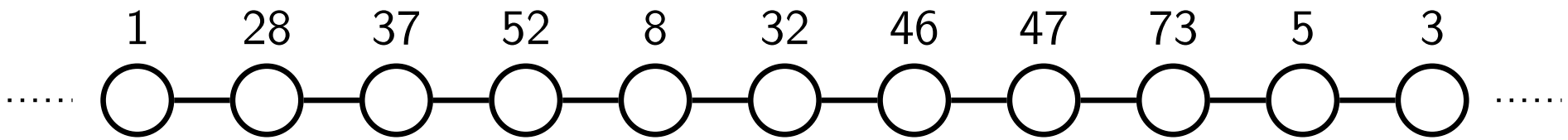
Decidability of distributed complexity of locally checkable problems on paths

Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti,
Mikaël Rabie, Jukka Suomela

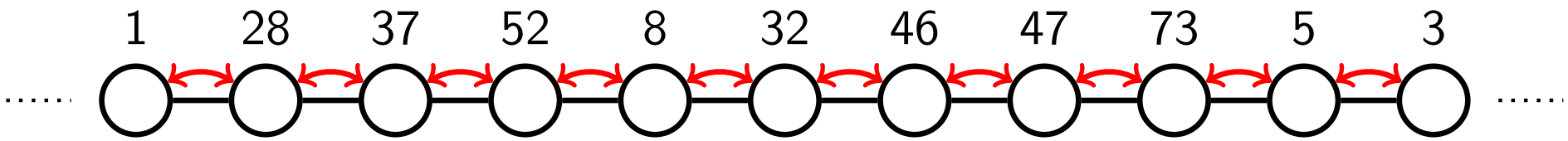
ANR DESCARTES/ESTATE

Tuesday, April 2

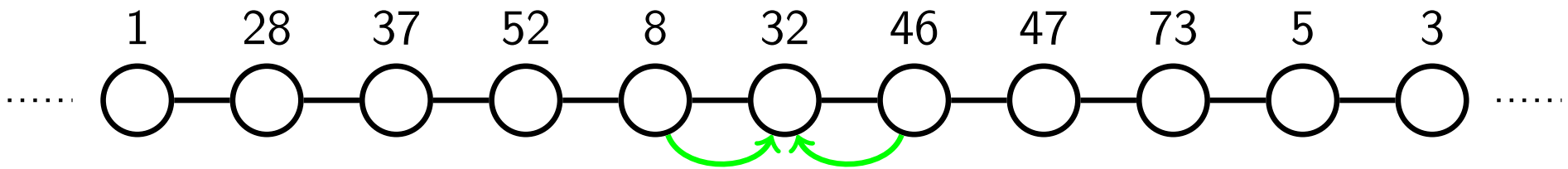
LCL Problems on Paths



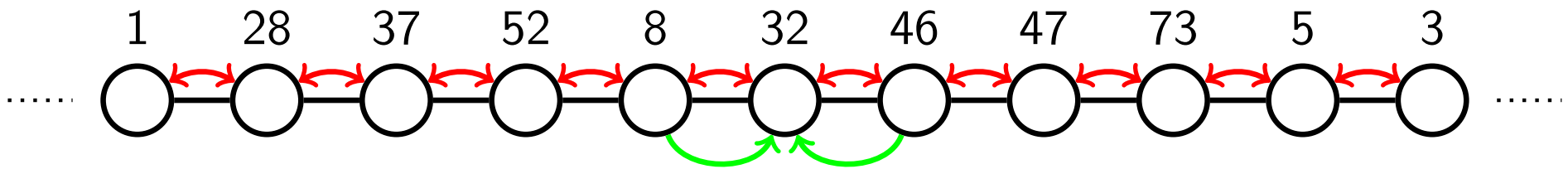
LCL Problems on Paths



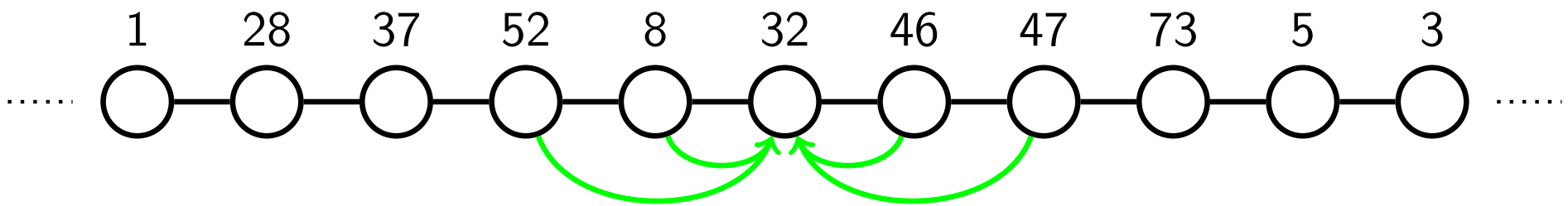
LCL Problems on Paths



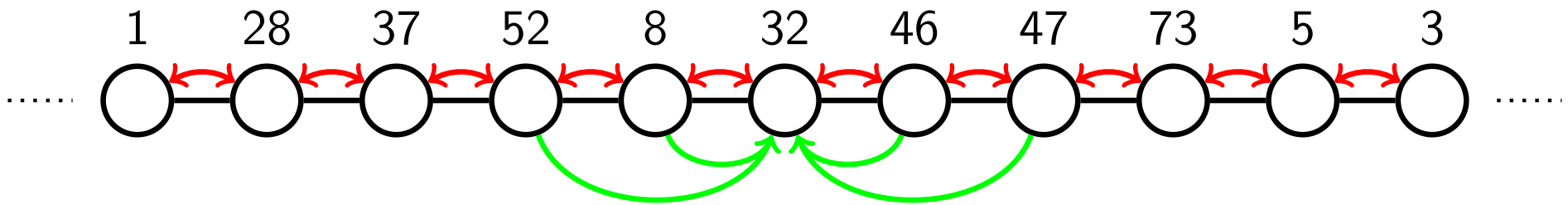
LCL Problems on Paths



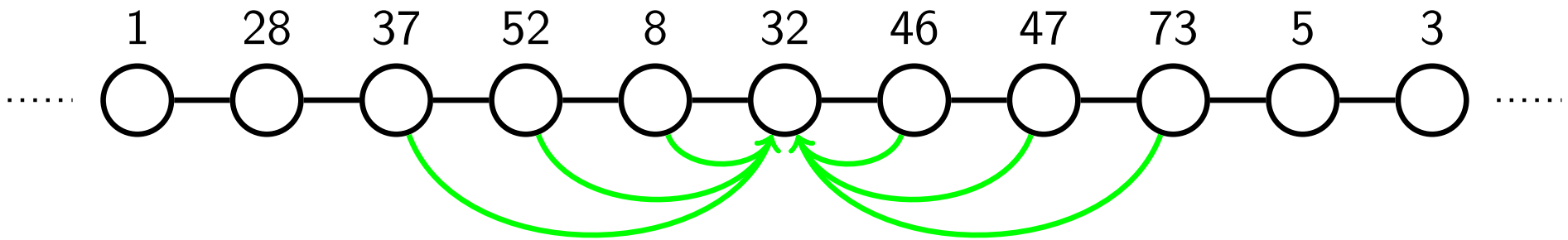
LCL Problems on Paths



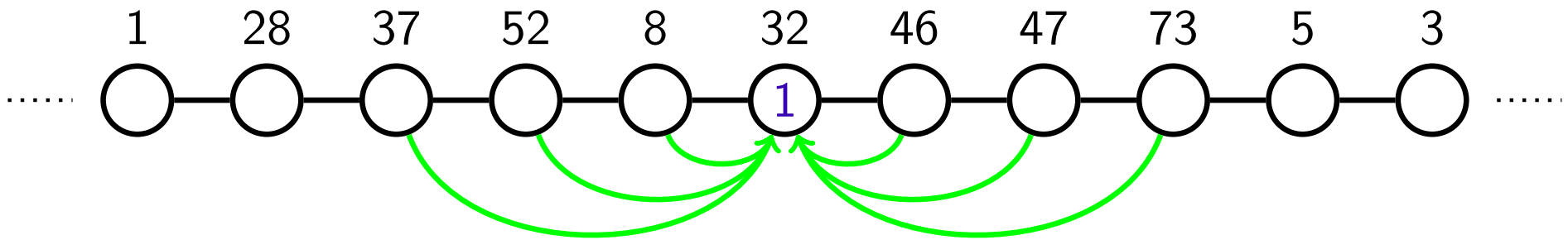
LCL Problems on Paths



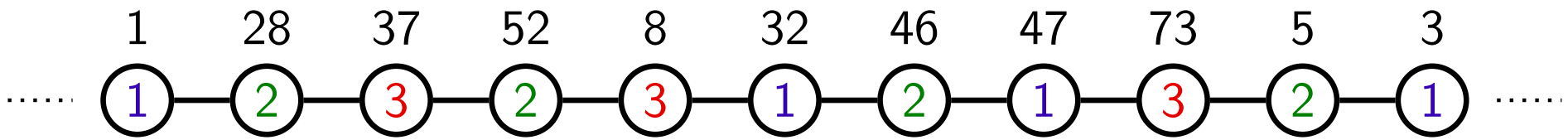
LCL Problems on Paths



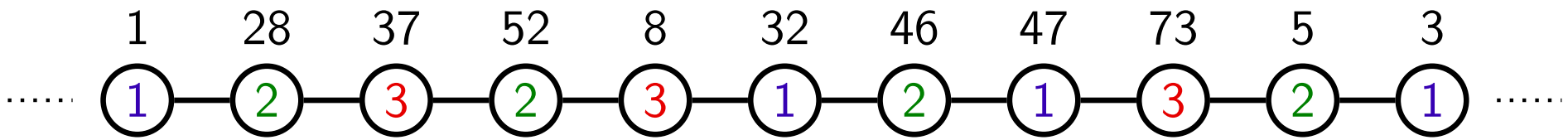
LCL Problems on Paths



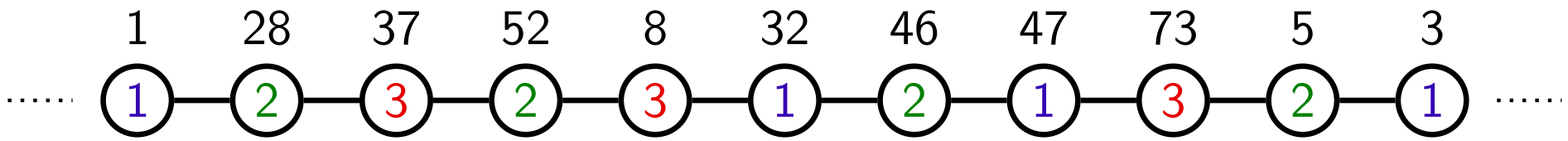
LCL Problems on Paths



LCL Problems on Paths

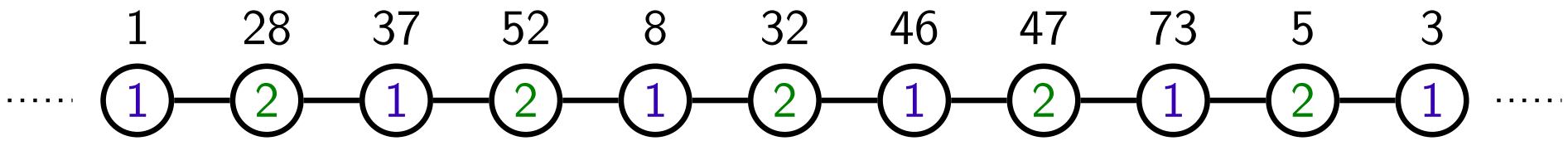


LCL Problems on Paths



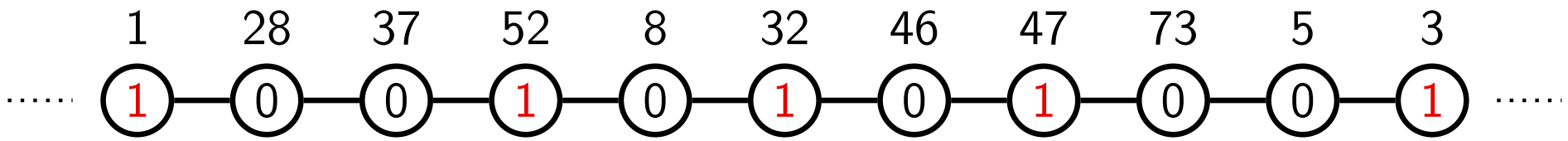
- 3 Coloring.

LCL Problems on Paths



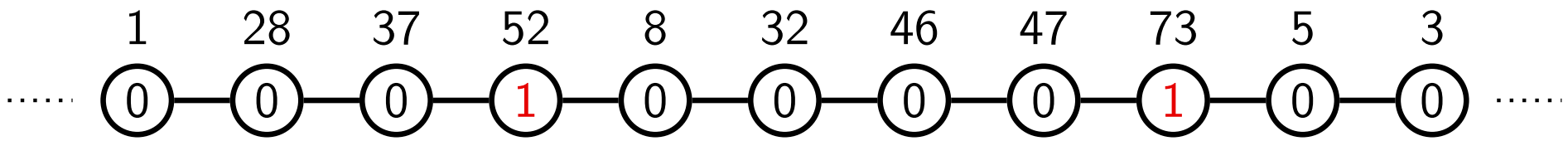
- 3 Coloring.
- 2 Coloring.

LCL Problems on Paths



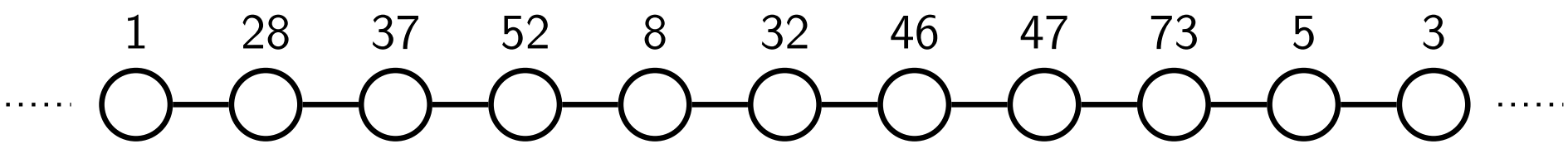
- 3 Coloring.
- 2 Coloring.
- Maximal Independent Set.

LCL Problems on Paths

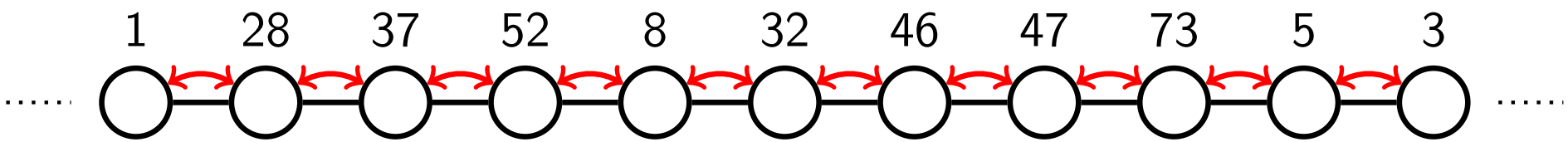


- 3 Coloring.
- 2 Coloring.
- Maximal Independent Set.
- Independent Set.

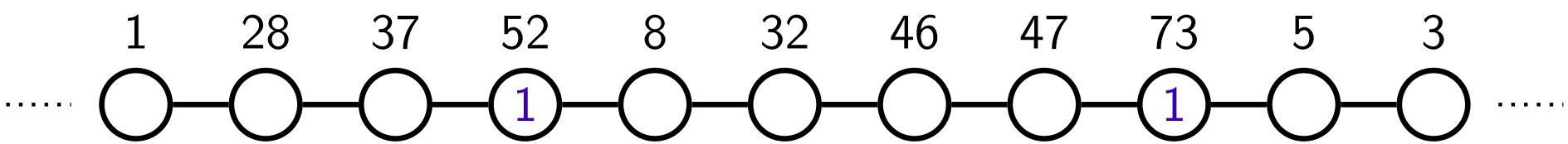
3 Coloring a Path



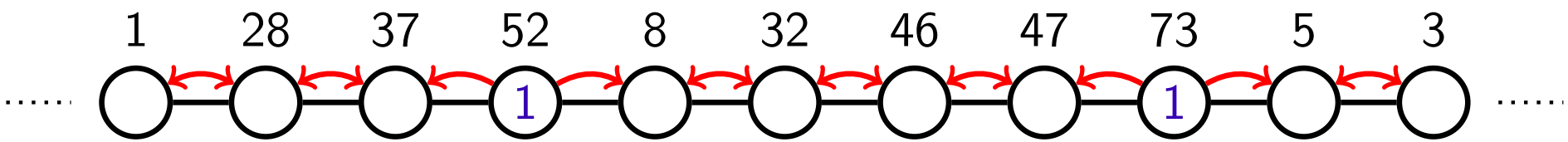
3 Coloring a Path



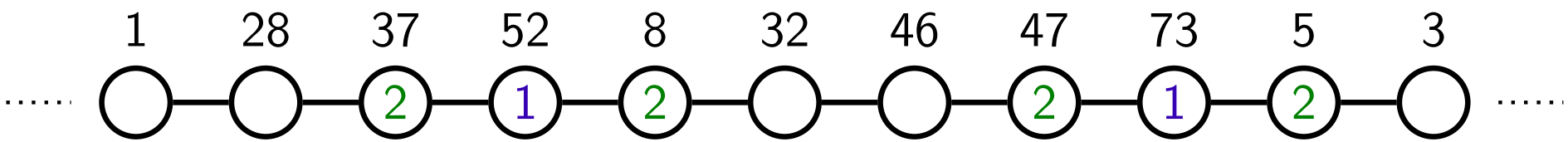
3 Coloring a Path



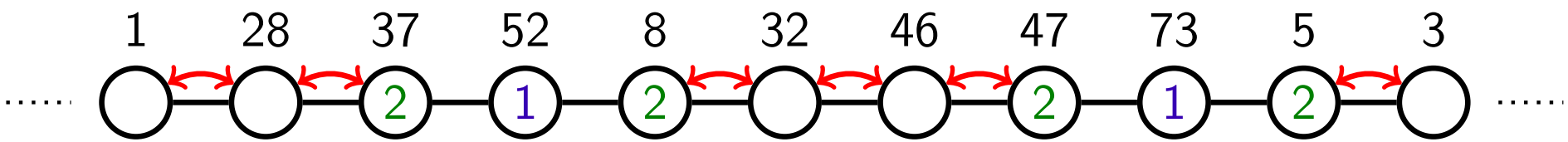
3 Coloring a Path



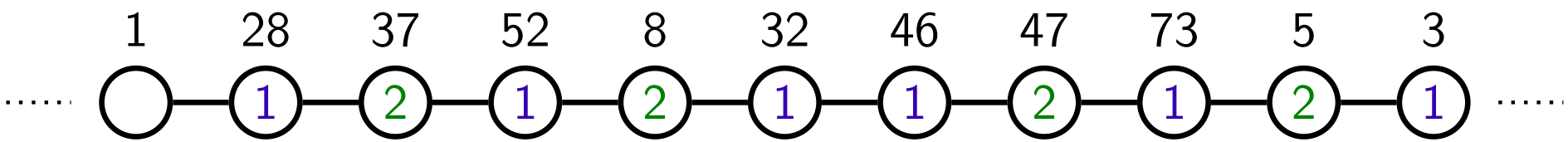
3 Coloring a Path



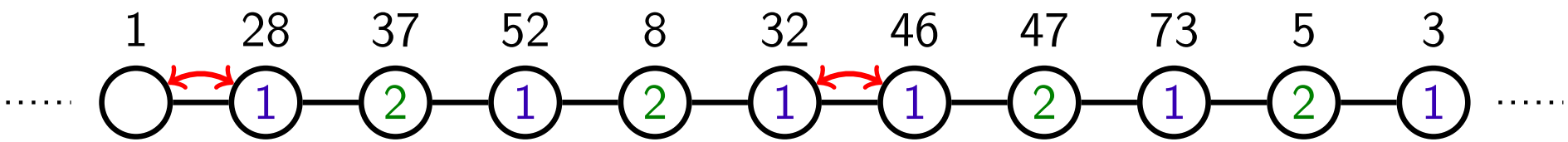
3 Coloring a Path



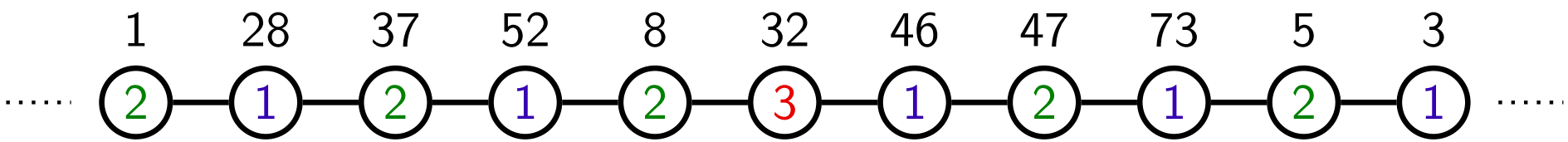
3 Coloring a Path



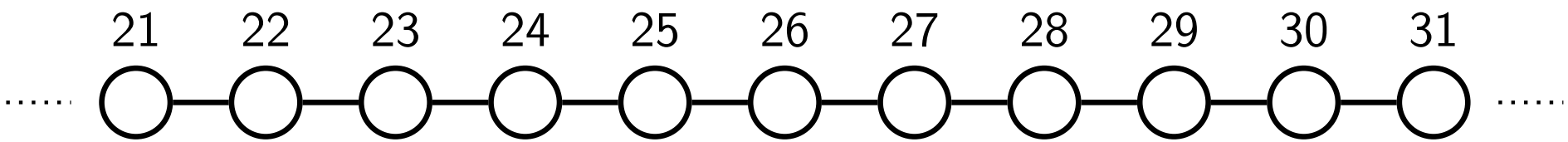
3 Coloring a Path



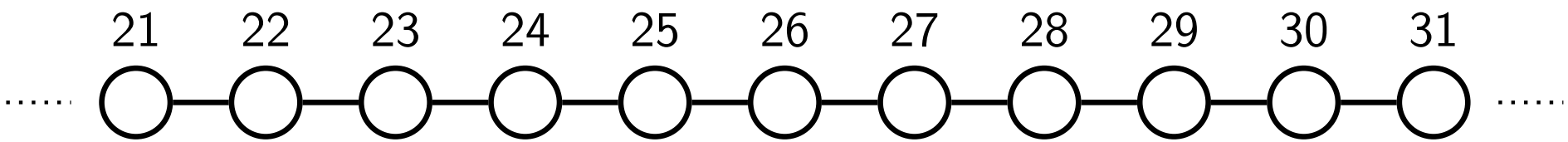
3 Coloring a Path



3 Coloring a Path



3 Coloring a Path



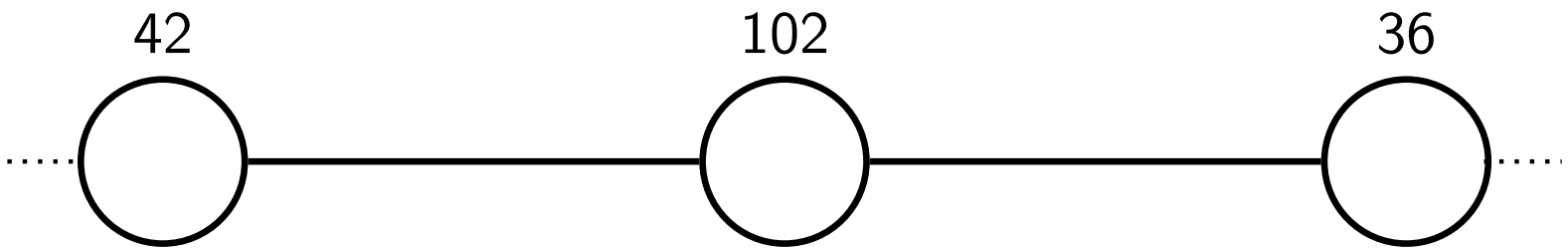
Worst case communication complexity : $\Theta(n)$.

3-coloring in $O(\log^* n)$ Communications

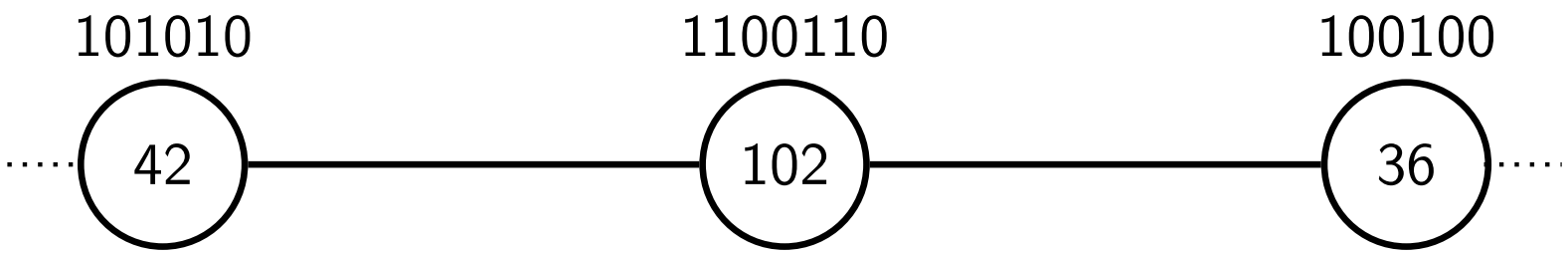
Cole, Vishkin (1986)

There exists a LCL algorithm to 3-color a path in $O(\log^* n)$ communications.

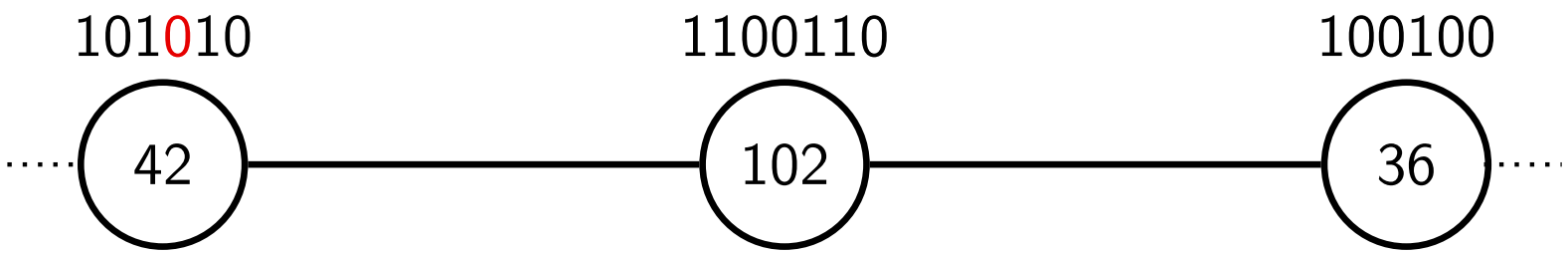
From n colors to $\log n$ colors



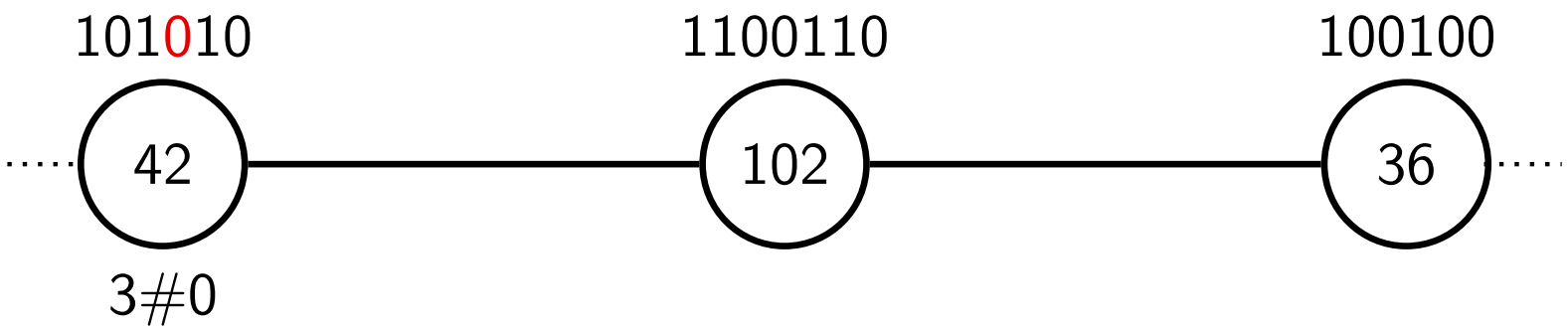
From n colors to $\log n$ colors



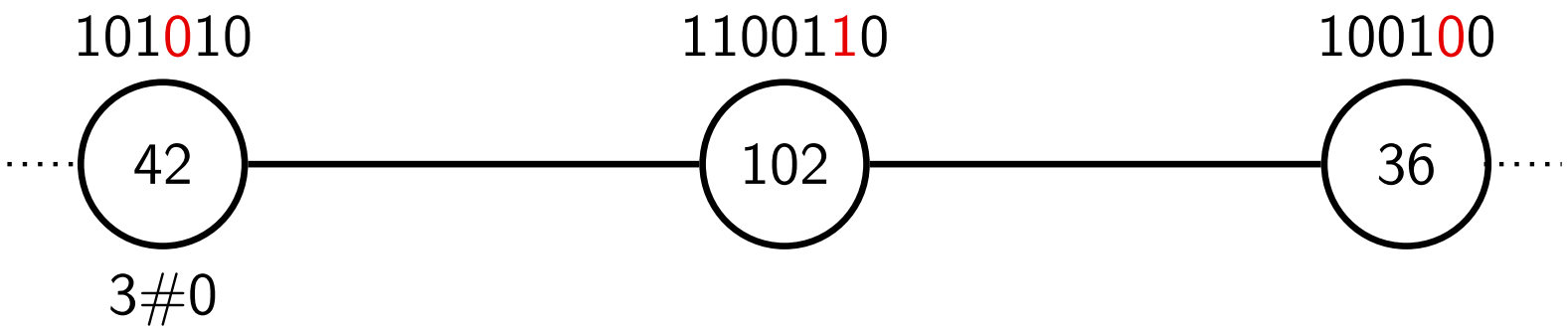
From n colors to $\log n$ colors



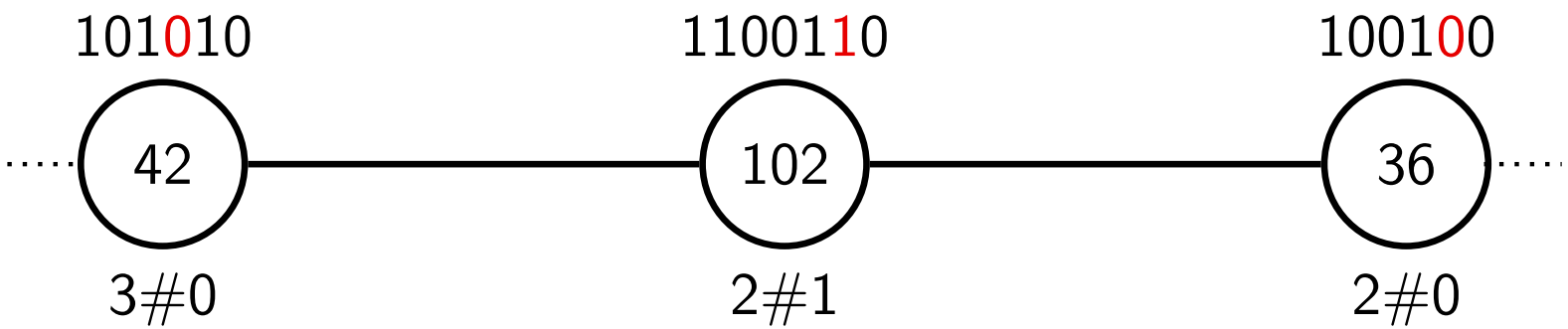
From n colors to $\log n$ colors



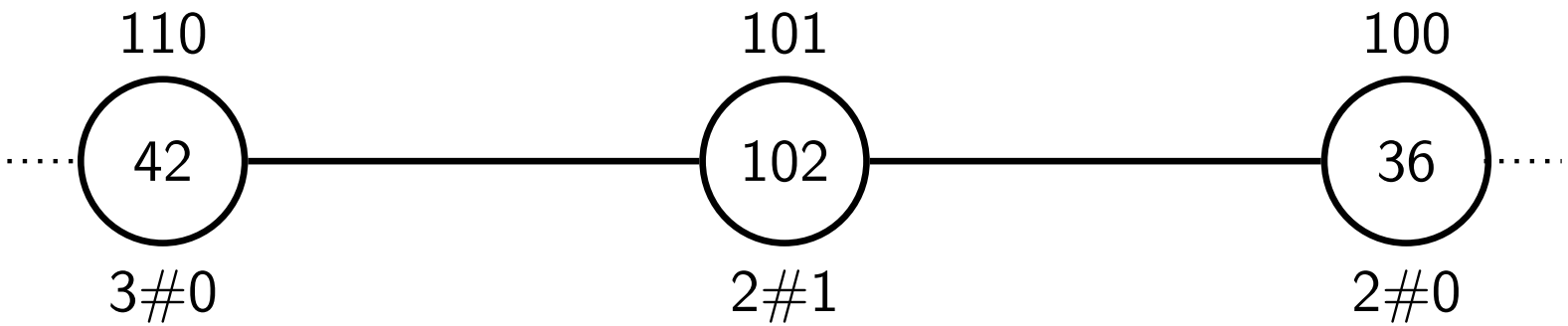
From n colors to $\log n$ colors



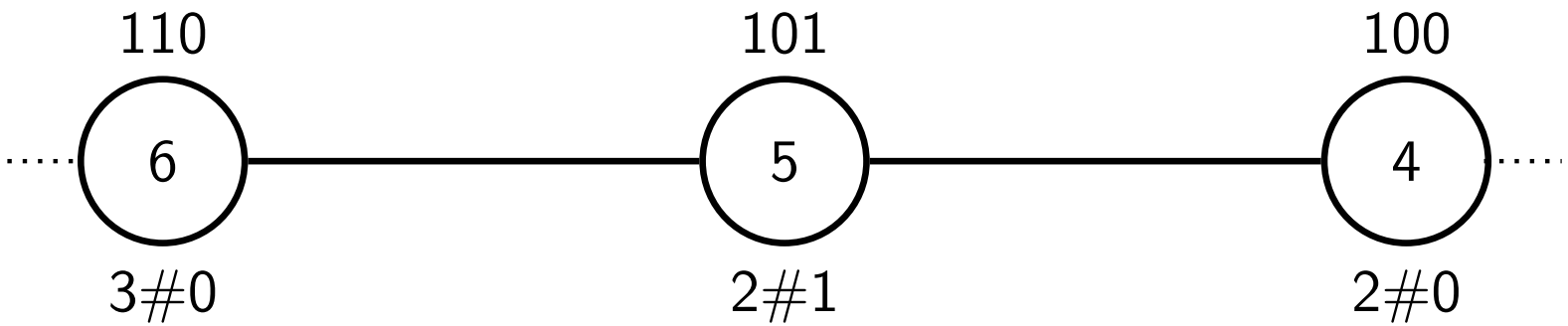
From n colors to $\log n$ colors



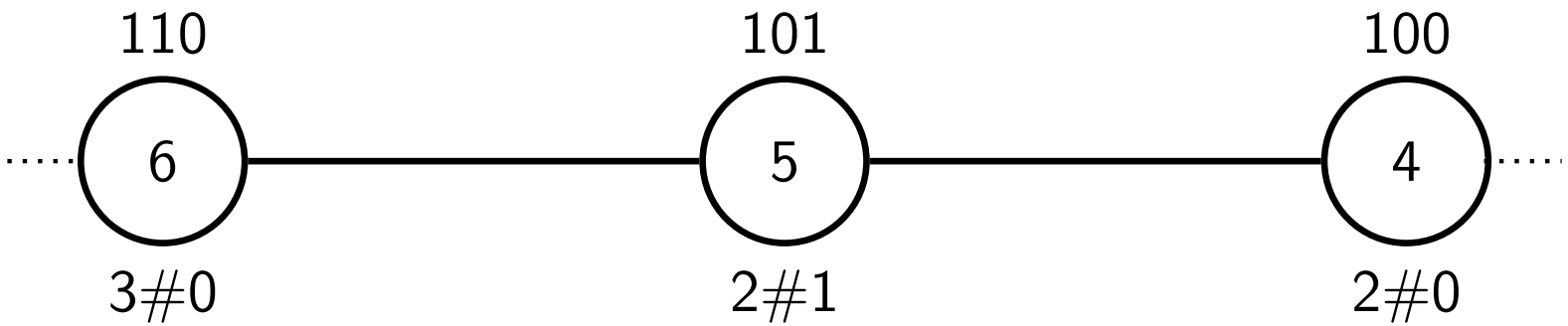
From n colors to $\log n$ colors



From n colors to $\log n$ colors

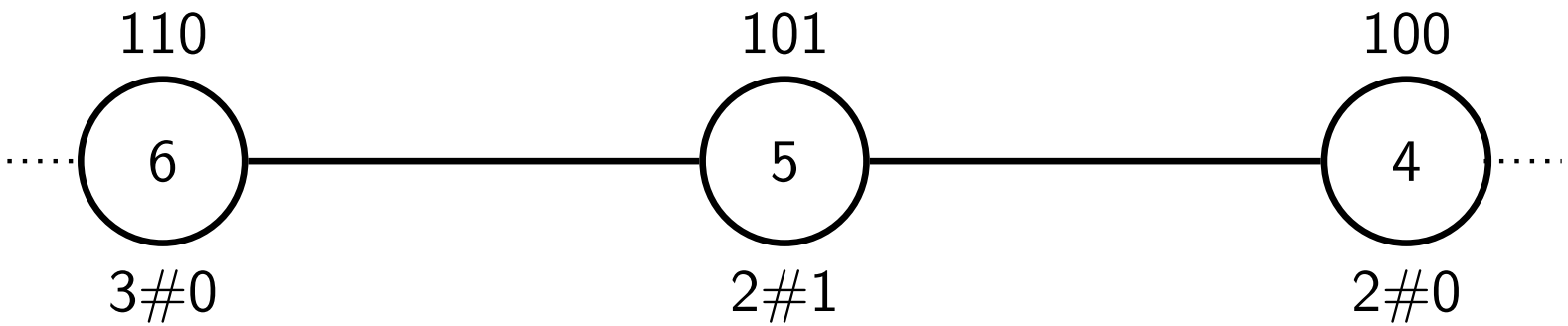


From n colors to $\log n$ colors



n colors $\Rightarrow \log n$ bits $\Rightarrow 2 \log n$ new colors $\Rightarrow \log \log n + 1$ bits

From n colors to $\log n$ colors



n colors $\Rightarrow \log n$ bits $\Rightarrow 2 \log n$ new colors $\Rightarrow \log \log n + 1$ bits

After $\log^* n$ iterations, $O(1)$ bits.

Coloration Lower Bound

Linial (1992)

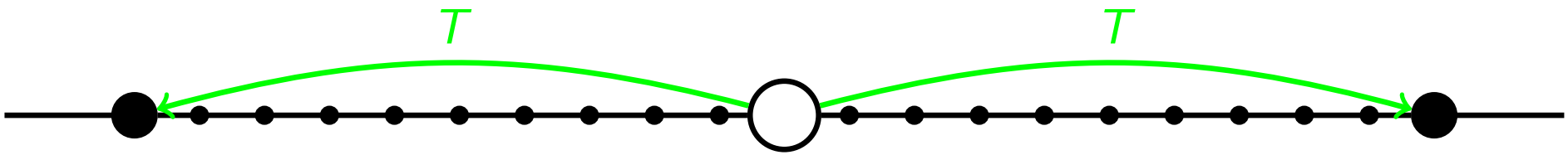
An algorithm which colors the n-cycle with three colors requires time at least $\frac{1}{2}(\log^* n - 3)$. The same bound holds also for randomized algorithms.

Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.

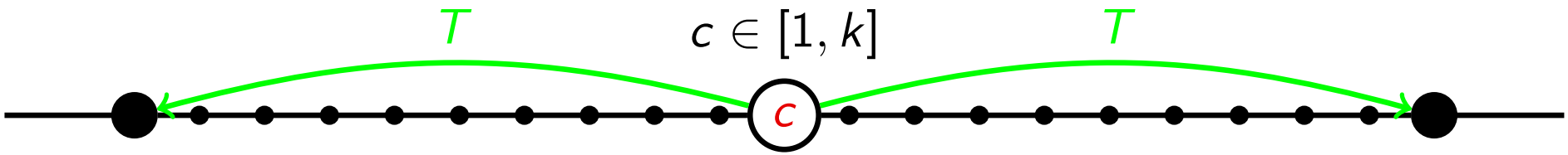
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.



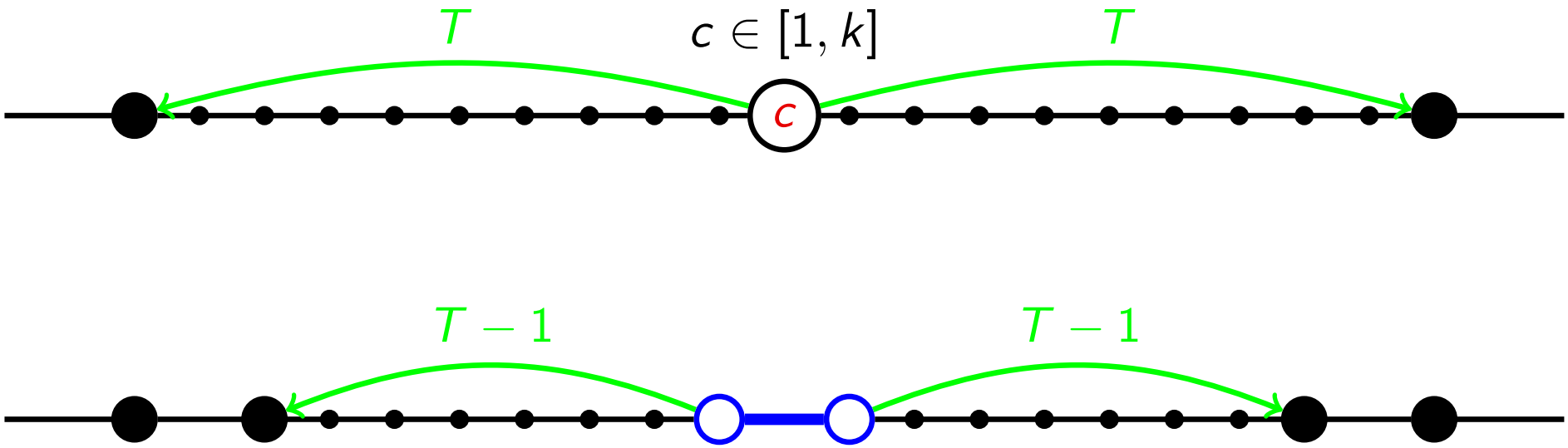
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.



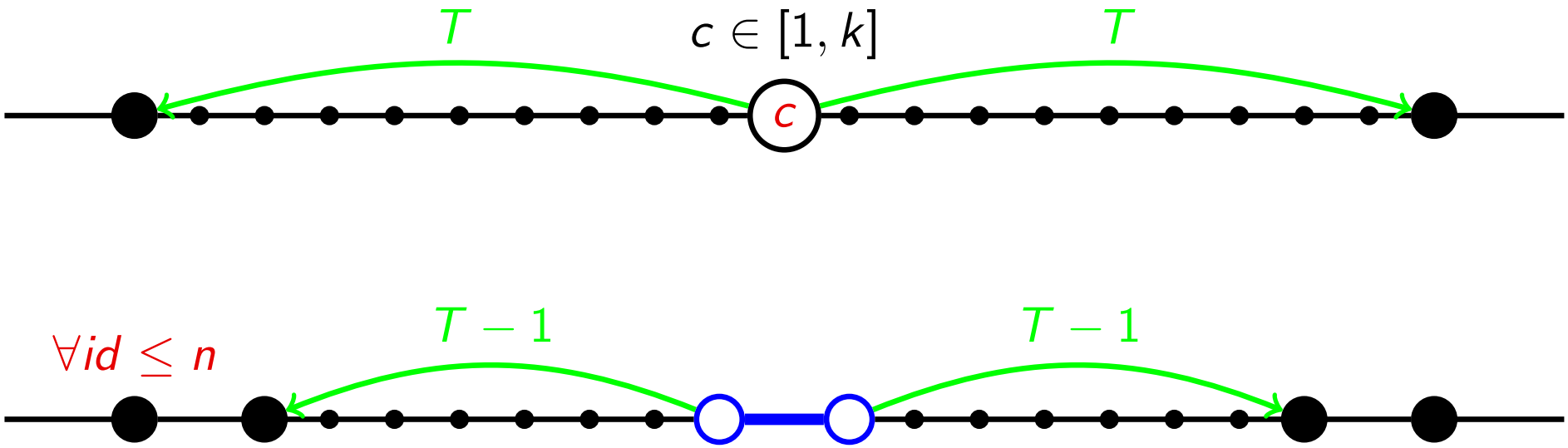
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.



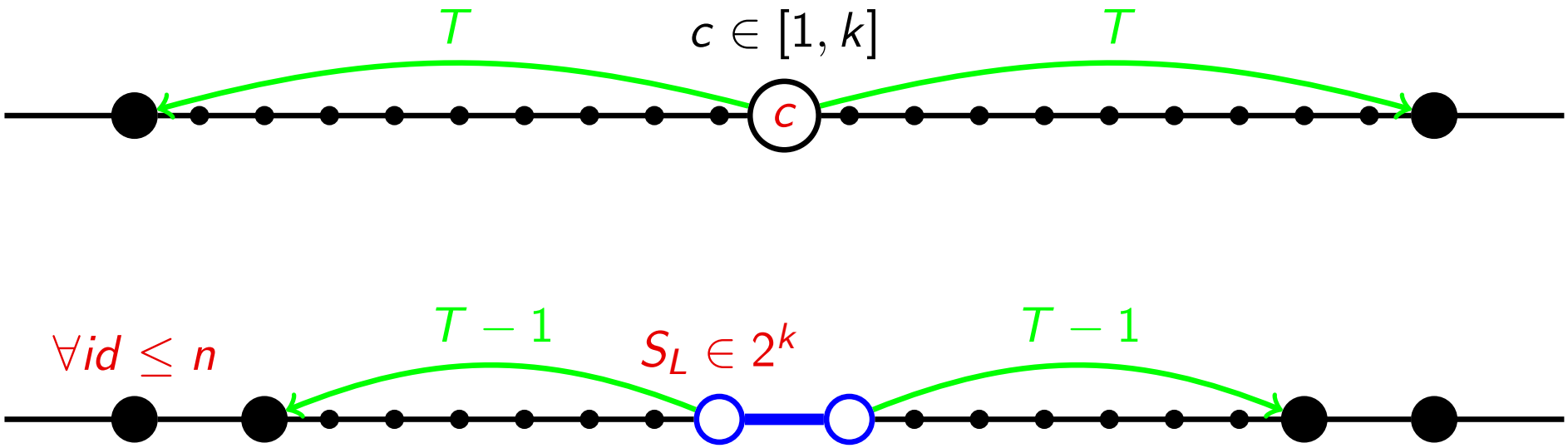
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.



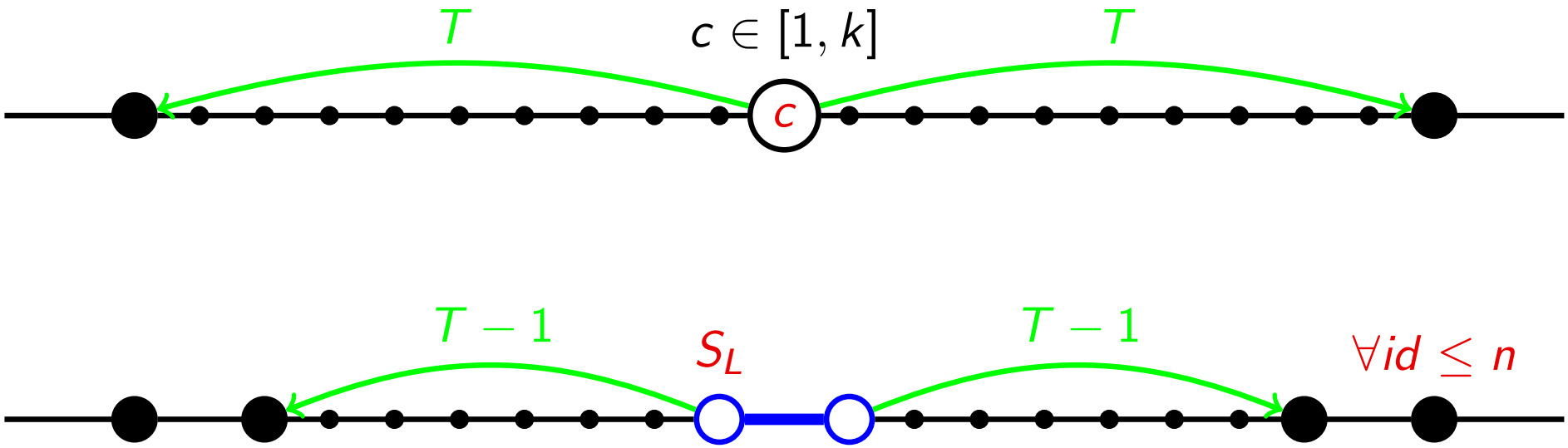
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.



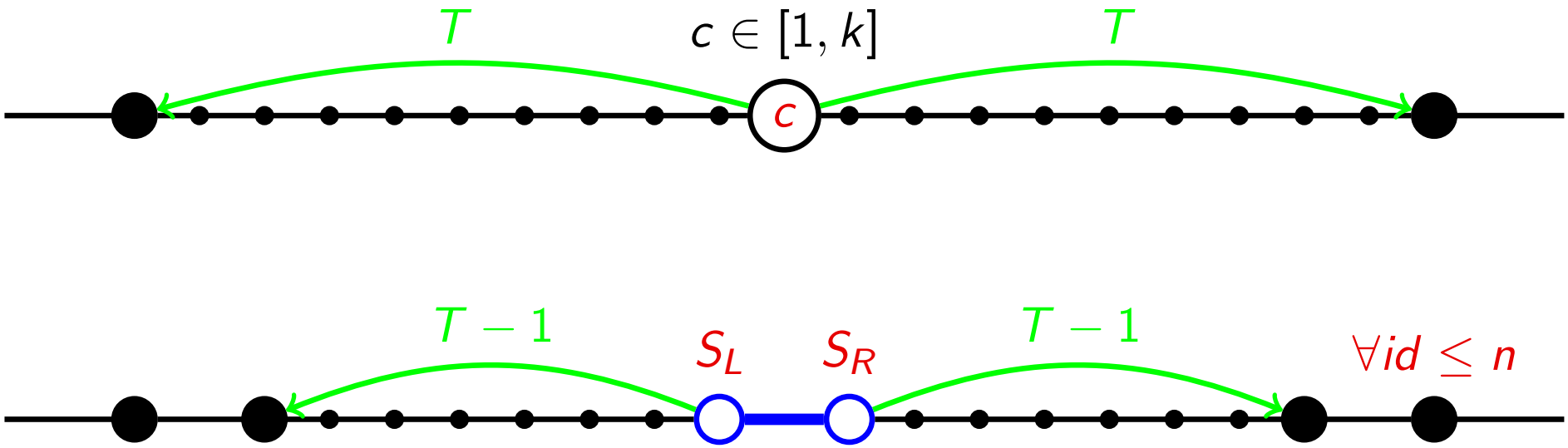
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.



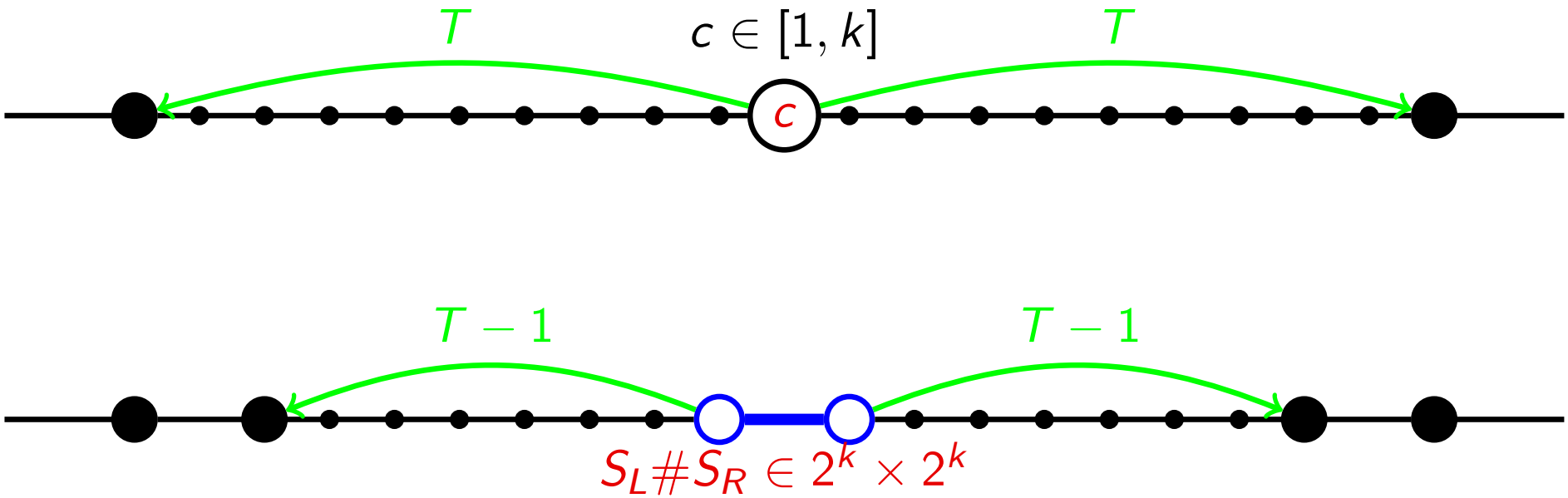
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.



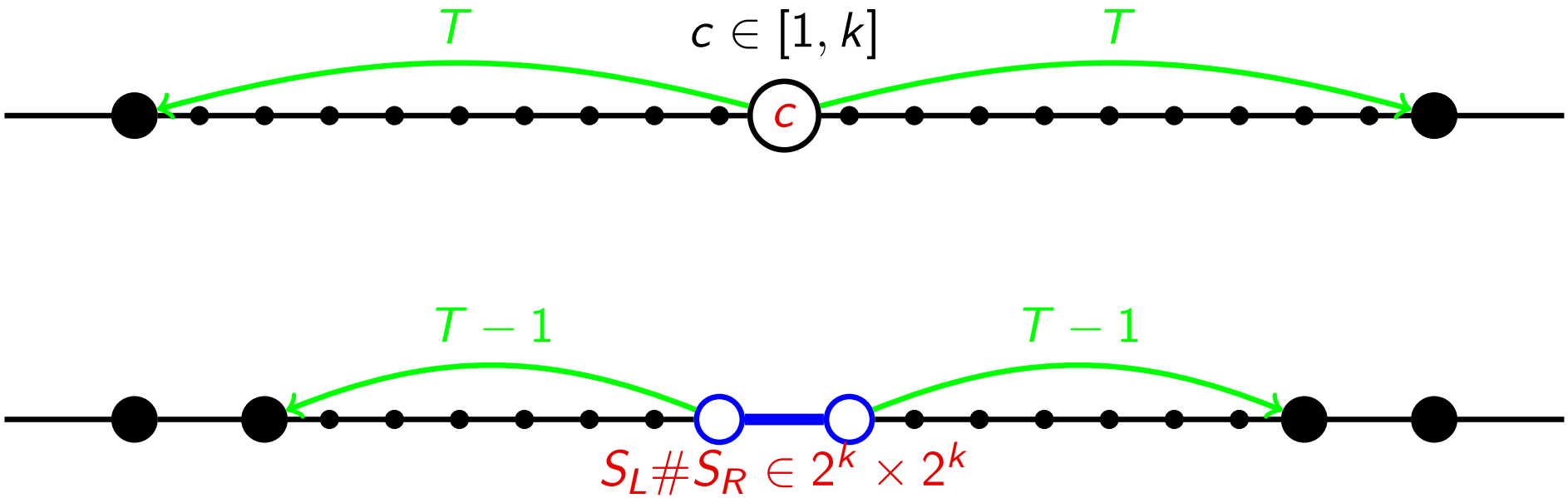
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.



Speed up Algorithm

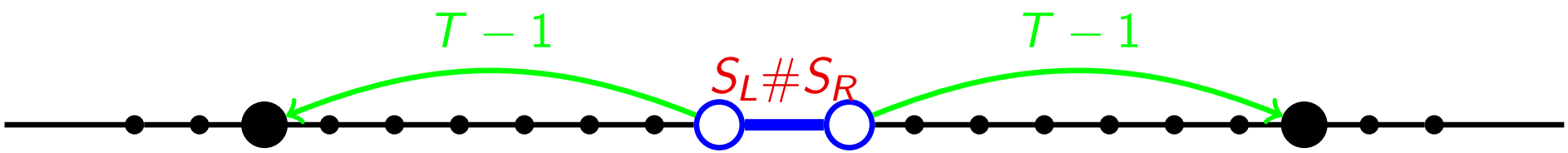
A : algorithm that k-colors nodes in T rounds.



$$S_L \cap S_R = \emptyset$$

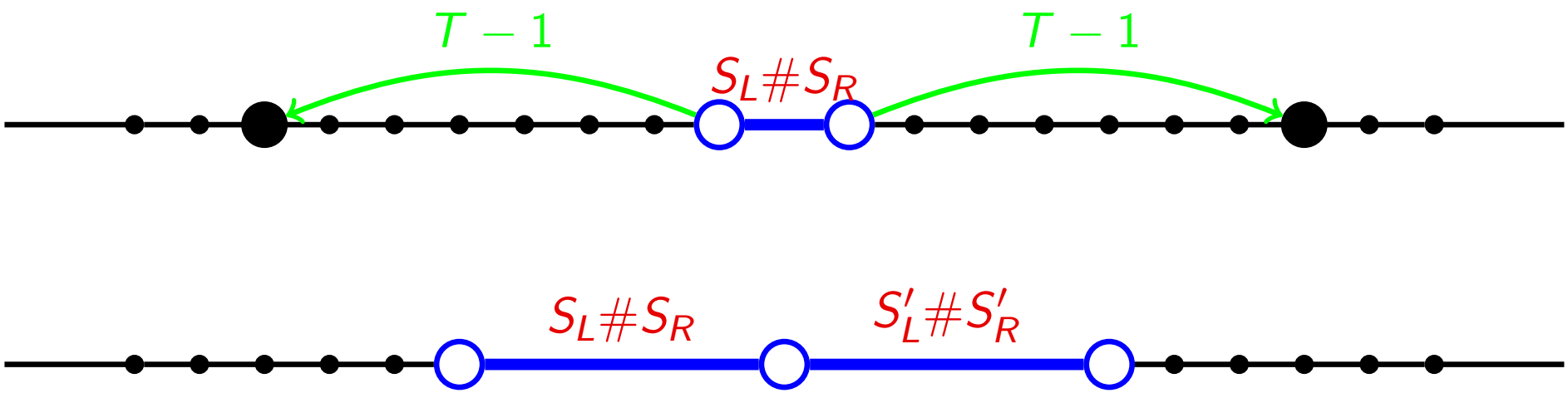
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.



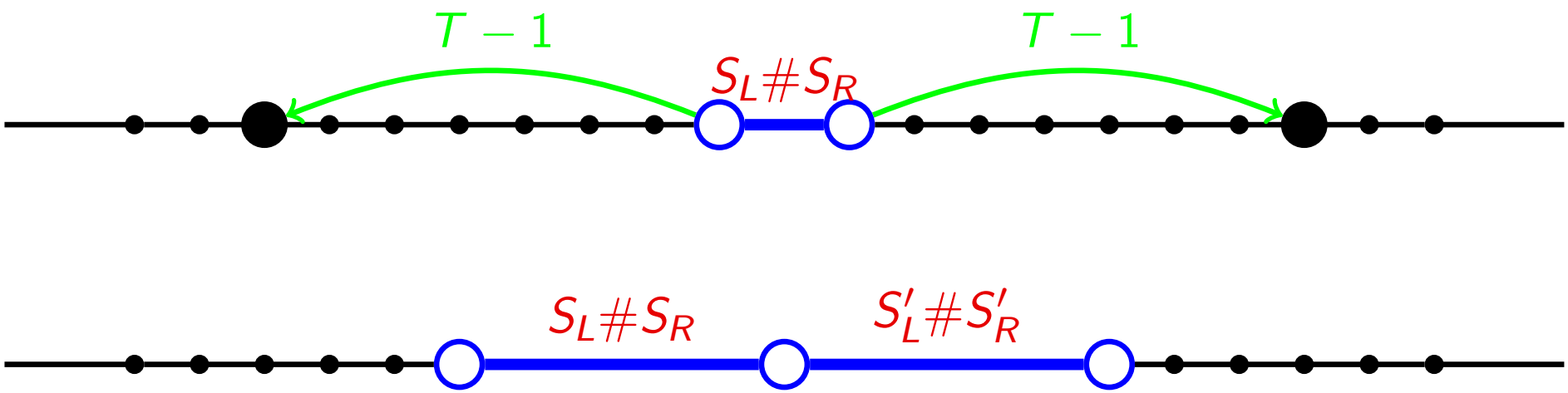
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.



Speed up Algorithm

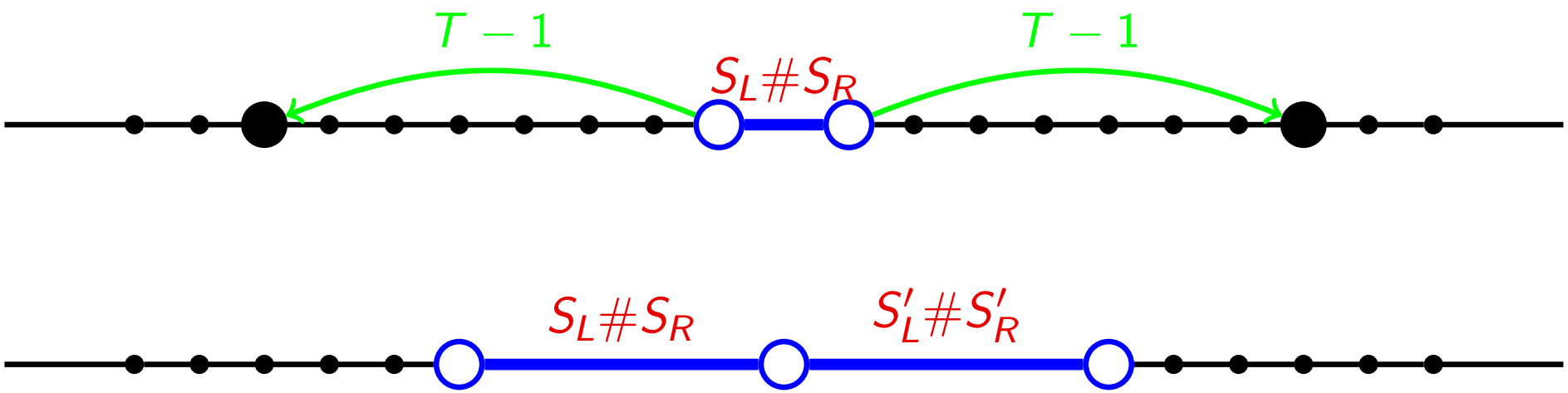
A : algorithm that k-colors nodes in T rounds.



$$S_L \cap S_R = \emptyset \ \& \ S'_L \cap S'_R = \emptyset$$

Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.

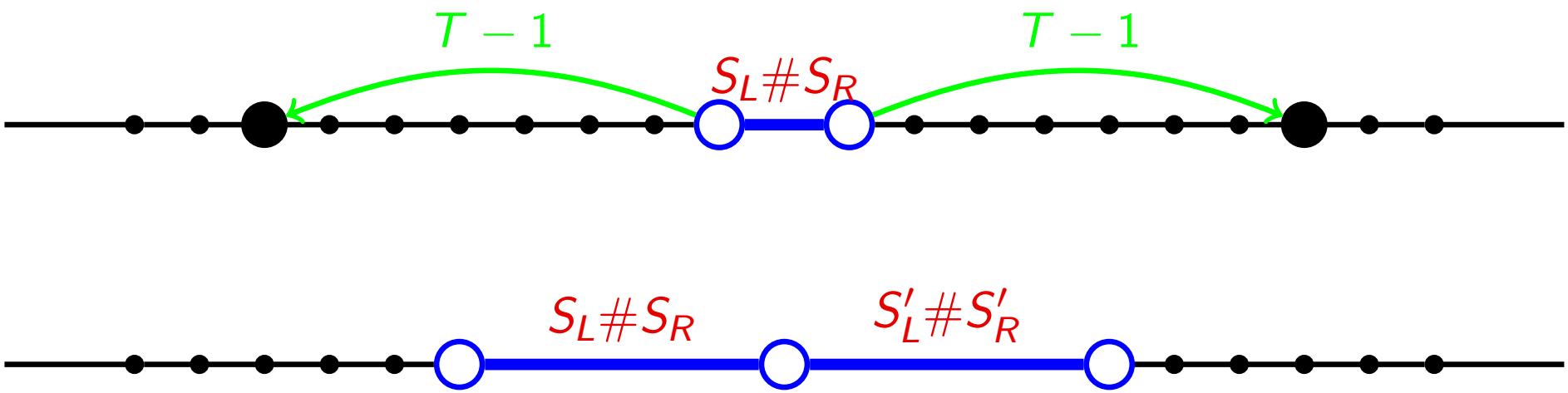


$$S_L \cap S_R = \emptyset \ \& \ S'_L \cap S'_R = \emptyset$$

$$S'_L \cap S_R \neq \emptyset$$

Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.



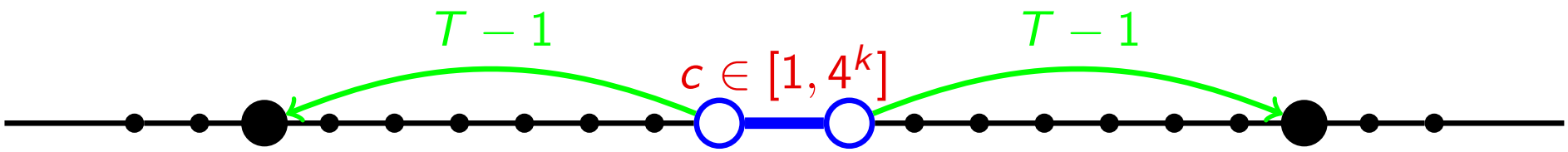
$$S_L \cap S_R = \emptyset \ \& \ S'_L \cap S'_R = \emptyset$$

$$S'_L \cap S_R \neq \emptyset$$

$$S_L \# S_R \neq S'_L \# S'_R$$

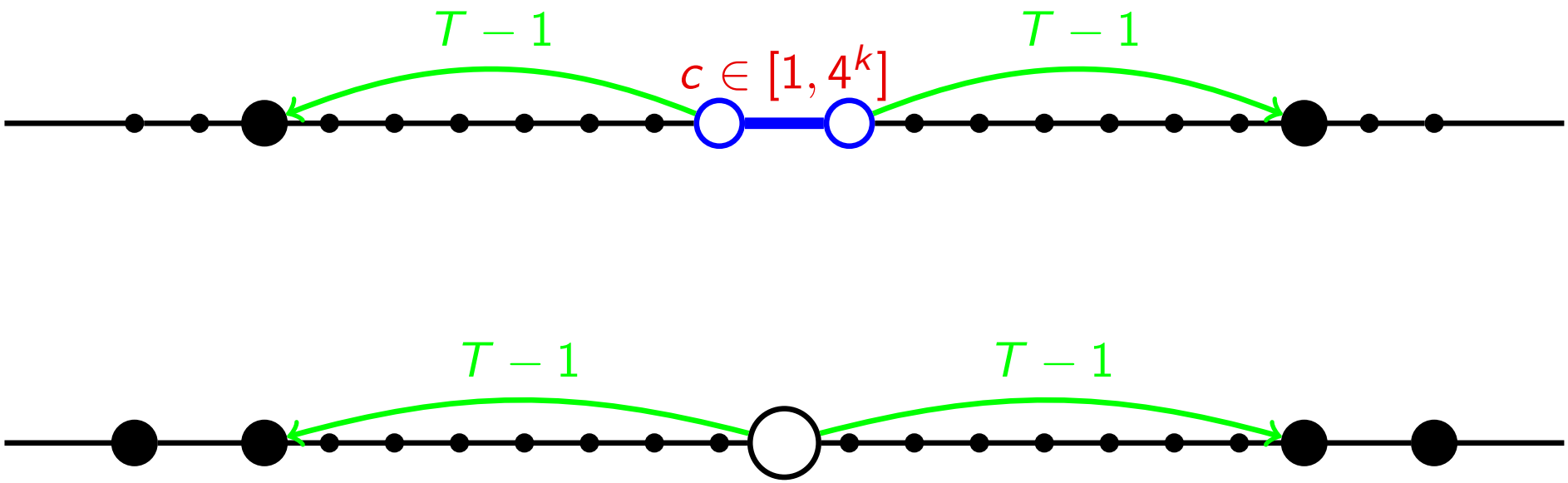
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.
A_1 : algorithm that 4^k-colors edges in $T - 1$ rounds.



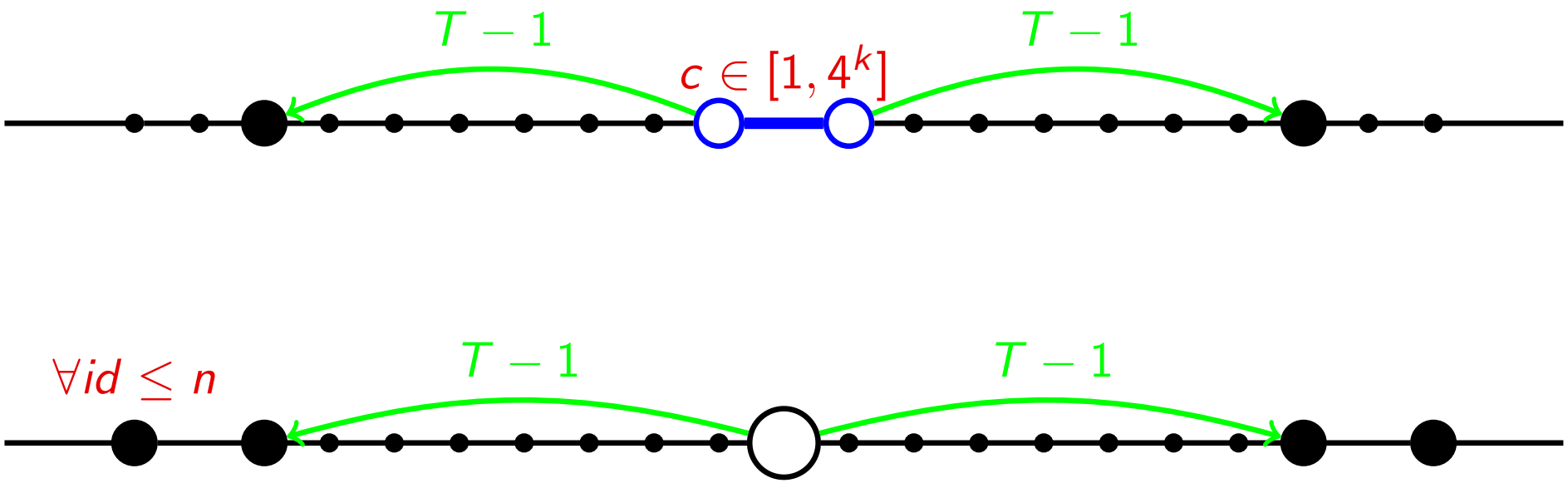
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.
A_1 : algorithm that 4^k-colors edges in $T - 1$ rounds.



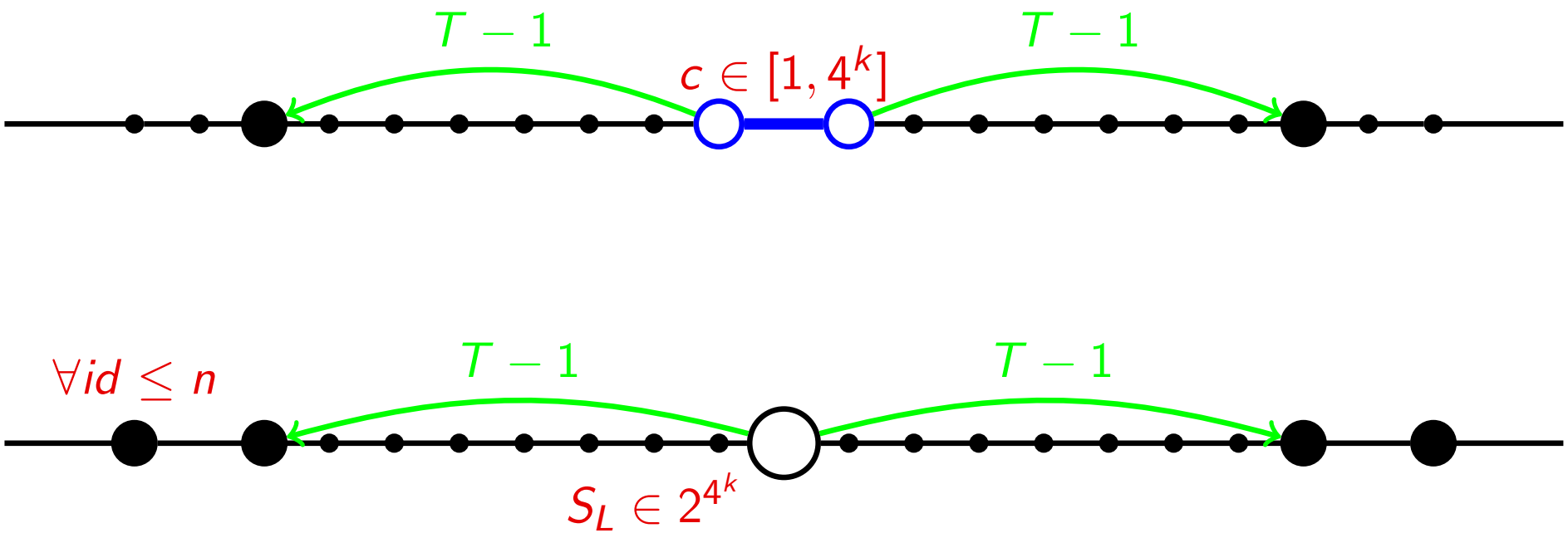
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.
A_1 : algorithm that 4^k-colors edges in $T - 1$ rounds.



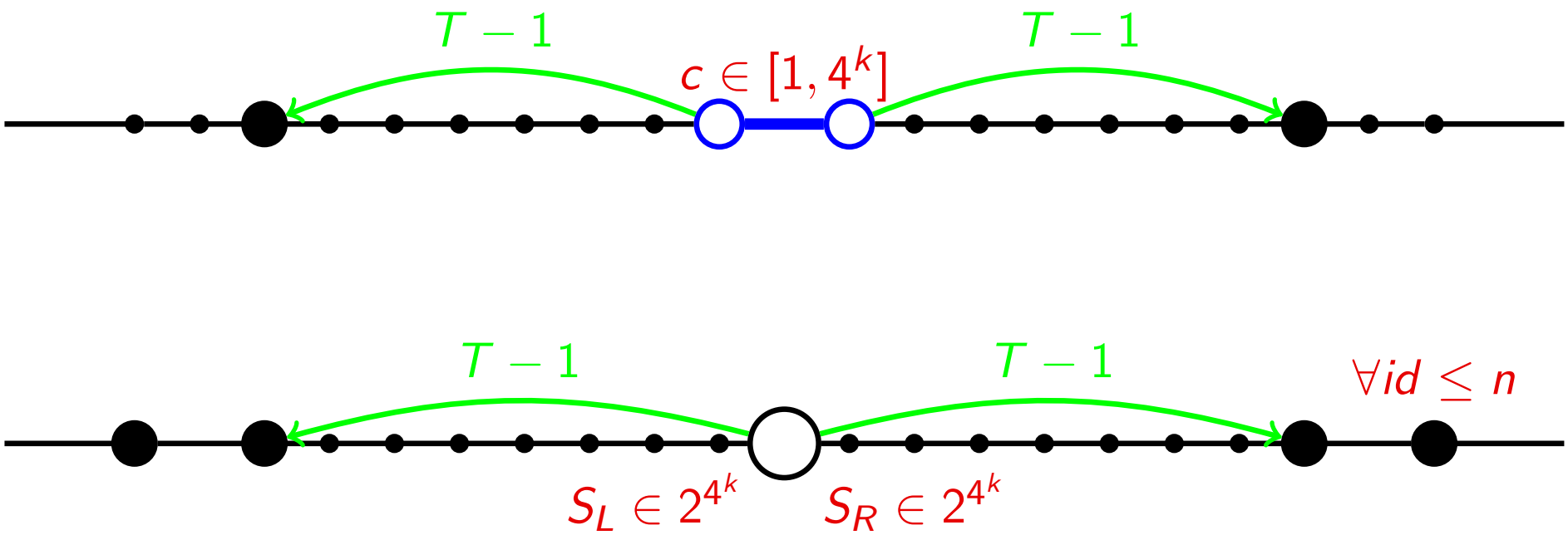
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.
A_1 : algorithm that 4^k-colors edges in $T - 1$ rounds.



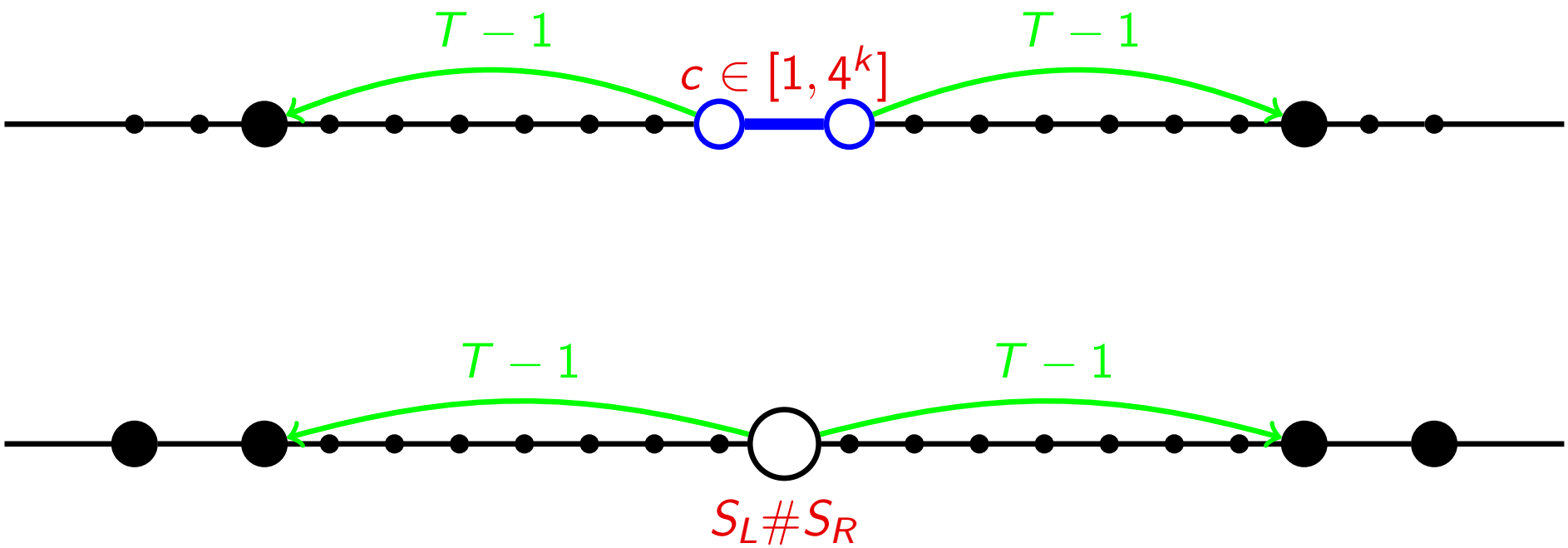
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.
A_1 : algorithm that 4^k-colors edges in $T - 1$ rounds.



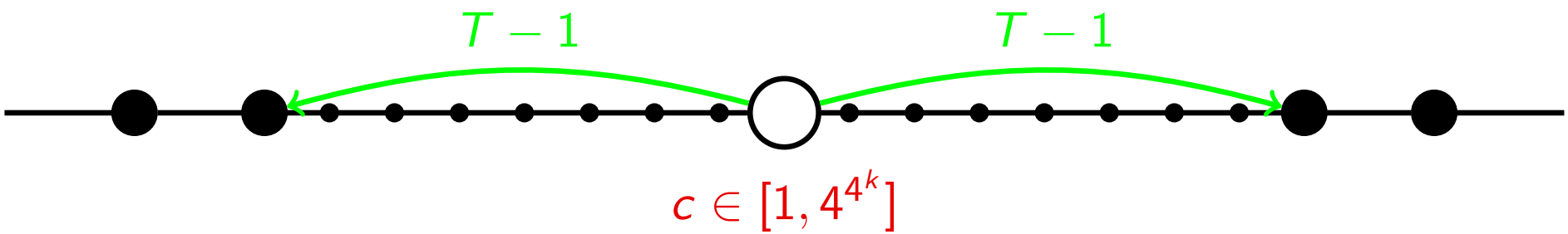
Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.
A_1 : algorithm that 4^k-colors edges in $T - 1$ rounds.



Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.
A_1 : algorithm that 4^k-colors edges in $T-1$ rounds.
A_2 : algorithm that 4^{4^k}-colors nodes in $T-1$ rounds.



Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.
A_1 : algorithm that 4^k-colors edges in $T - 1$ rounds.
A_2 : algorithm that 4^{4^k}-colors nodes in $T - 1$ rounds.
...
$A_{\log^* n}$: algorithm that n-colors nodes in $T - \frac{log^* n}{2}$ rounds.

Speed up Algorithm

A : algorithm that k-colors nodes in T rounds.
A_1 : algorithm that 4^k-colors edges in $T - 1$ rounds.
A_2 : algorithm that 4^{4^k}-colors nodes in $T - 1$ rounds.
...
$A_{\log^* n}$: algorithm that n-colors nodes in $T - \frac{log^* n}{2}$ rounds.
$\Rightarrow$ any 3-coloring algorithm needs at least $\frac{log^* n}{2}$ rounds.

Independent Set at some Distance

- From 3−coloring to Maximal Independent Set.

Independent Set at some Distance

- From 3−coloring to Maximal Independent Set.
  Nodes at distance 2 or 3.

Independent Set at some Distance

- From 3−coloring to Maximal Independent Set.
  Nodes at distance 2 or 3.
- Reiterate on the Independent Node.
  Nodes at distance between 4 and 9.

Independent Set at some Distance

- From 3−coloring to Maximal Independent Set.
  Nodes at distance 2 or 3.
- Reiterate on the Independent Node.
  Nodes at distance between 4 and 9.
  ...
- Reiterate k times.
  Nodes at distance between 2^k and 3^k.

Independent Set at some Distance

- From 3−coloring to Maximal Independent Set.
  Nodes at distance 2 or 3.
- Reiterate on the Independent Node.
  Nodes at distance between 4 and 9.
  ...
- Reiterate k times.
  Nodes at distance between 2^k and 3^k.
- Cut the segments in sizes between s and $2s$ for some $s \leq 2^k$.

Independent Set at some Distance

- From 3−coloring to Maximal Independent Set.
  Nodes at distance 2 or 3.
- Reiterate on the Independent Node.
  Nodes at distance between 4 and 9.
  ...
- Reiterate k times.
  Nodes at distance between 2^k and 3^k.
- Cut the segments in sizes between s and $2s$ for some $s \leq 2^k$.

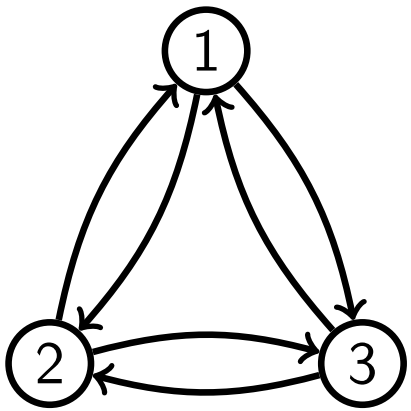
Time complexity : $O(s \log^* n)$.

Transition Automata

Node : Sequence of outputs.
Edge : Connecting to an admitable next output.

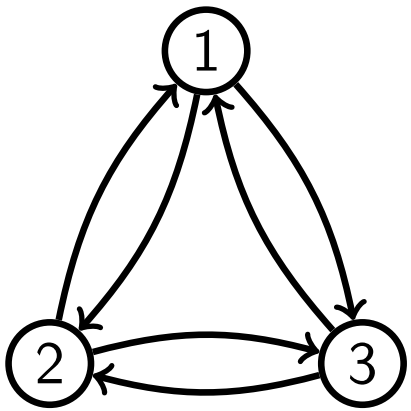
Transition Automata

Node : Sequence of outputs.
Edge : Connecting to an admitable next output.



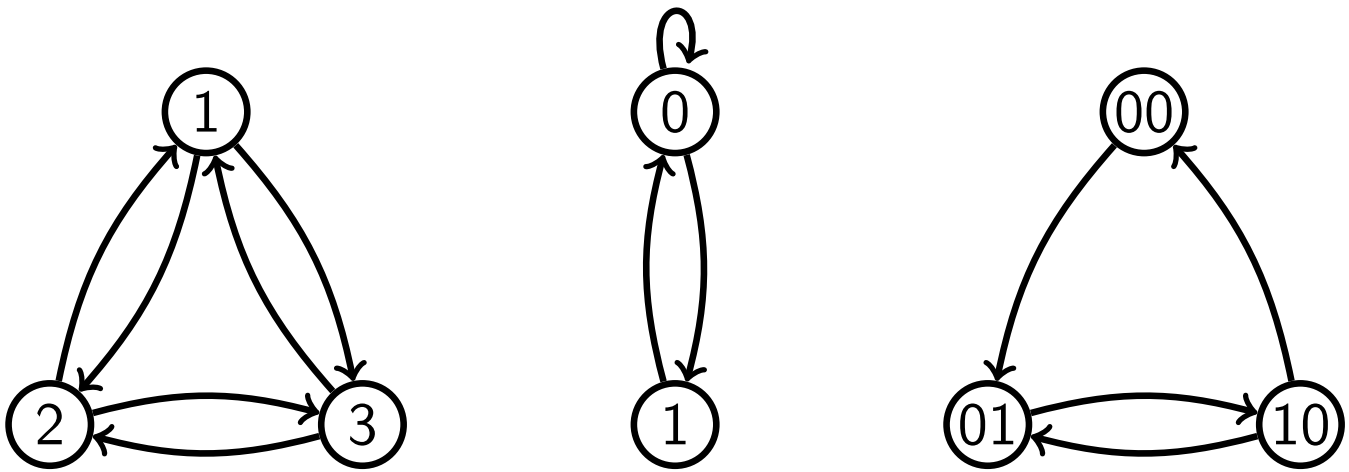
Transition Automata

Node : Sequence of outputs.
Edge : Connecting to an admitable next output.



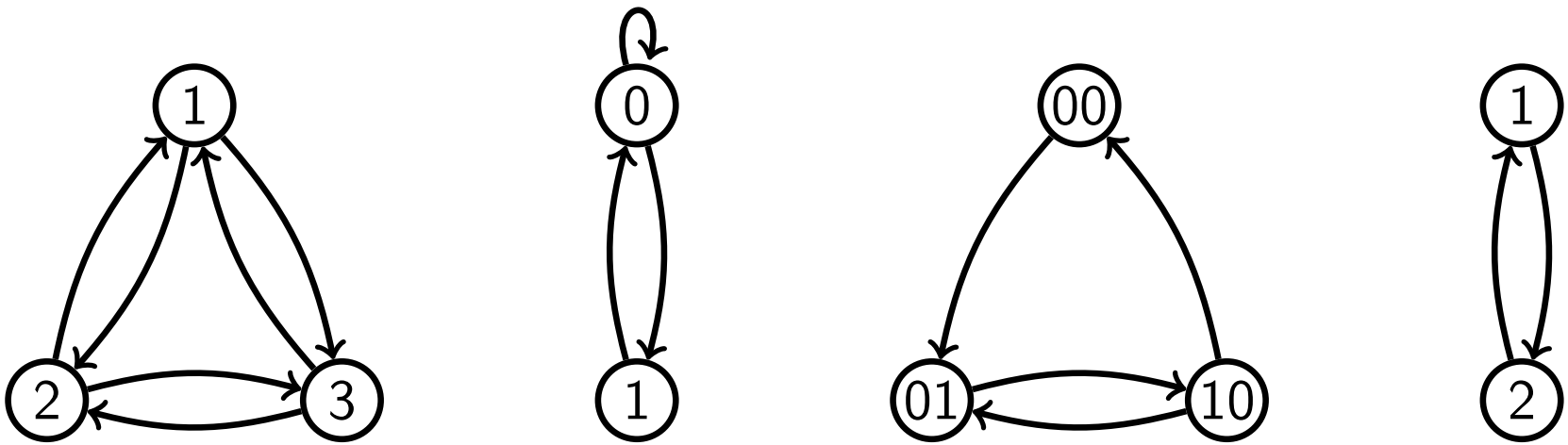
Transition Automata

Node : Sequence of outputs.
Edge : Connecting to an admitable next output.



Transition Automata

Node : Sequence of outputs.
Edge : Connecting to an admitable next output.



Complexity Separation on Paths

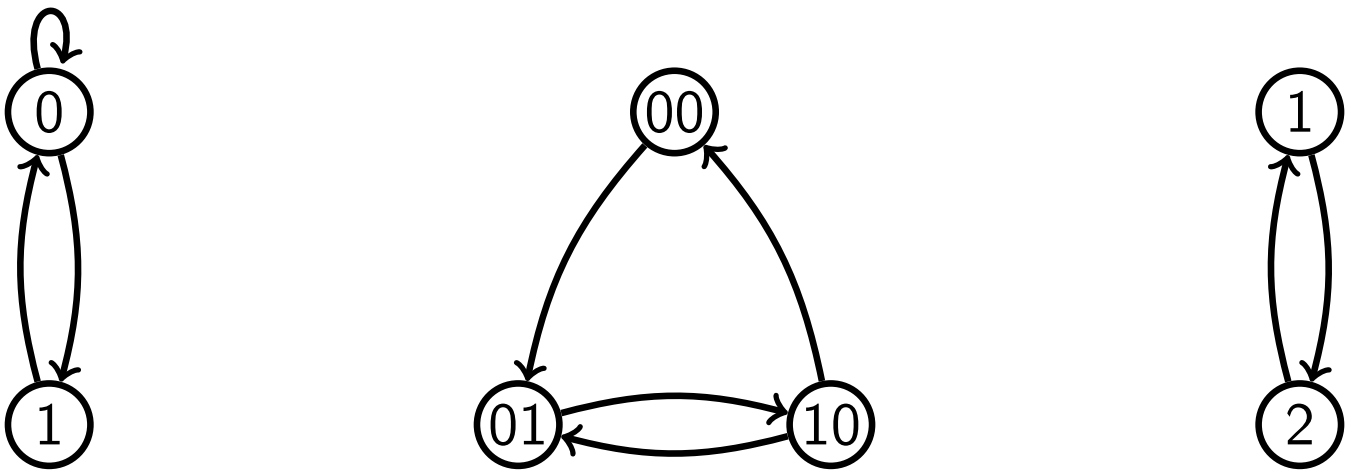
Naor, Sotckmeyer (1995)

If the input graph is an unlabeled path or cycle, the time complexity is decidable.
The different time complexities are $O(1)$, $\Theta(\log^* n)$ and $\Omega(n)$.

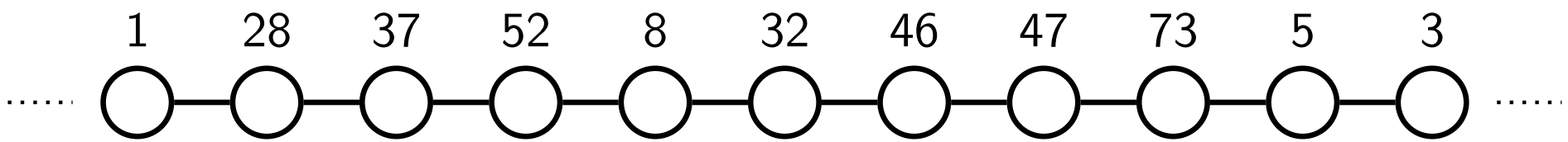
Complexity Separation on Paths

Naor, Sotckmeyer (1995)

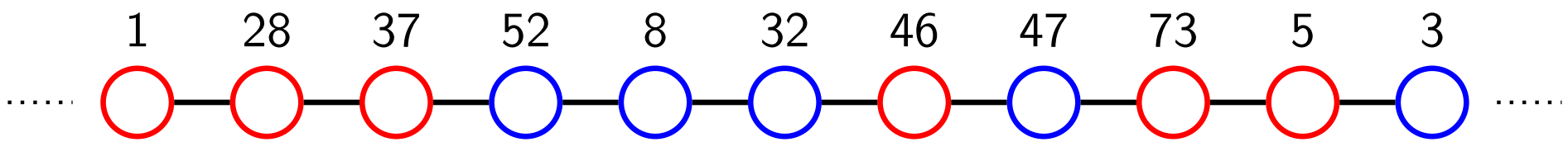
If the input graph is an unlabeled path or cycle, the time complexity is decidable.
The different time complexities are $O(1)$, $\Theta(\log^* n)$ and $\Omega(n)$.



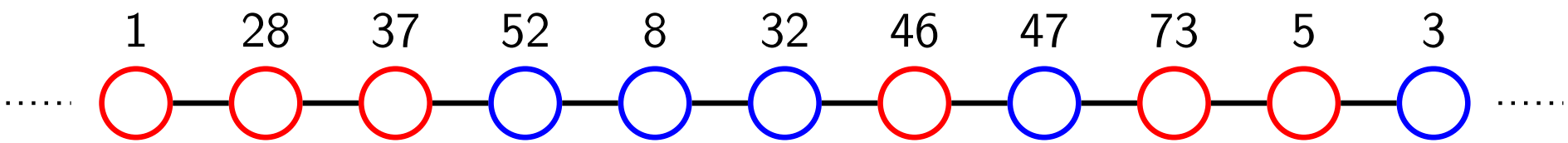
Problem on Paths with Inputs



Problem on Paths with Inputs

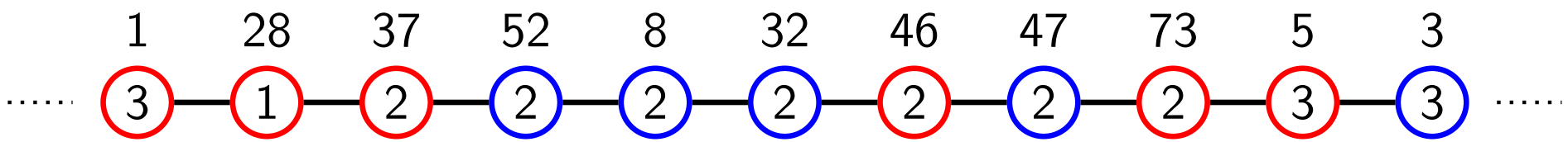


Problem on Paths with Inputs



- 3-color the red nodes.
- Carry the color through the blue nodes.

Problem on Paths with Inputs



- 3-color the red nodes.
- Carry the color through the blue nodes.

Complexity Separation on Paths with Inputs

Balliu, Brandt, Chang, Olivetti, Rabie, Suomela (2018)

For any LCL problem on cycle graphs, its complexity is either $\Omega(n)$ or $O(\log^* n)$. Moreover, there is an algorithm that decides whether the problem has complexity $\Omega(n)$ or $O(\log^* n)$ on cycle graphs; for the case the complexity is $O(\log^* n)$, the algorithm outputs a description of an $O(\log^* n)$-round deterministic LOCAL algorithm that solves it.

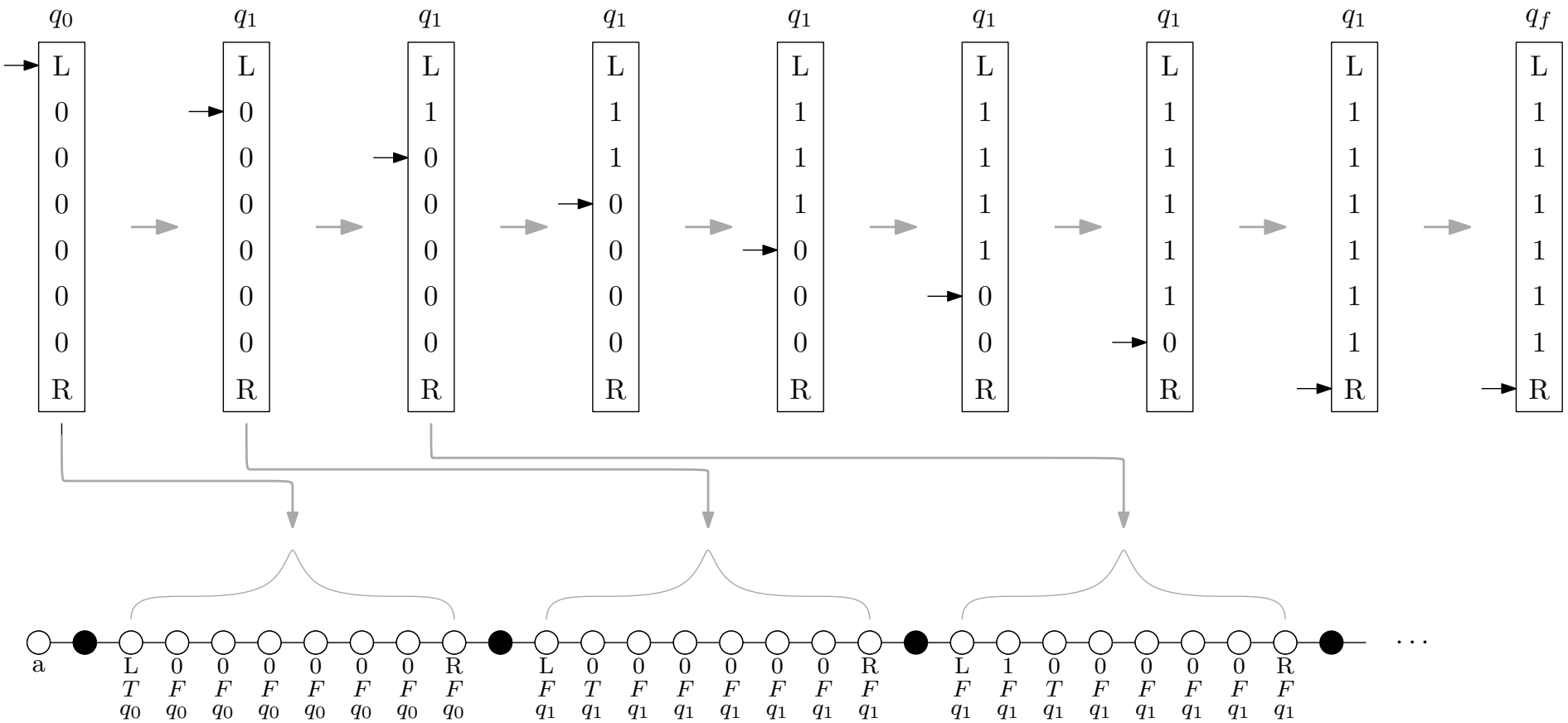
For any LCL problem on cycle graphs, its complexity is either $\Omega(\log^* n)$ or $O(1)$. Moreover, there is an algorithm that decides whether the problem has complexity $\Omega(\log^* n)$ or $O(1)$ on cycle graphs; for the case the complexity is $O(1)$, the algorithm outputs a description of an $O(1)$-round deterministic LOCAL algorithm that solves it.

Complexity Separation on Paths with Inputs

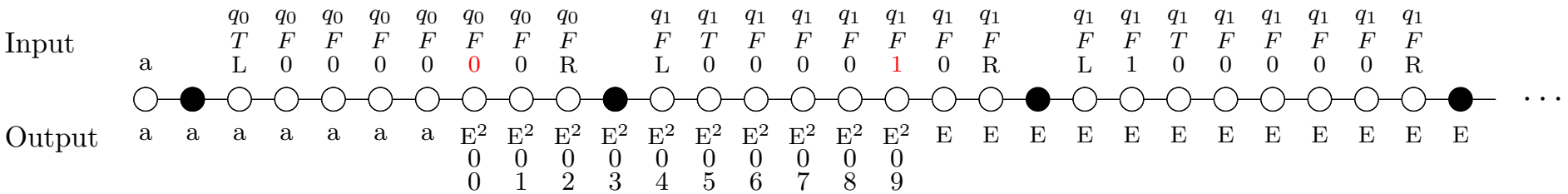
Balliu, Brandt, Chang, Olivetti, Rabie, Suomela (2018)

It is PSPACE-hard to distinguish whether a given LCL problem with input labels can be solved in $O(1)$ time or needs $\Omega(n)$ time on globally oriented path graphs.

Turing Machine Encoding



Error Detection



Complexity Separation on Trees

Balliu, Brandt, Chang, Olivetti, Rabie, Suomela (2018)

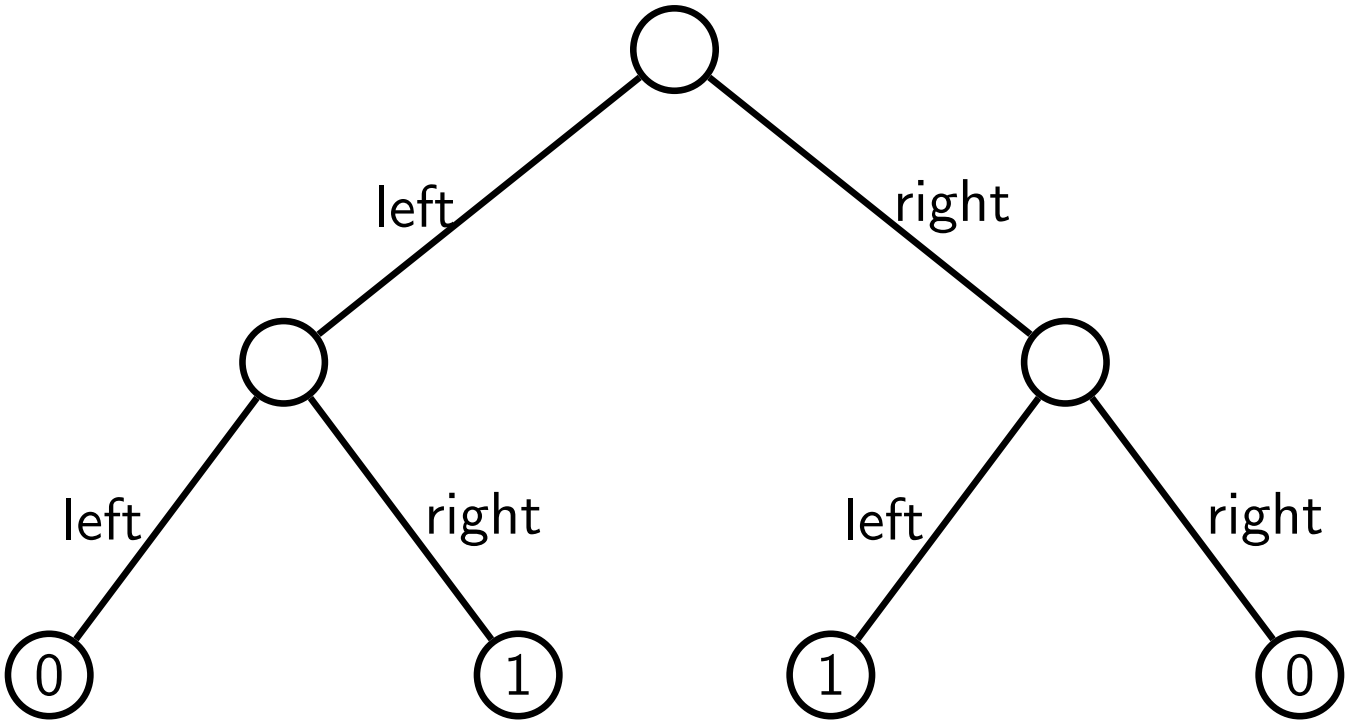
It is PSPACE-hard to distinguish whether a given LCL problem without input labels can be solved in $O(1)$ time or needs $\Omega(n)$ time on trees with degree $\Delta = 3$.

Encoding a Number in a Tree

Encoding $6{=}0110_2$

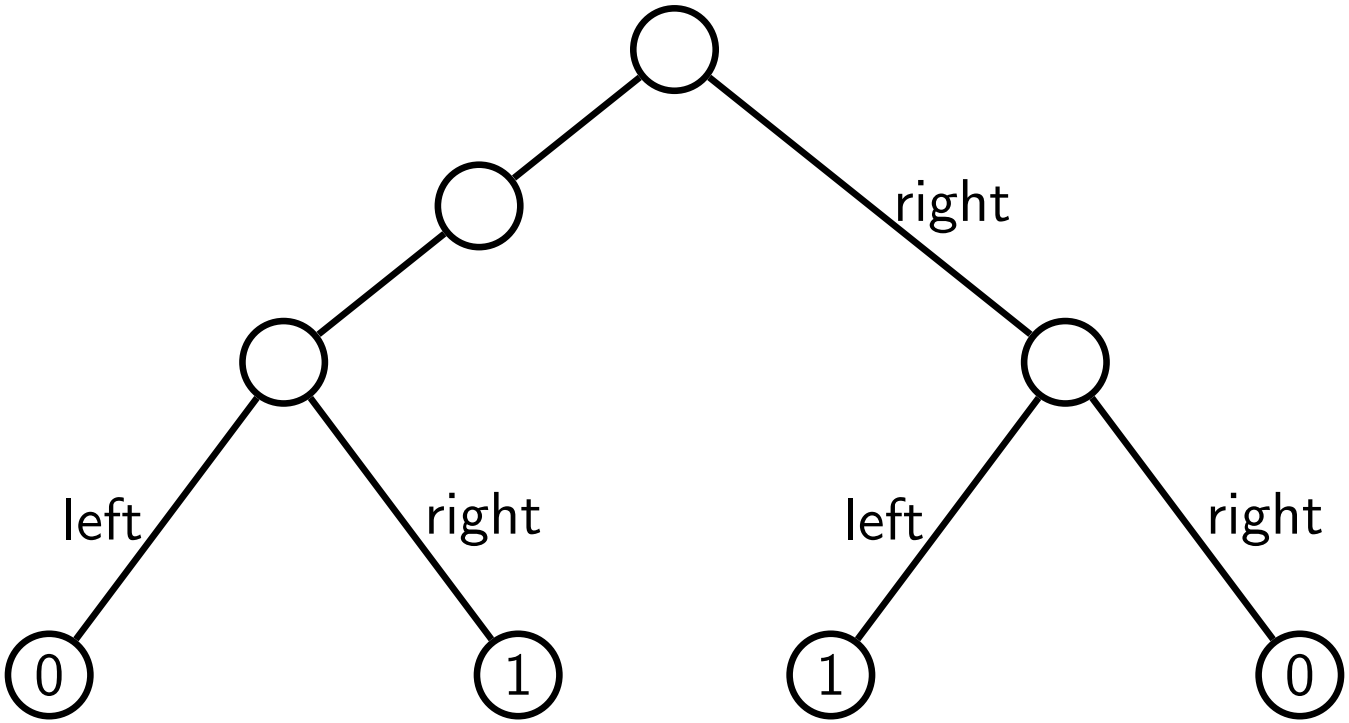
Encoding a Number in a Tree

Encoding $6{=}0110_2$



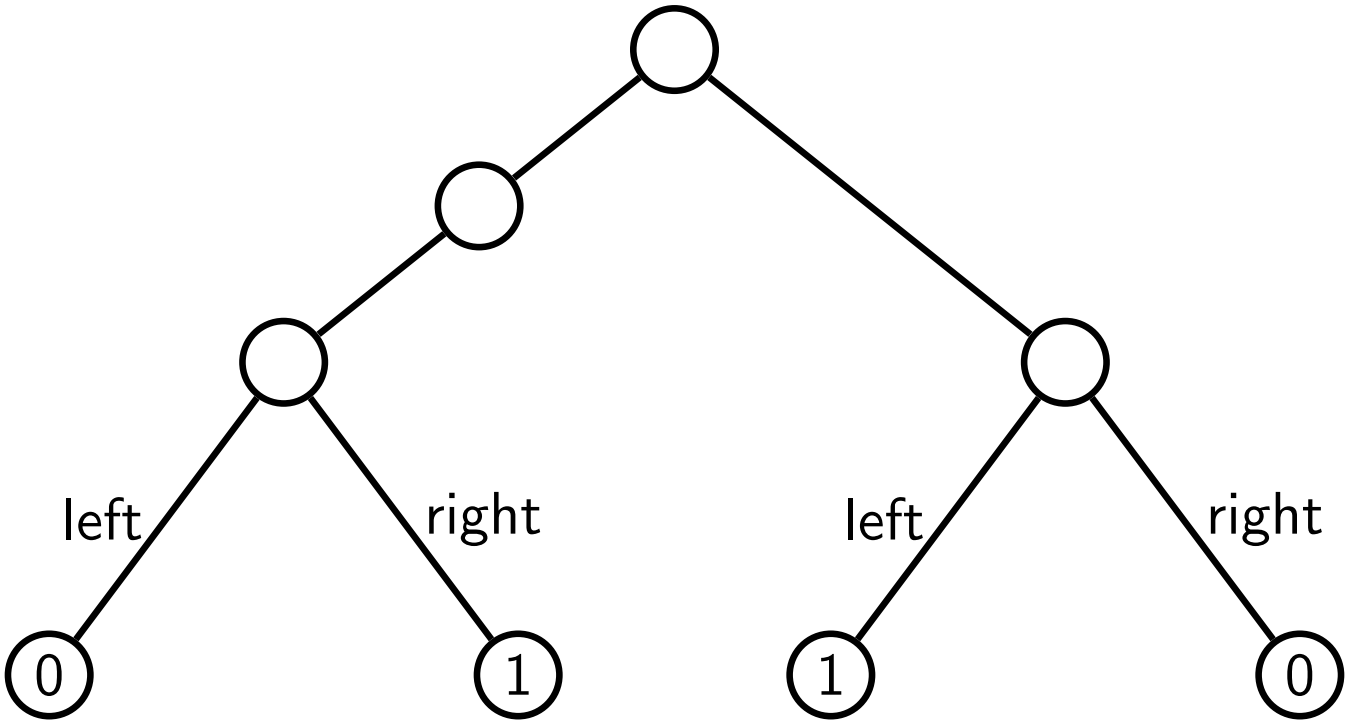
Encoding a Number in a Tree

Encoding $6{=}0110_2$



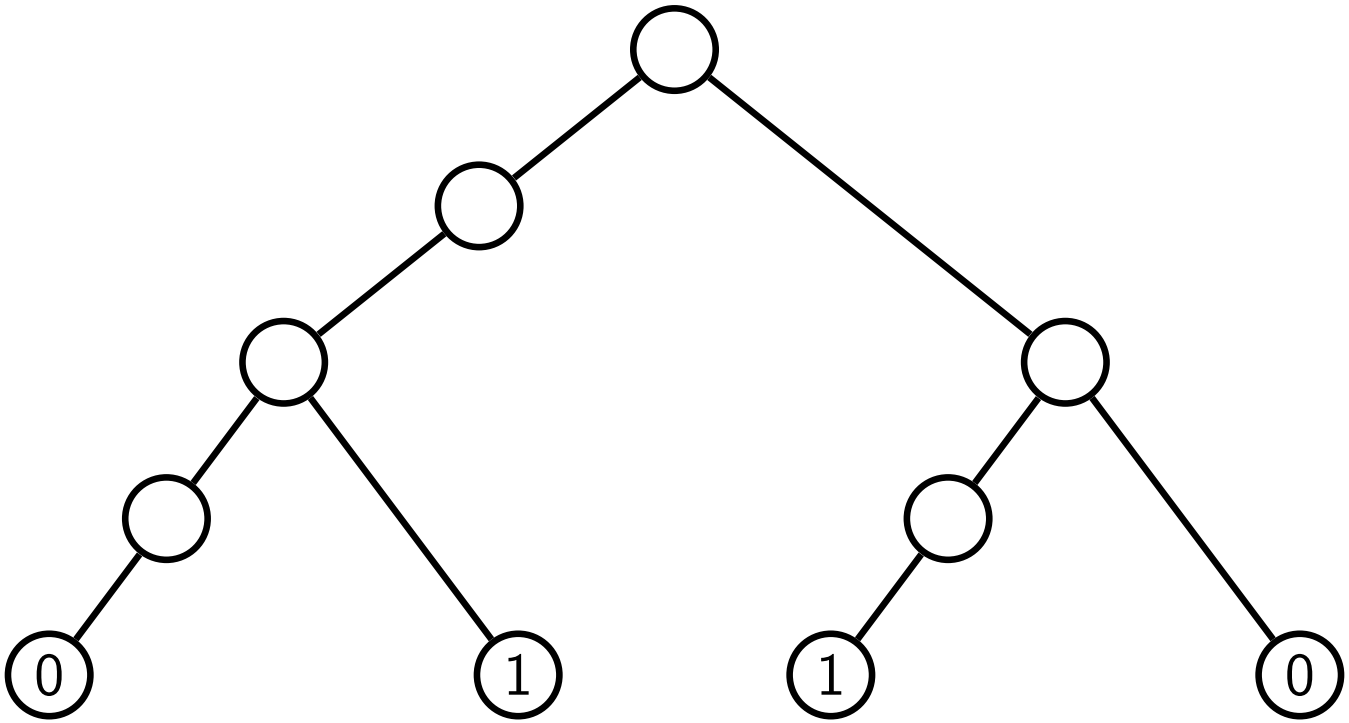
Encoding a Number in a Tree

Encoding $6{=}0110_2$



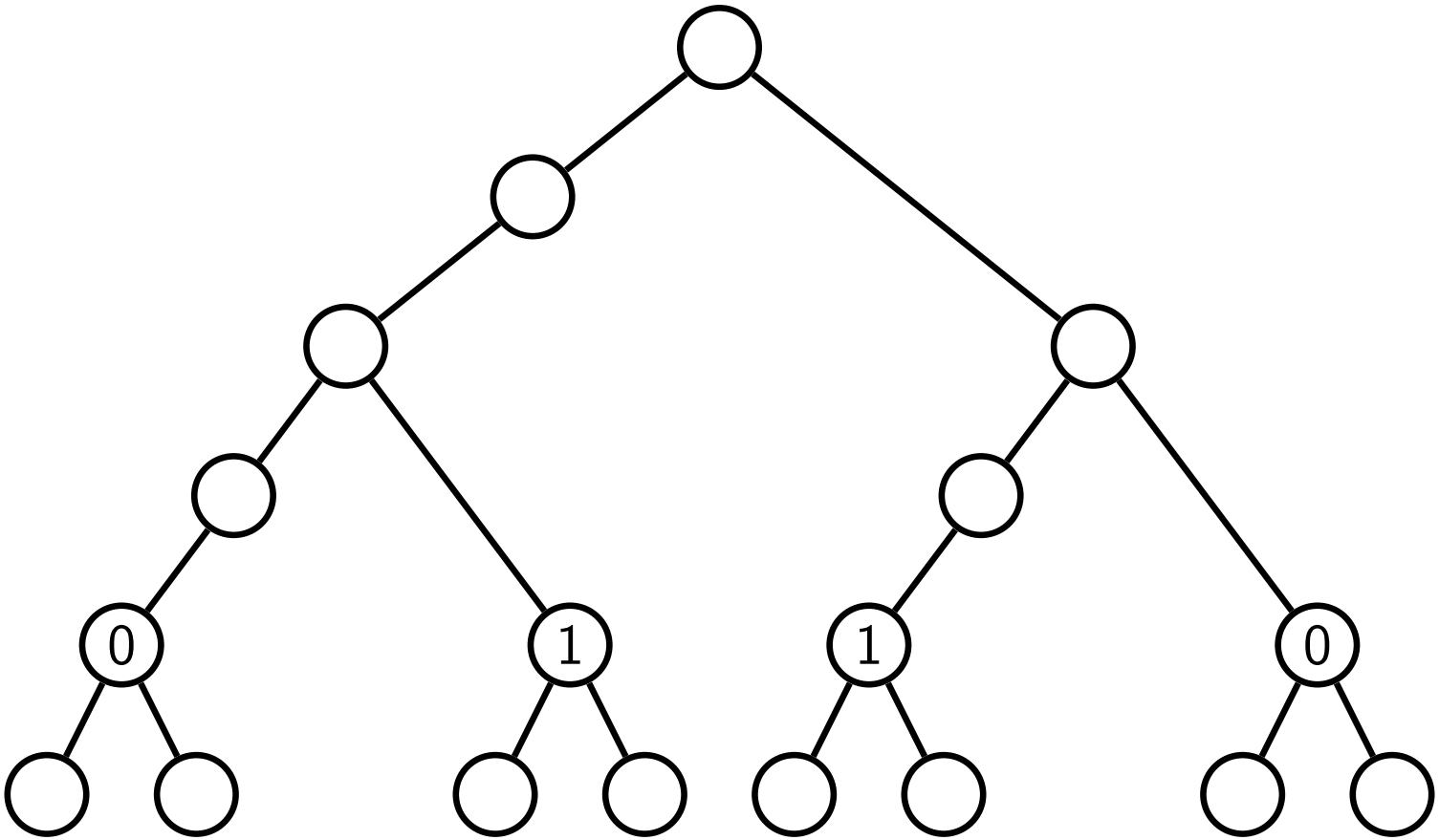
Encoding a Number in a Tree

Encoding $6{=}0110_2$



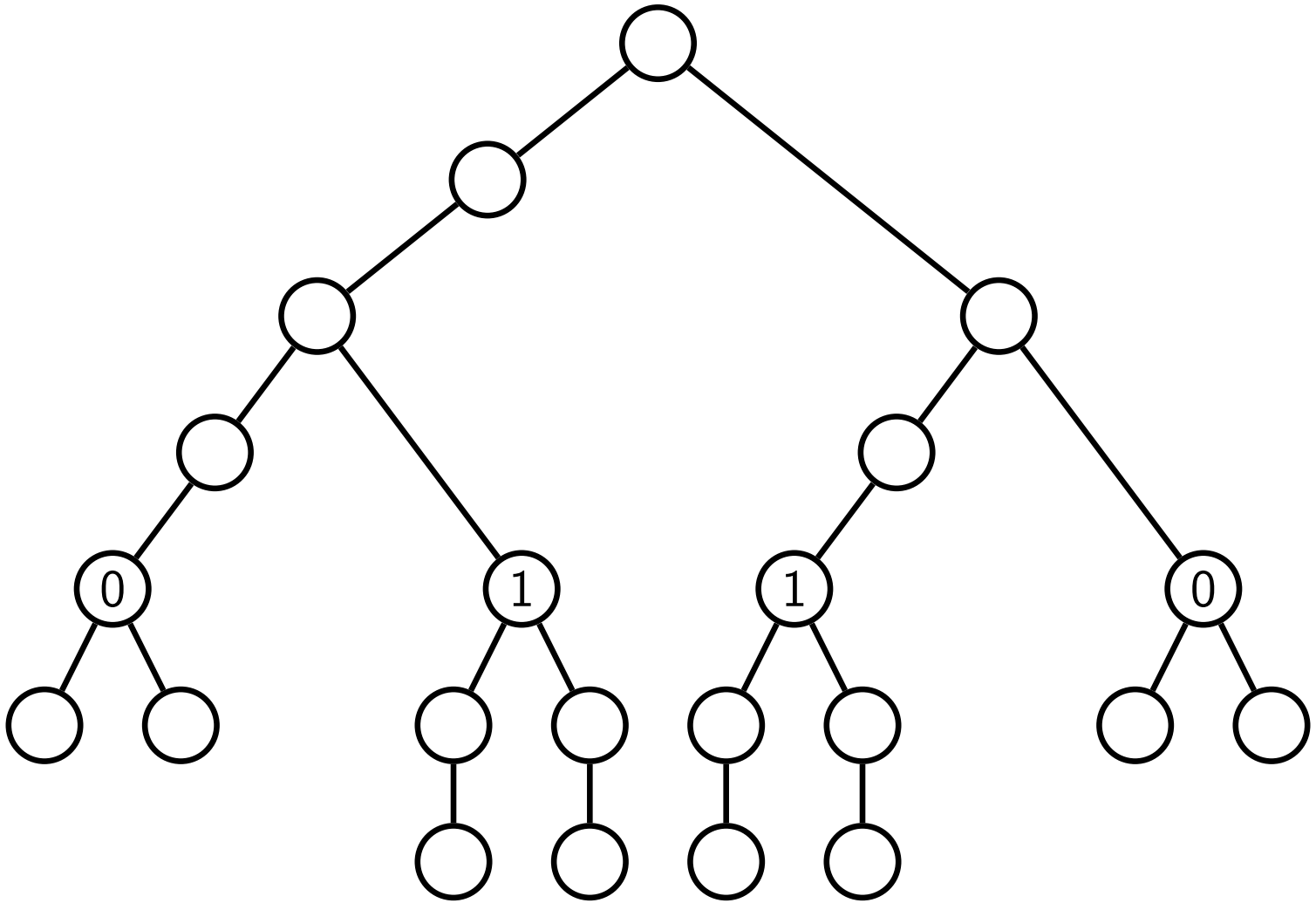
Encoding a Number in a Tree

Encoding $6{=}0110_2$



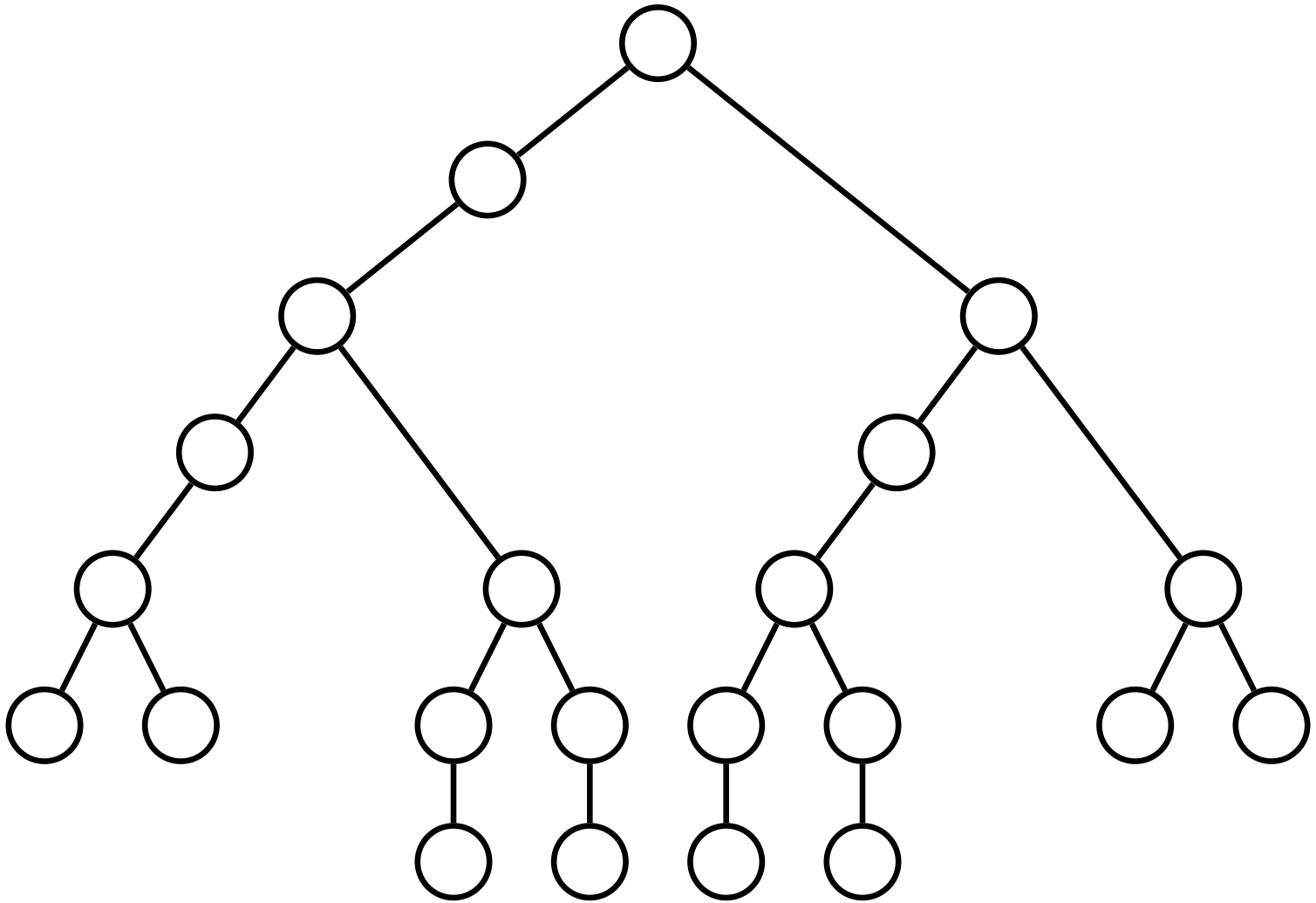
Encoding a Number in a Tree

Encoding $6{=}0110_2$

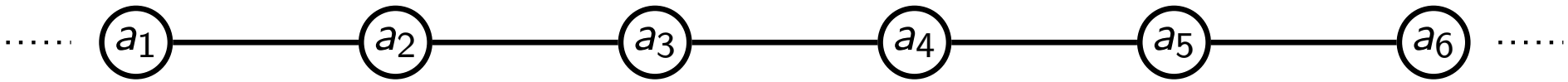


Encoding a Number in a Tree

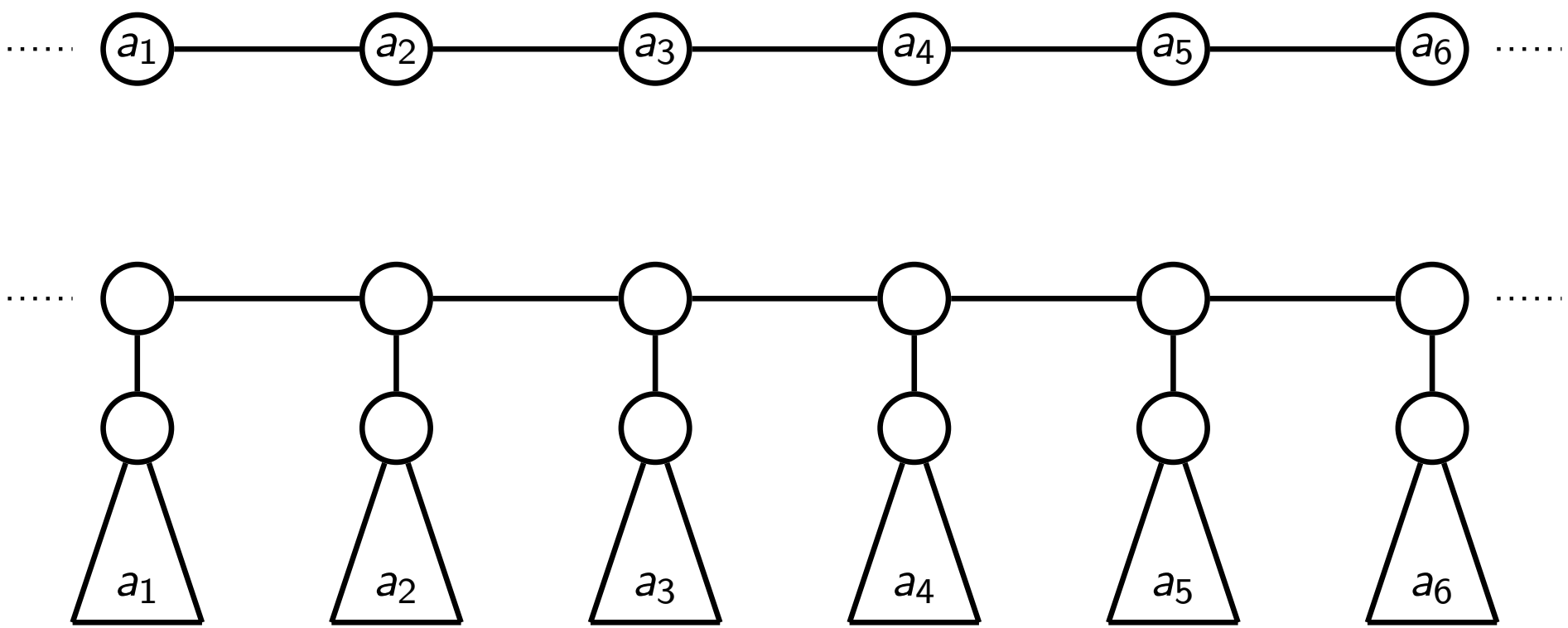
Encoding $6{=}0110_2$

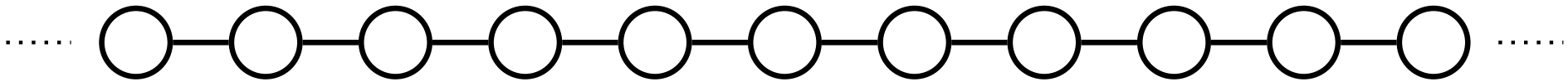


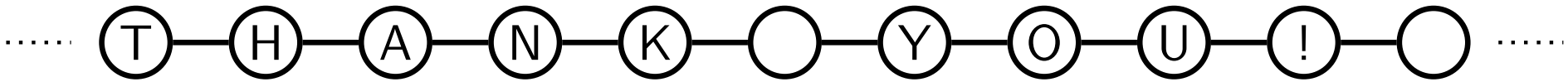
Encoding the Input of the Path



Encoding the Input of the Path






T
H
A
N
K
Y
O
U
!