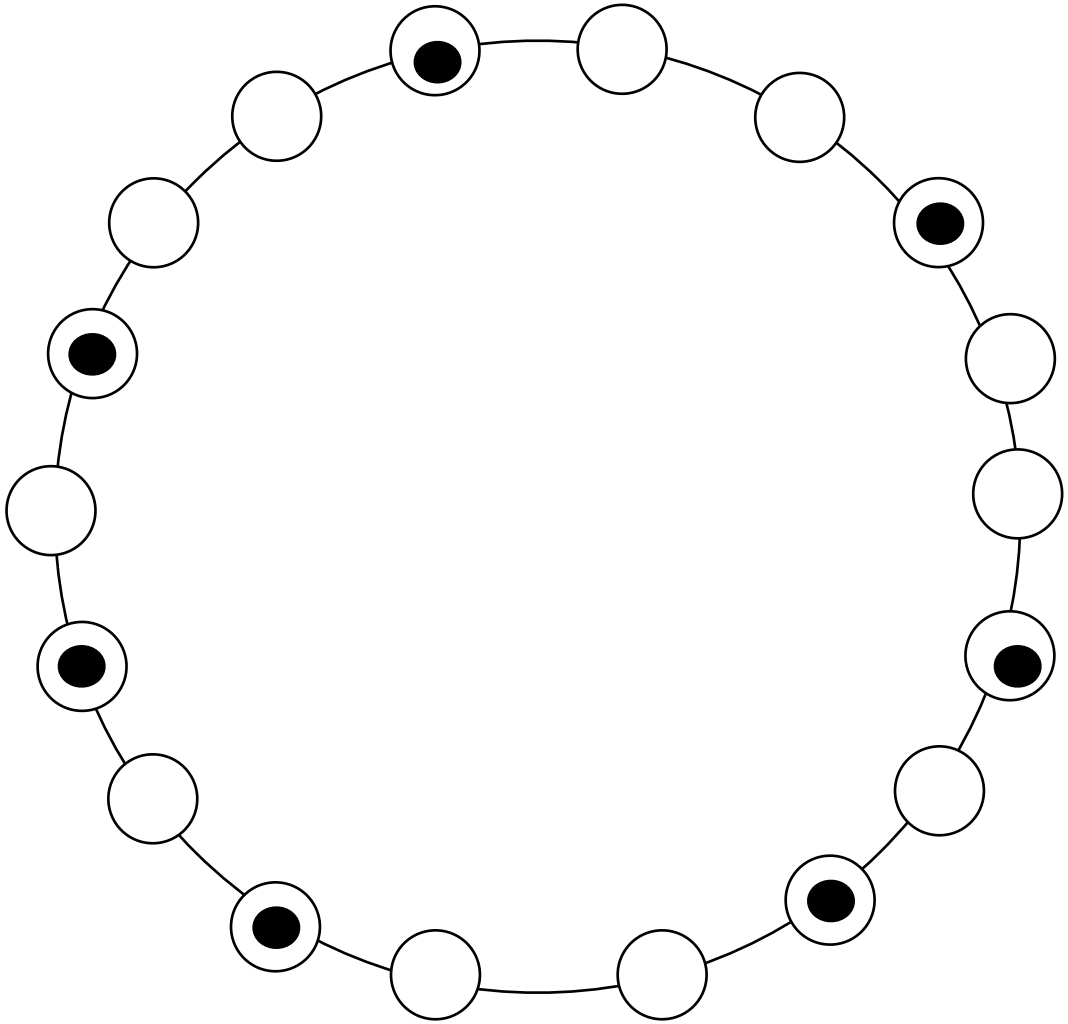
Deterministic Algorithm

- $k \neq n$, $k > \log n$, and $n > 16$



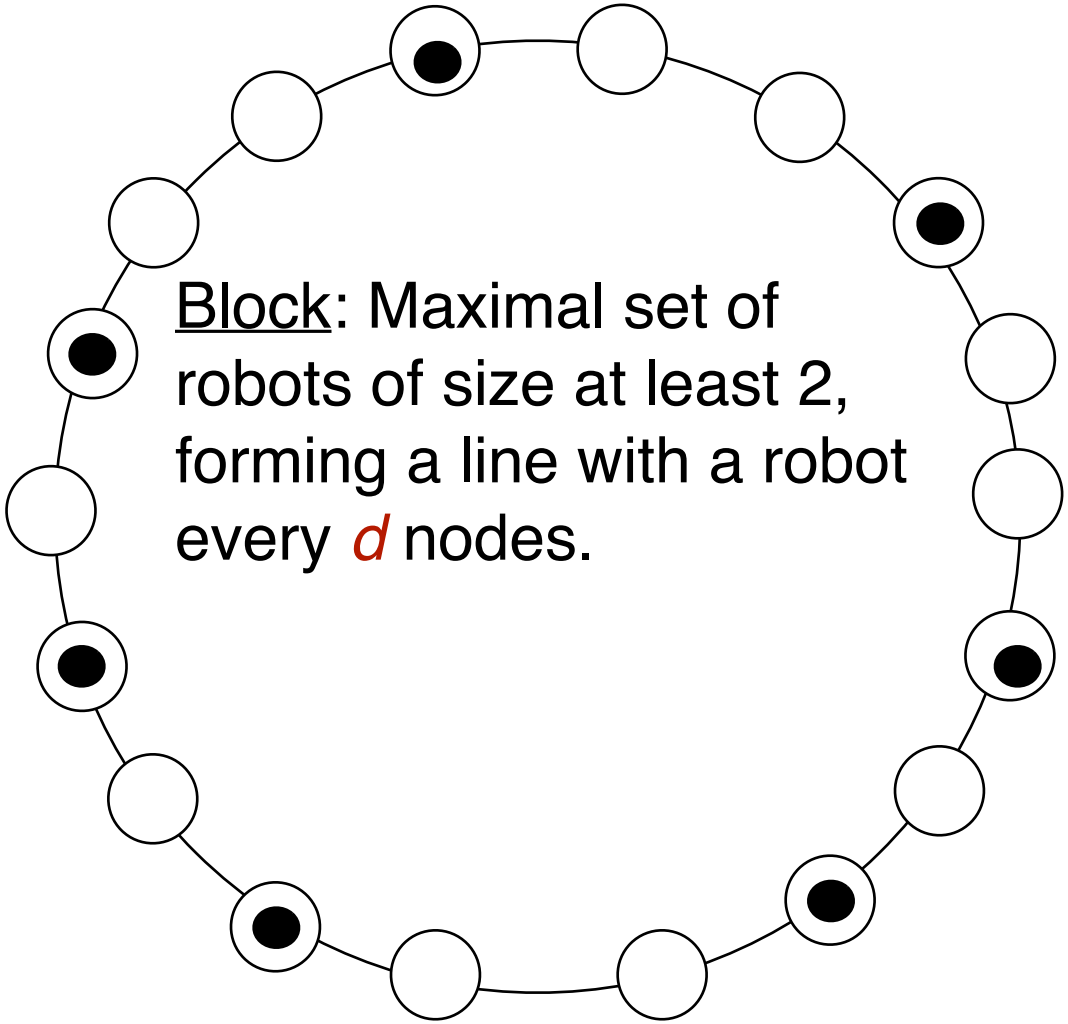
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



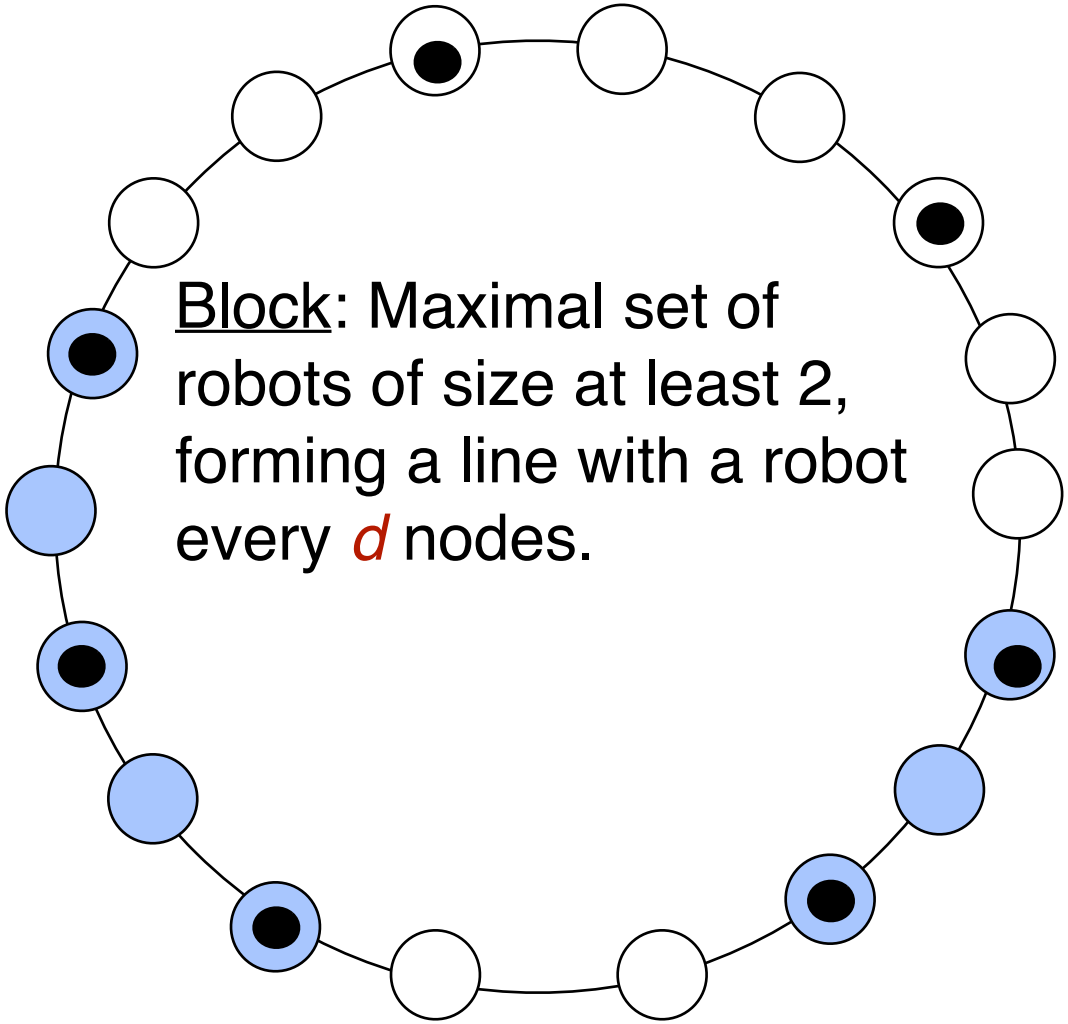
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



Deterministic Algorithm

Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$

Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$
- Setup Phase:

  Goal: Transform the (arbitrary) initial configuration into a configuration of interdistance 1 where there is a single block or two blocks of the same size.

  Method: Decrease the number of blocks whenever possible. Otherwise, decrease the interdistance.

Deterministic Algorithm

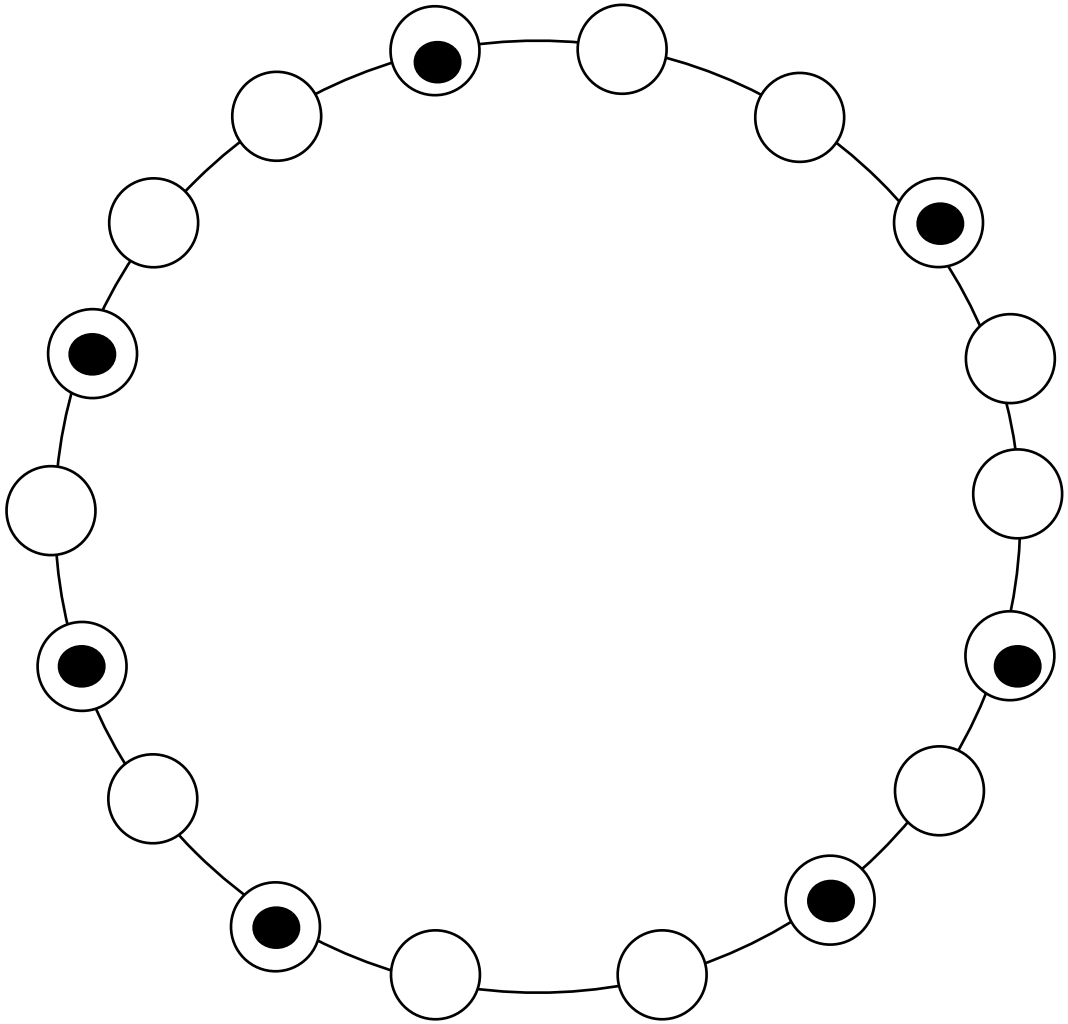
- $k \nmid n$, $k > \log n$, and $n > 16$
- Setup Phase:
  Goal: Transform the (arbitrary) initial configuration into a configuration of interdistance 1 where there is a single block or two blocks of the same size.
  Method: Decrease the number of blocks whenever possible. Otherwise, decrease the interdistance.
- Tower Phase:
  Goal: Create one or two multiplicities inside each block.

Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$
- Setup Phase:
  Goal: Transform the (arbitrary) initial configuration into a configuration of interdistance 1 where there is a single block or two blocks of the same size.
  Method: Decrease the number of blocks whenever possible. Otherwise, decrease the interdistance.
- Tower Phase:
  Goal: Create one or two multiplicities inside each block.
- Exploration Phase:
  Goal: Perform exploration until reaching an identified final configuration.

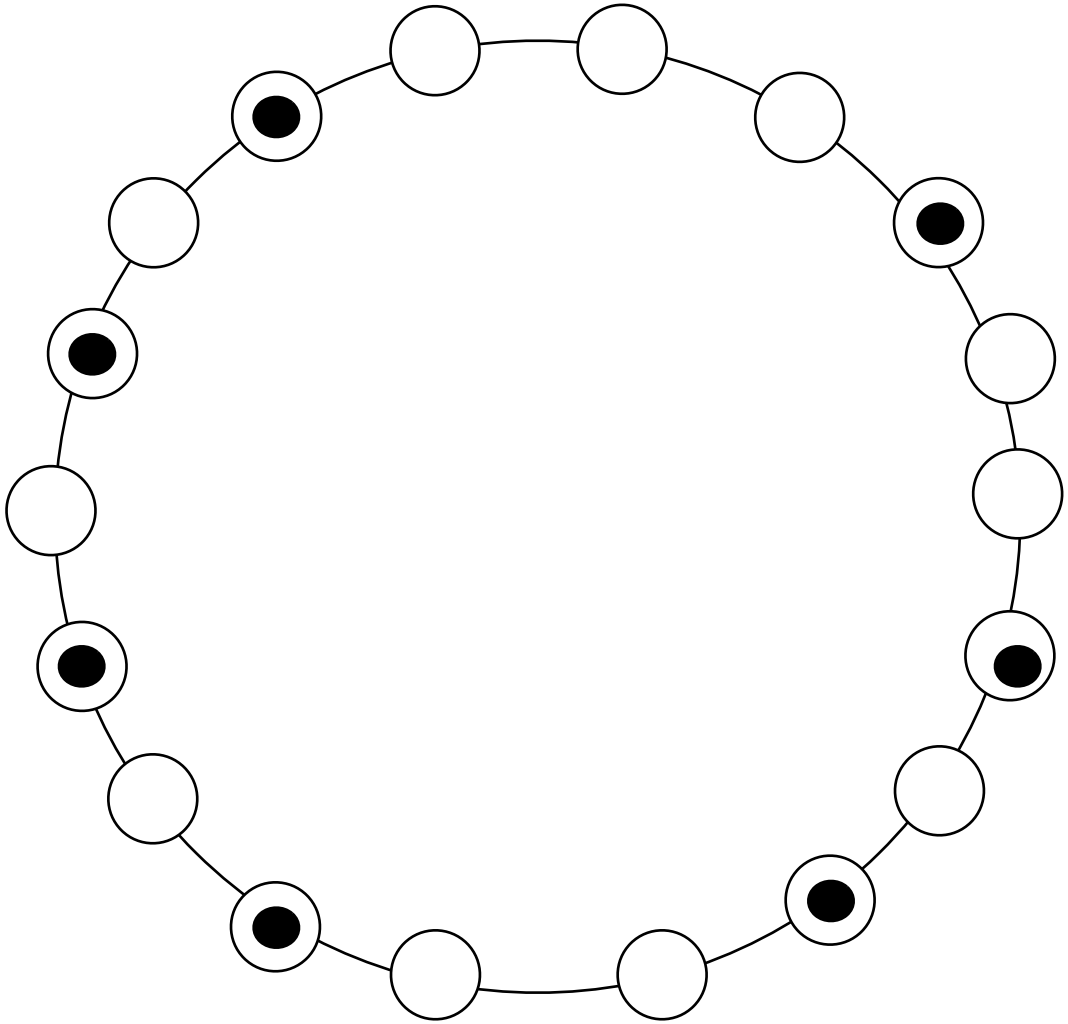
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



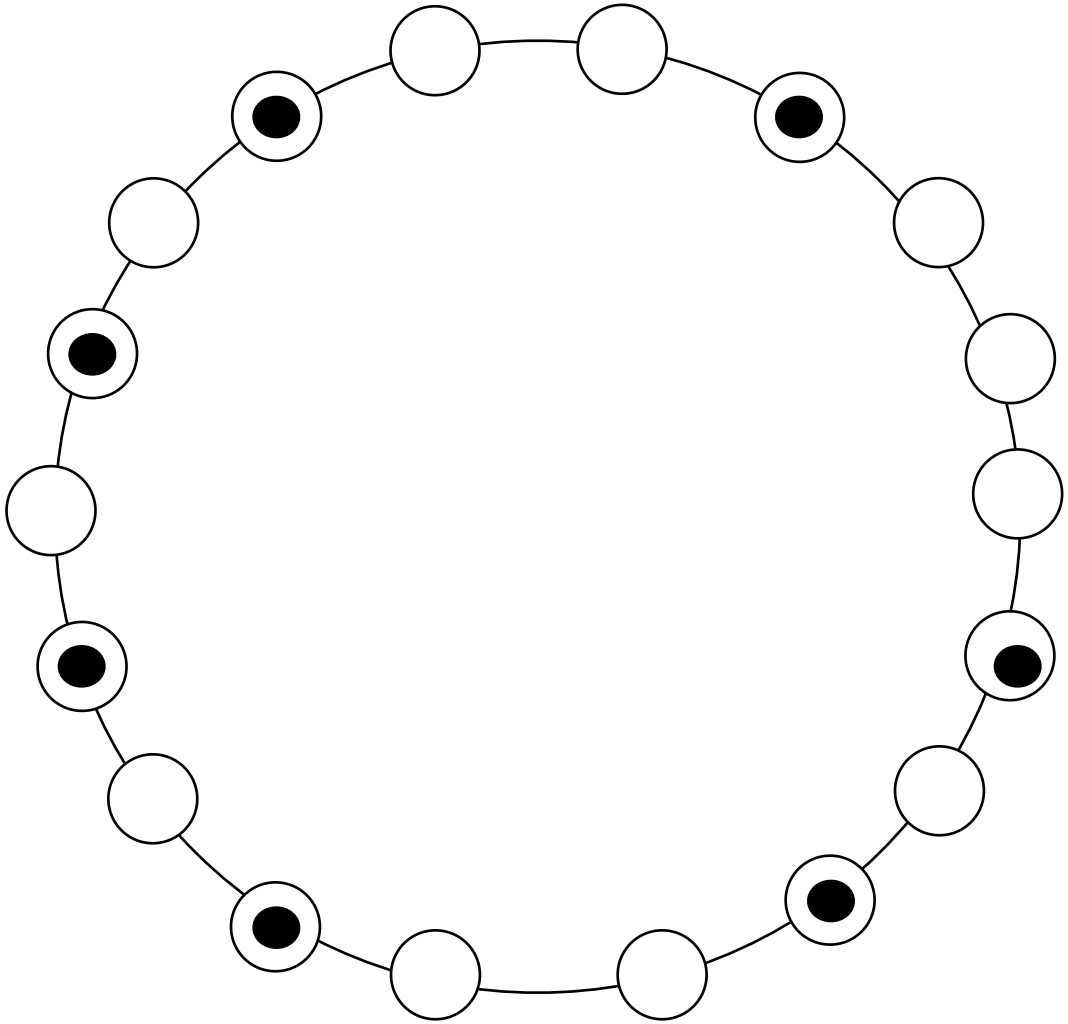
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



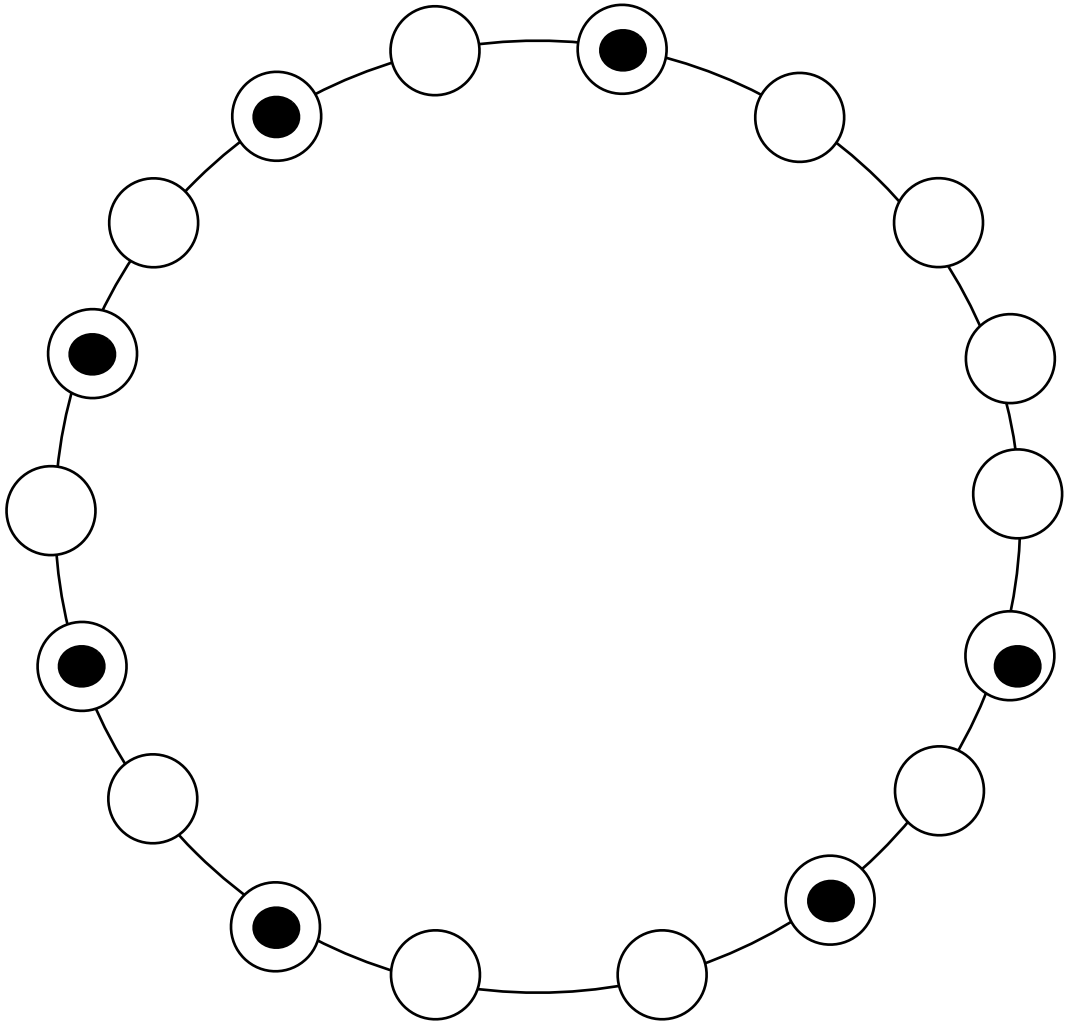
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



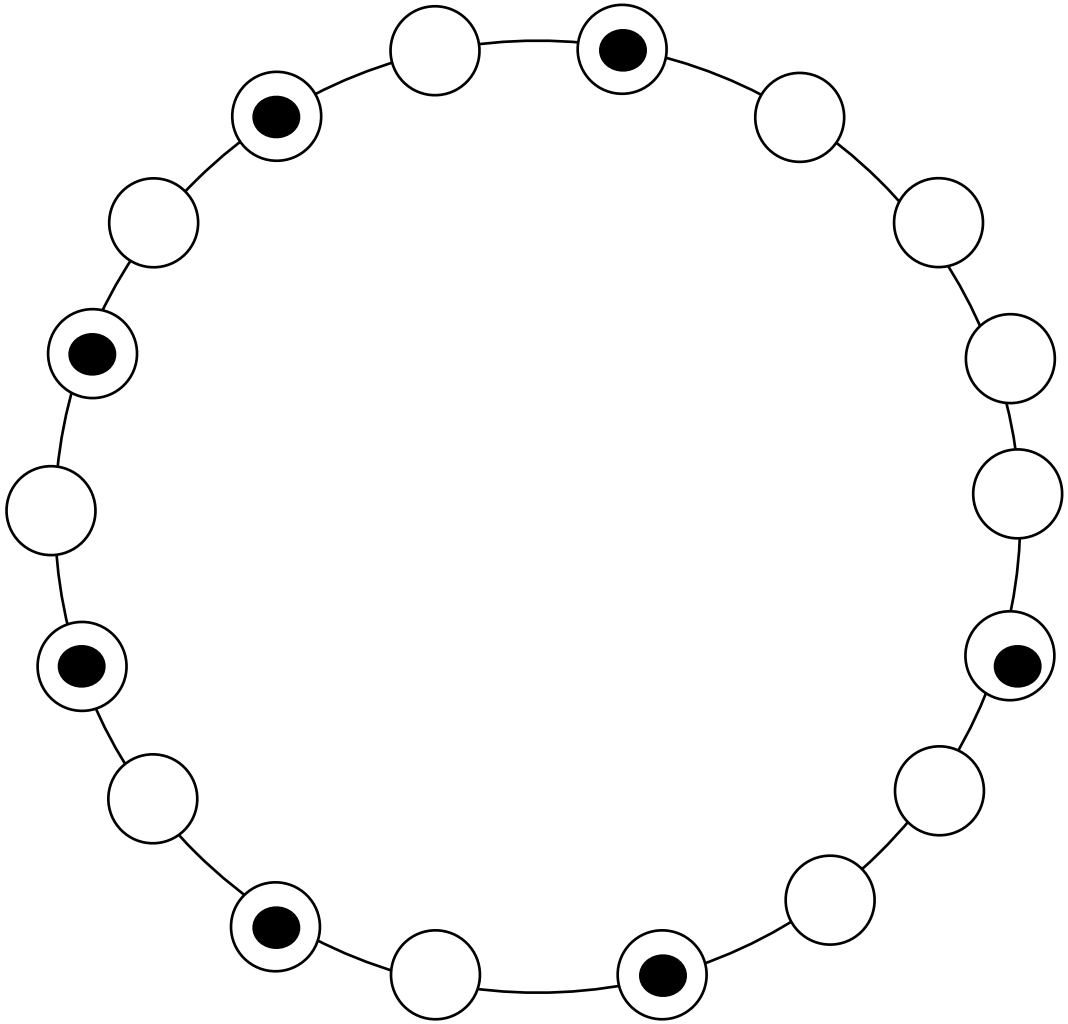
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



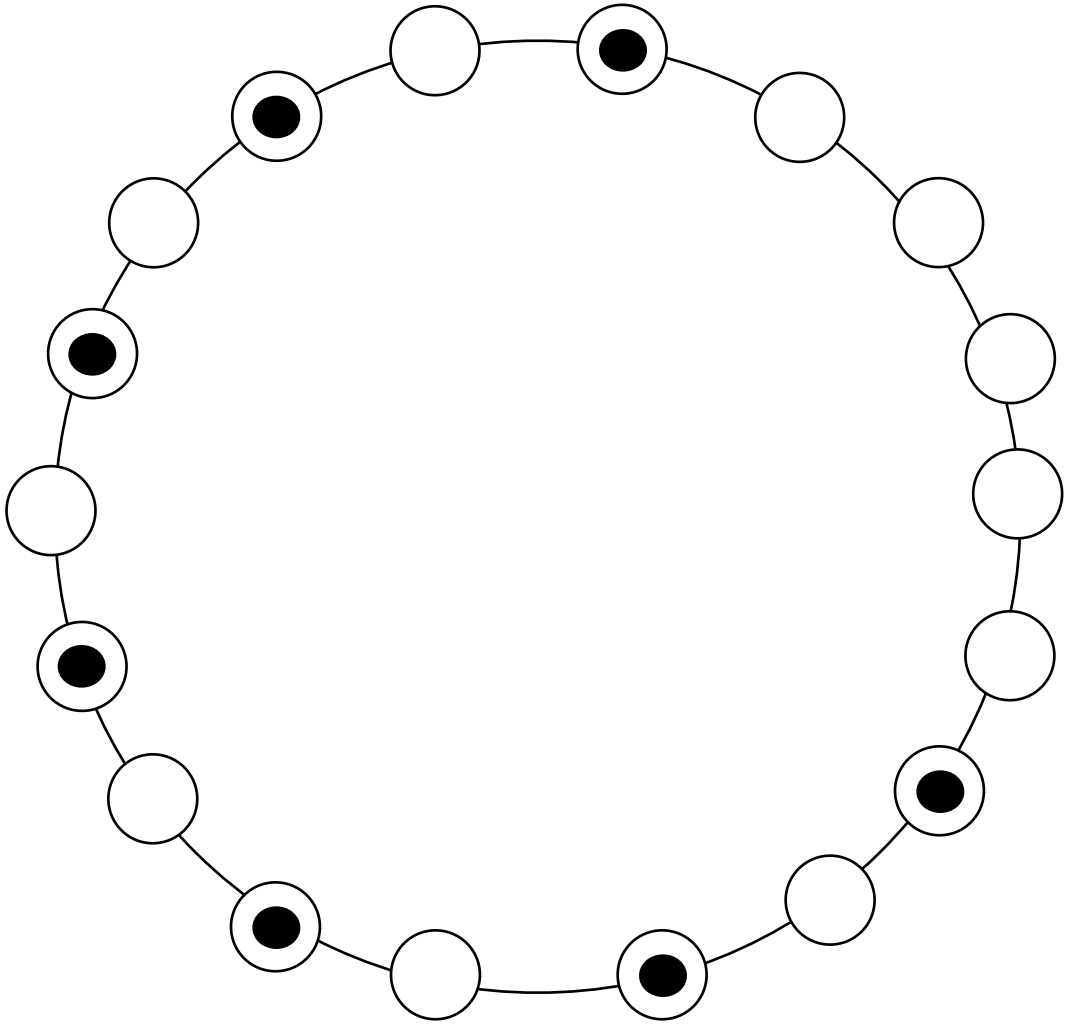
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



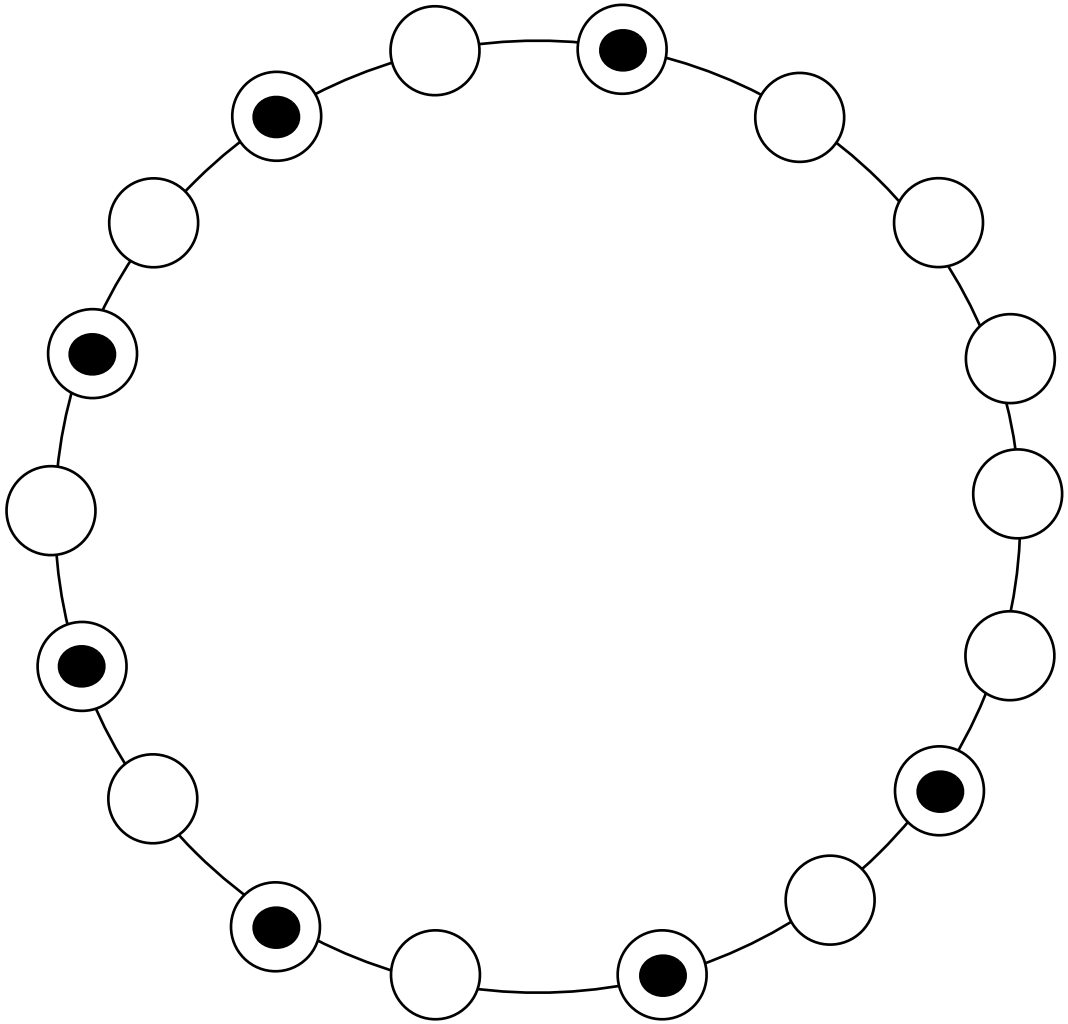
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



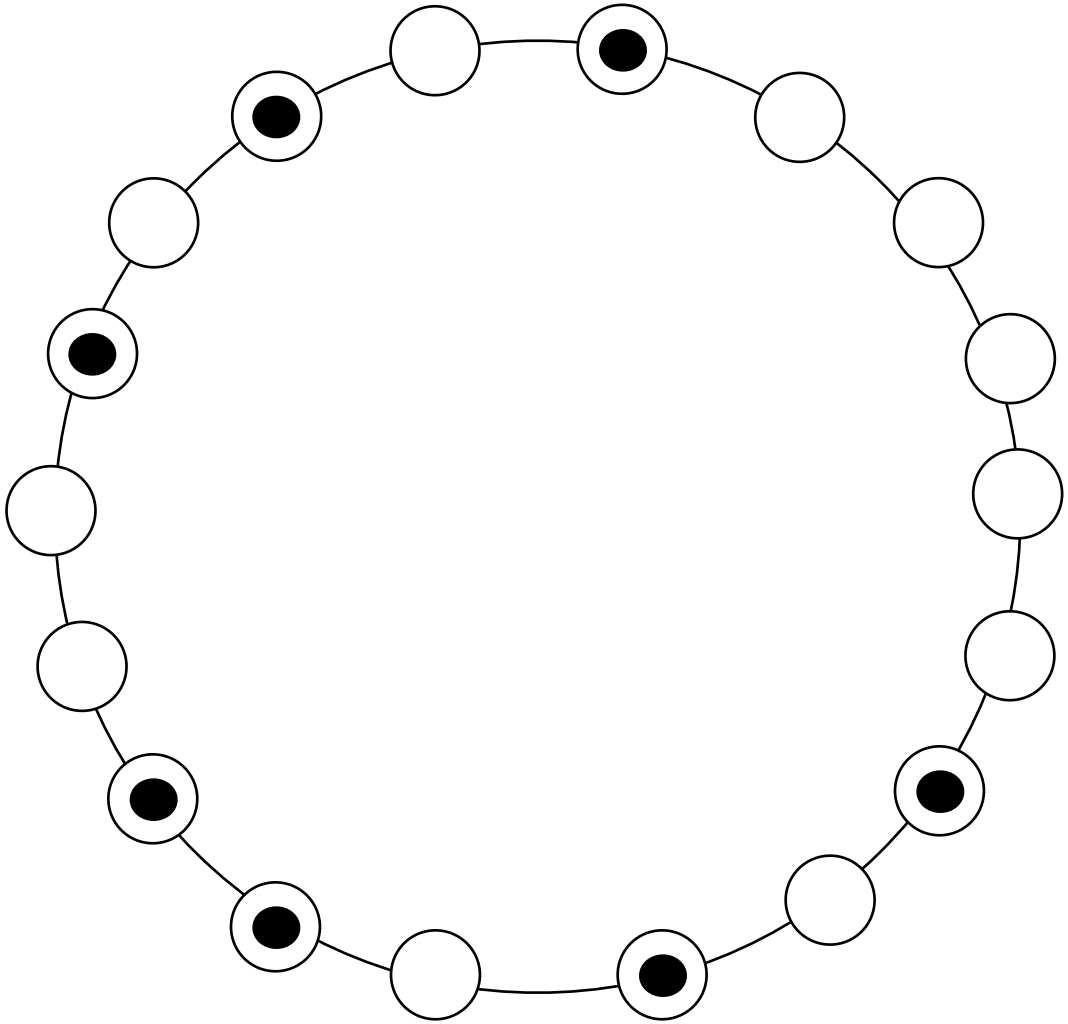
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



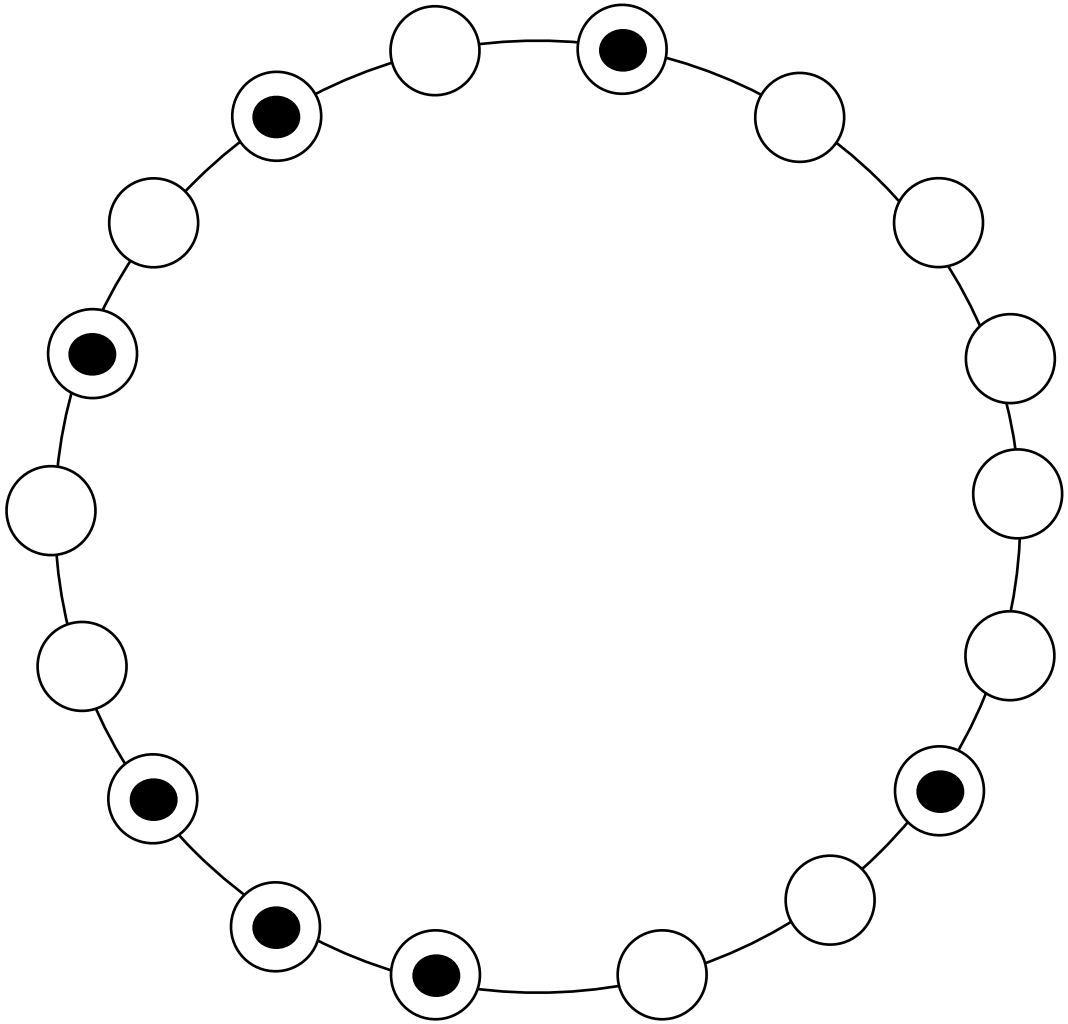
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



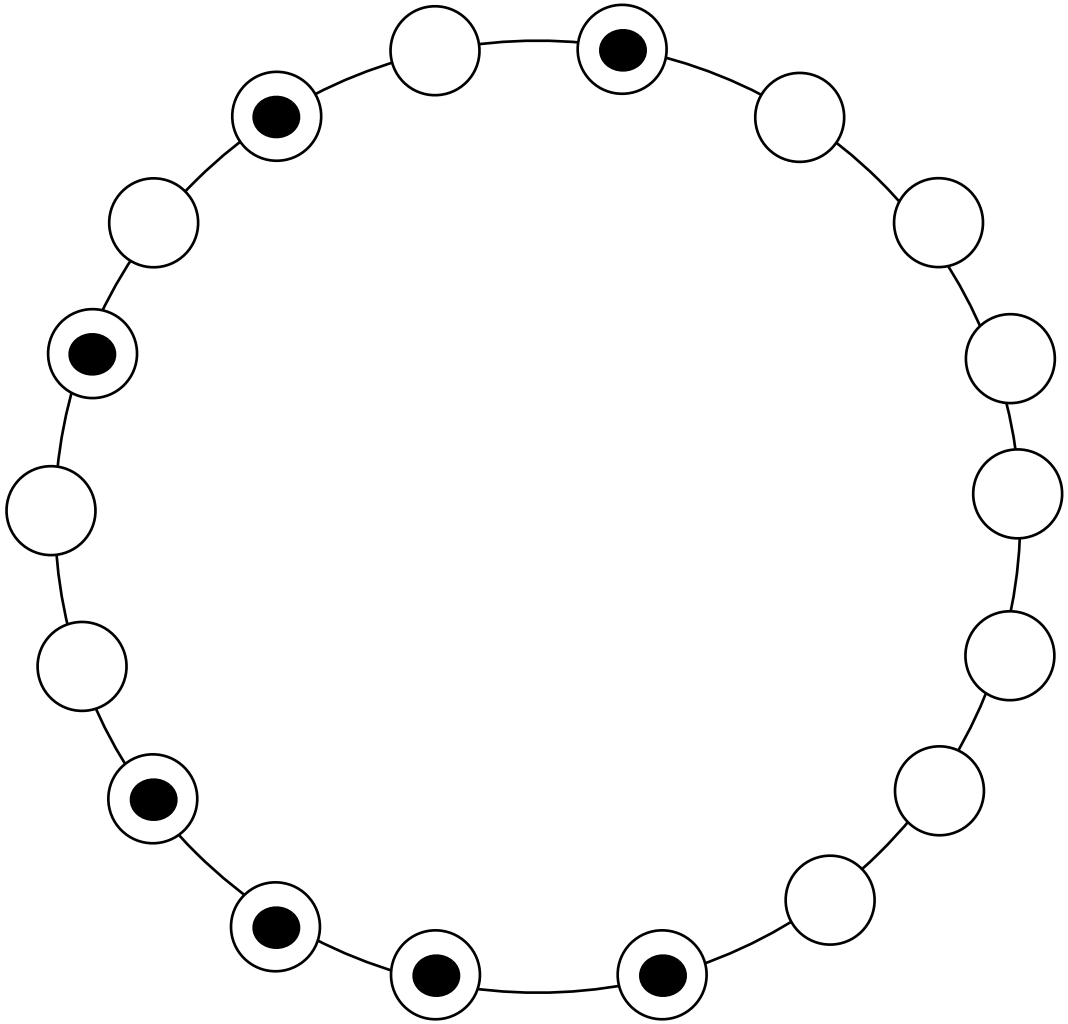
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



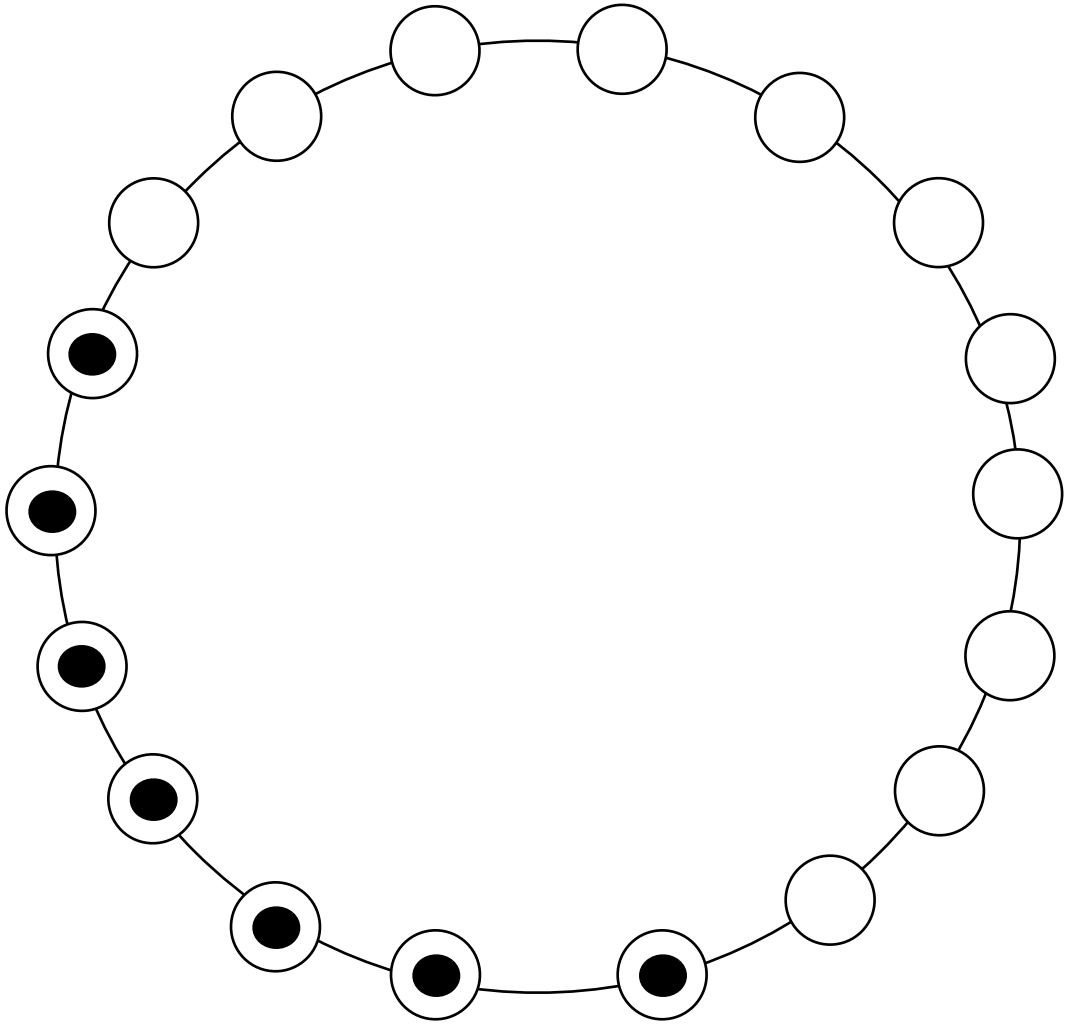
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



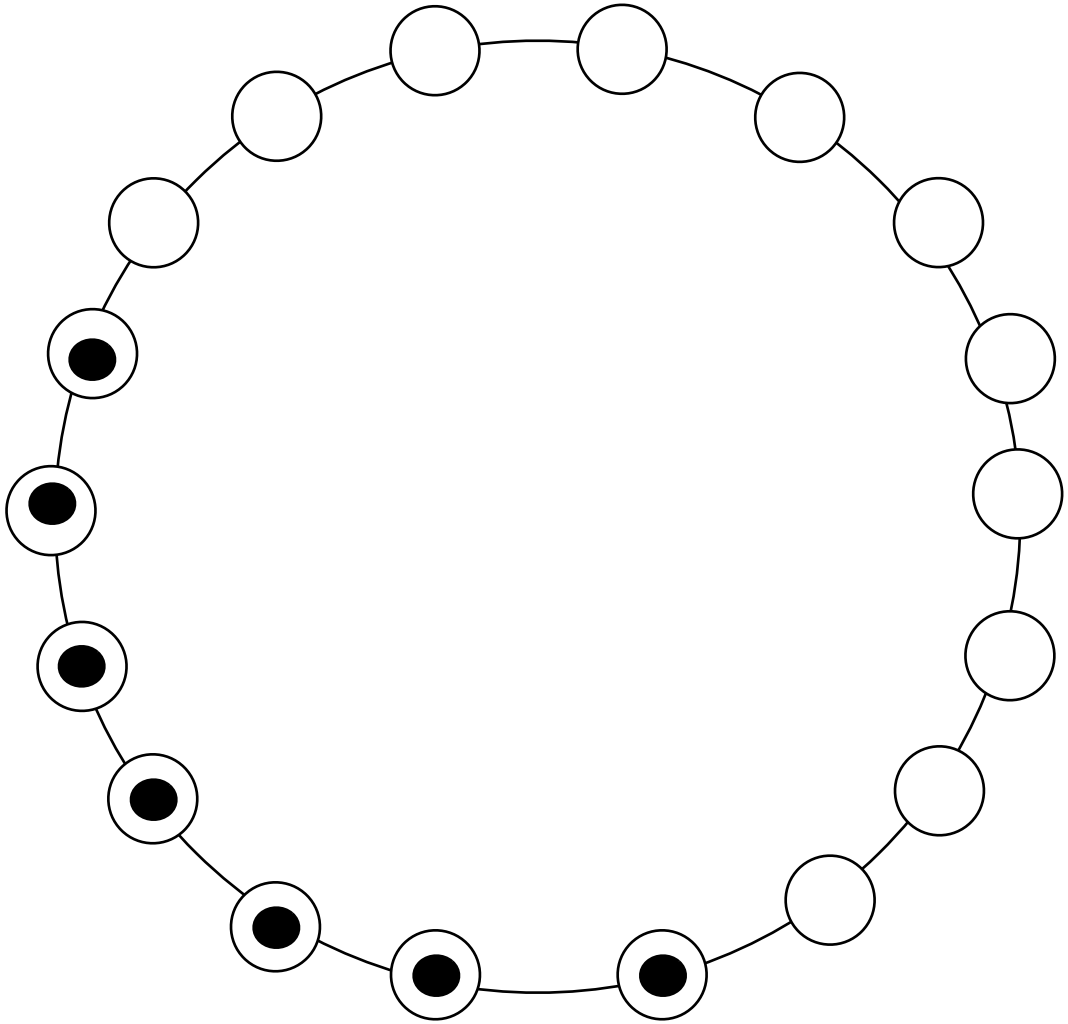
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



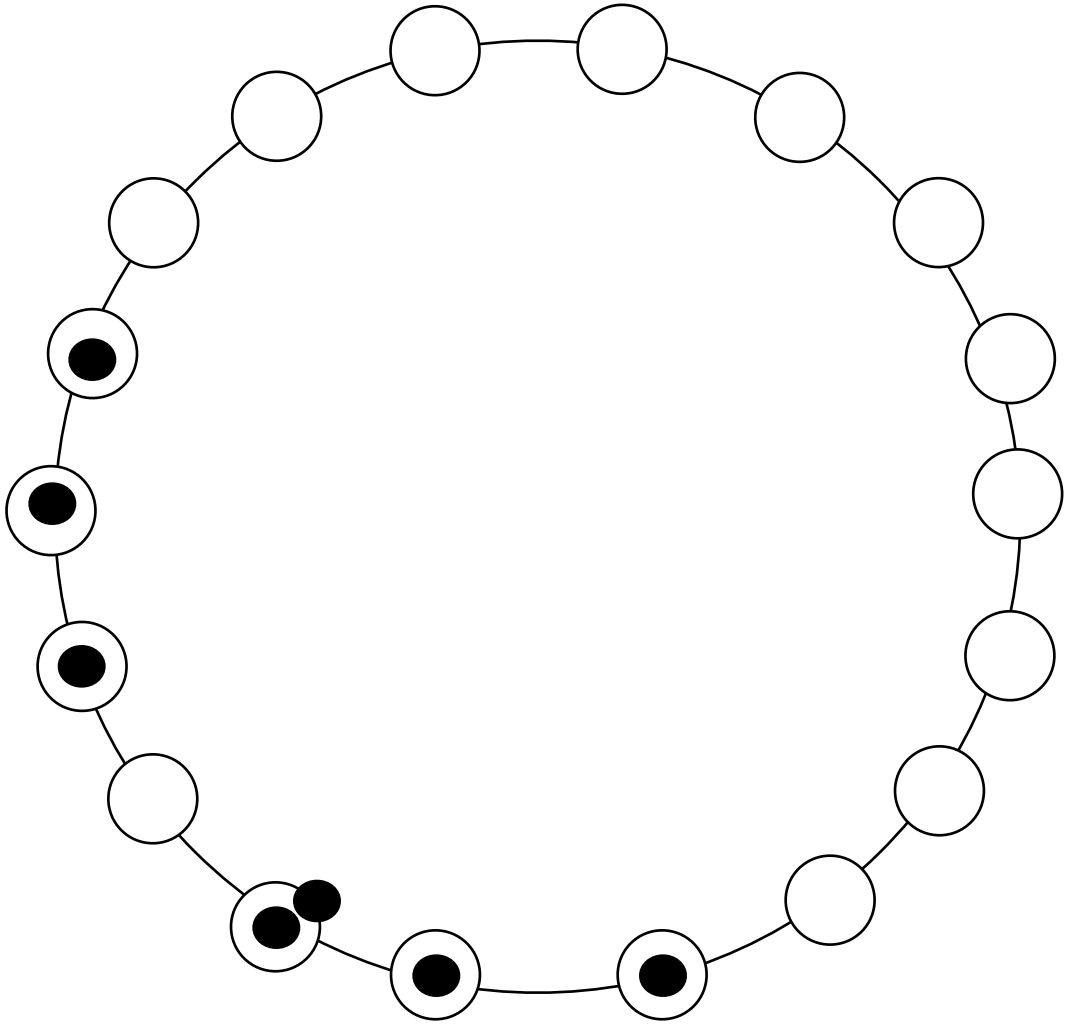
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



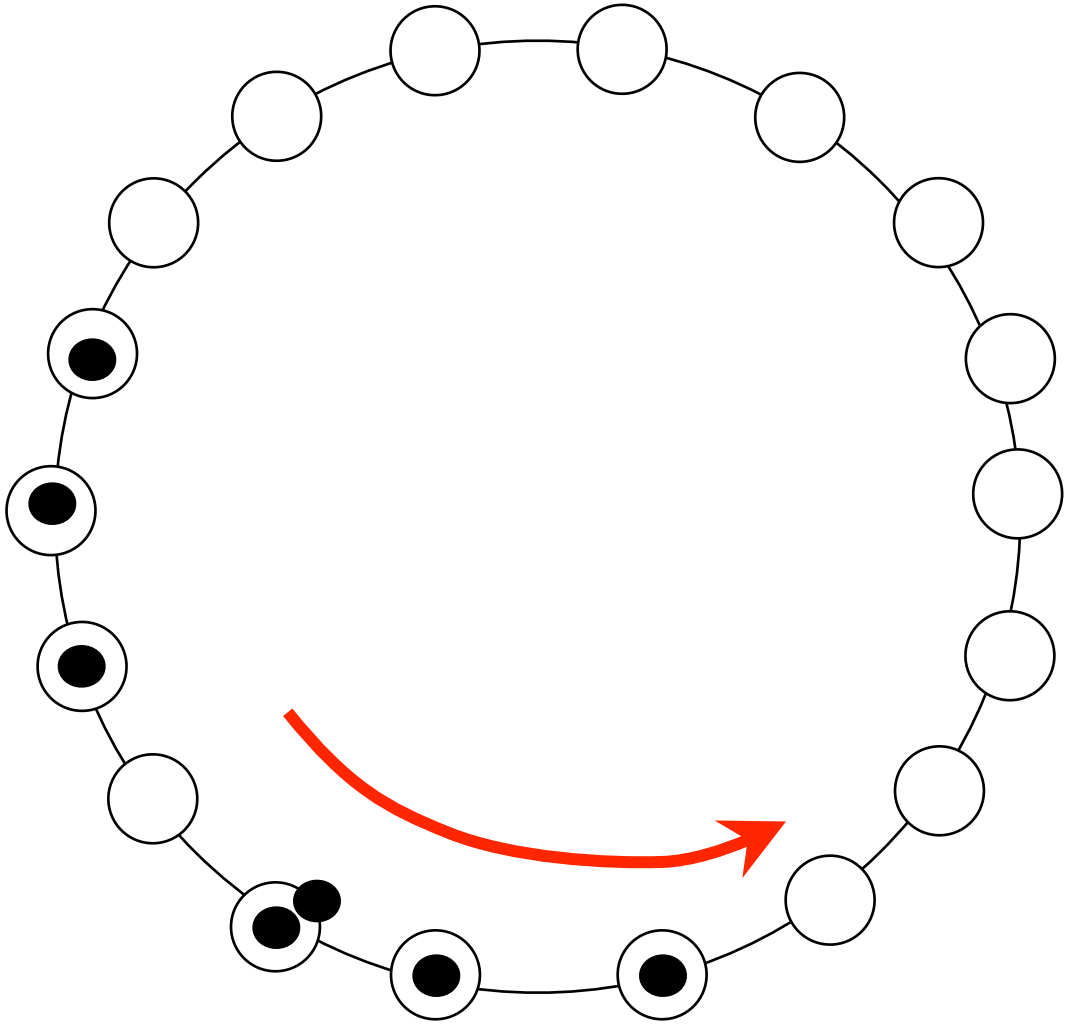
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



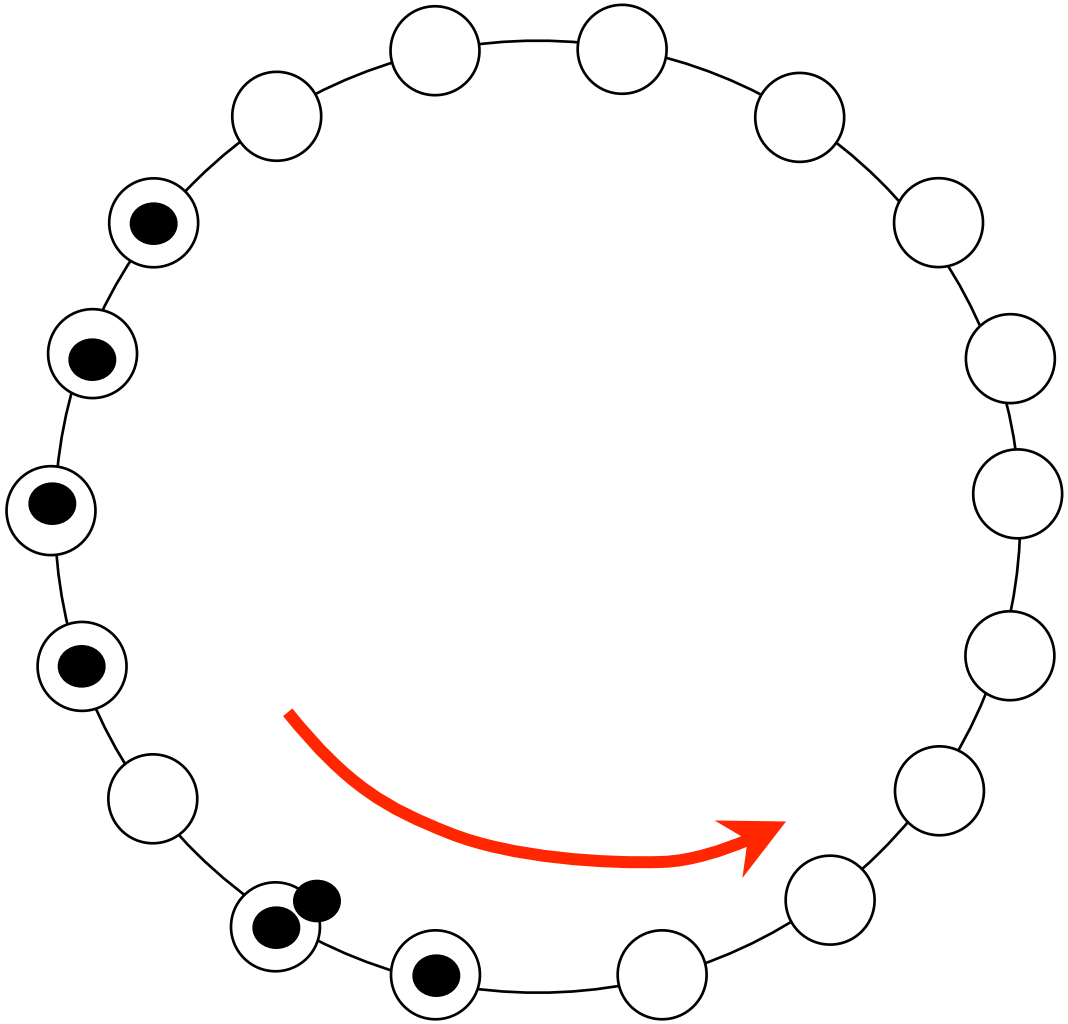
Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



Deterministic Algorithm

- $k \nmid n$, $k > \log n$, and $n > 16$



Optimality

Optimality

Optimality

- Minimal Number of Robots?

Optimality

Optimality

Theorem.
4 **probabilistic** robots are necessary and sufficient, provided that $n > 4$

Optimality

Theorem.
4 **probabilistic** robots are necessary and sufficient, provided that $n > 4$

- The theorem holds even if k divides n.

Optimality

Theorem.
4 **probabilistic** robots are necessary and sufficient, provided that $n > 4$

- The theorem holds even if k divides n.

Optimality

Theorem.
4 **probabilistic** robots are necessary and sufficient, provided that $n > 4$

- The theorem holds even if k divides n.

1. Exploration impossible with less than *4* robots

Optimality

Theorem.
4 **probabilistic** robots are necessary and sufficient, provided that $n > 4$

- The theorem holds even if k divides n.

1. Exploration impossible with less than *4* robots
2. An algorithm working with *4* probabilistic robots ($n > 4$)

Optimality

Theorem.
4 **probabilistic** robots are necessary and sufficient, provided that $n > 4$

- The theorem holds even if k divides n.

1. Exploration impossible with less than *4* robots
2. An algorithm working with *4* probabilistic robots ($n > 4$)

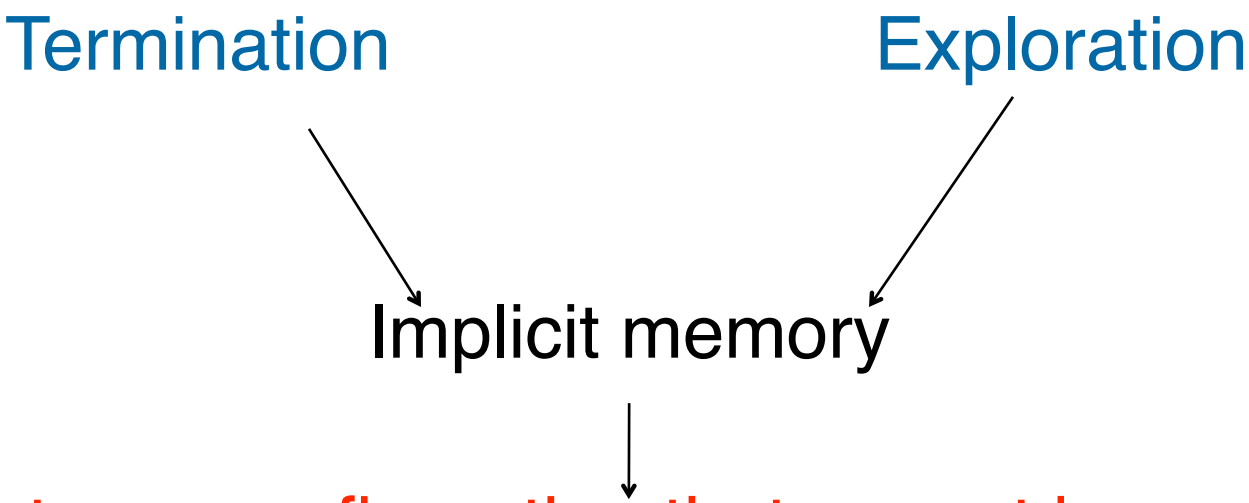
Oblivious Robots

Termination

Exploration

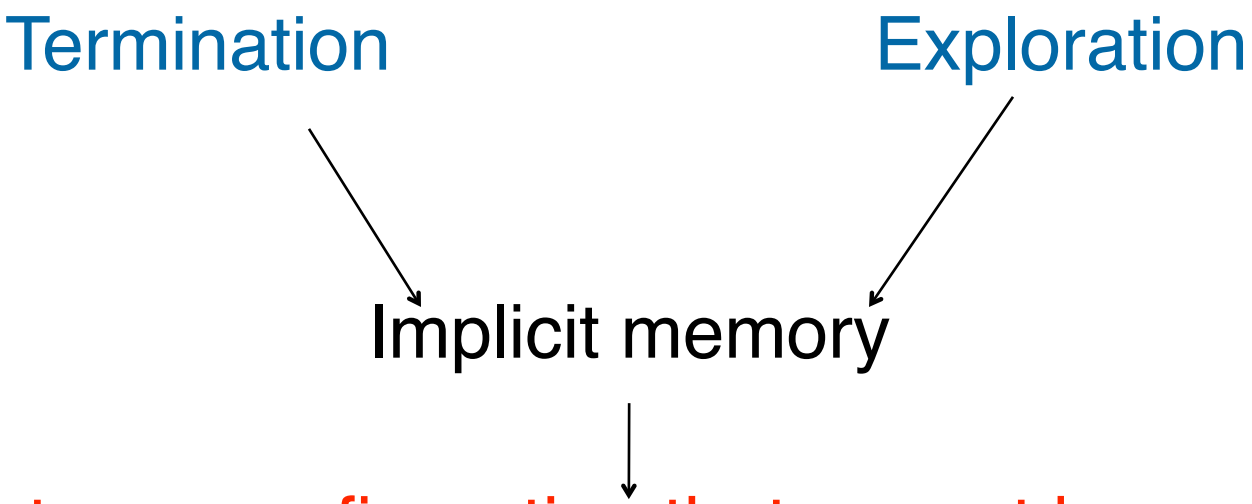
Oblivious Robots

Oblivious Robots



At least one configuration that cannot be an initial configuration

Oblivious Robots



Remark.
If $n > k$, any terminal configuration of any protocol contains at least one *tower*.

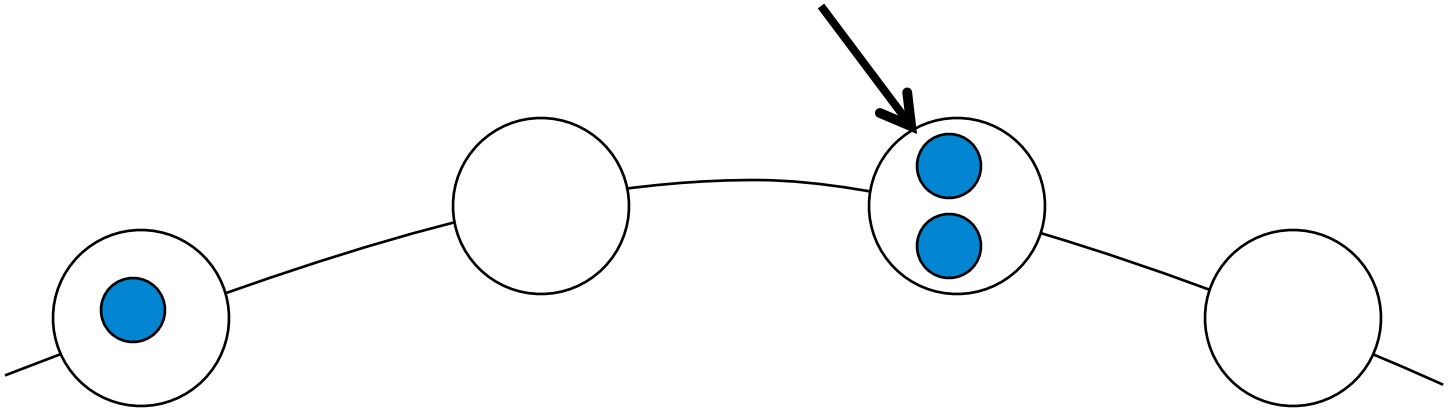
Tower

Tower

Definition.
A node with at least two robots.

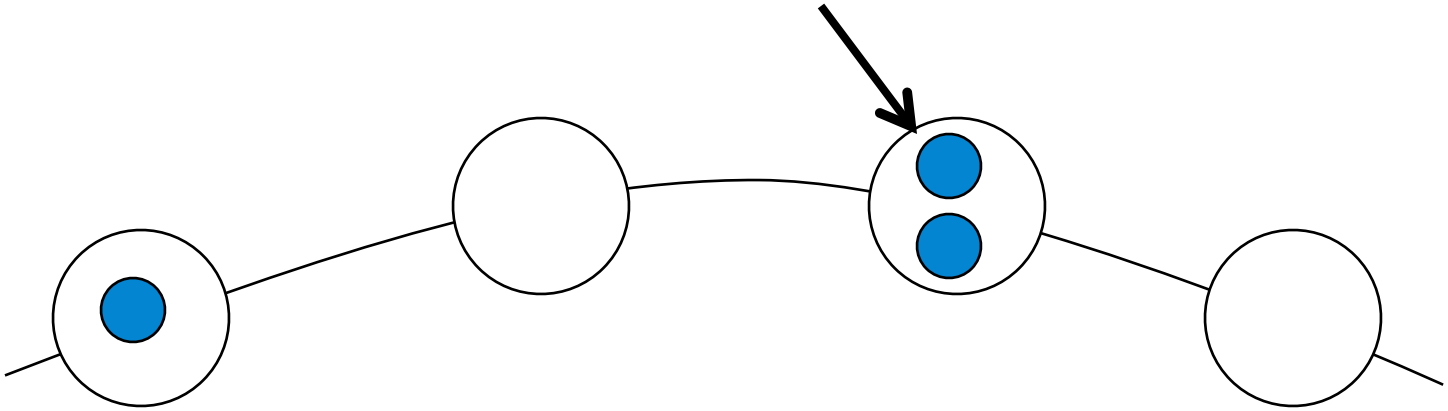
Tower

Definition.
A node with at least two robots.



Tower

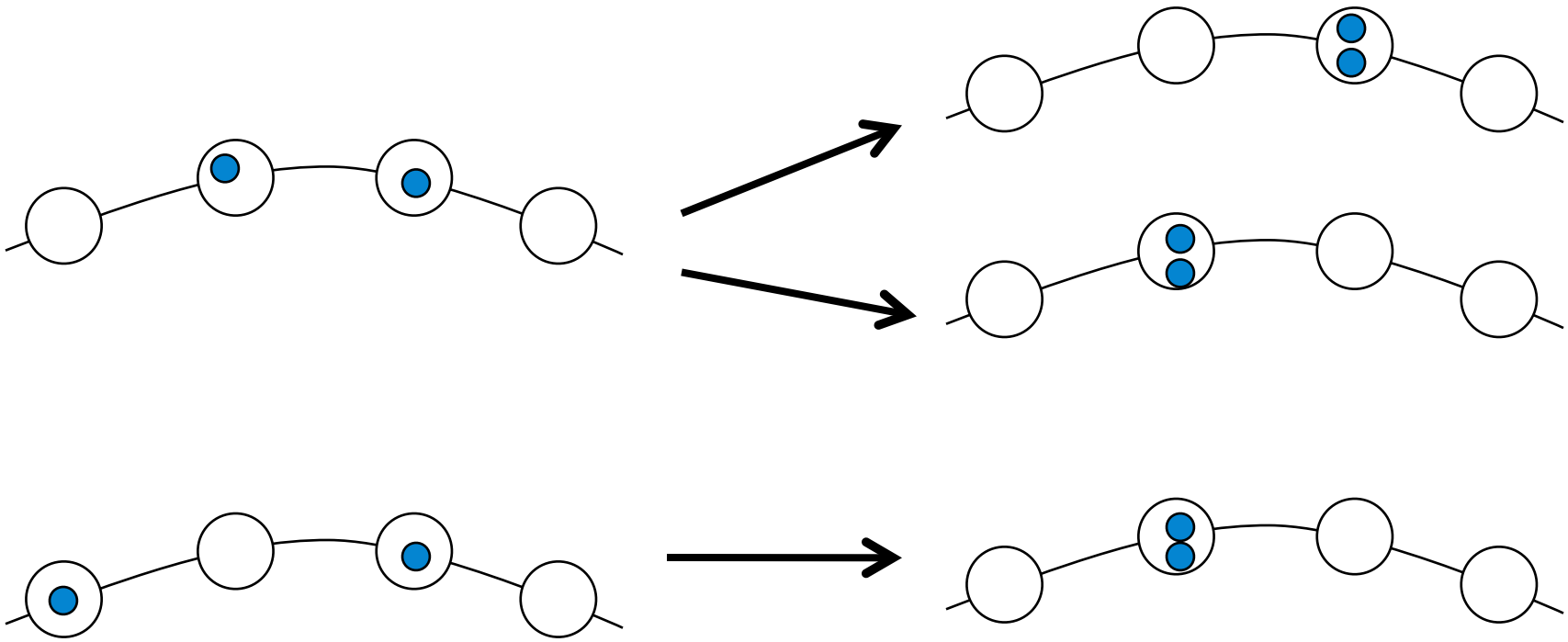
Definition.
A node with at least two robots.



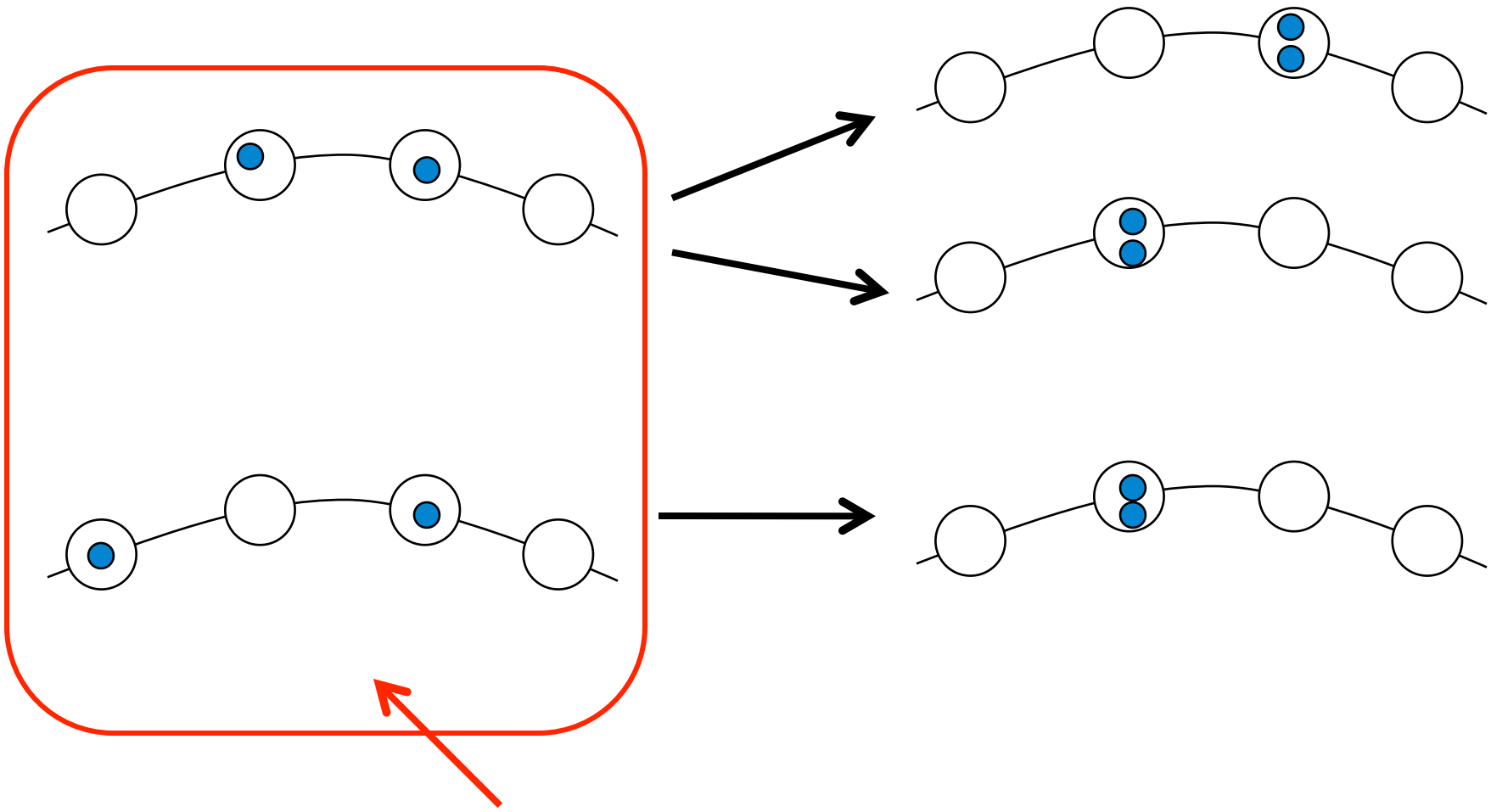
$k \geq 2$

Tower Building

Tower Building

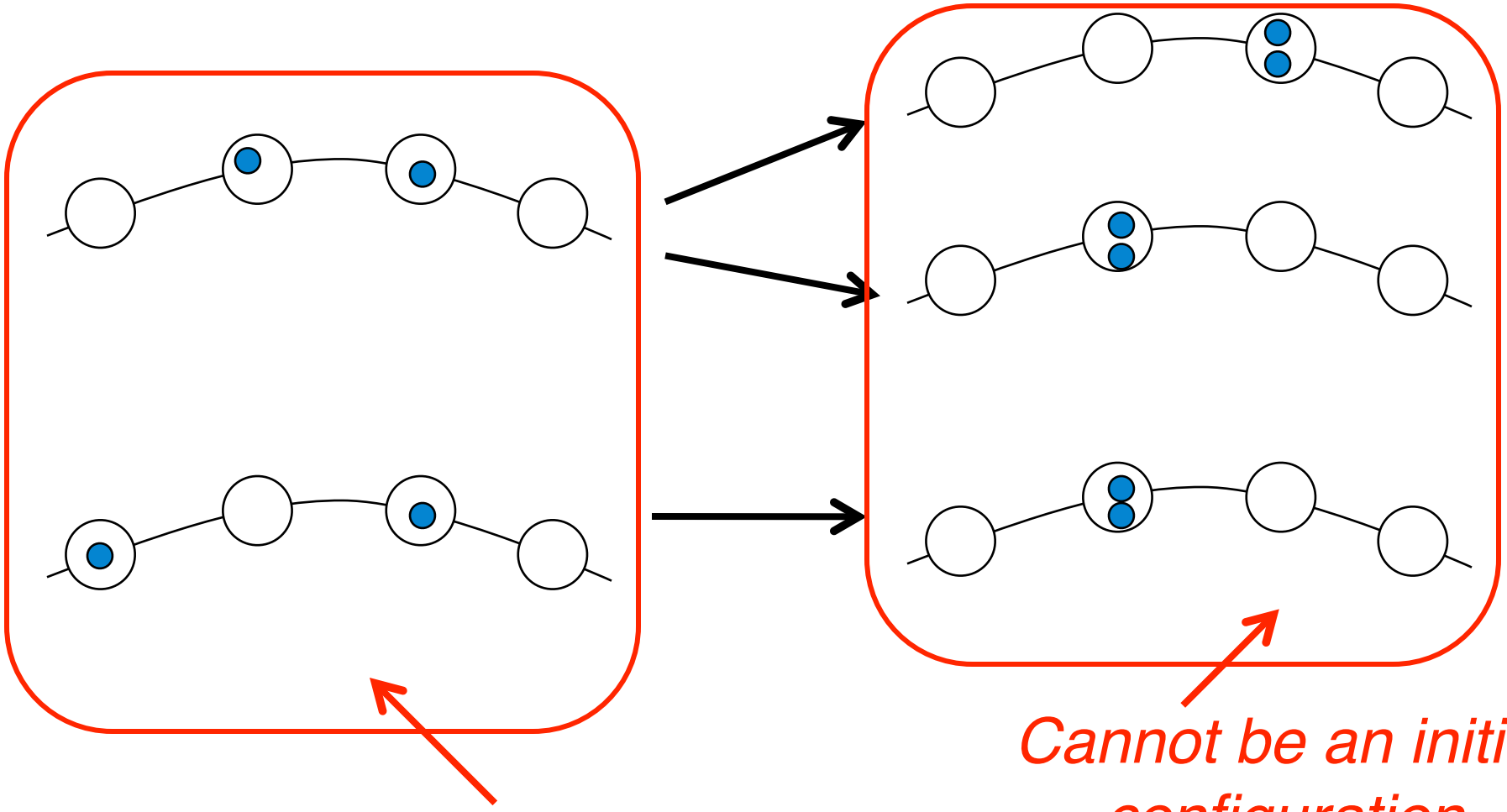


Tower Building

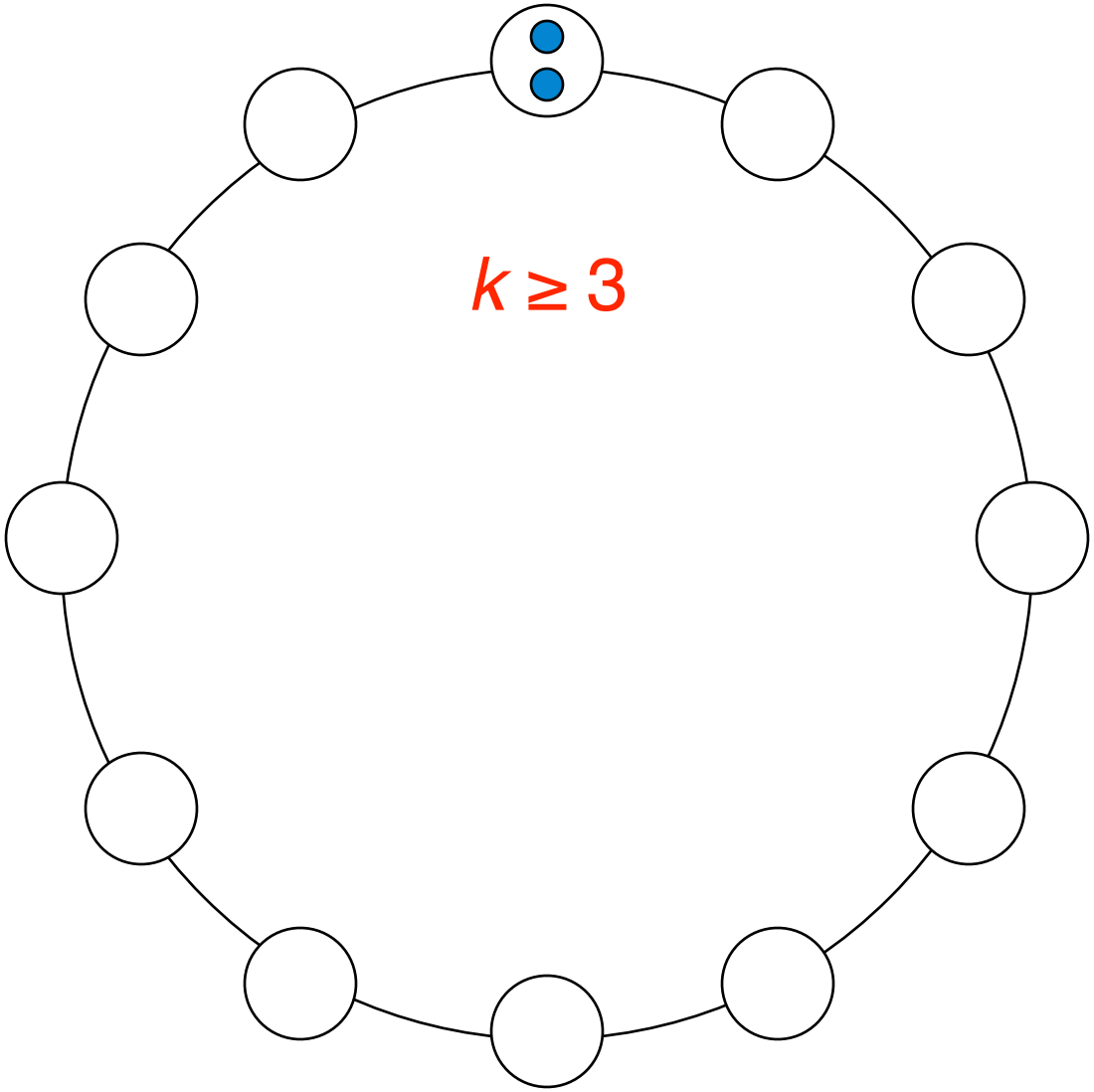


Can be an initial configuration

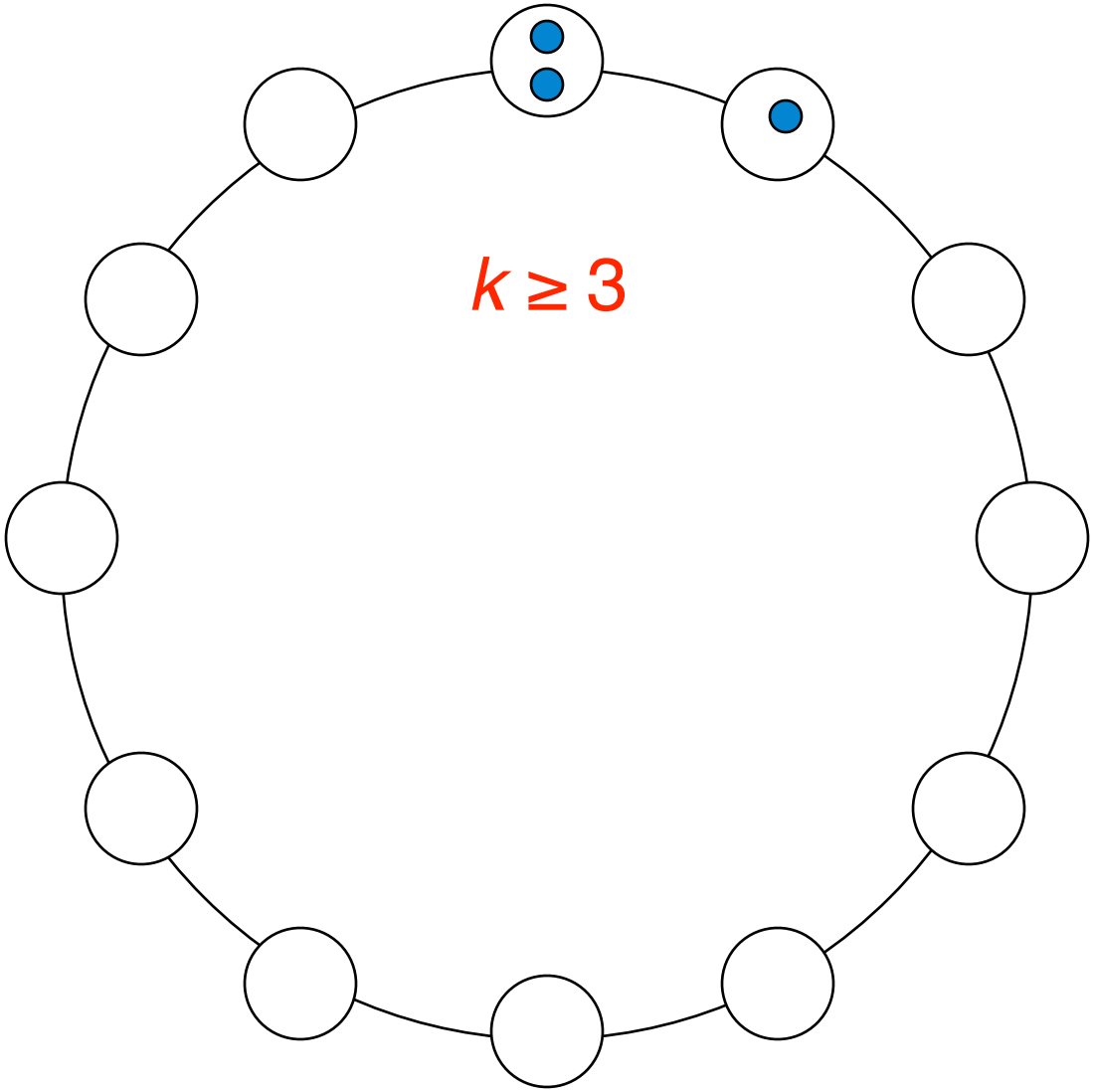
Tower Building



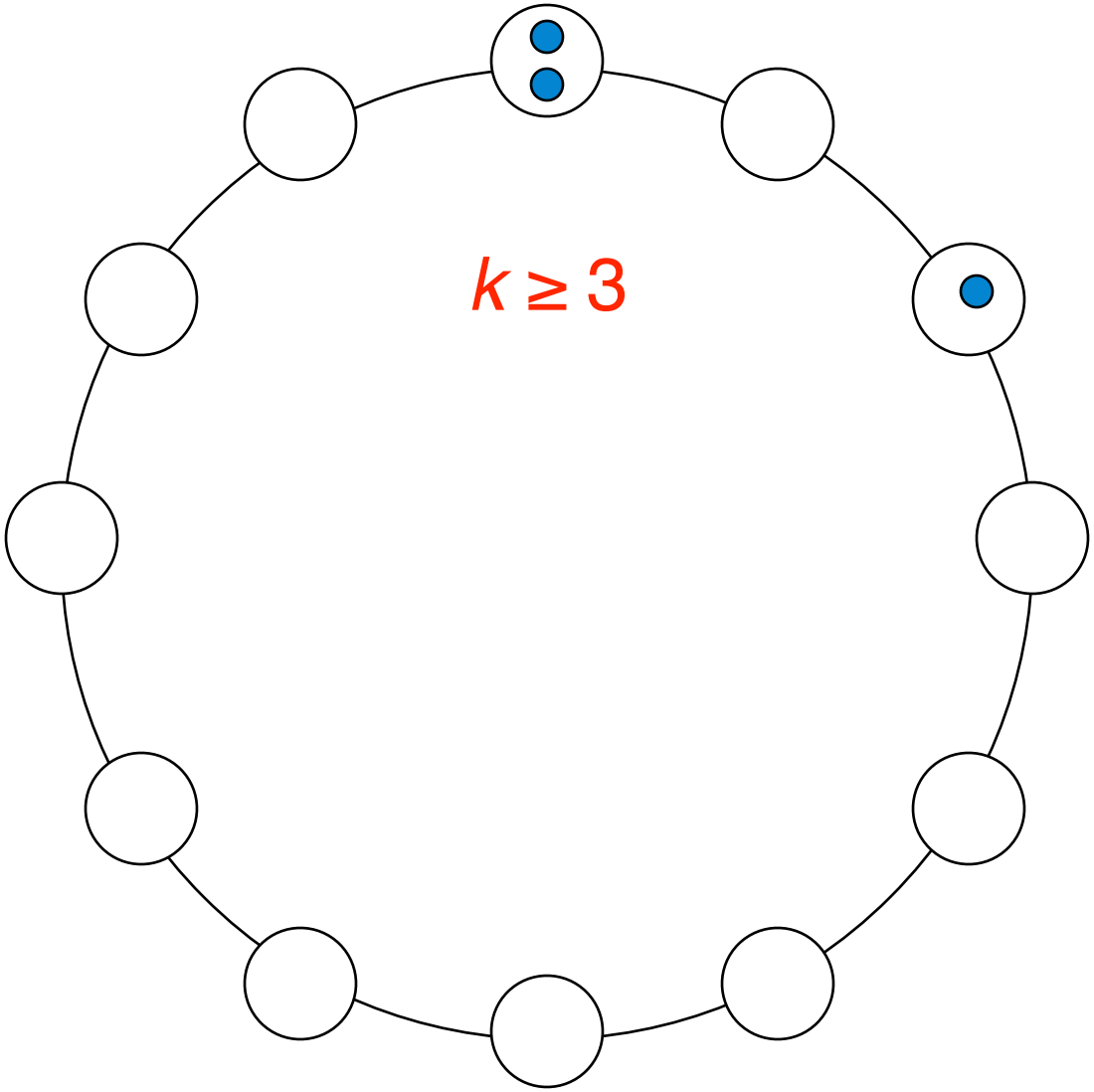
Enabling Exploration



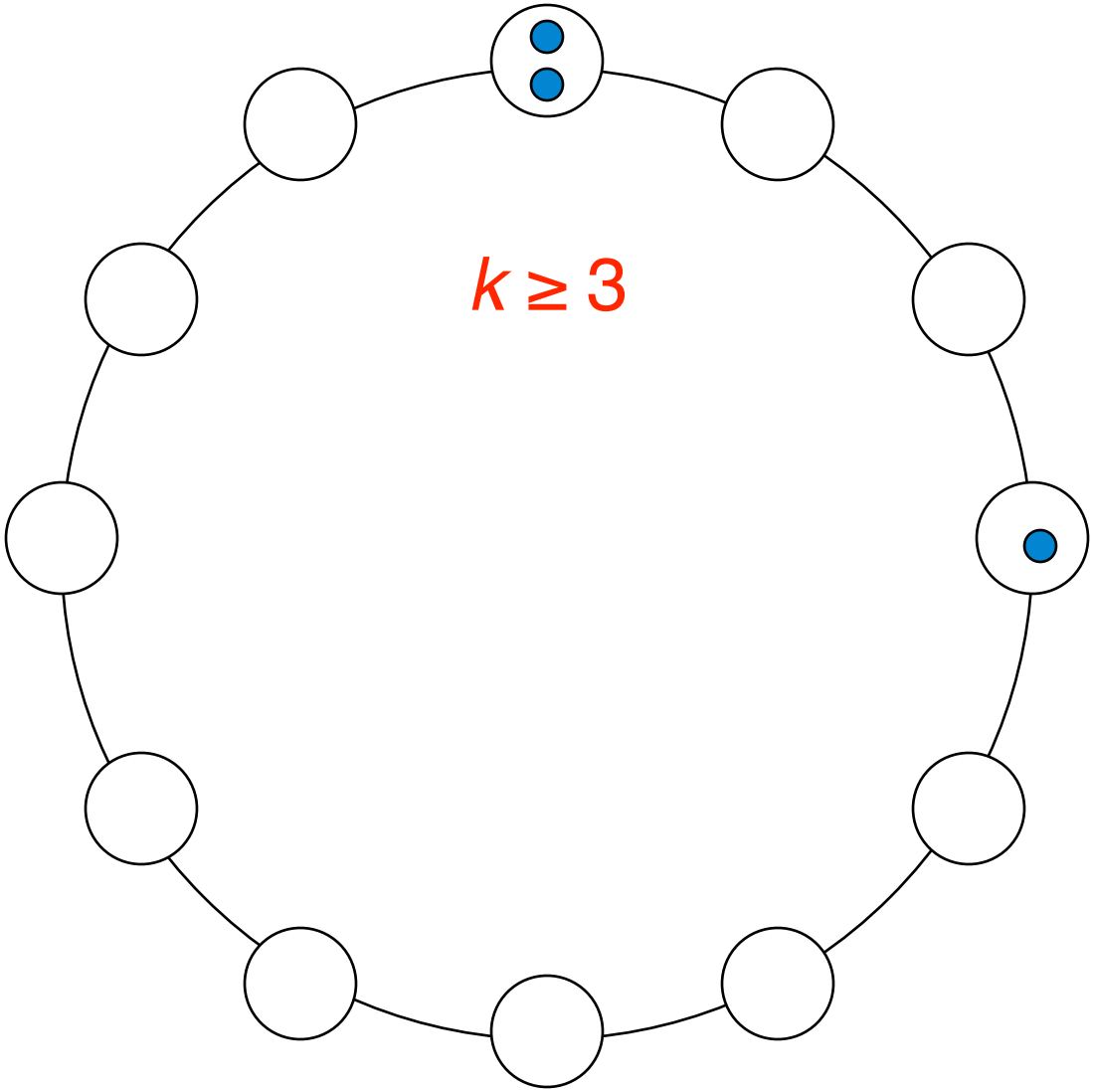
Enabling Exploration



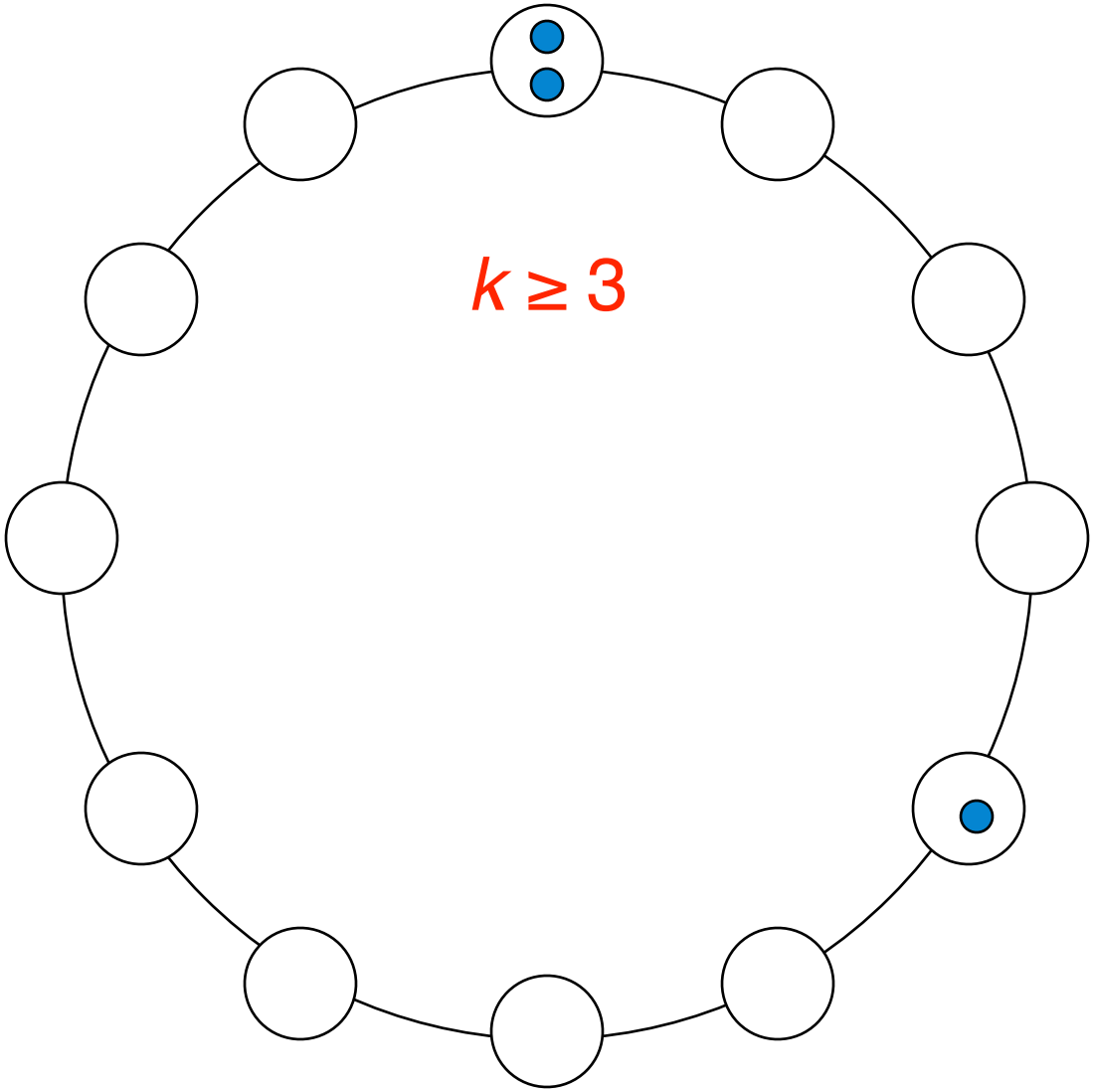
Enabling Exploration



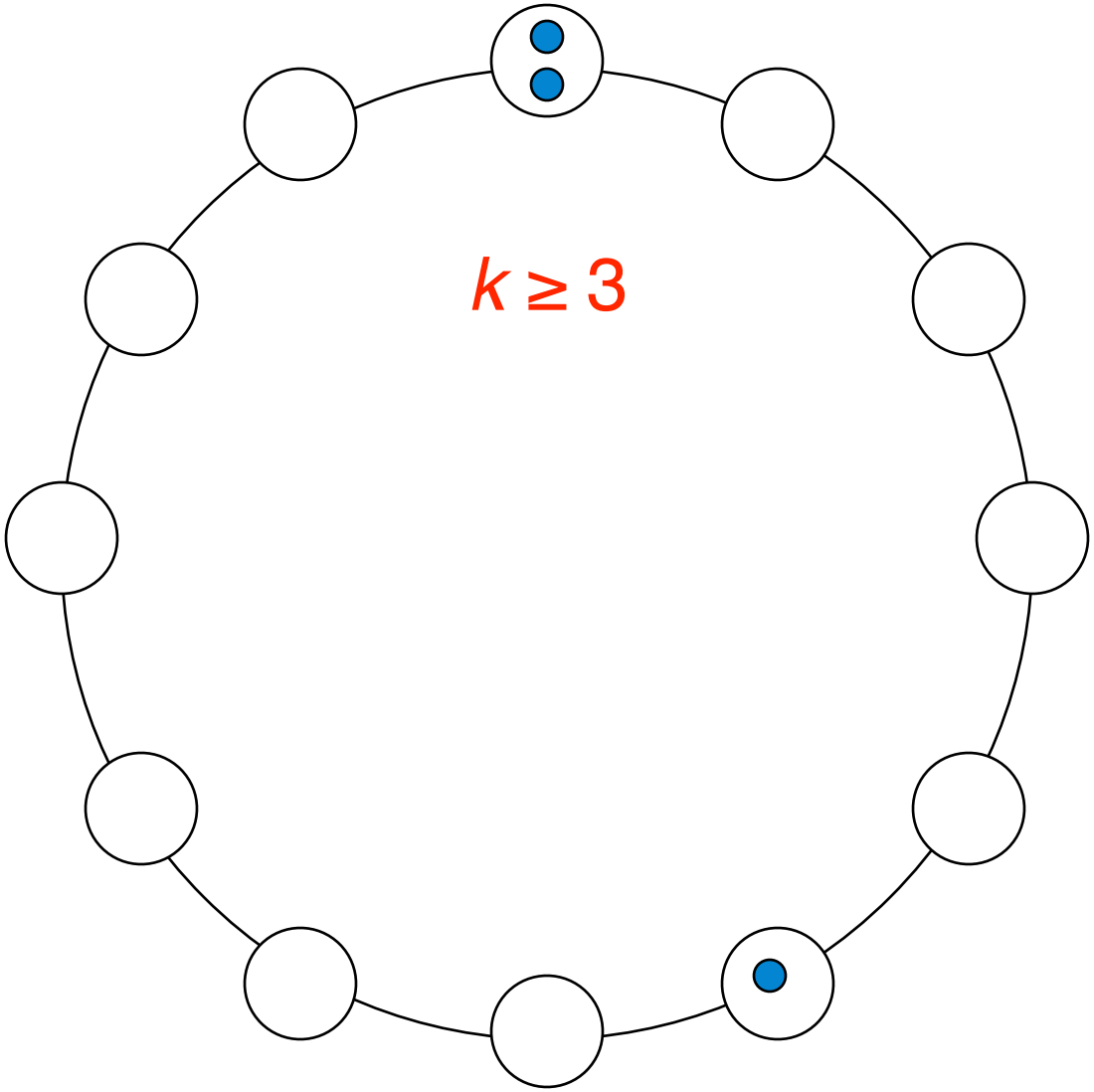
Enabling Exploration



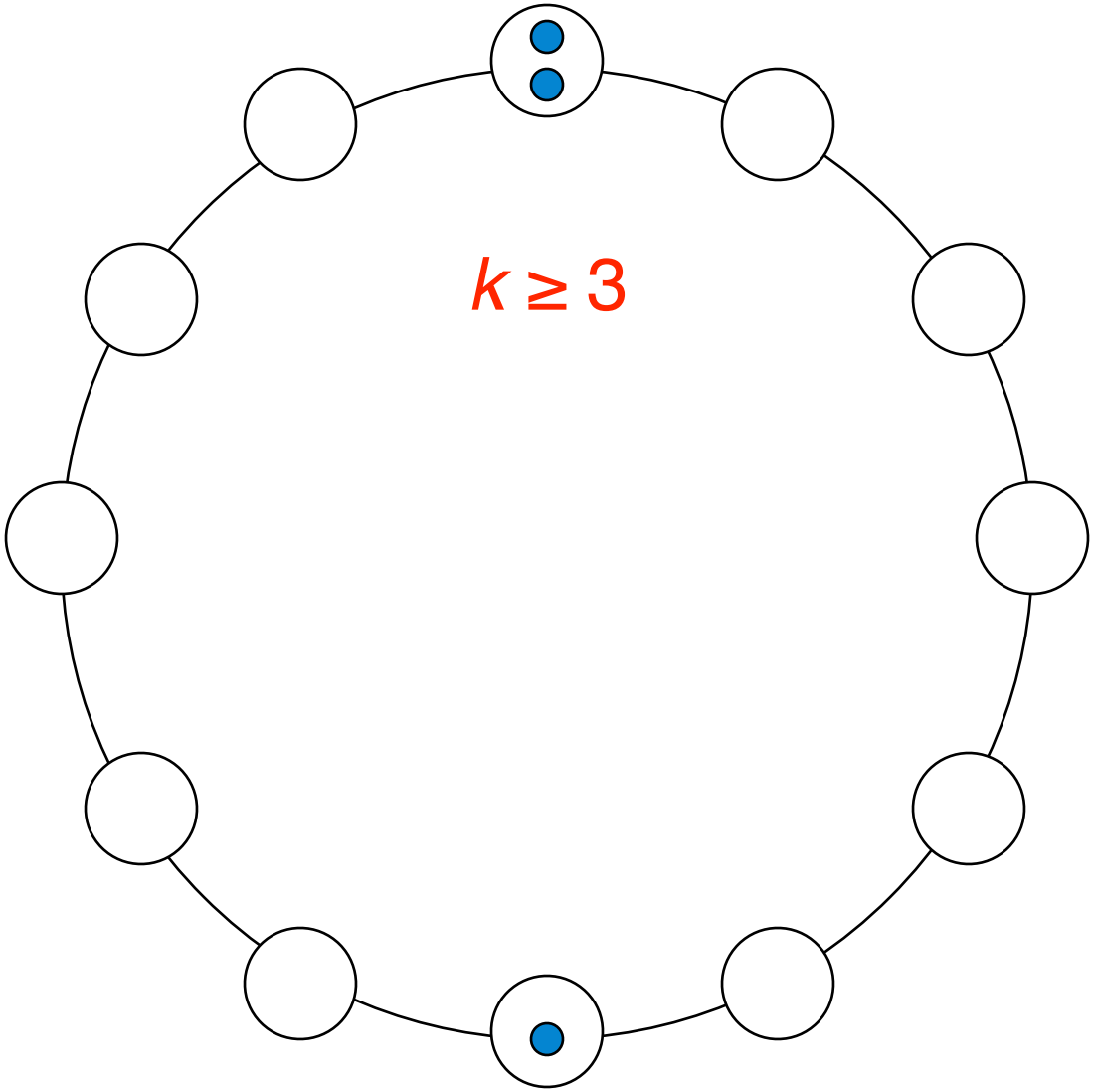
Enabling Exploration



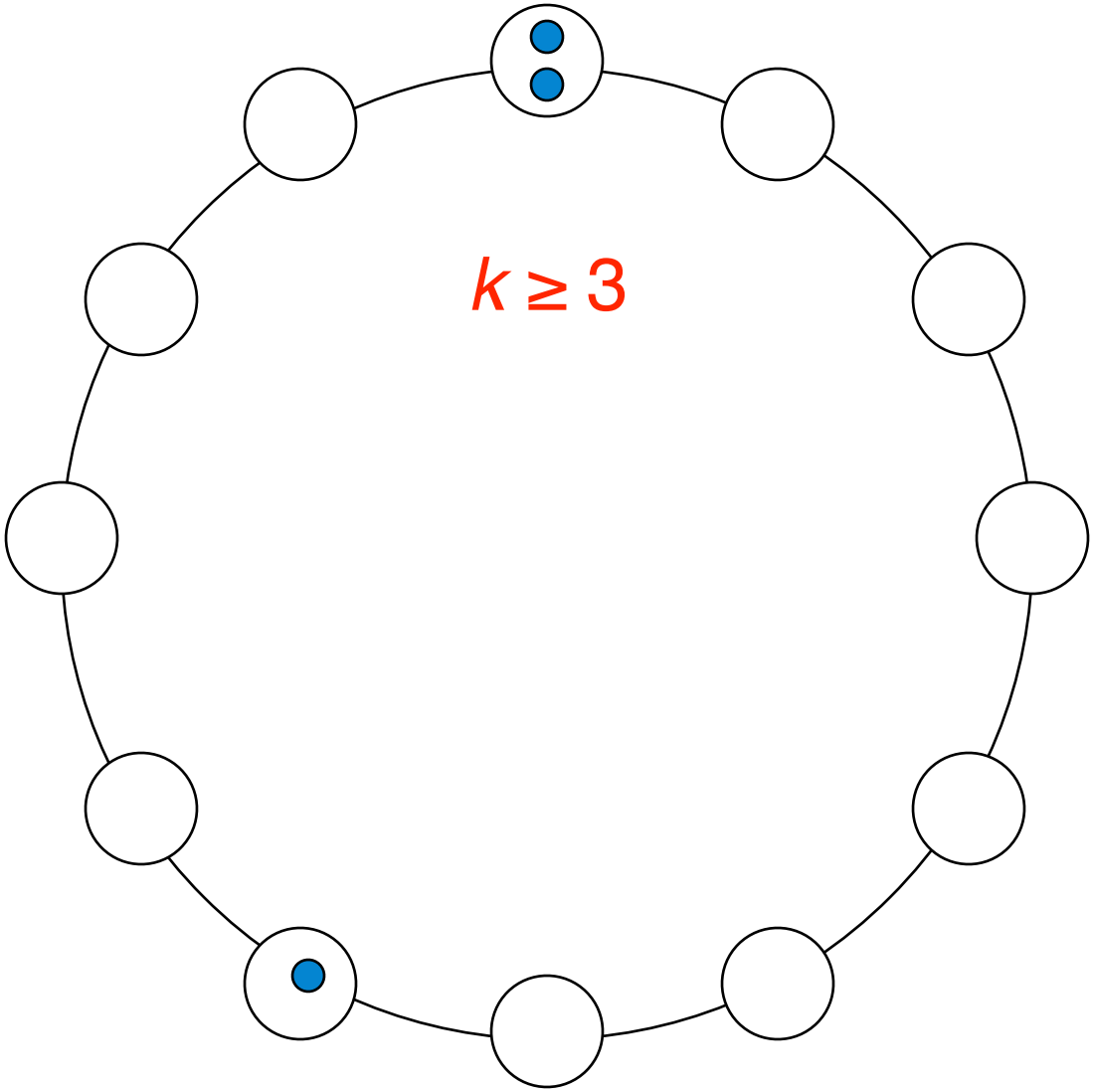
Enabling Exploration



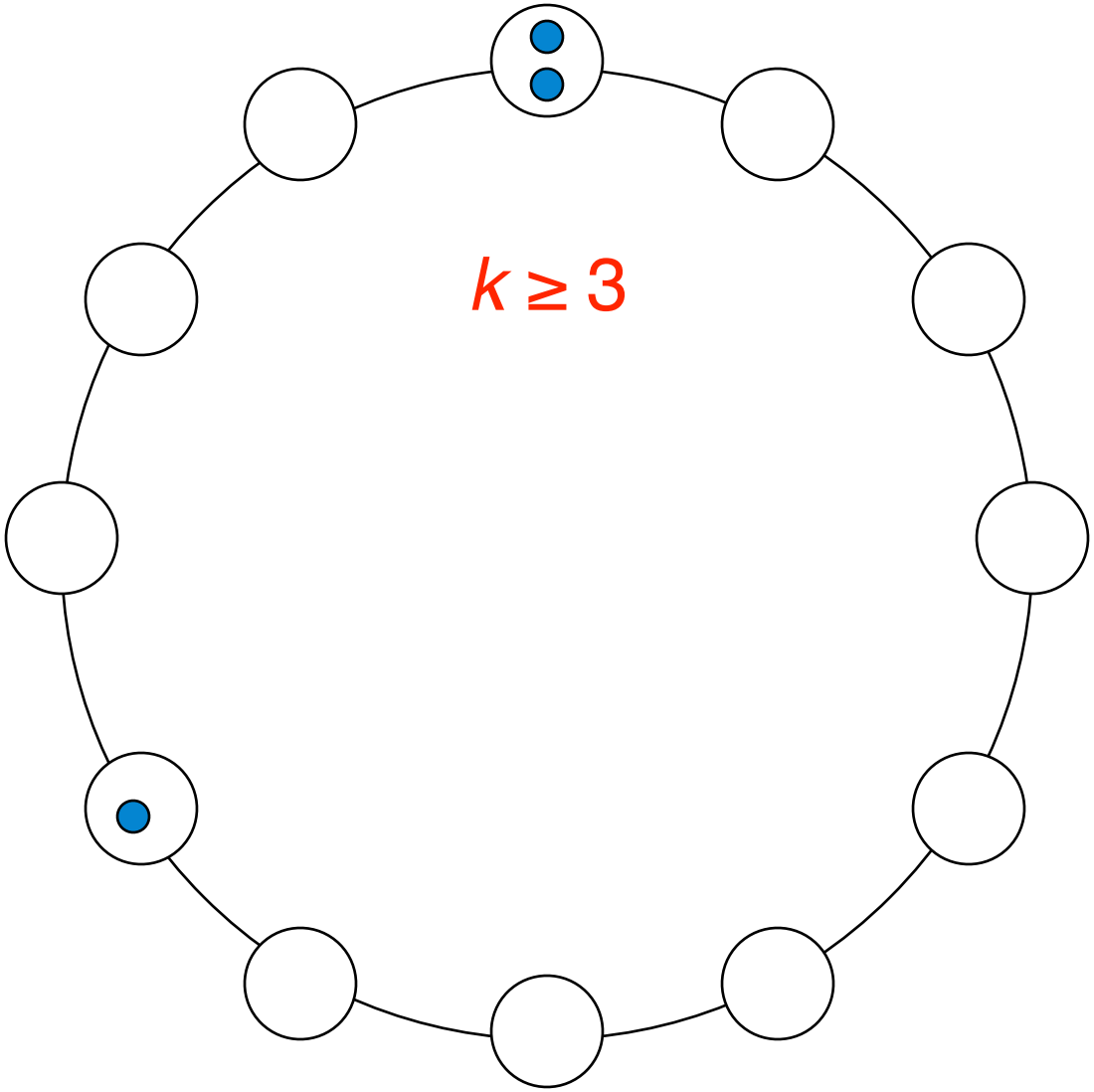
Enabling Exploration



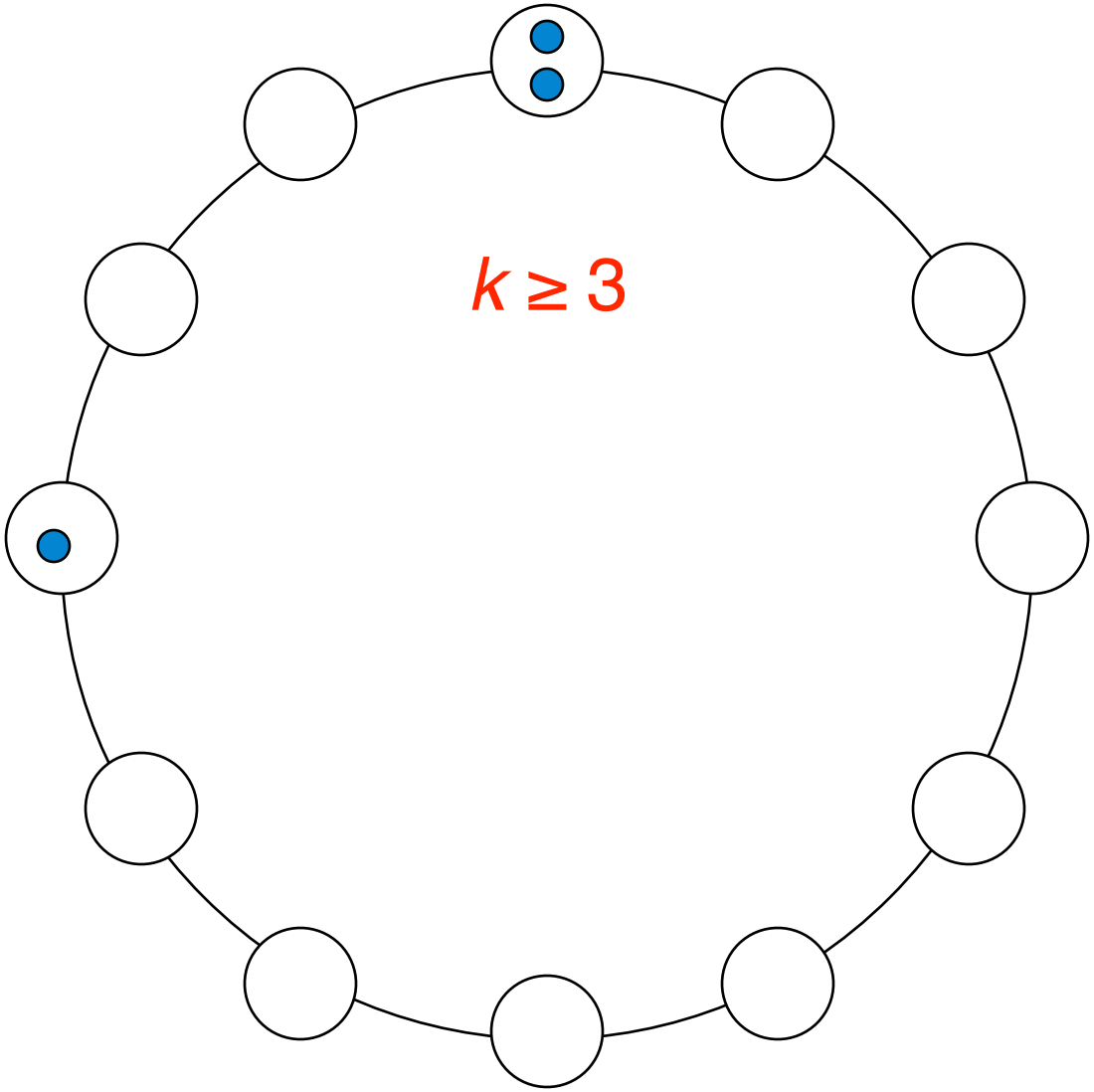
Enabling Exploration



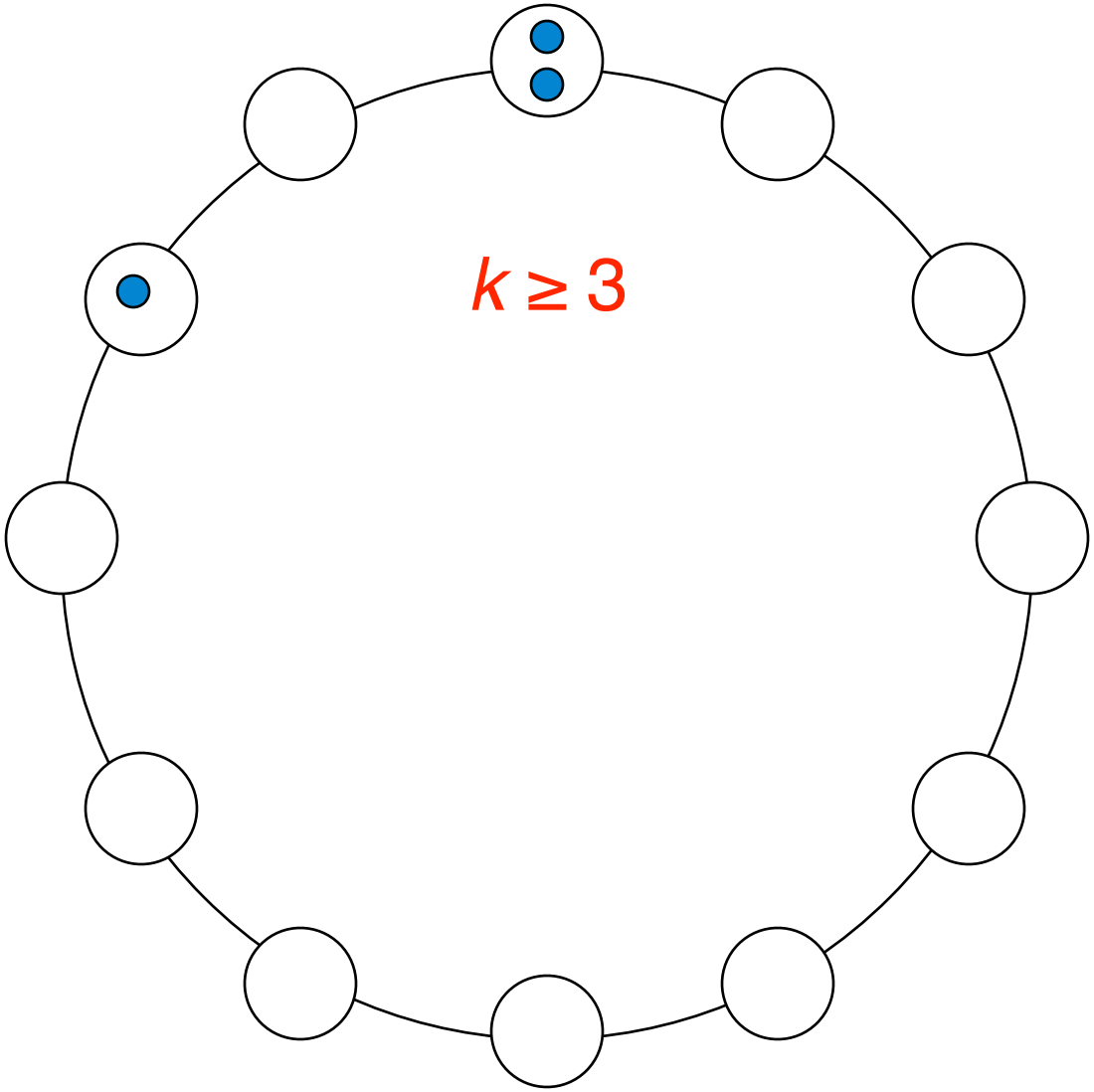
Enabling Exploration



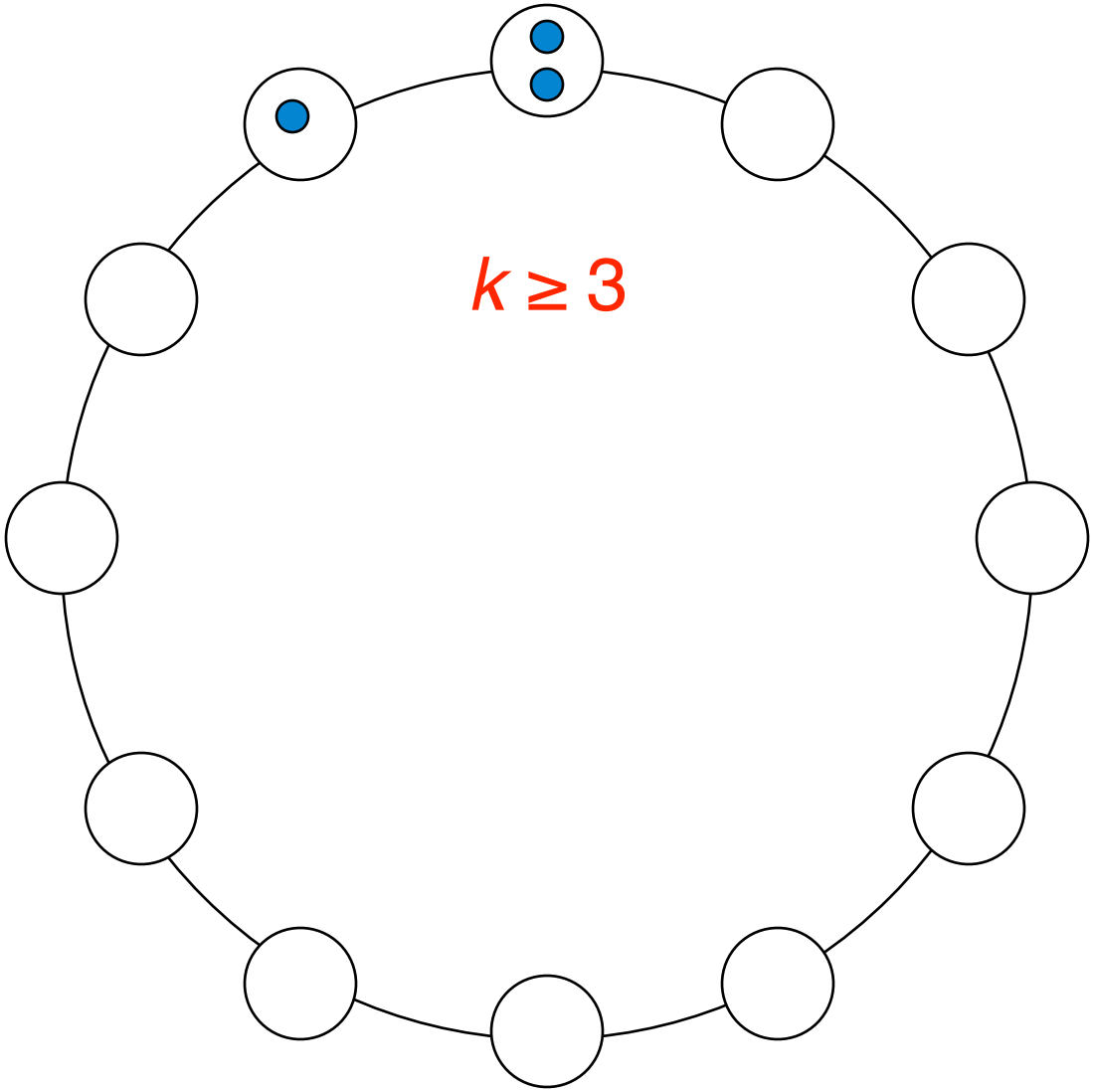
Enabling Exploration



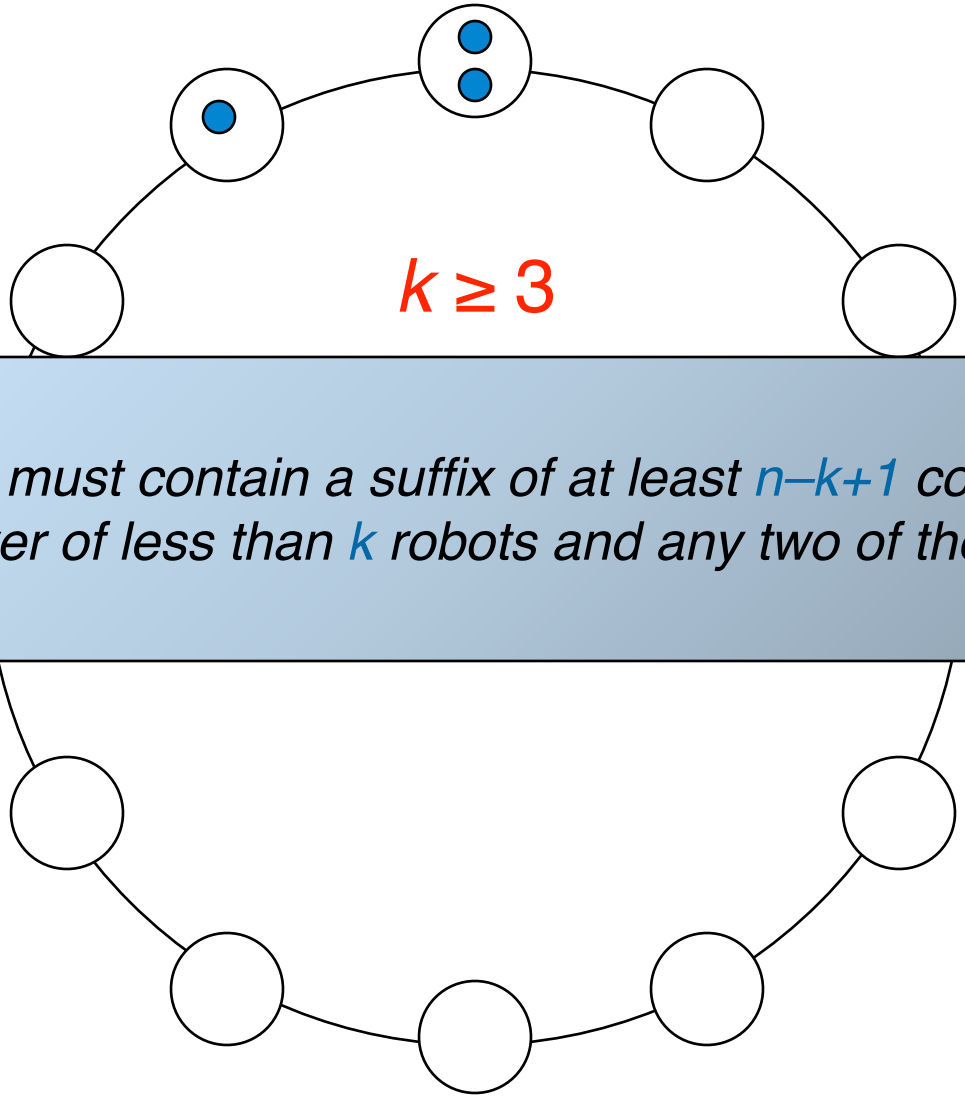
Enabling Exploration



Enabling Exploration

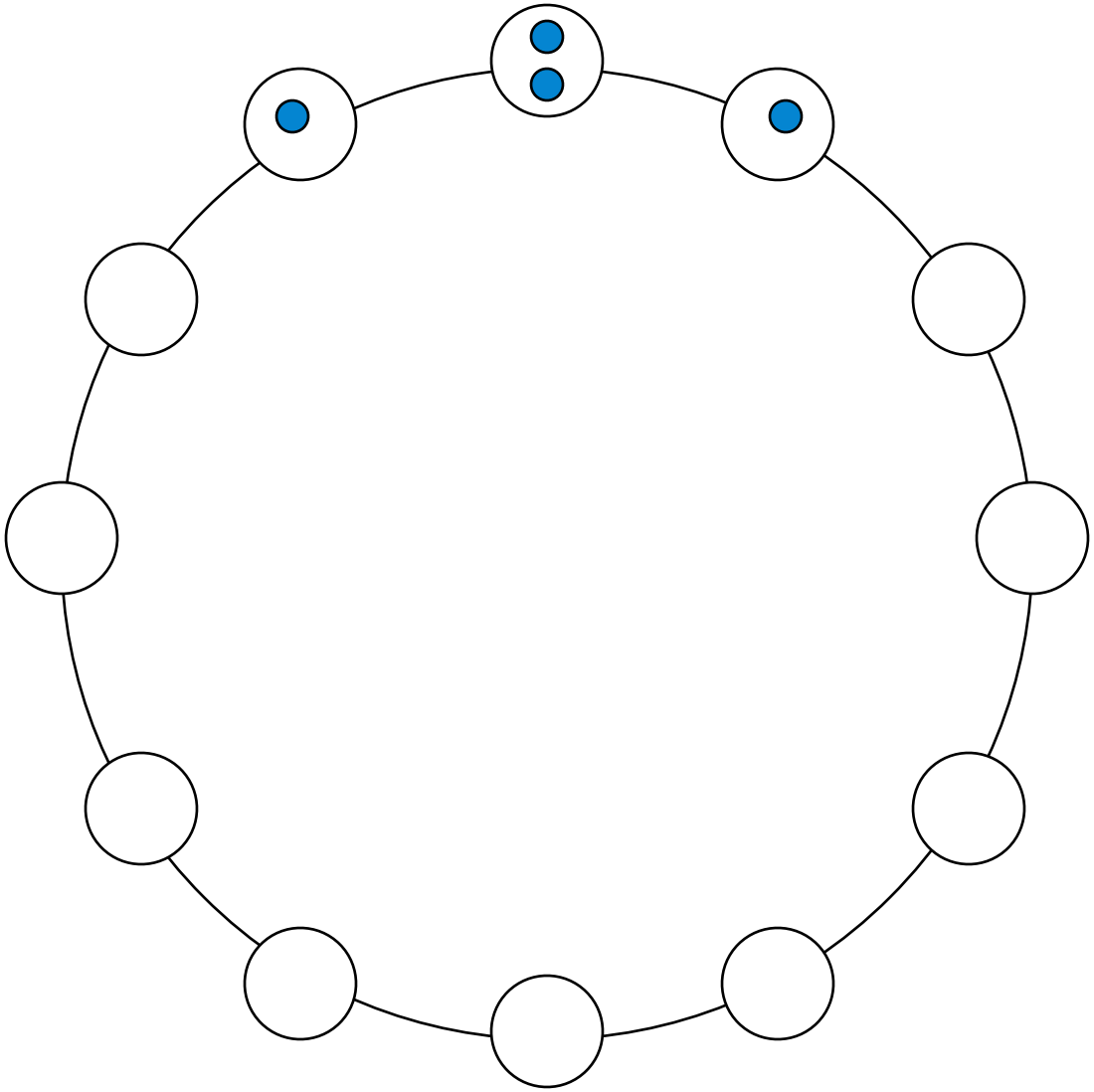


Enabling Exploration

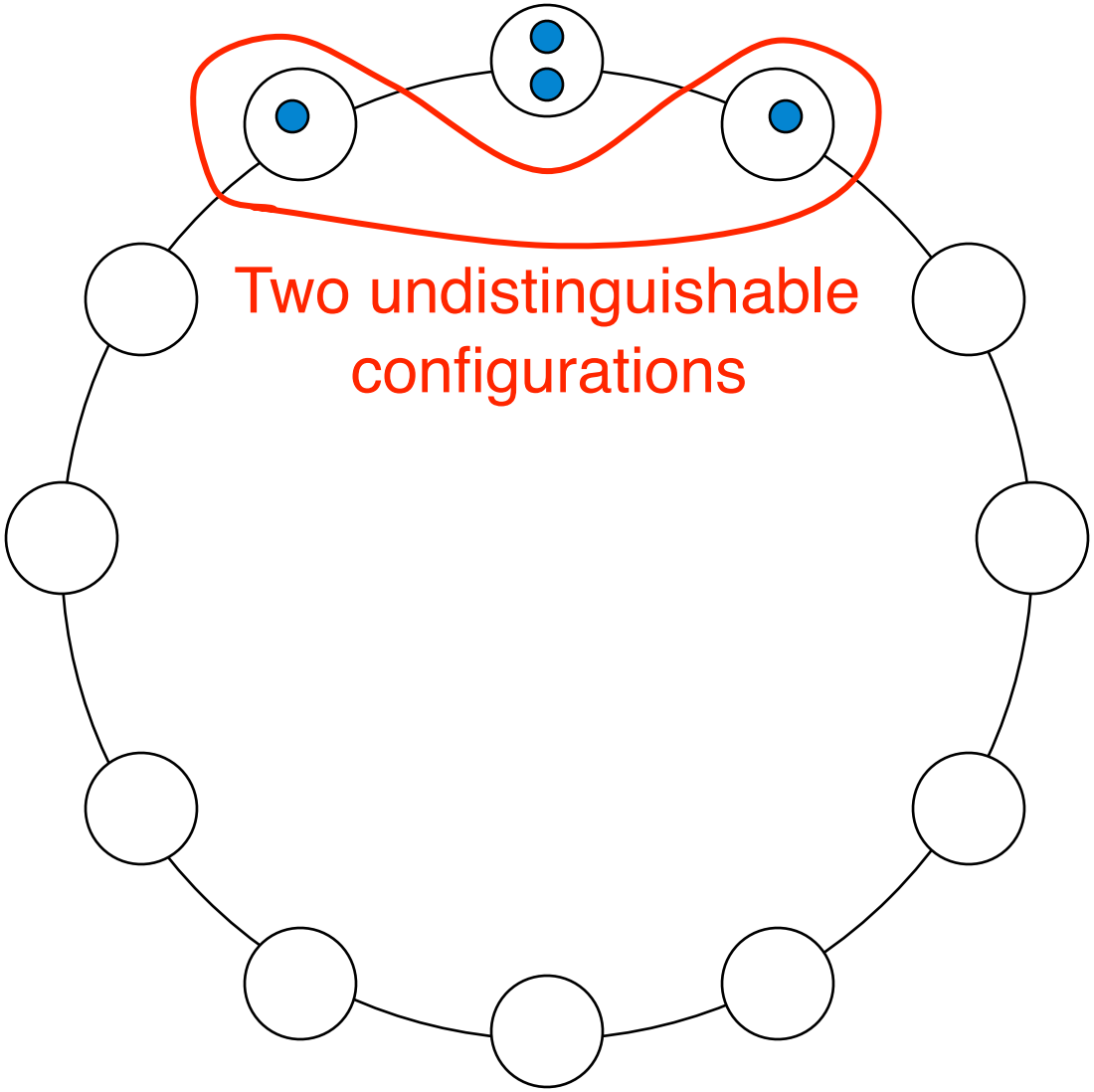


Lemma.
Every execution must contain a suffix of at least $n-k+1$ configurations containing a tower of less than k robots and any two of them are distinguishable.

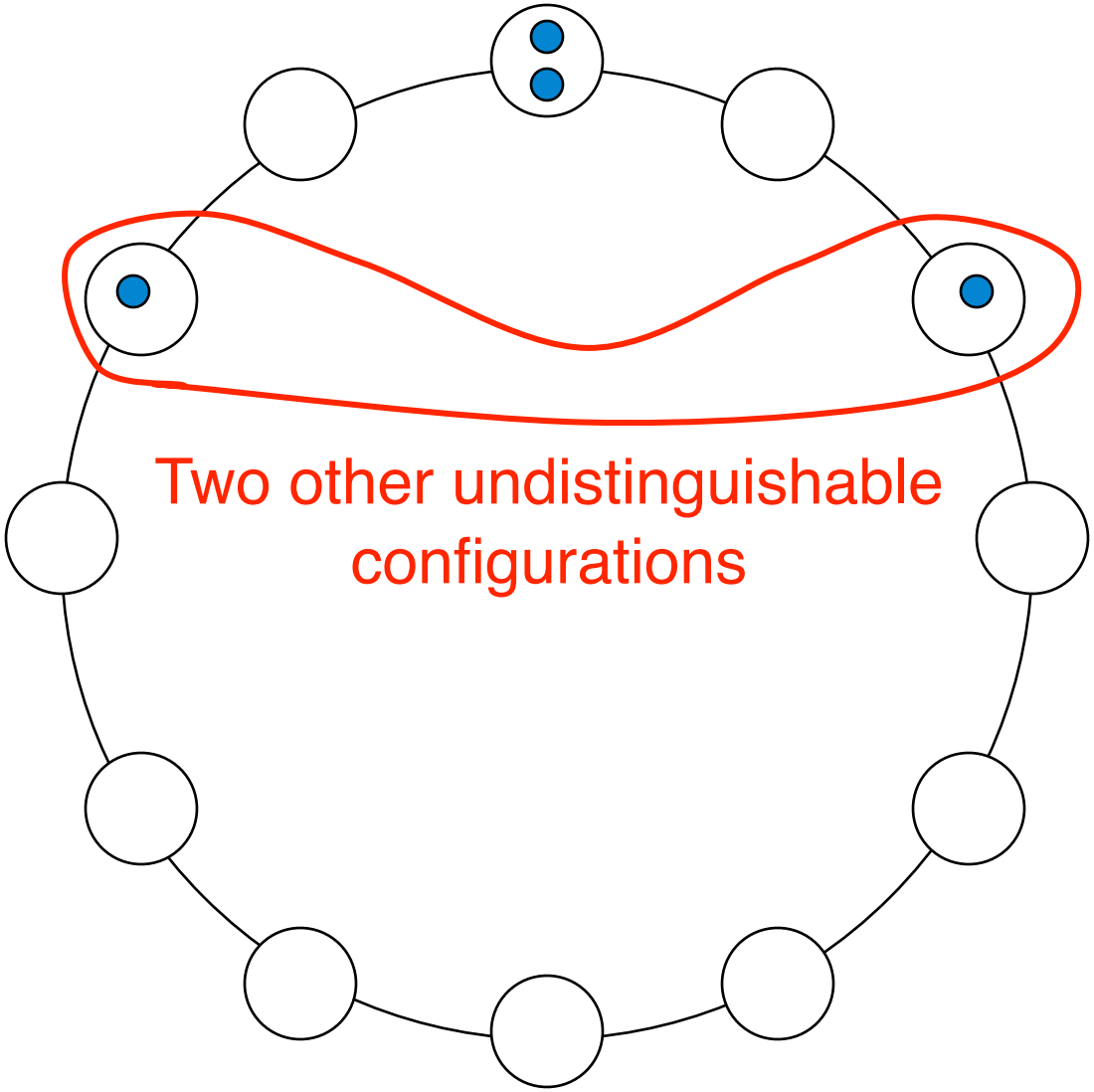
Enabling Exploration



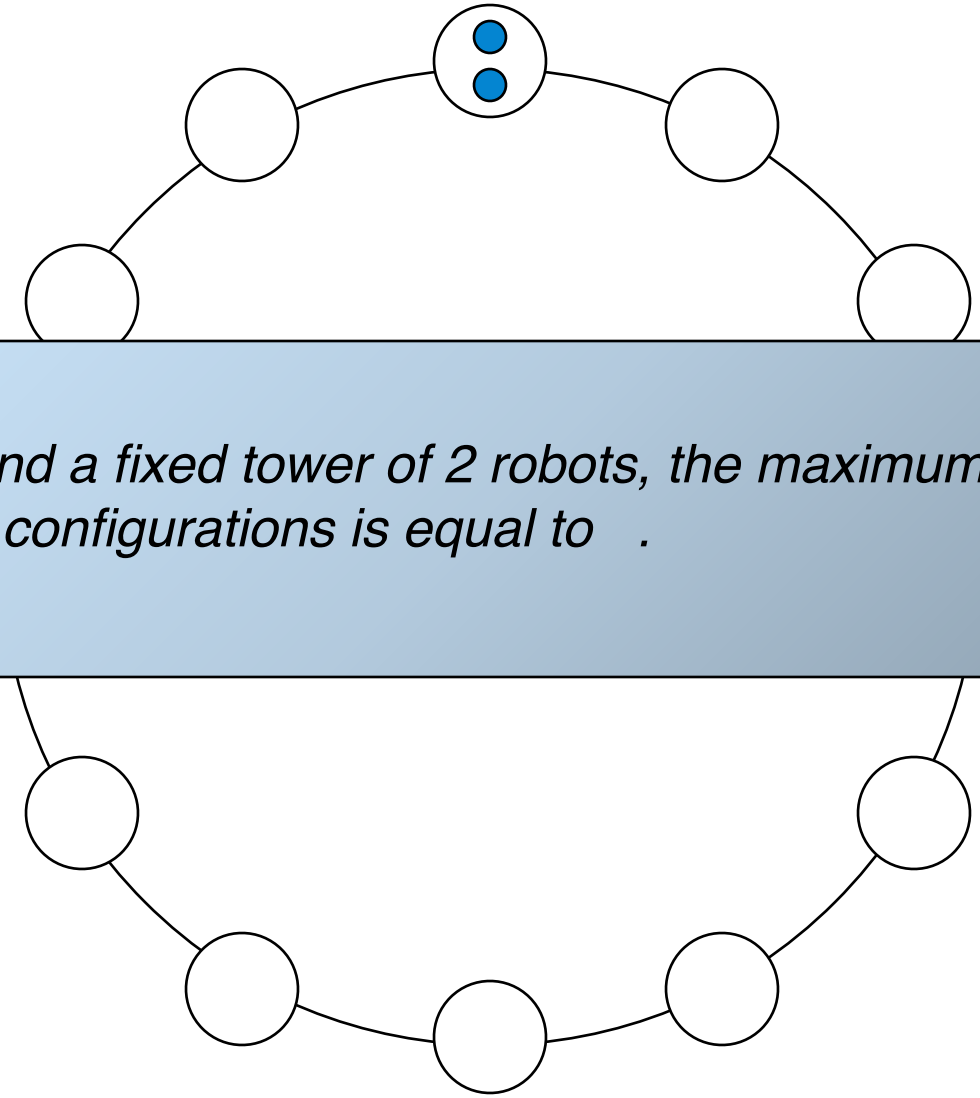
Enabling Exploration



Enabling Exploration



Enabling Exploration



Lemma.
With 3 robots and a fixed tower of 2 robots, the maximum number of distinguishable configurations is equal to $\left\lfloor \frac{n}{2} \right\rfloor$.

Enabling Exploration

Enabling Exploration

Theorem.
For every $n > 4$, there exists no exploration protocol (even probabilistic) of an n-size ring with 3 robots.

Enabling Exploration

Theorem.
For every $n > 4$, there exists no exploration protocol (even probabilistic) of an n-size ring with 3 robots.

Proof :

$$\left\lfloor \frac{n}{2} \right\rfloor \geq n - k + 1 \Rightarrow n \leq 4$$

Contribution

Theorem.
4 **probabilistic** robots are necessary and sufficient, provided that $n > 4$

- The theorem holds even if k divides n.

1. Exploration impossible with less than *4* robots
2. Give an algorithm working with *4* probabilistic robots

Contribution

Theorem.
4 **probabilistic** robots are necessary and sufficient, provided that $n > 4$

- The theorem holds even if k divides n.

1. Exploration impossible with less than *4* robots
2. Give an algorithm working with *4* probabilistic robots

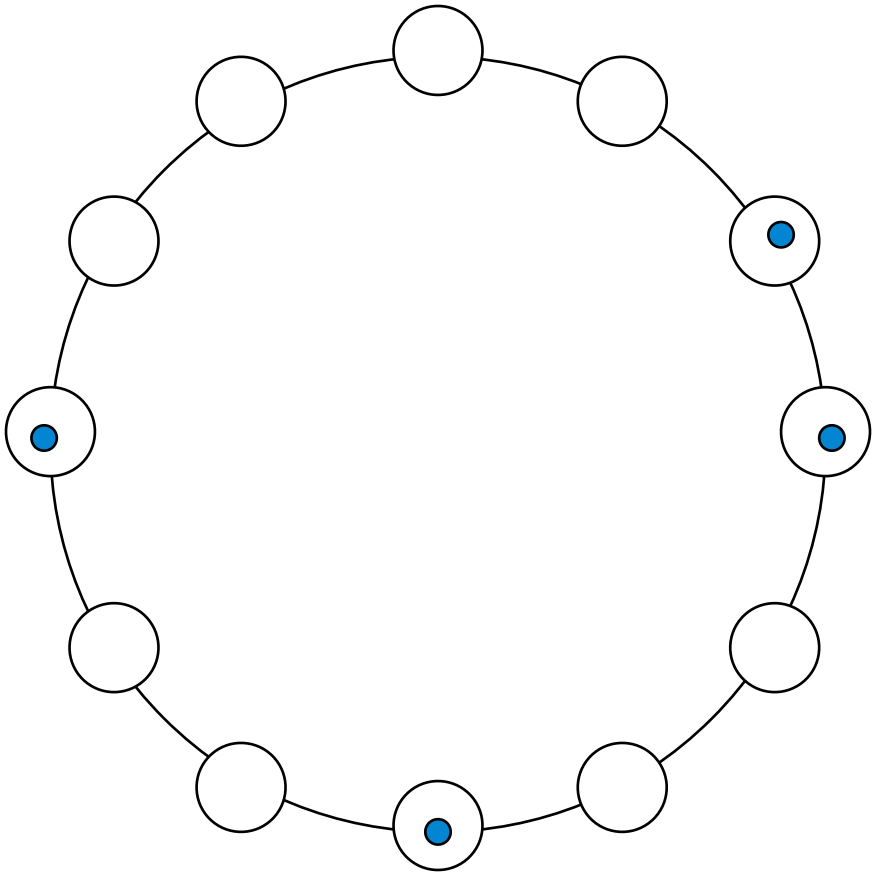
Definitions

Definitions

Segment.
A maximal non-empty elementary path of occupied nodes.

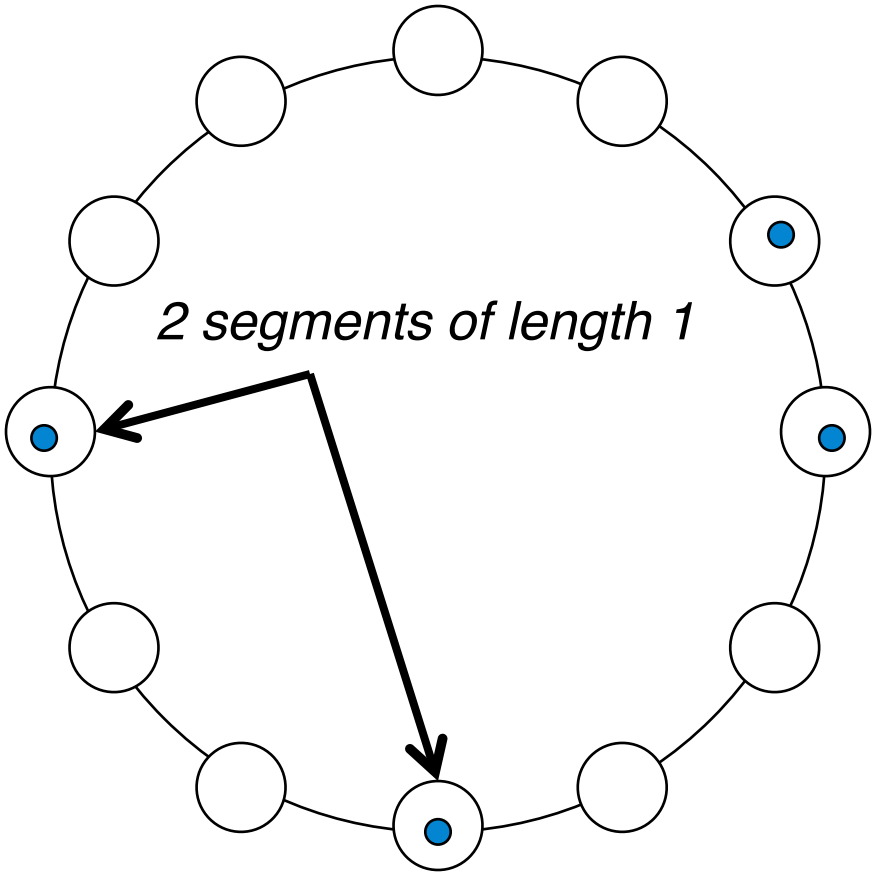
Definitions

Segment.
A maximal non-empty elementary path of occupied nodes.



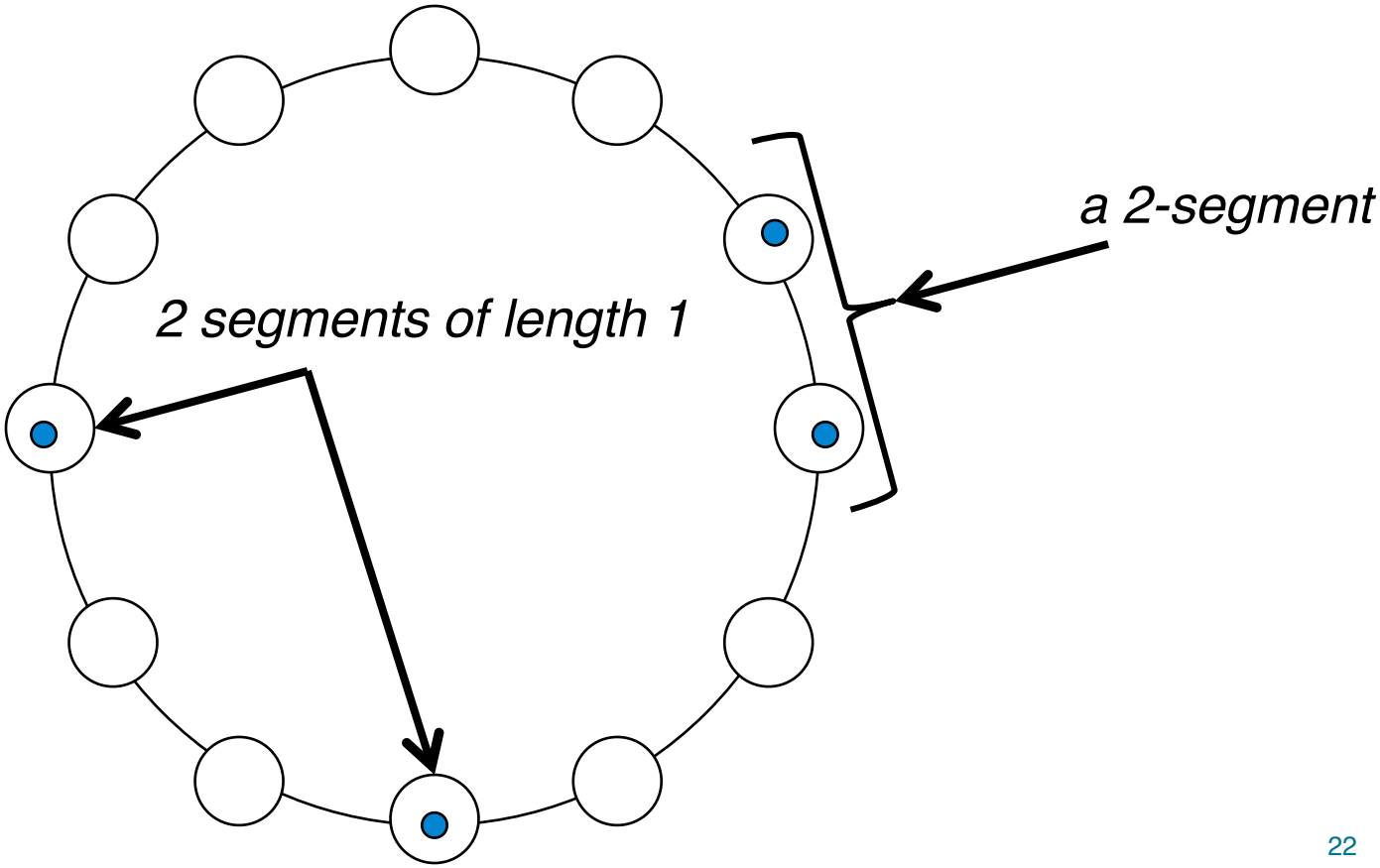
Definitions

Segment.
A maximal non-empty elementary path of occupied nodes.

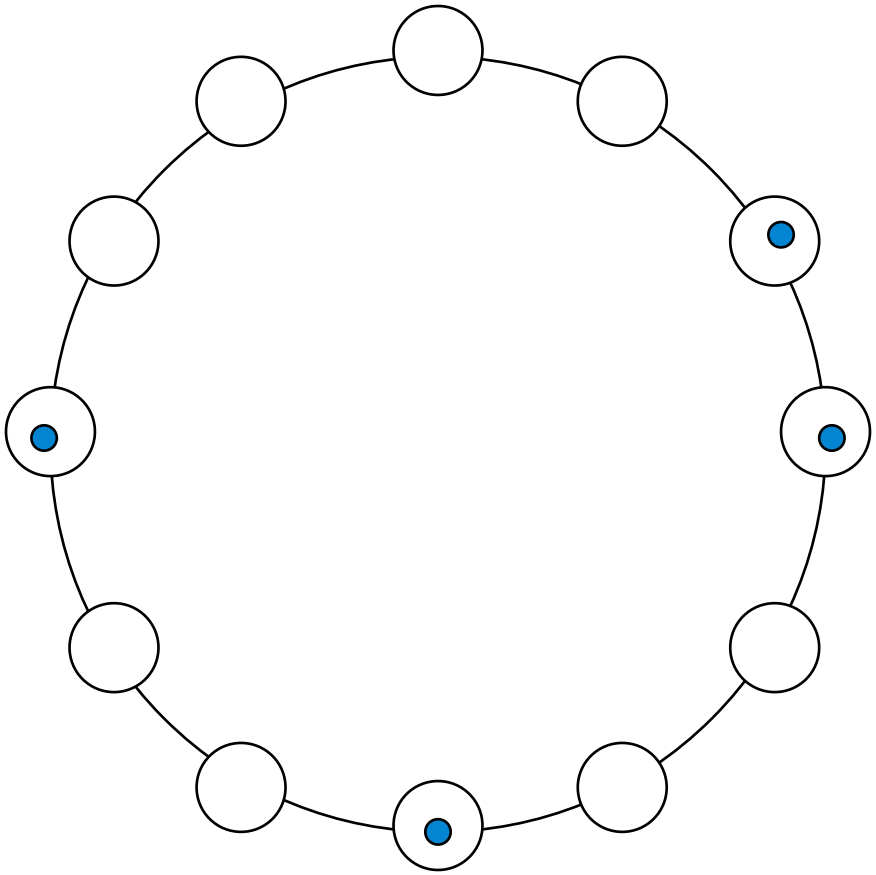


Definitions

Segment.
A maximal non-empty elementary path of occupied nodes.

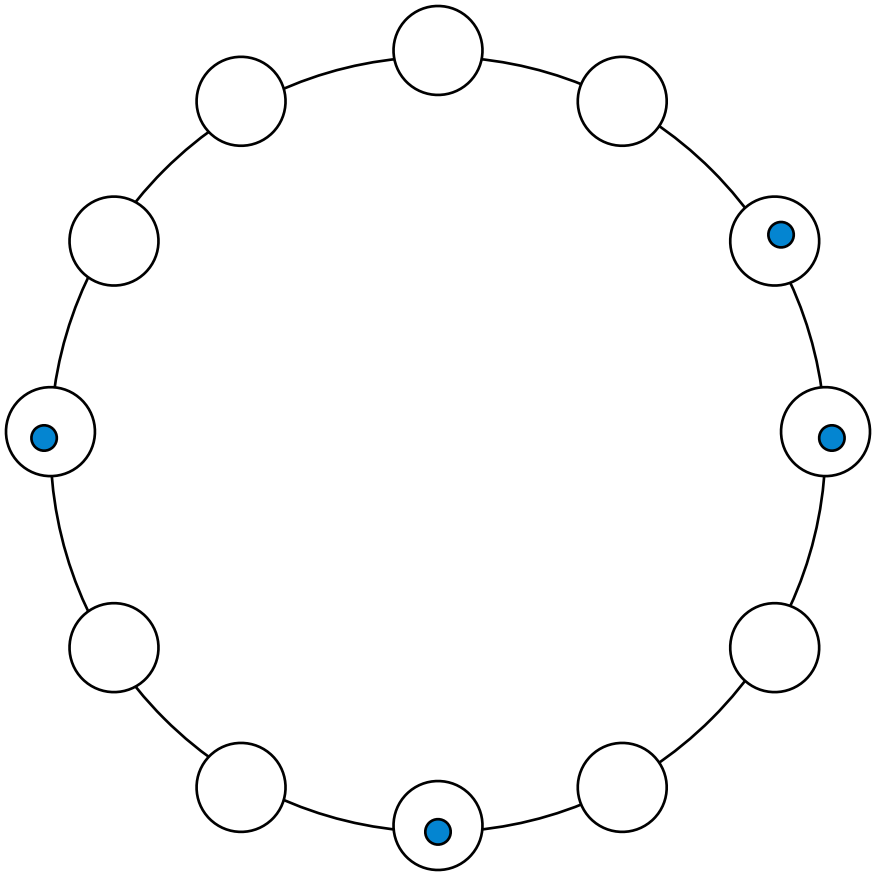


Definitions



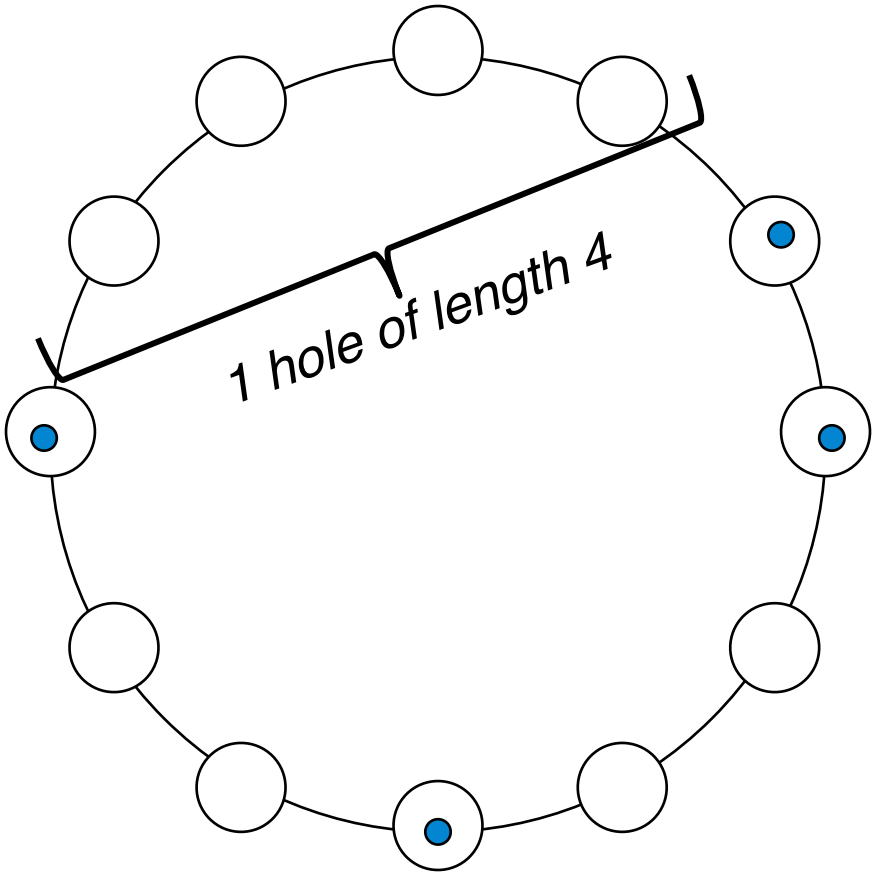
Definitions

Hole.
A maximal non-empty elementary path of free nodes.



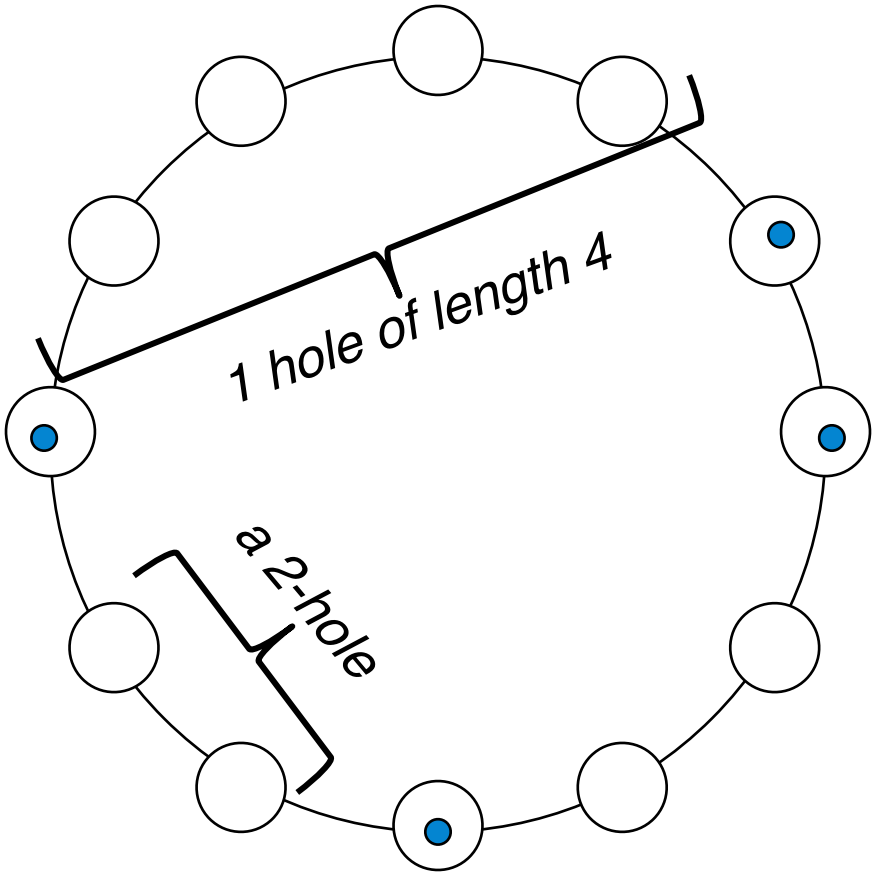
Definitions

Hole.
A maximal non-empty elementary path of free nodes.

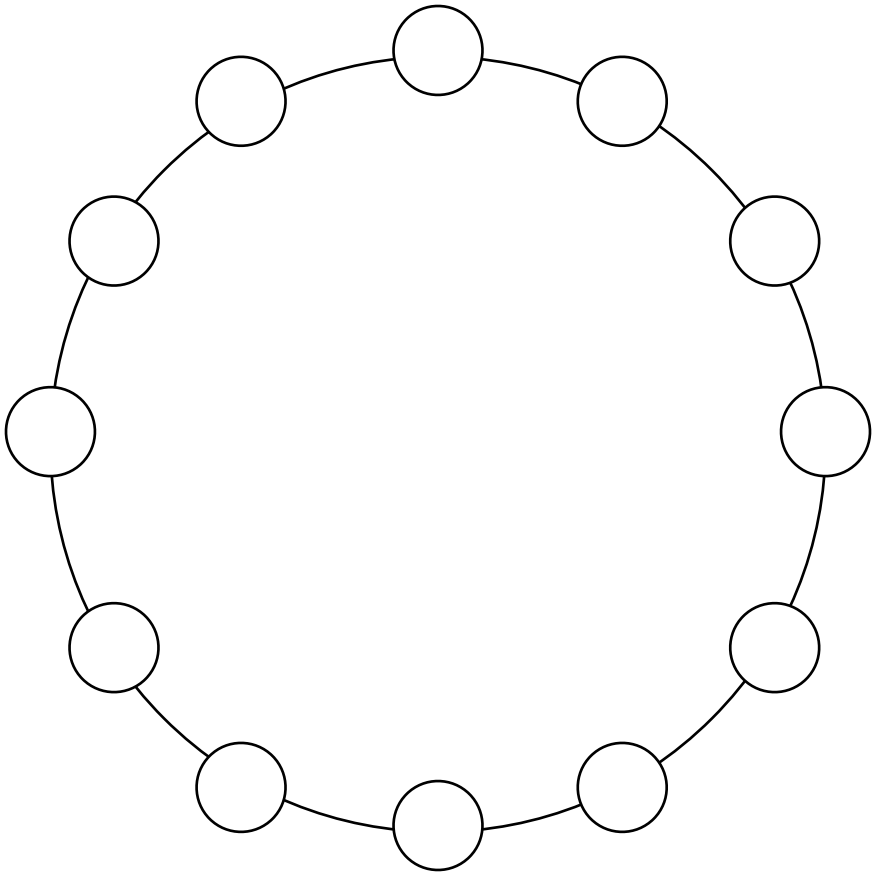


Definitions

Hole.
A maximal non-empty elementary path of free nodes.

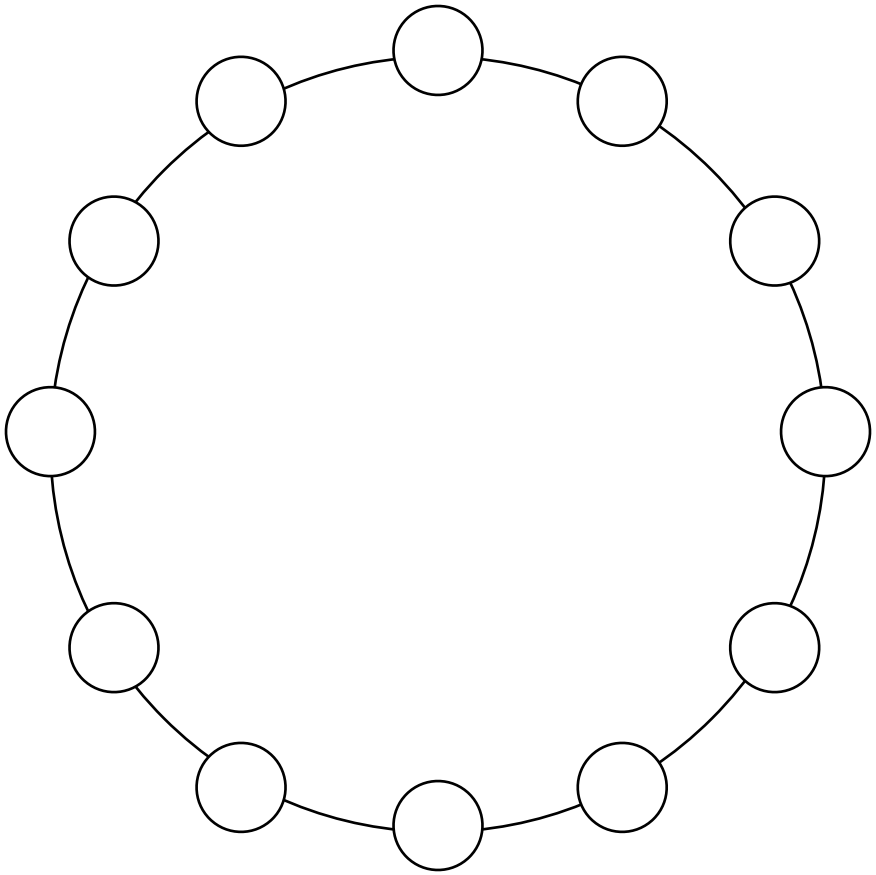


Definitions



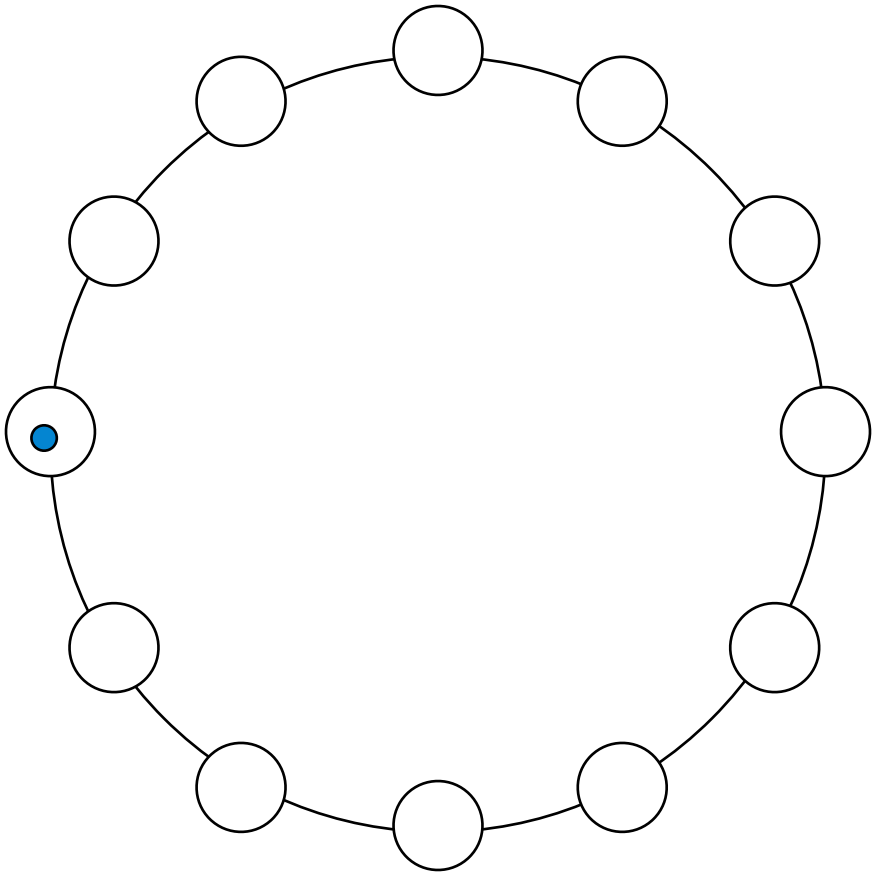
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



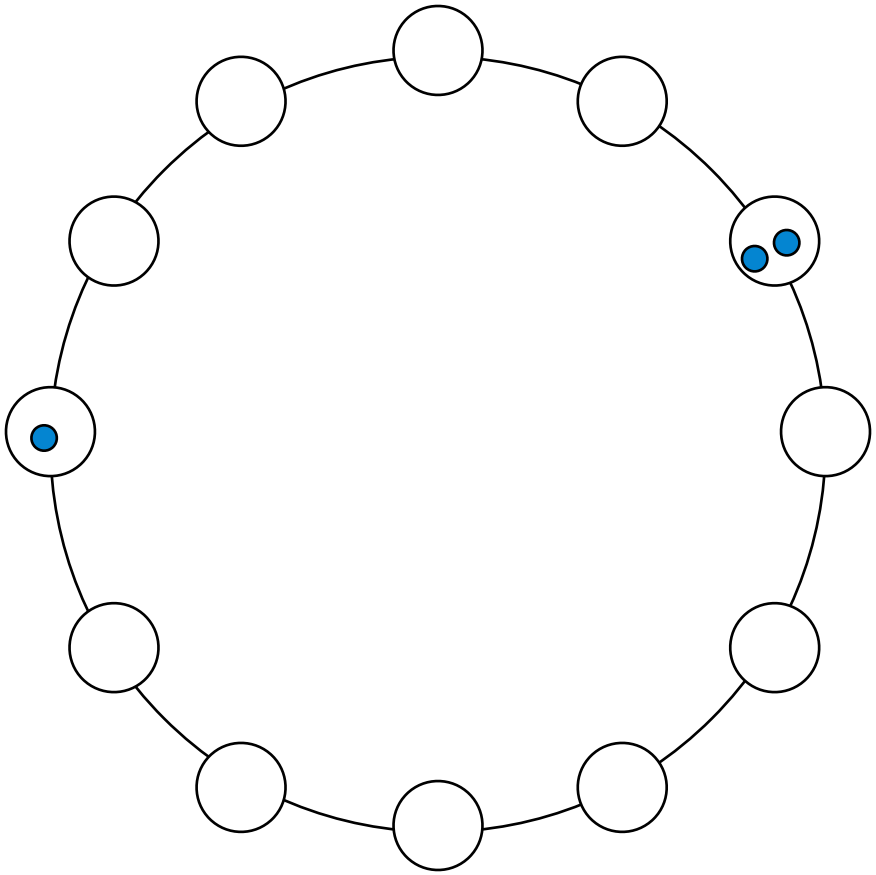
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



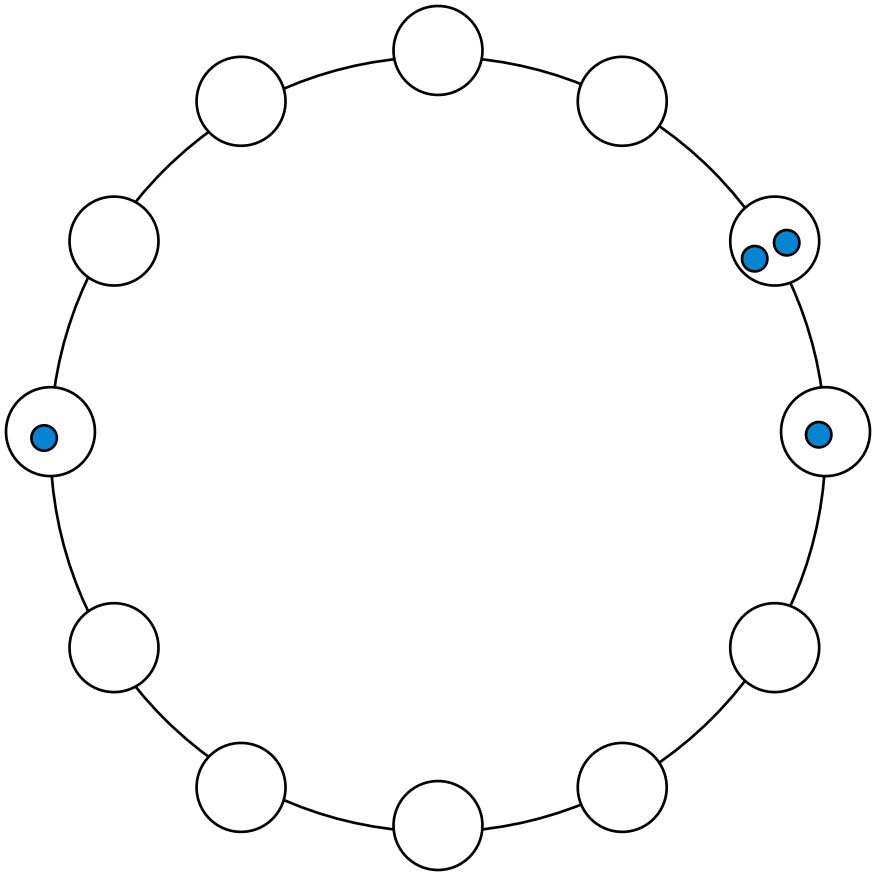
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



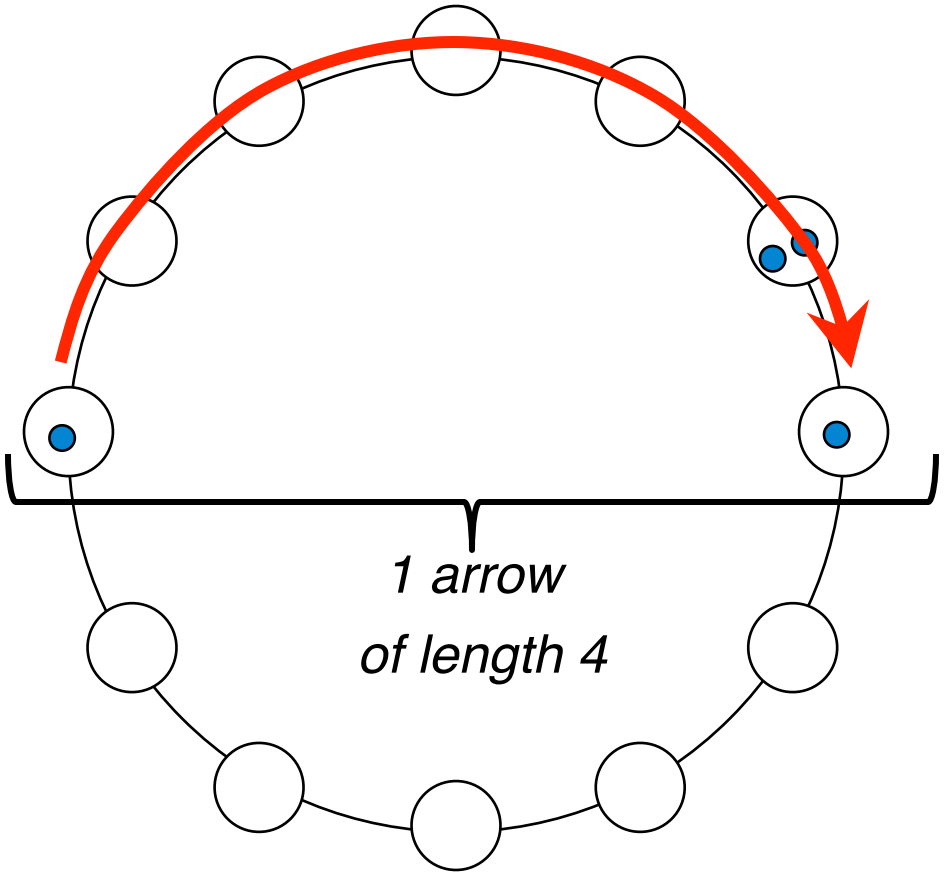
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



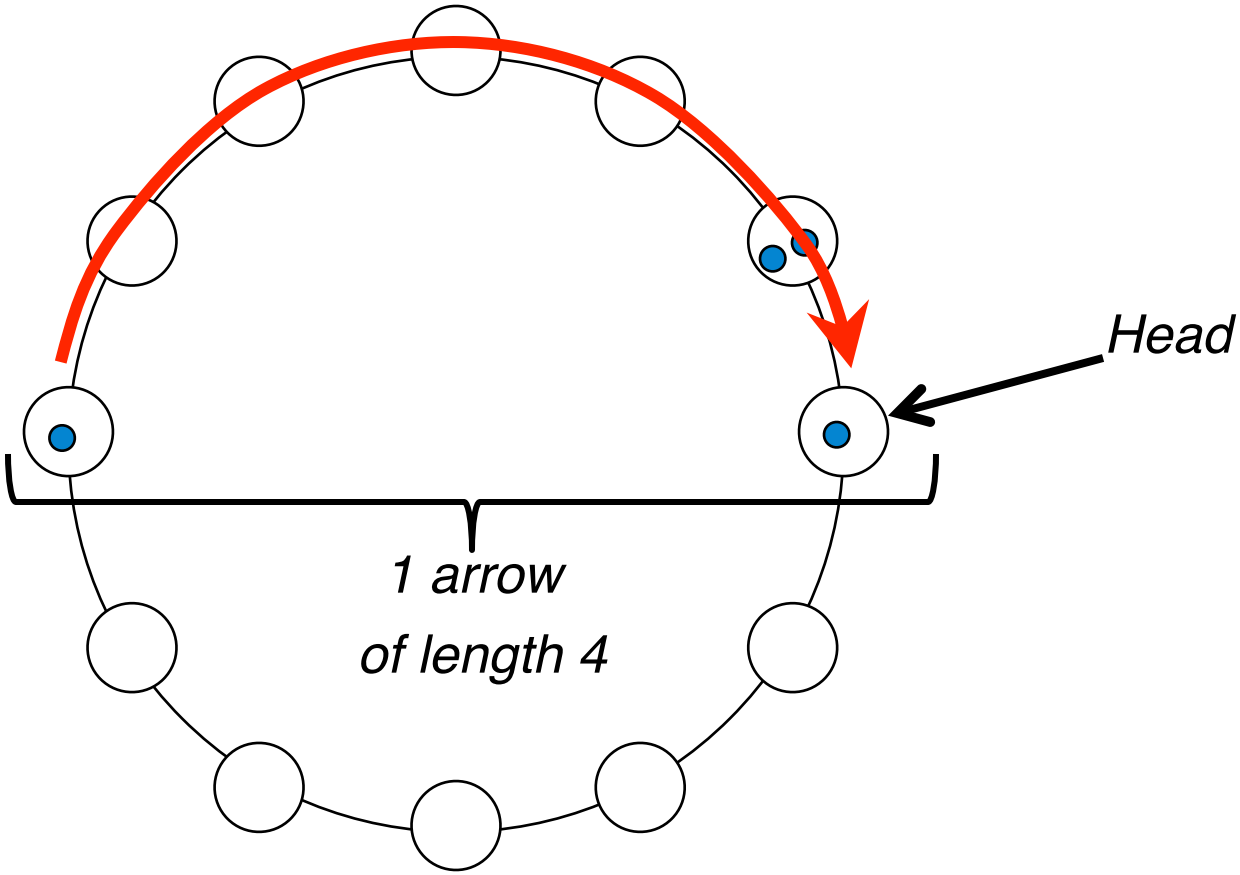
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



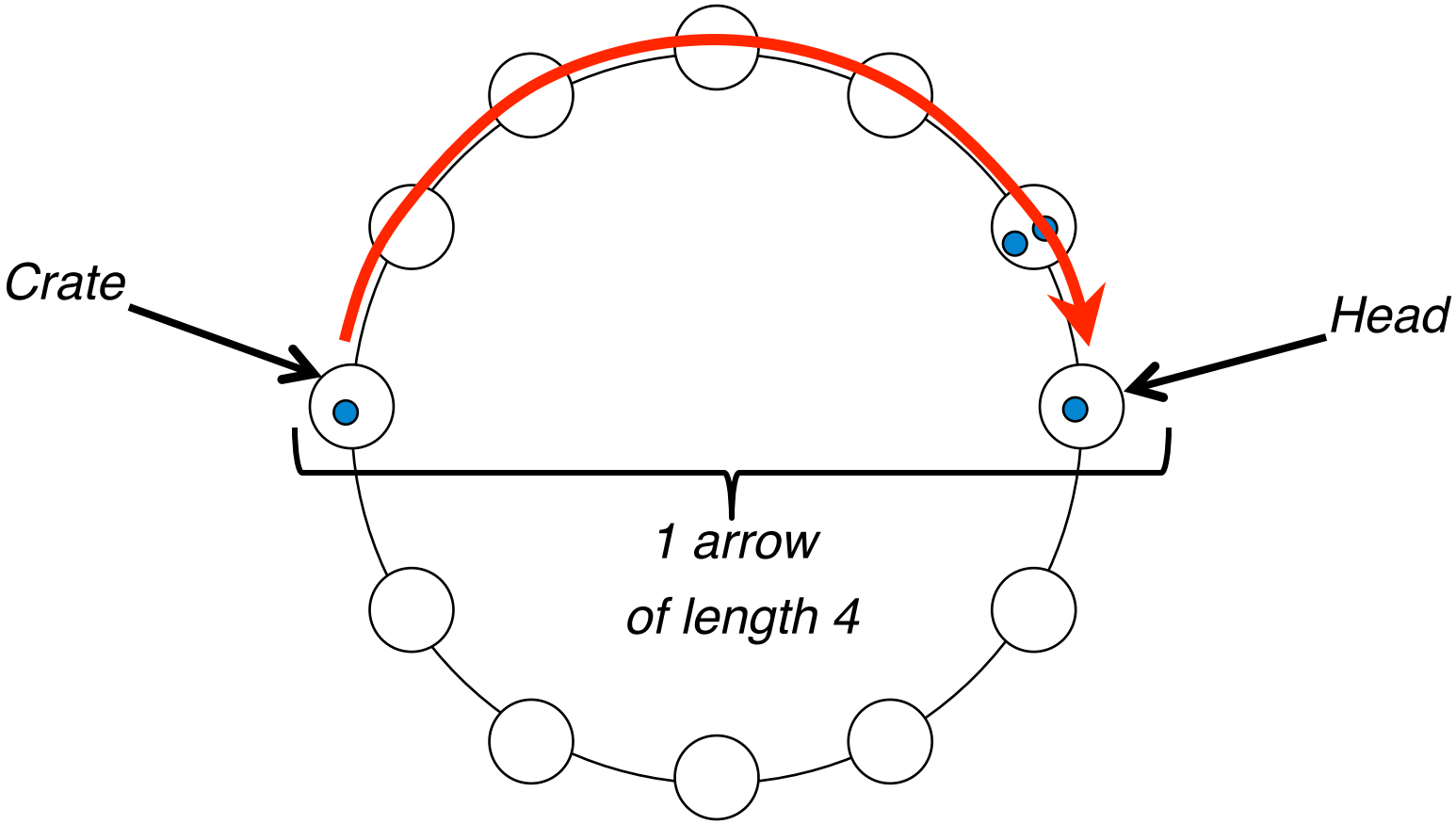
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



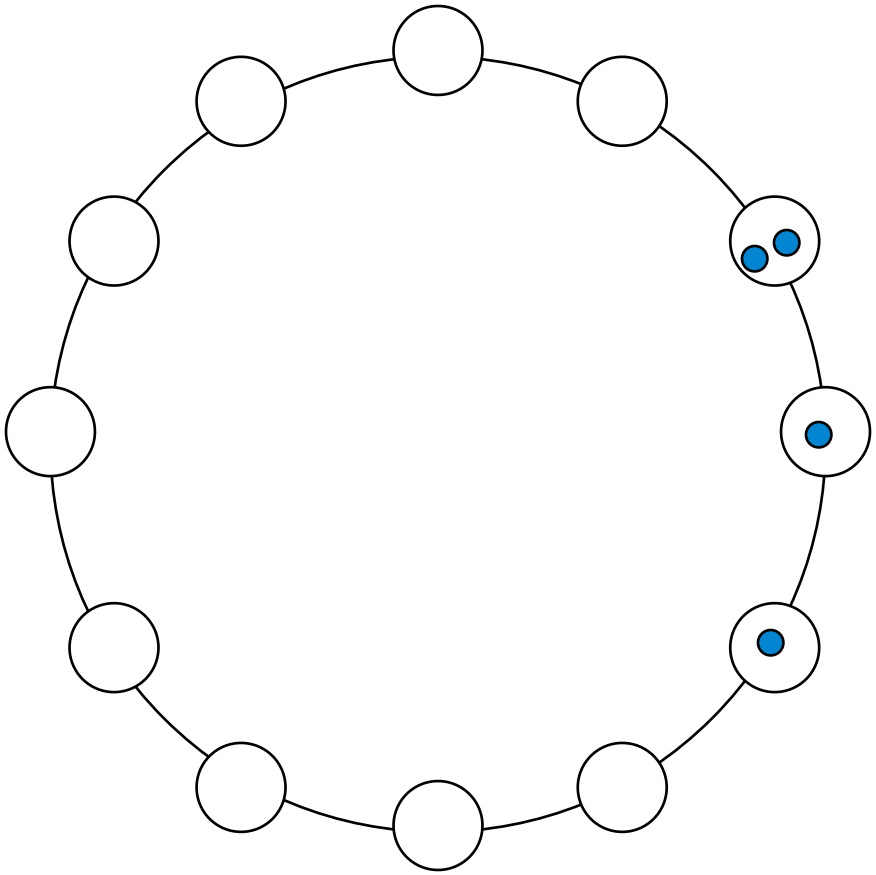
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



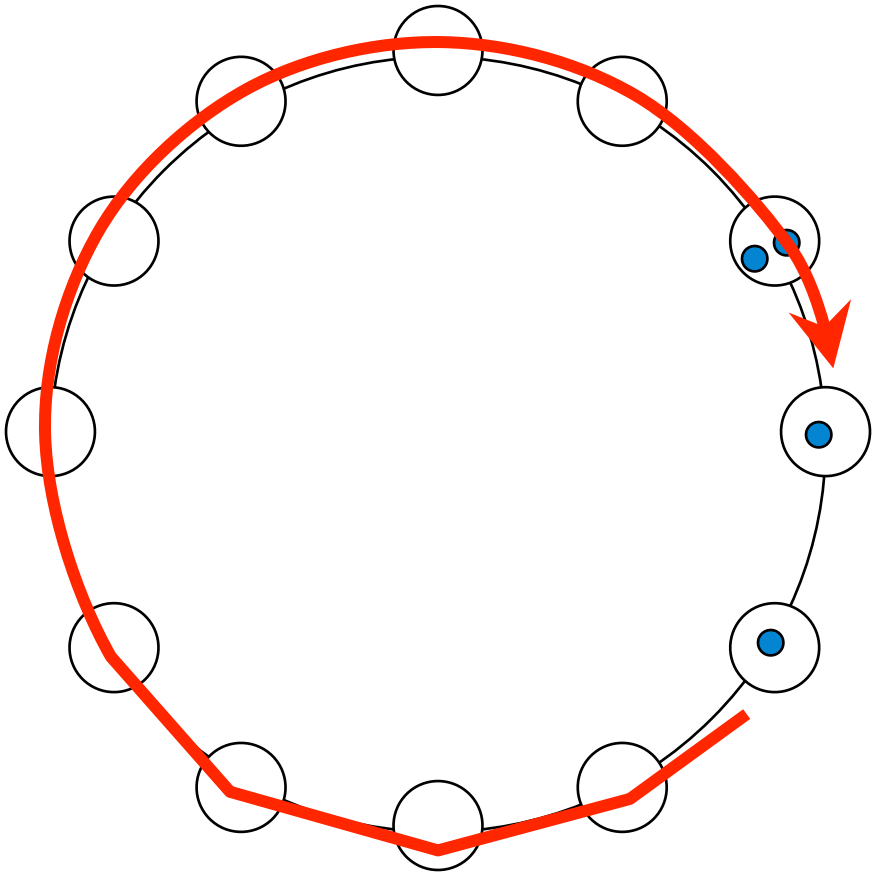
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



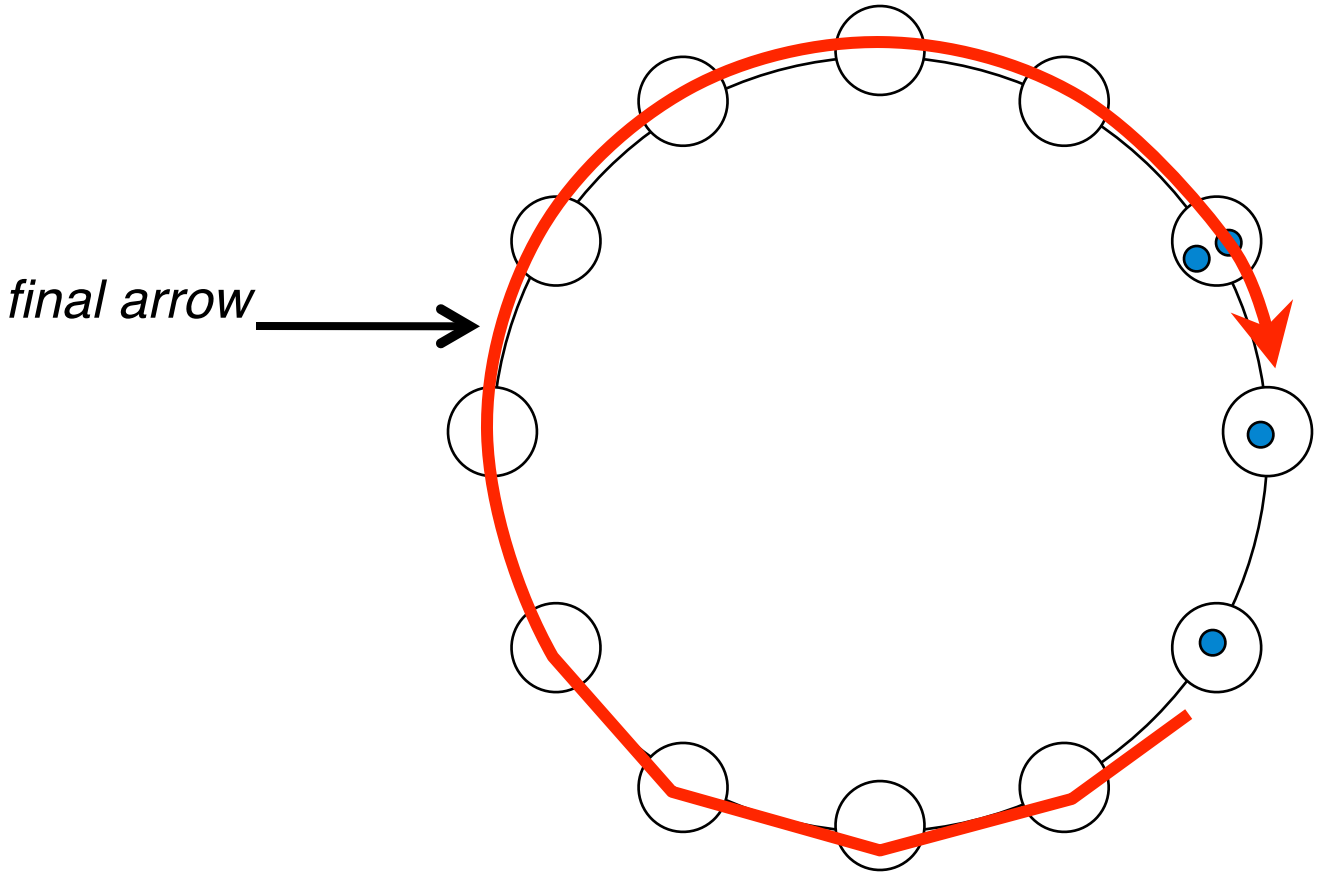
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



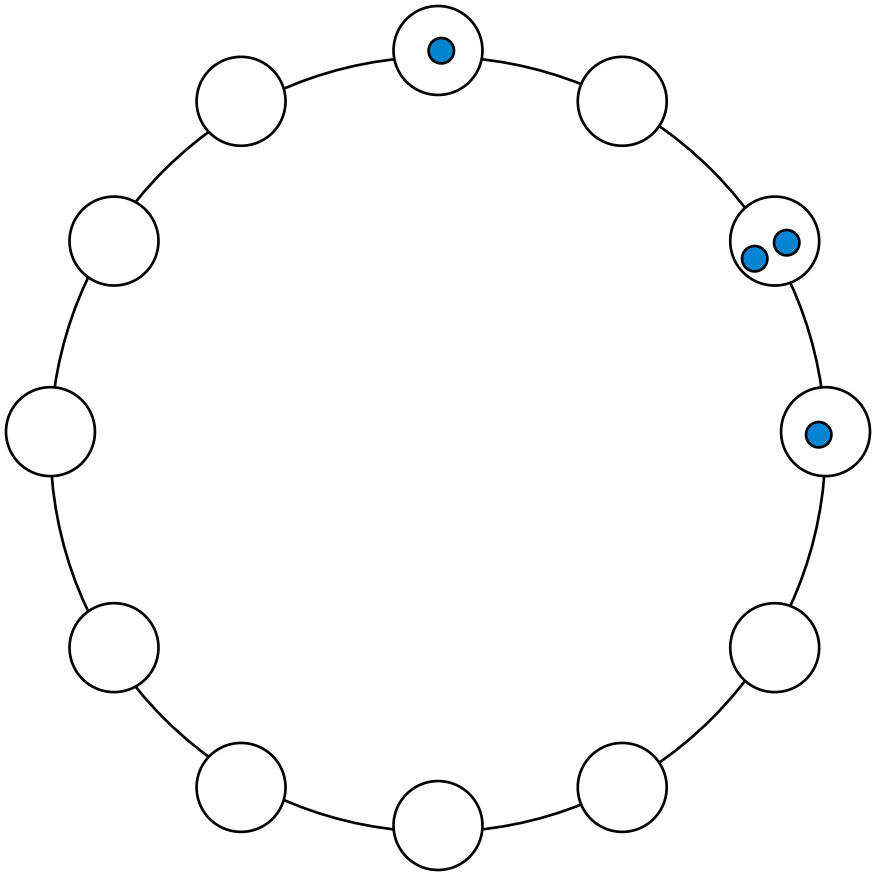
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



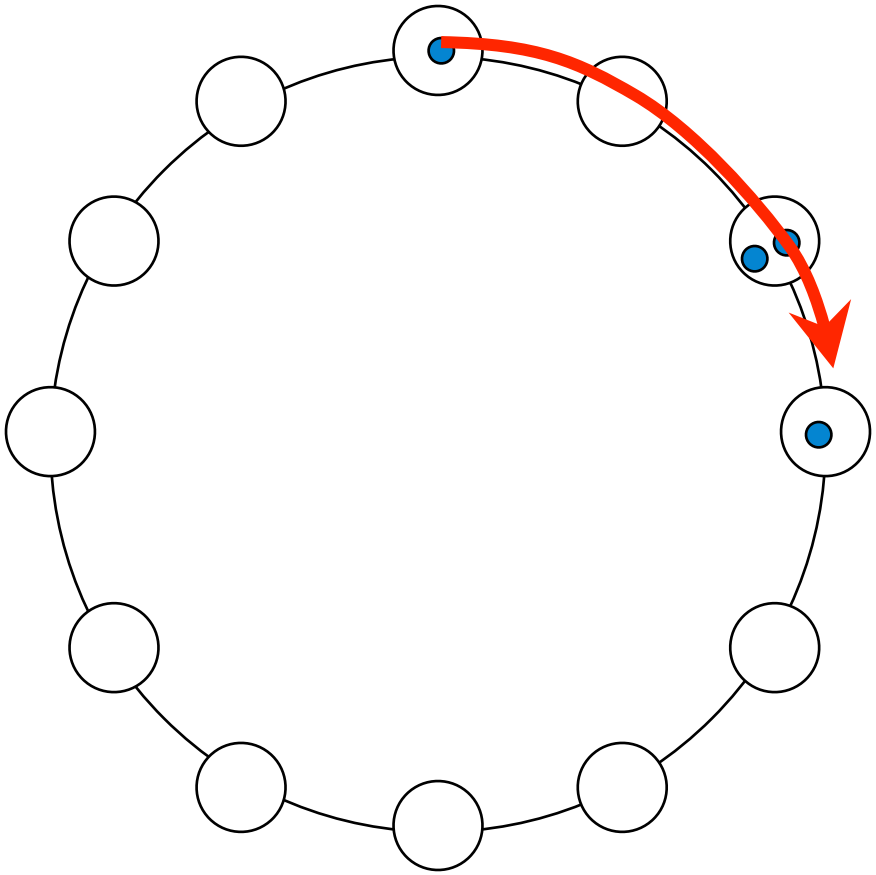
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



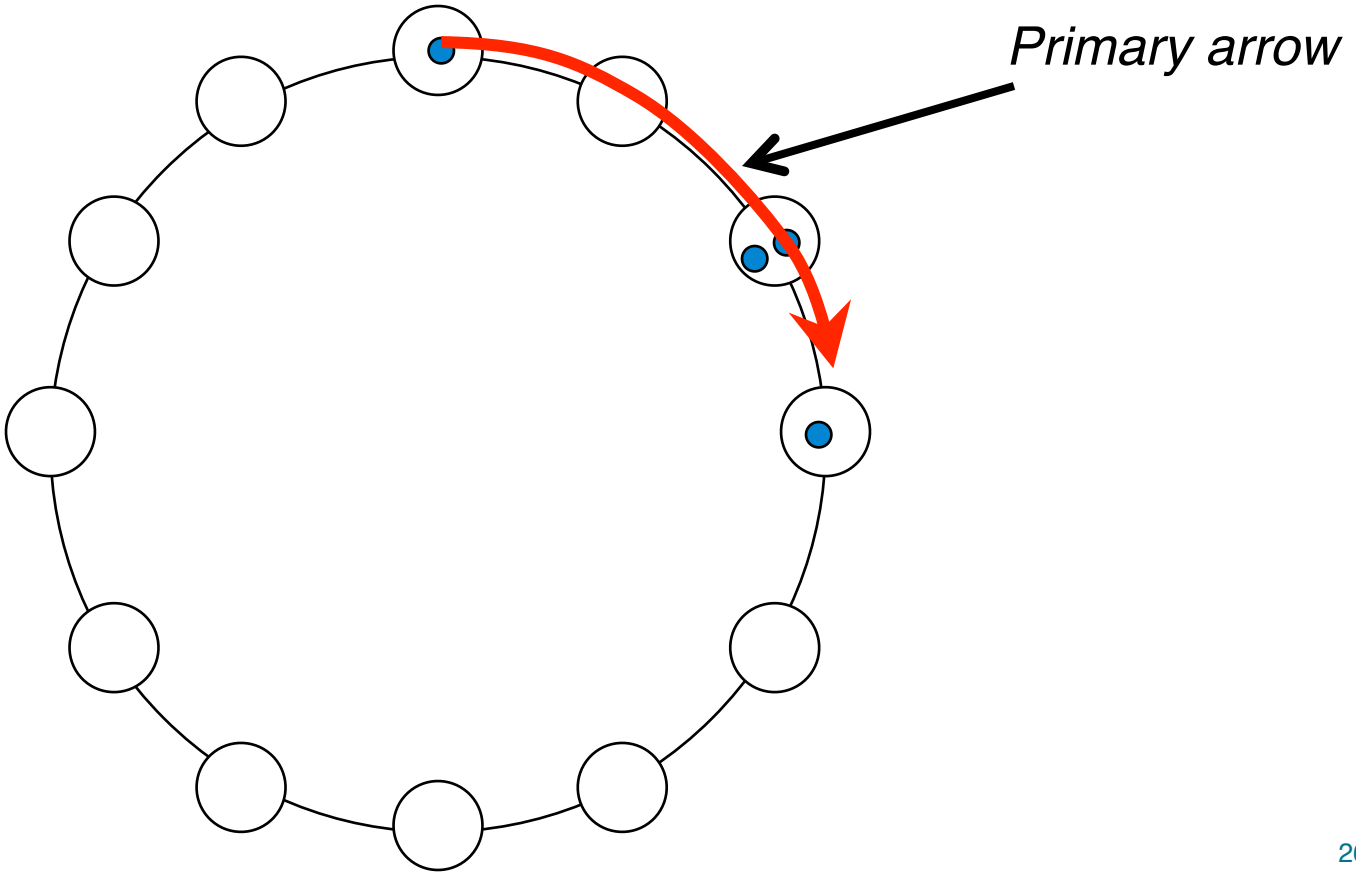
Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



Definitions

Arrow. A 1-segment, followed by a non-empty elementary path of free nodes, a tower, and a 1-segment.



Probabilistic Algorithm

Probabilistic Algorithm

Initially, there is no tower

Probabilistic Algorithm

Initially, there is no tower

Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment

Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower

Probabilistic Algorithm

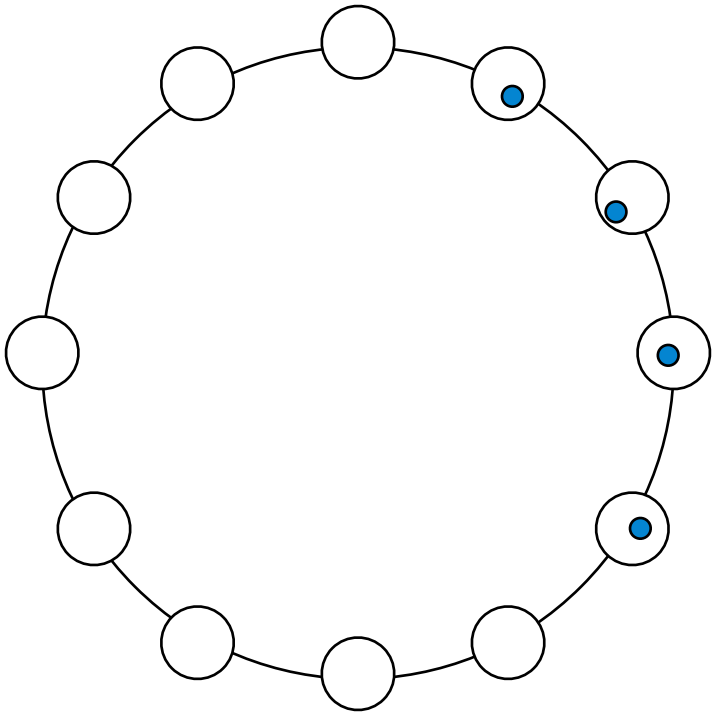
- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower
3. Visit the ring and terminate

Probabilistic Algorithm

- *Initially, there is no tower*

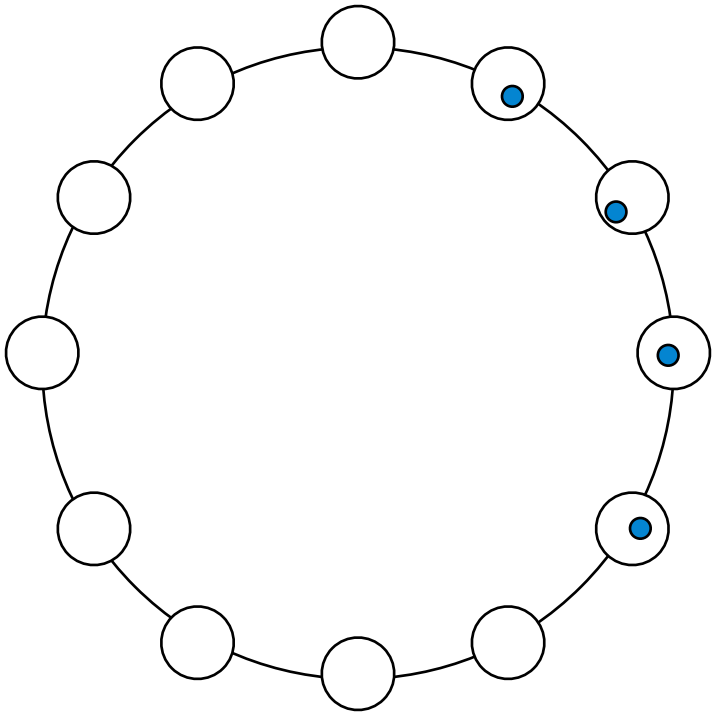
1. Converge toward a *4*-segment
2. Build a tower
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower
3. Visit the ring and terminate

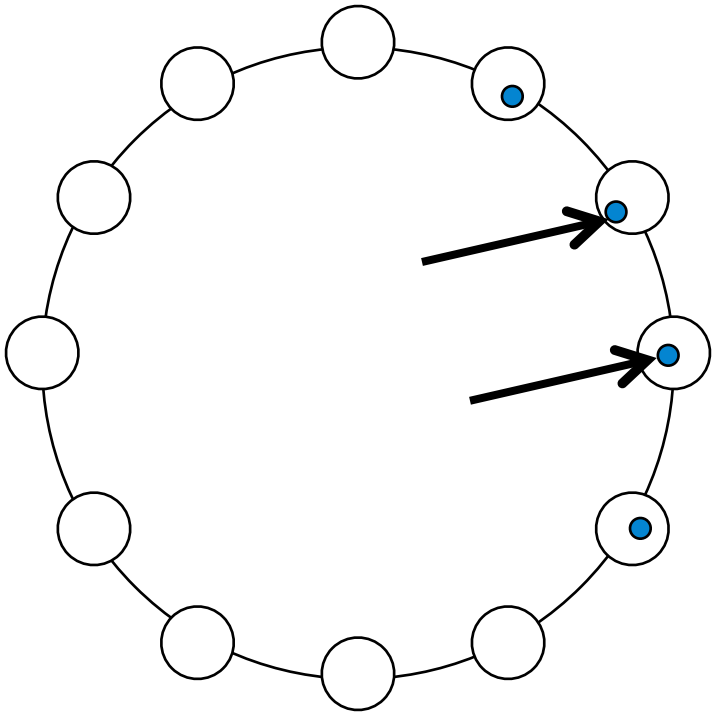


Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower
3. Visit the ring and terminate

If I am an internal node, then I try to move on the other internal node.

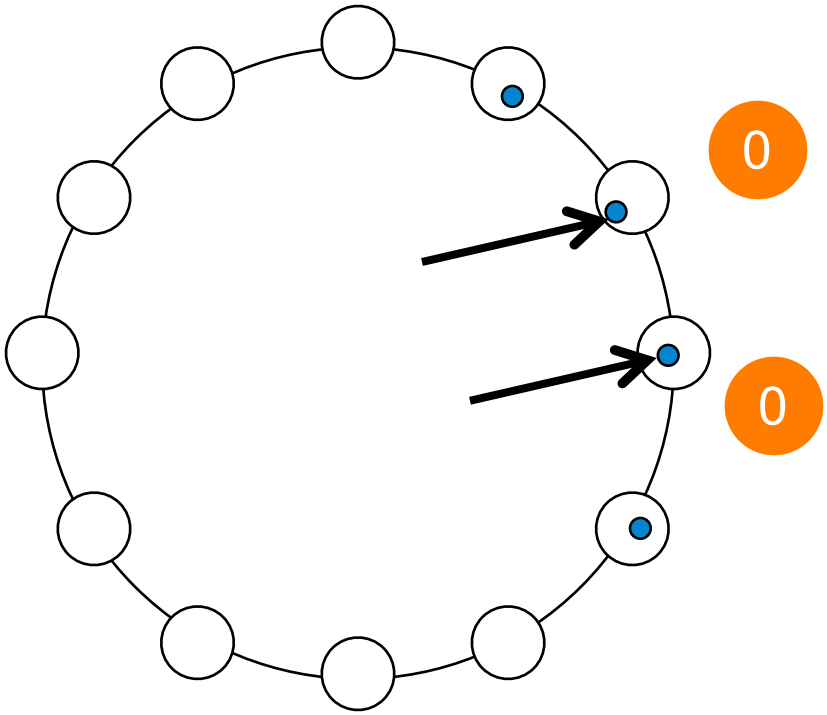


Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower
3. Visit the ring and terminate

If I am an internal node, then I try to move on the other internal node.

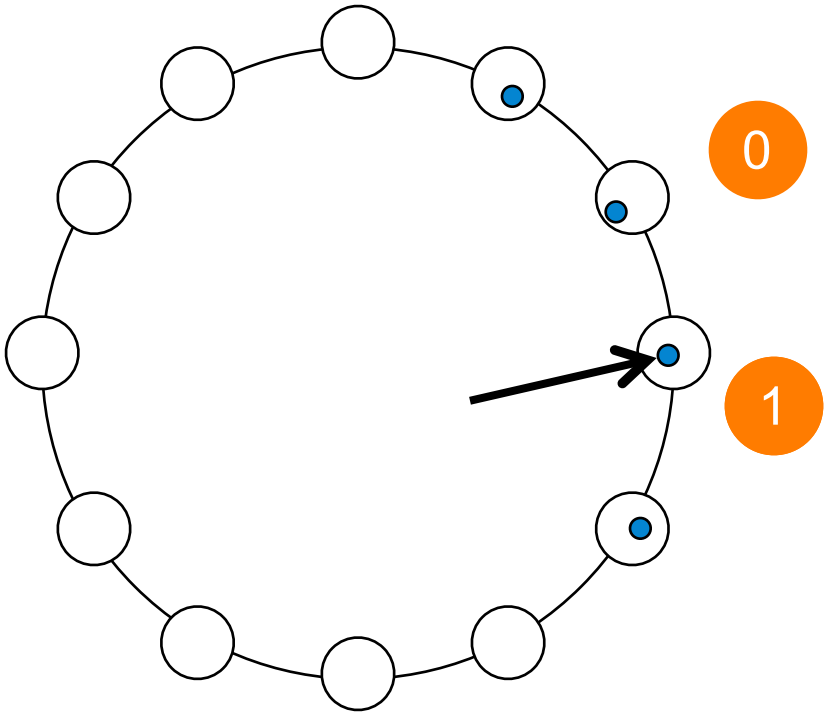


Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower
3. Visit the ring and terminate

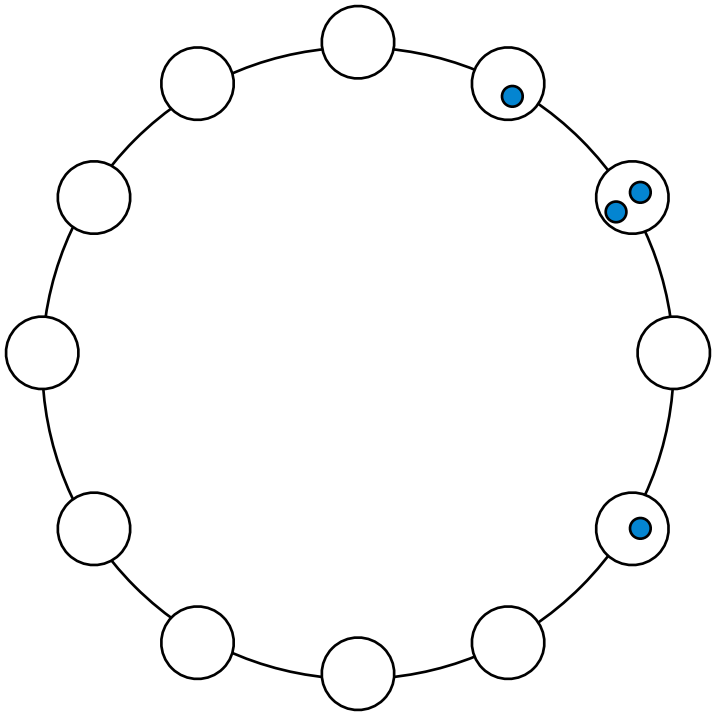
If I am an internal node, then I try to move on the other internal node.



Probabilistic Algorithm

- *Initially, there is no tower*

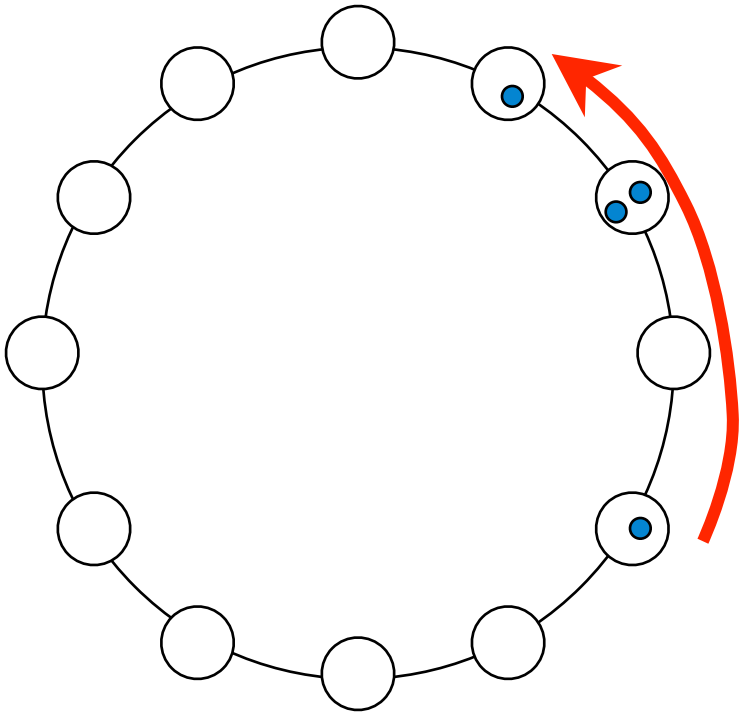
1. Converge toward a *4*-segment
2. Build a tower
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

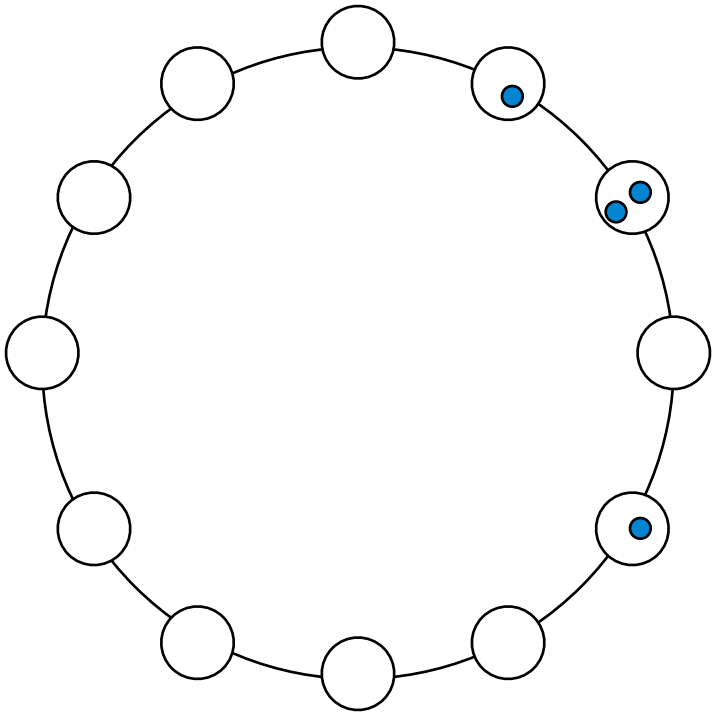
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

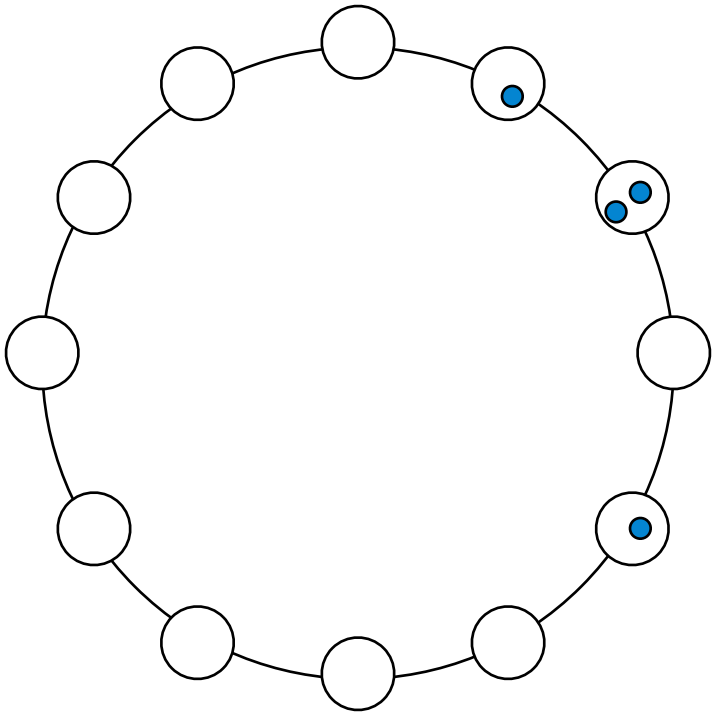
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

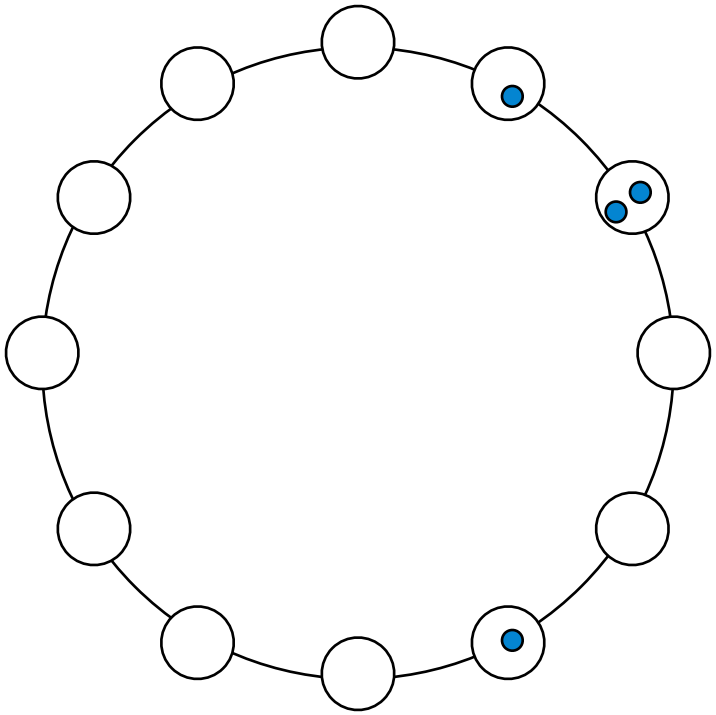
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

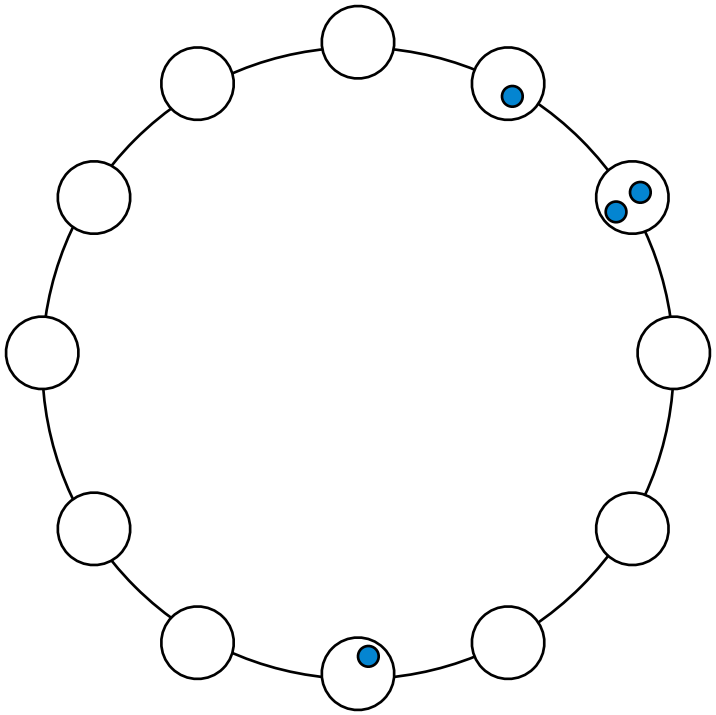
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

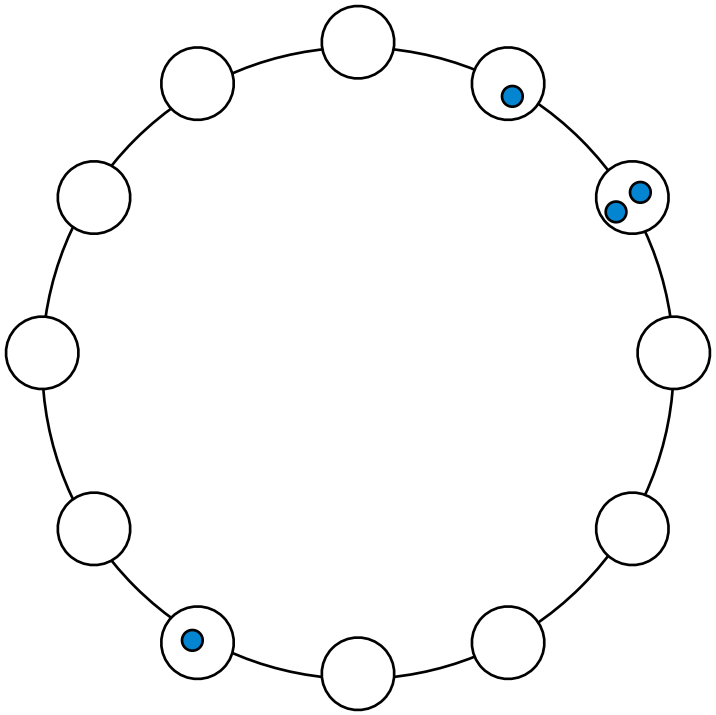
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

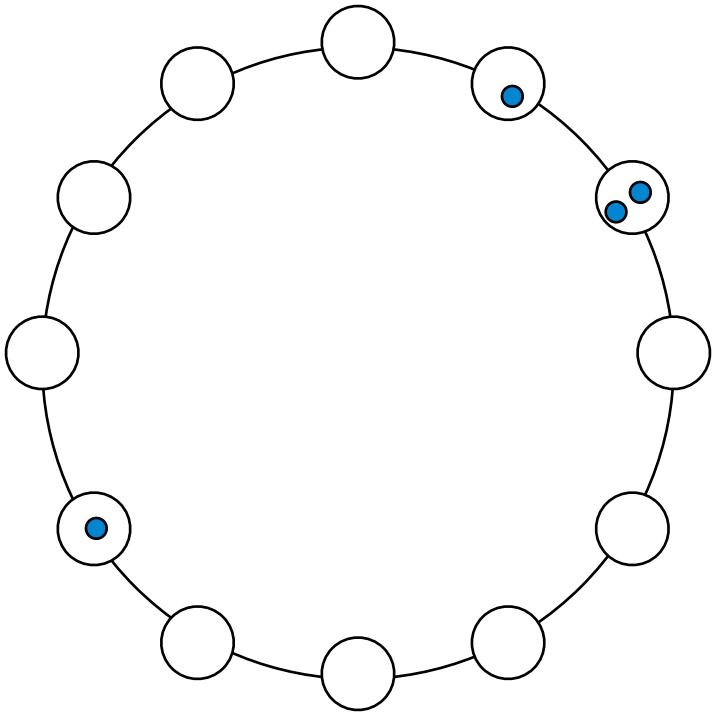
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

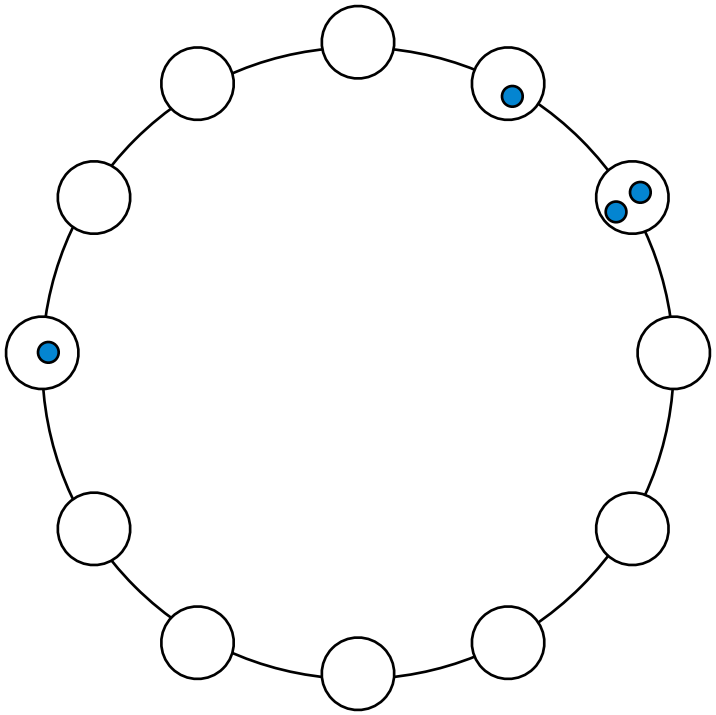
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

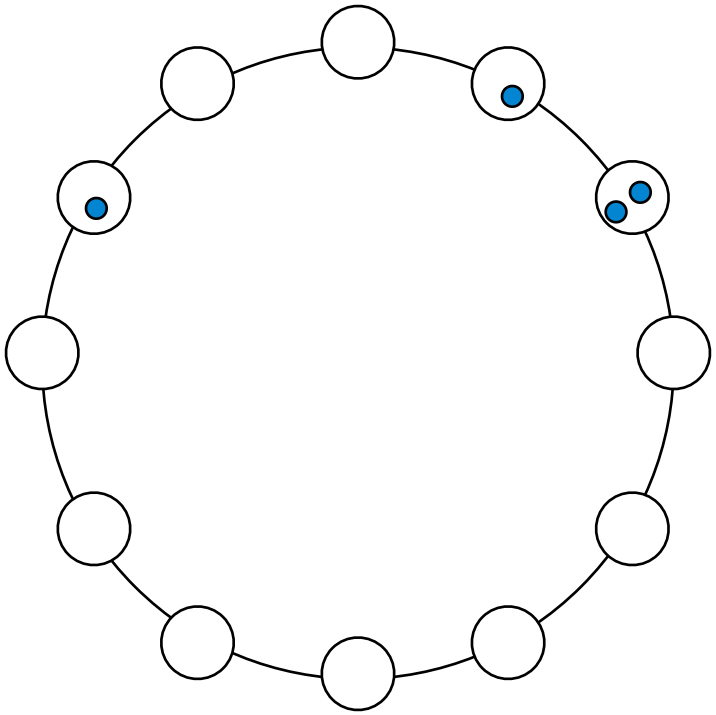
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

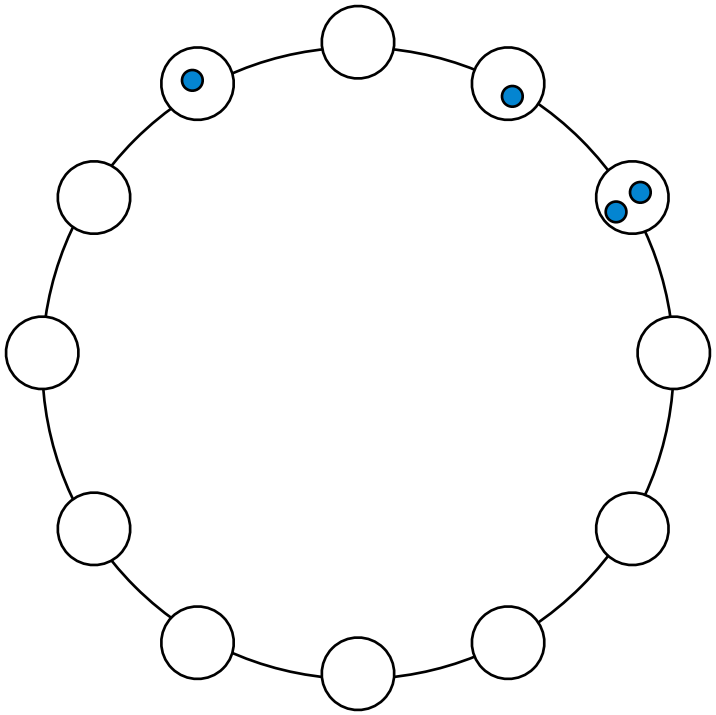
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

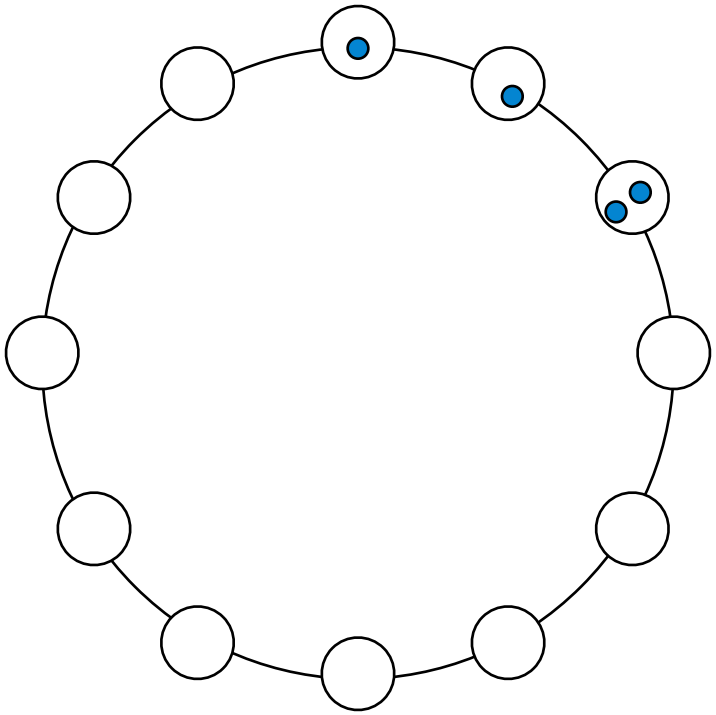
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

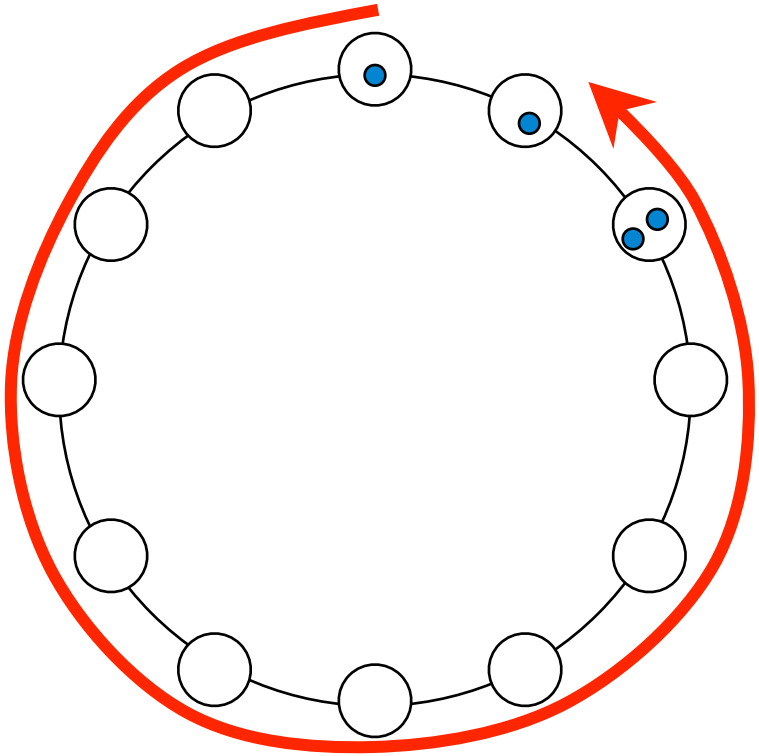
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate



Probabilistic Algorithm

- *Initially, there is no tower*

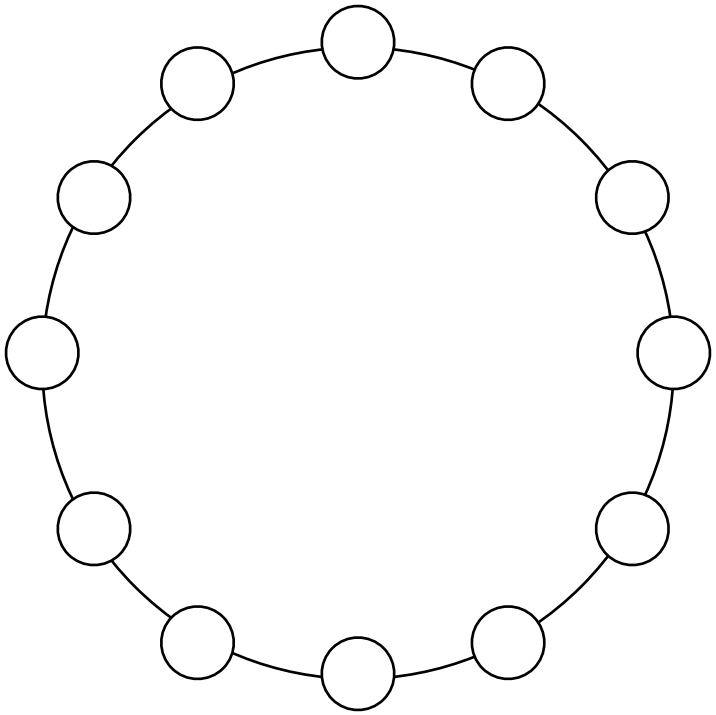
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow



Probabilistic Algorithm

- *Initially, there is no tower*

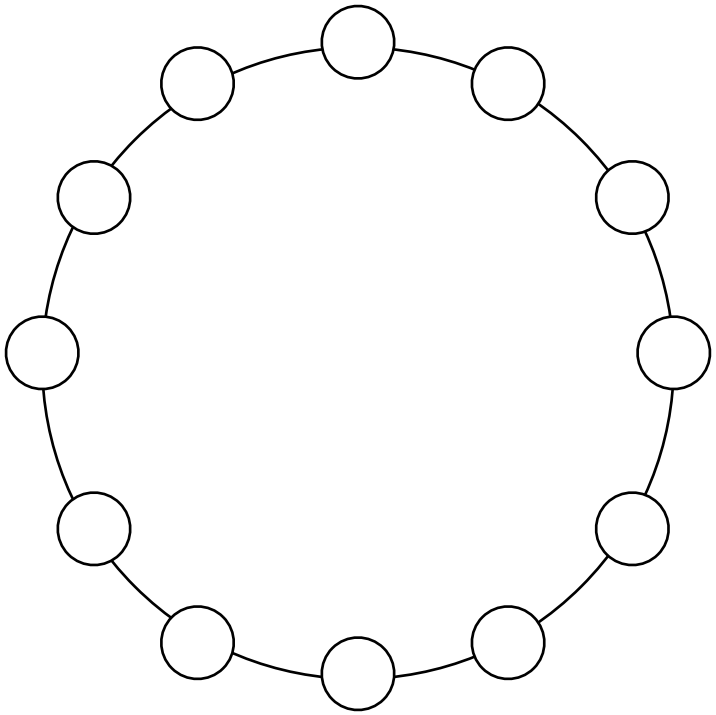
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow



Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

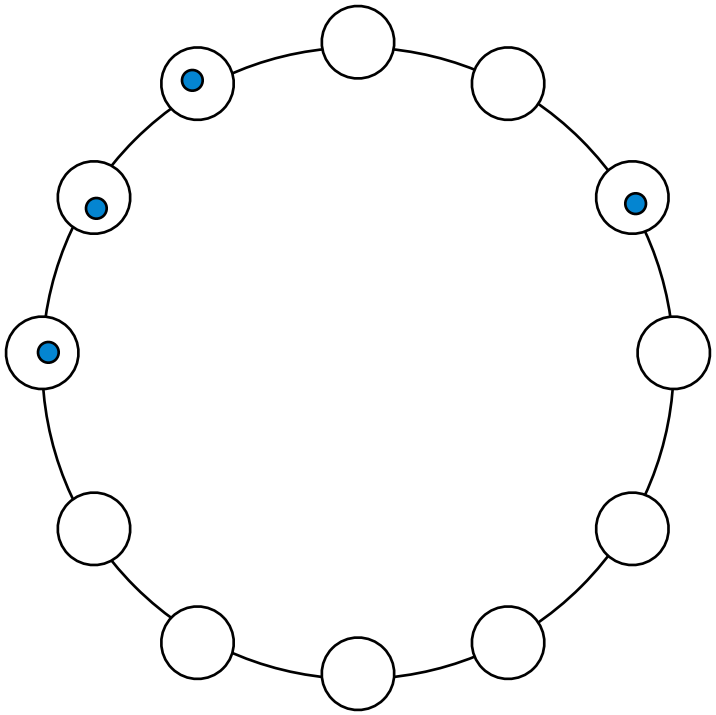


Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment

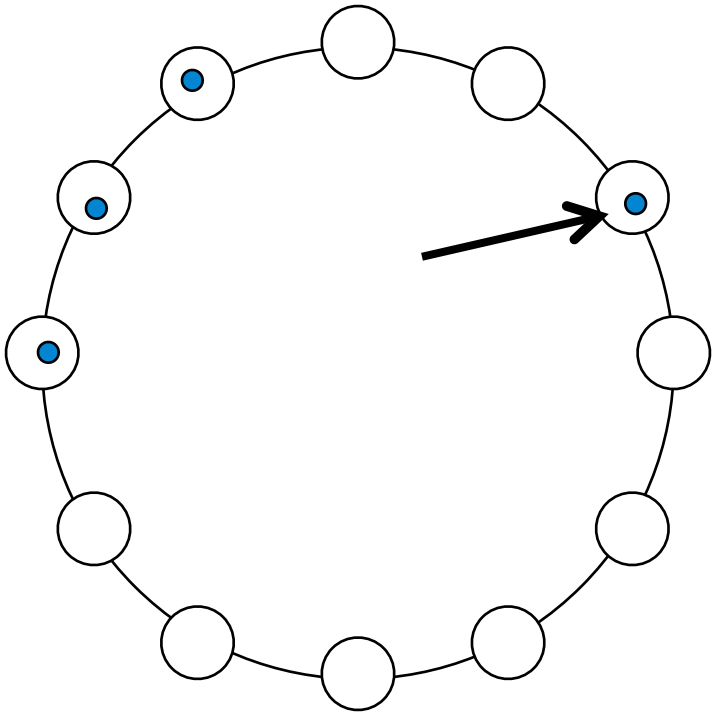


Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment



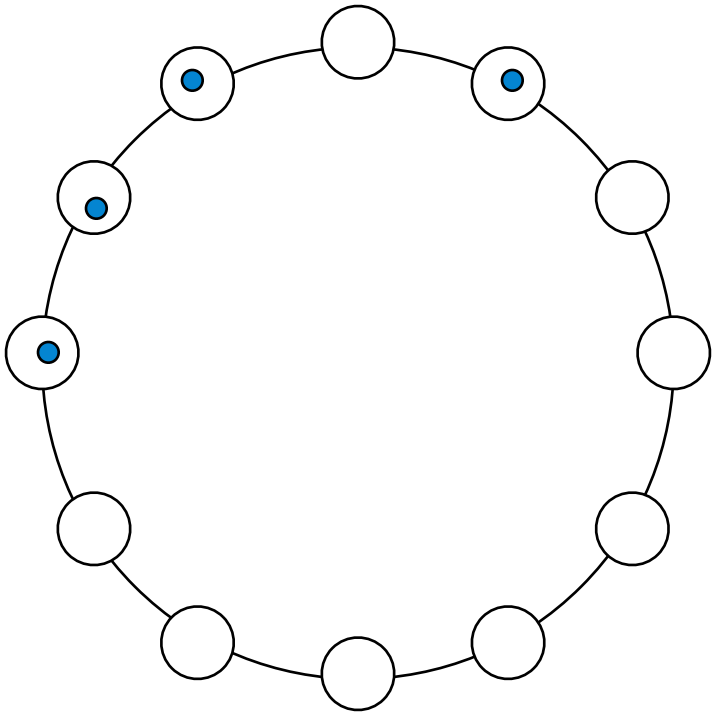
If I am the isolated node, then I move through a shortest hole.

Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment



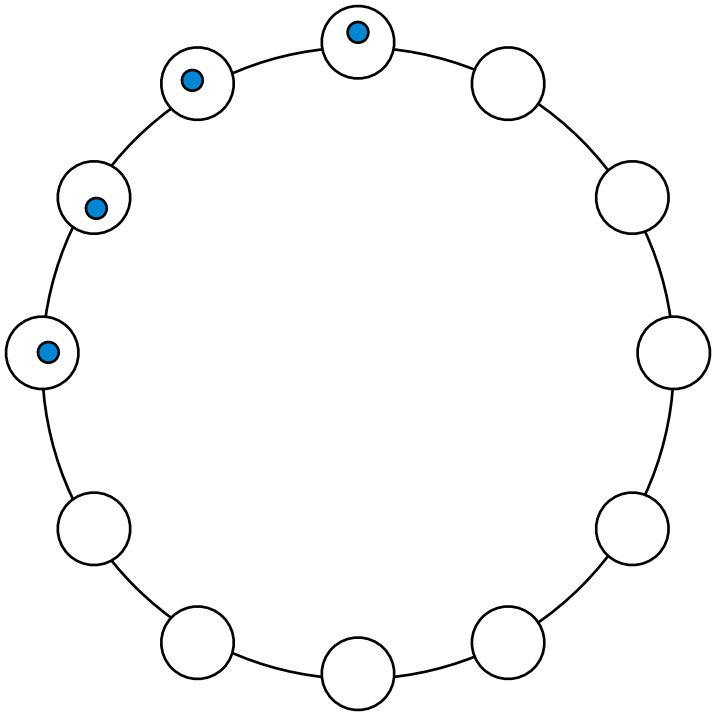
If I am the isolated node, then I move through a shortest hole.

Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment



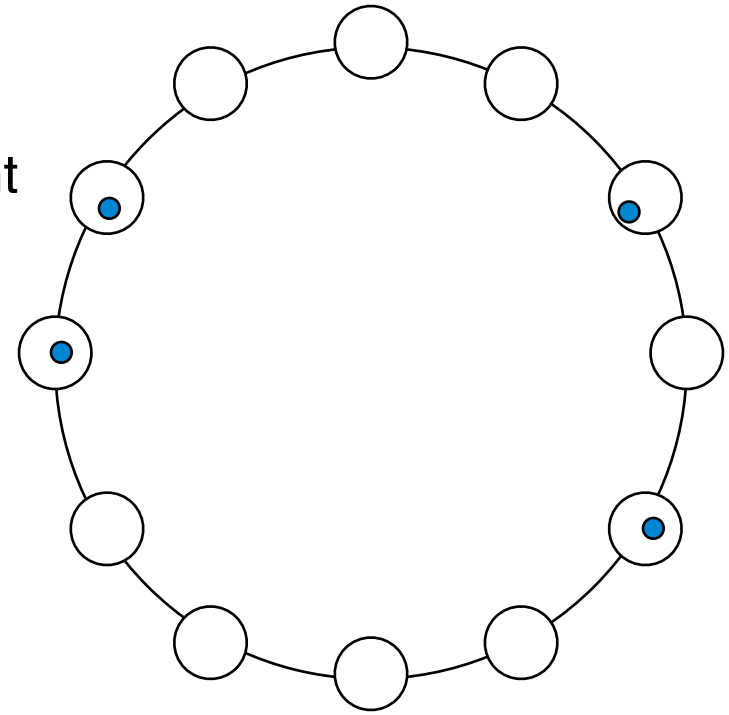
If I am the isolated node, then I move through a shortest hole.

Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment



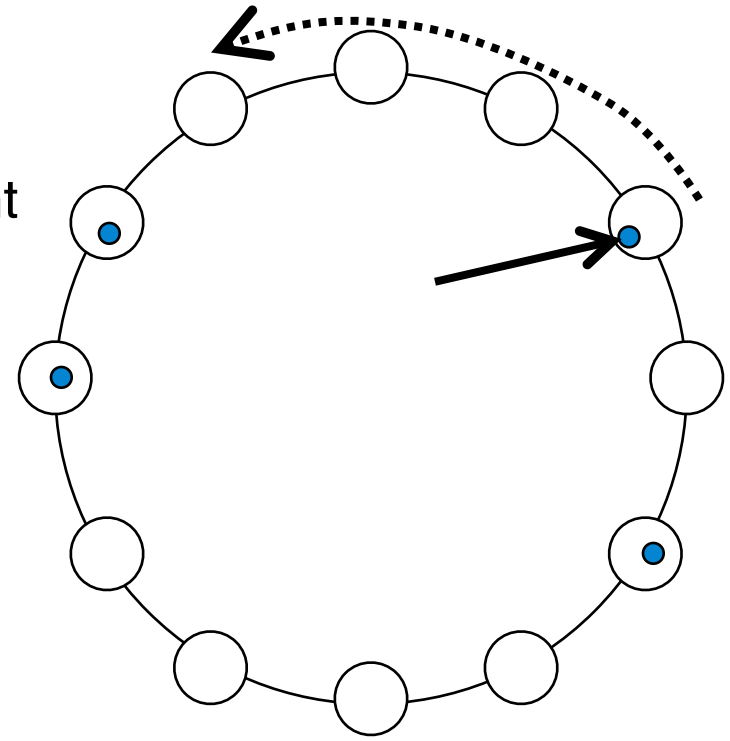
Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment

If I am at the closest distance from the 2-segment, then I move toward the closest extremity.



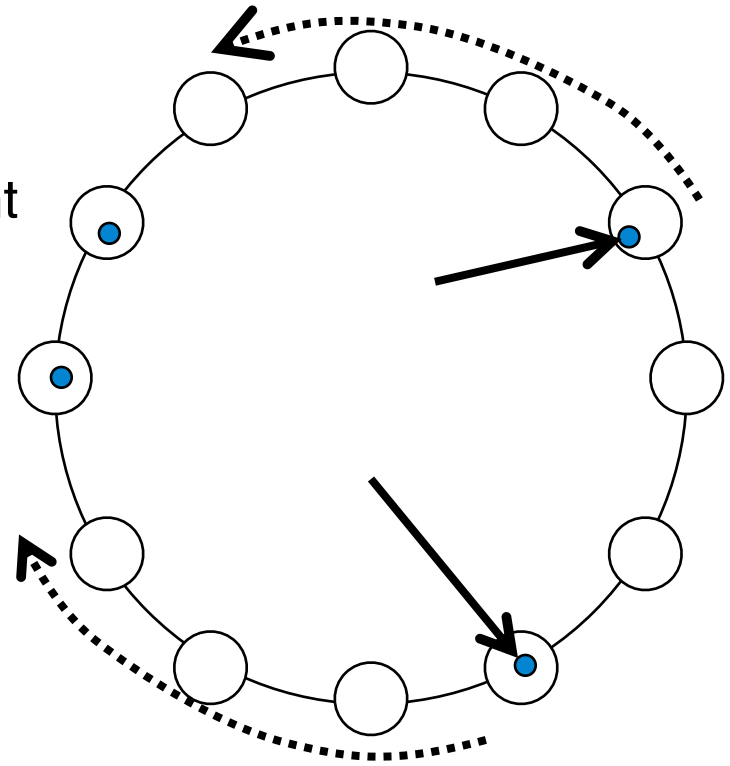
Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment

If I am at the closest distance from the 2-segment, then I move toward the closest extremity.

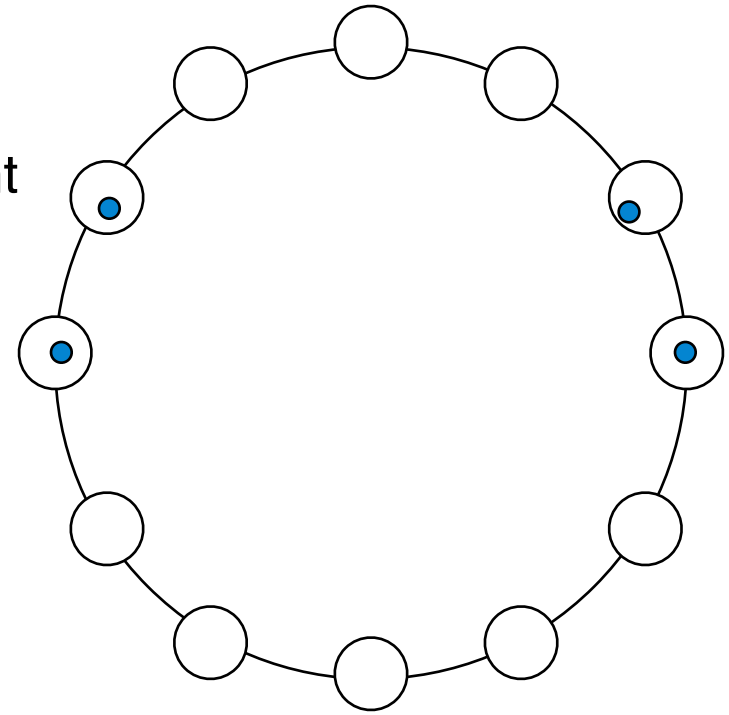


Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments



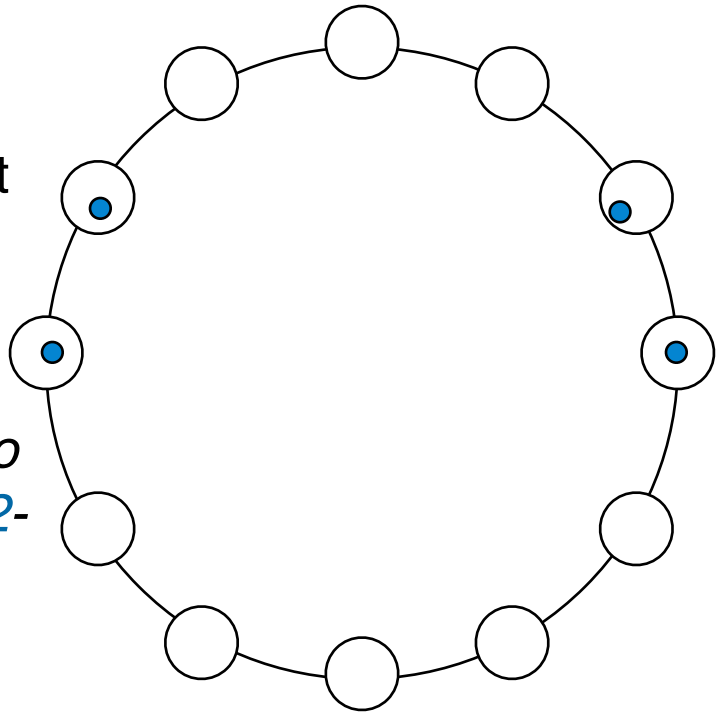
Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments

If I am a neighbor of the longest hole, then I try to move toward the other 2-segment.



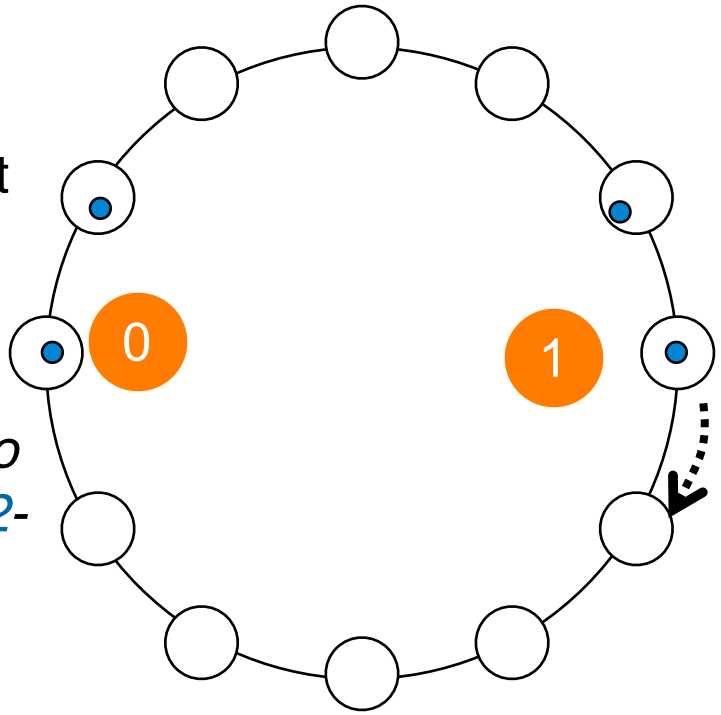
Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments

If I am a neighbor of the longest hole, then I try to move toward the other 2-segment.



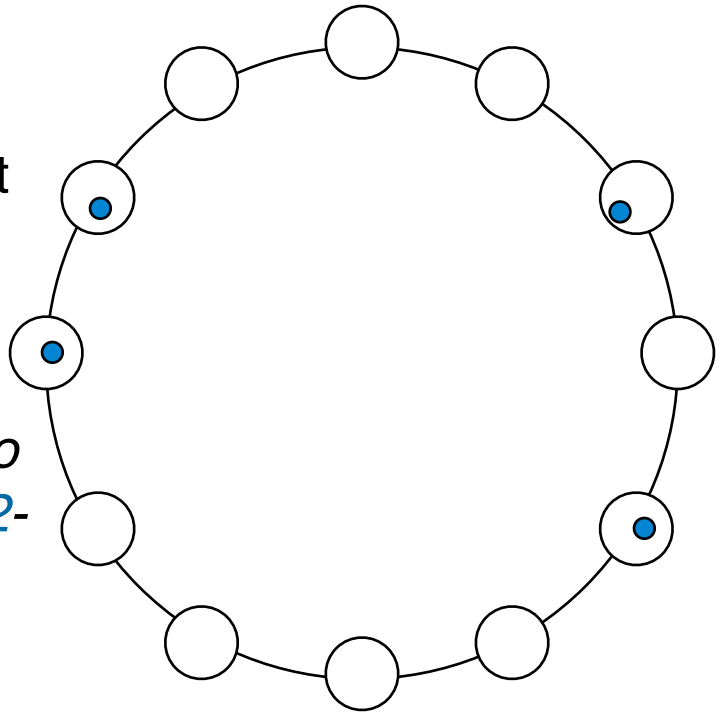
Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments

If I am a neighbor of the longest hole, then I try to move toward the other 2-segment.

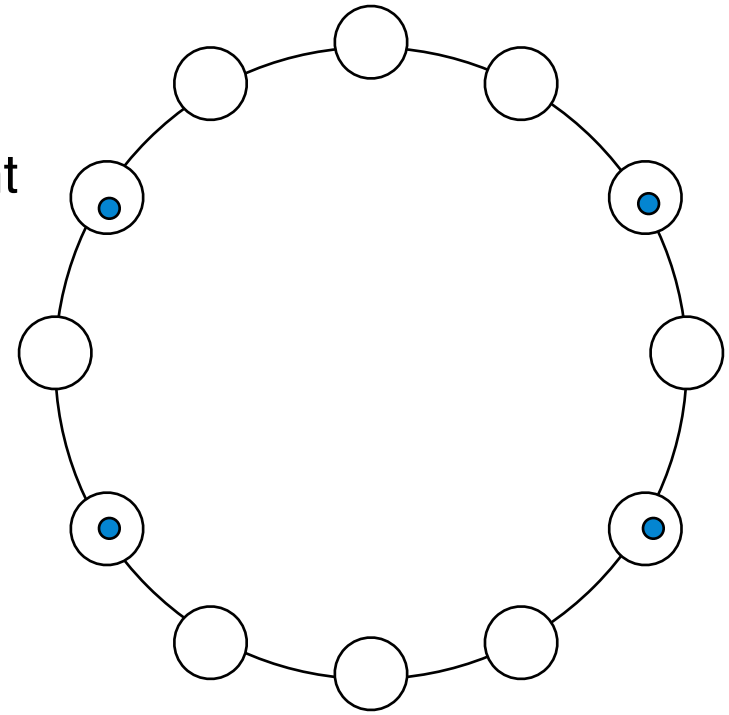


Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments
d) four isolated nodes

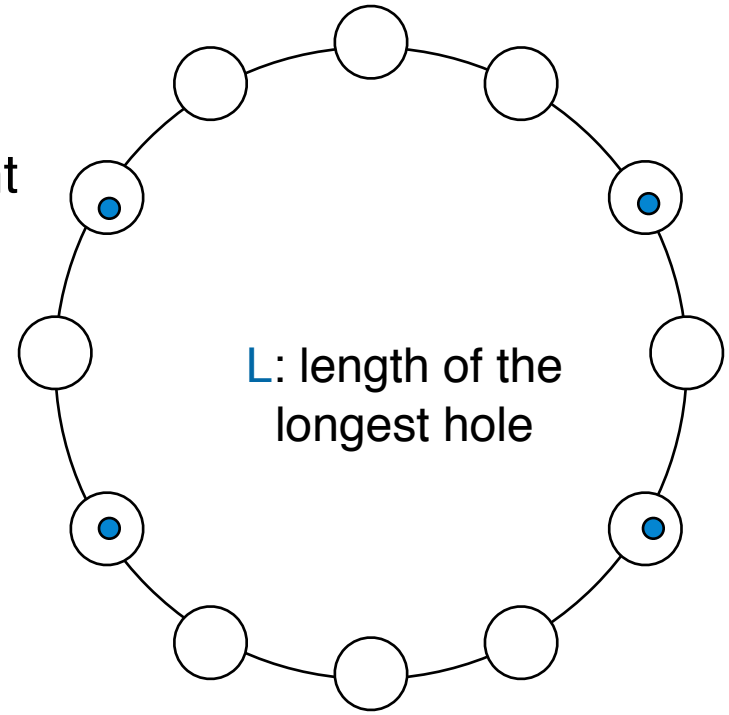


Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments
d) four isolated nodes



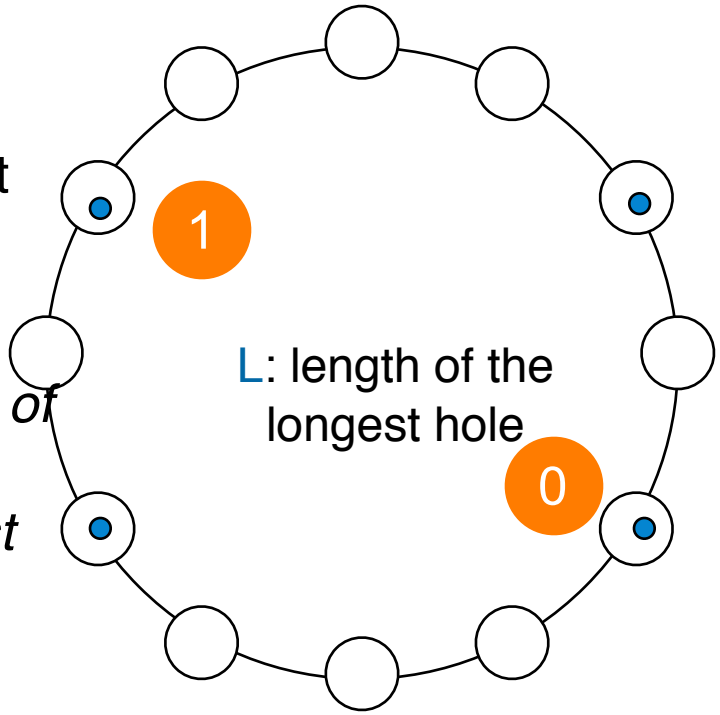
Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments
d) four isolated nodes

If 4 robots are neighbors of an L-hole, then I try to move through my longest neighboring hole.



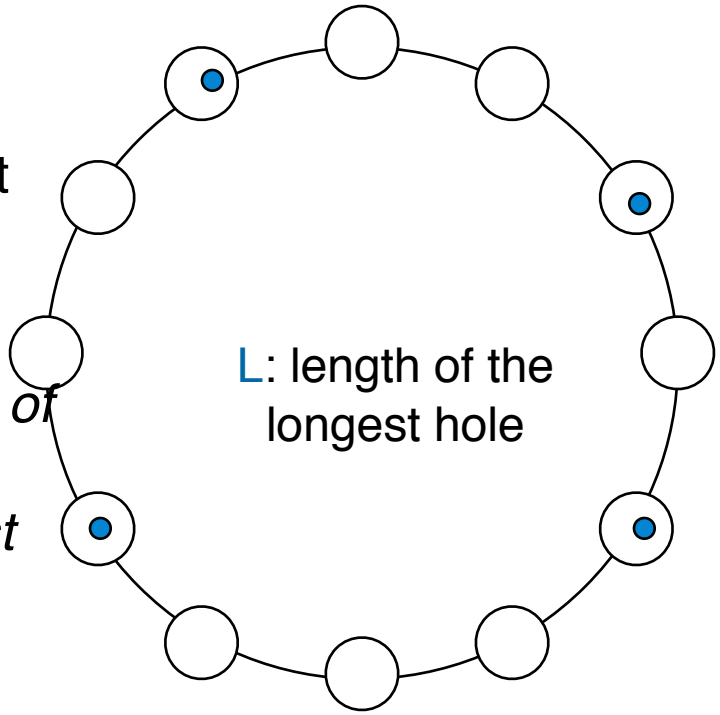
Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments
d) four isolated nodes

If 4 robots are neighbors of an L-hole, then I try to move through my longest neighboring hole.

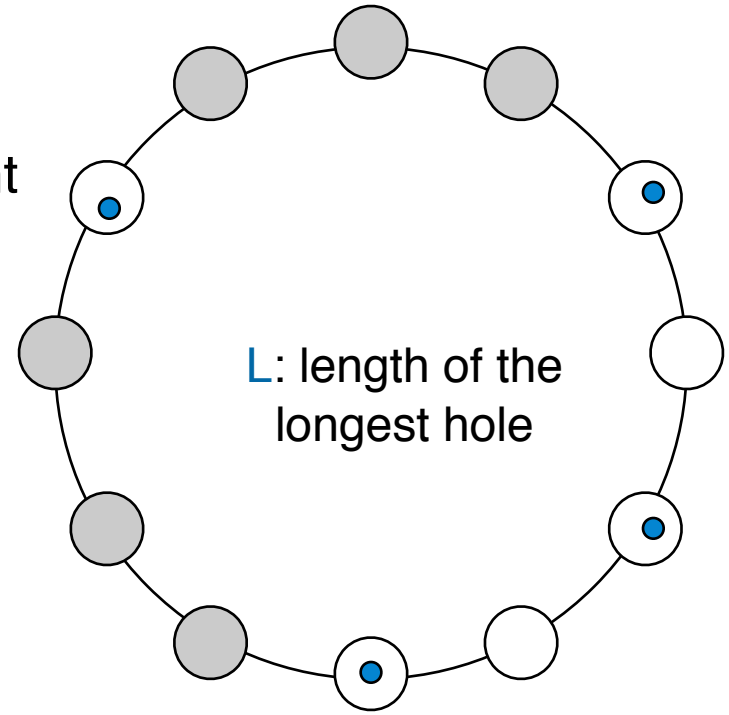


Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments
d) four isolated nodes



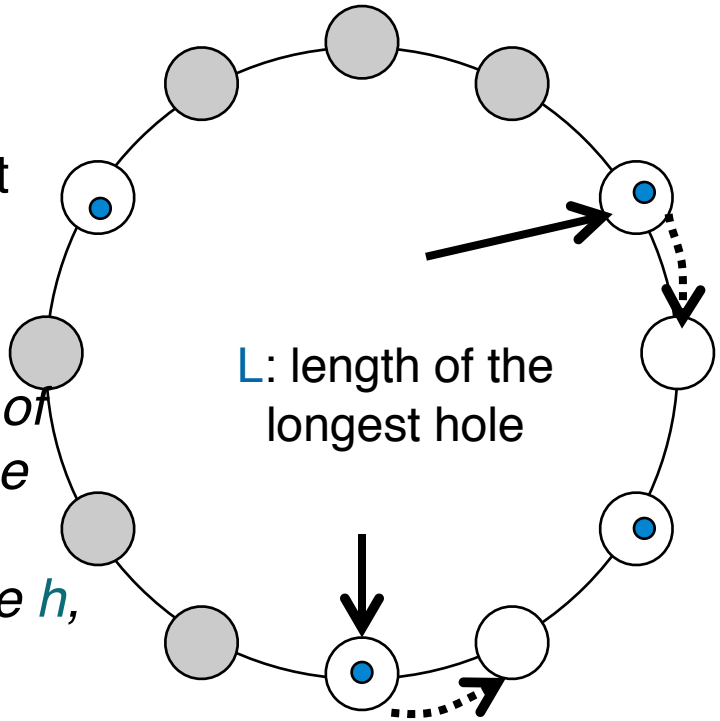
Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments
d) four isolated nodes

If 3 robots are neighbors of an L-hole, then if I am one of this 3 robots and a neighbor of a smaller hole h, then I move through h.

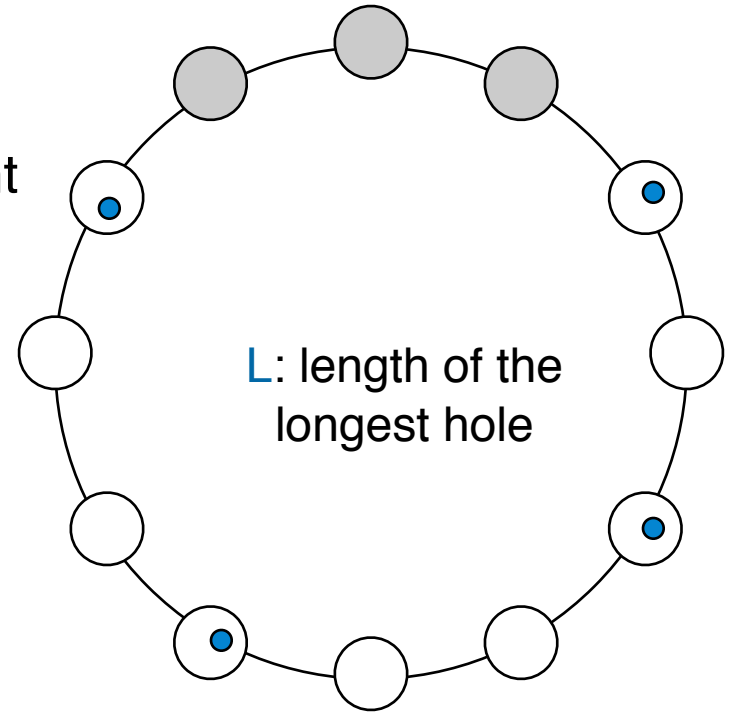


Probabilistic Algorithm

- *Initially, there is no tower*

1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments
d) four isolated nodes

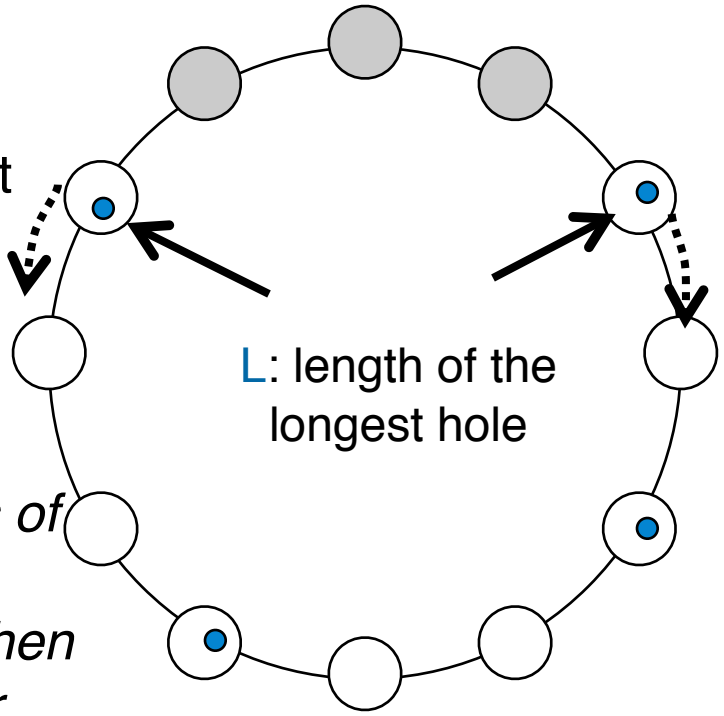


Probabilistic Algorithm

- *Initially, there is no tower*

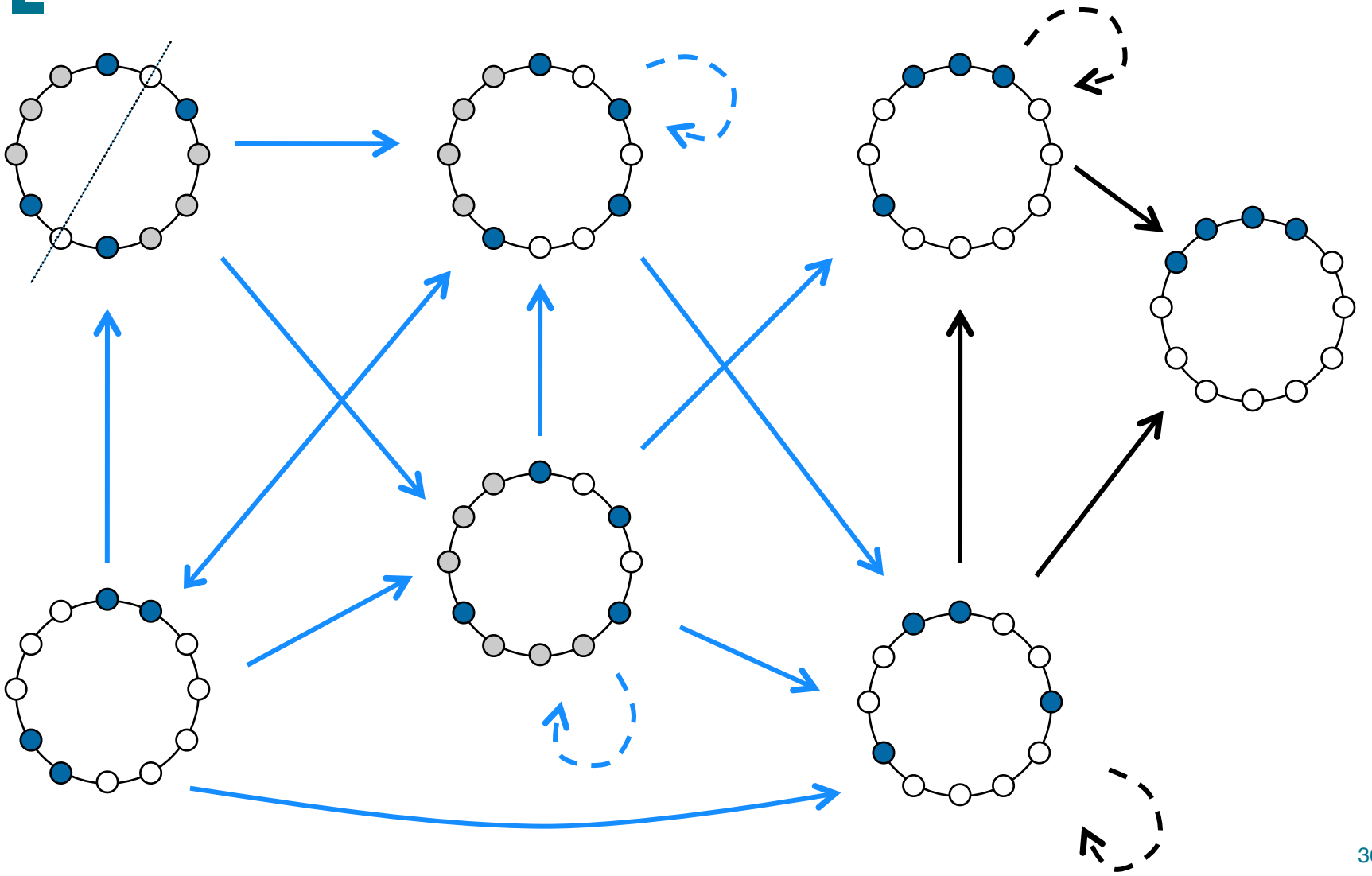
1. Converge toward a *4*-segment
2. Build a tower ➔ Primary arrow
3. Visit the ring and terminate ➔ Final arrow

a) 3-segment
b) a unique 2-segment
c) two 2-segments
d) four isolated nodes



If 2 robots are neighbors of an L-hole, then if I am neighbor of the L-hole, then I move through the other

Phase 1, Summary



Proof

Proof

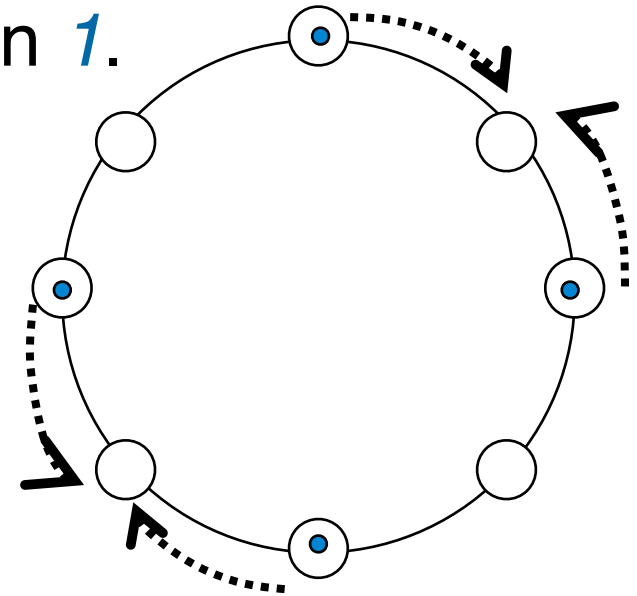
Lemma.
No tower is created during Phase 1 in a n-ring with $n > 8$.

Proof

Lemma.
No tower is created during Phase 1 in a n-ring with n > 8.

Proof Bas:

With $n > 8$ and 4 robots, there always exists a hole of length greater than 1.



Proof

Lemma.
No tower is created during Phase 1 in a n-ring with $n > 8$.

cases n=5, 6, 7 are treated separately.

Proof

Lemma.
No tower is created during Phase 1 in a n-ring with $n > 8$.

Lemma.
Starting from any initial configuration, the system reaches in finite expected time a configuration containing a 4-segment.

cases n=5, 6, 7 are treated separately.

Proof

Lemma.
No tower is created during Phase 1 in a n-ring with $n > 8$.

Lemma.
Starting from any initial configuration, the system reaches in finite expected time a configuration containing a 4-segment.

Theorem.
The algorithm (Phases 1 to 3) is a probabilistic exploration protocol for 4 robots in a ring of $n > 8$ nodes.

cases n=5, 6, 7 are treated separately.

Conclusion

n: Number of nodes
k: Number of agents

- ✓ *Ring*

Conclusion

n: Number of nodes
k: Number of agents

✓ *Ring* [*Flocchini et al.*, OPODIS 2007]

- Deterministic exploration impossible if *k* divides *n* (except if *k* = *n*)
- Asynchronous deterministic algorithm with *k* >16
- Deterministic or probabilistic exploration impossible if *k* < *4*
- Probabilistic algorithm impossible in asynchronous settings
- Optimal Semi-synchronous Probabilistic Algorithm
- Deterministic exploration impossible if *k* < 5 and *n* even
- Optimal asynchronous deterministic algorithm, k = 5 and n even
- Optimal semi-synchronous deterministic algorithm, k = 4 and n odd

Conclusion

n: Number of nodes
k: Number of agents

✓ *Ring* [*Flocchini et al.*, OPODIS 2007] [*Devismes et al.*, SIROCCO 2009]

- Deterministic exploration impossible if *k* divides *n* (except if *k* = *n*)
- Asynchronous deterministic algorithm with *k* >16
- Deterministic or probabilistic exploration impossible if *k* < *4*
- Probabilistic algorithm impossible in asynchronous settings
- Optimal Semi-synchronous Probabilistic Algorithm
- Deterministic exploration impossible if *k* < 5 and *n* even
- Optimal asynchronous deterministic algorithm, k = 5 and n even
- Optimal semi-synchronous deterministic algorithm, k = 4 and n odd

Conclusion

n: Number of nodes
k: Number of agents

✓ *Ring* [*Flocchini et al.*, OPODIS 2007] [*Devismes et al.*, SIROCCO 2009]
[*Lamani et al.*, SIROCCO 2010]

- Deterministic exploration impossible if *k* divides *n* (except if *k* = *n*)
- Asynchronous deterministic algorithm with *k* >16
- Deterministic or probabilistic exploration impossible if *k* < *4*
- Probabilistic algorithm impossible in asynchronous settings
- Optimal Semi-synchronous Probabilistic Algorithm
- Deterministic exploration impossible if *k* < 5 and *n* even
- Optimal asynchronous deterministic algorithm, k = 5 and n even
- Optimal semi-synchronous deterministic algorithm, k = 4 and n odd

Conclusion

n: Number of nodes
k: Number of agents

✓ *Ring* [*Flocchini et al.*, OPODIS 2007] [*Devismes et al.*, SIROCCO 2009]
[*Lamani et al.*, SIROCCO 2010]

Conclusion

n: Number of nodes
k: Number of agents

✓ *Ring* [*Flocchini et al.*, OPODIS 2007] [*Devismes et al.*, SIROCCO 2009]
[*Lamani et al.*, SIROCCO 2010]

✓ *Tree* [*Flocchini et al.*, SIROCCO 2008]

- Asynchronous deterministic algorithm for trees with maximum degree equal to 3: $k \in \Theta(\log n / \log \log n)$
- Arbitrary tree: $k \in \Theta(\log n)$

✓ *Chain* [*Flocchini et al.*, IPL 2011]

- Characterization of k: $k = 3$, $k > 4$, or $k = 4$ and n odd

Conclusion

n: Number of nodes
k: Number of agents

✓ *Ring* [*Flocchini et al.*, OPODIS 2007] [*Devismes et al.*, SIROCCO 2009]
[*Lamani et al.*, SIROCCO 2010]

✓ *Tree* [*Flocchini et al.*, SIROCCO 2008]

- Asynchronous deterministic algorithm for trees with maximum degree equal to 3: $k \in \Theta(\log n / \log \log n)$
- Arbitrary tree: $k \in \Theta(\log n)$

✓ *Chain* [*Flocchini et al.*, IPL 2011]

- Characterization of *k*: $k = 3$, $k > 4$, or $k = 4$ and n odd

✓ *Grid* [*Devismes et al.*, SSS 2012]

- Deterministic or probabilistic exploration impossible if $k < 2$
- Optimal Semi-synchronous Deterministic Algorithm, $k = 3$