Label-Guided Graph Exploration by a Finite Automaton

Reuven Cohen[1] Pierre Fraigniaud[2] David Ilcinkas[2]
Amos Korman[1] David Peleg[1]

[1]Dept. of Computer Science, Weizmann Institute, Israel

[2]CNRS, LRI, Université Paris-Sud, France

ICALP '05
July 12, 2005

Graph exploration

Goal

A mobile entity has to traverse every edge of an unknown anonymous graph.

Motivations (1)

Exploration by mobile agents

- Physical robot: exploration of environments unreachable by humans
- Software agent: network maintenance

Equivalence between logic and automata (Engelfriet, Hoogeboom)

Through characterization of string, tree or graph languages

- Automata with nested pebbles
- First-order logic with transitive closure

Motivations (1)

Exploration by mobile agents

- Physical robot: exploration of environments unreachable by humans
- Software agent: network maintenance

Equivalence between logic and automata (Engelfriet, Hoogeboom)

Through characterization of string, tree or graph languages

- Automata with nested pebbles
- First-order logic with transitive closure

Motivations (2)

USTCON (undirected st-connectivity)

- $G = \{V, E\}$ an undirected graph
- $s, t \in V$ two vertices of G

Are s and t in the same connected component of G?

- L = class of problems solvable by deterministic log-space computations
- SL ($\supseteq$ L) = class of problems solvable by symmetric non-deterministic log-space computations

Motivations (2)

USTCON (undirected st-connectivity)

- $G = \{V, E\}$ an undirected graph
- $s, t \in V$ two vertices of G

Are s and t in the same connected component of G?

- L = class of problems solvable by deterministic log-space computations
- SL ($\supseteq$ L) = class of problems solvable by symmetric non-deterministic log-space computations

Motivations (2)

USTCON (undirected st-connectivity) SL-complete

- $G = \{V, E\}$ an undirected graph
- $s, t \in V$ two vertices of G

Are s and t in the same connected component of G?

- L = class of problems solvable by deterministic log-space computations
- SL ($\supseteq$ L) = class of problems solvable by symmetric non-deterministic log-space computations

Motivations (2)

USTCON (undirected st-connectivity) SL-complete

- $G = \{V, E\}$ an undirected graph
- $s, t \in V$ two vertices of G

Are s and t in the same connected component of G?

- L = class of problems solvable by deterministic log-space computations
- SL ($\supseteq$ L) = class of problems solvable by symmetric non-deterministic log-space computations

Reingold, STOC 2005
Undirected ST-Connectivity in Log-Space

USTCON $\in$ L $\Rightarrow$ SL=L

Unknown, anonymous

Unknown

- Unknown topology
- Unknown size (no upper bound)

Anonymous

- No node labeling
- Local edge labeling

Unknown, anonymous

Unknown

- Unknown topology
- Unknown size (no upper bound)

Anonymous

- No node labeling
- Local edge labeling

Unknown, anonymous

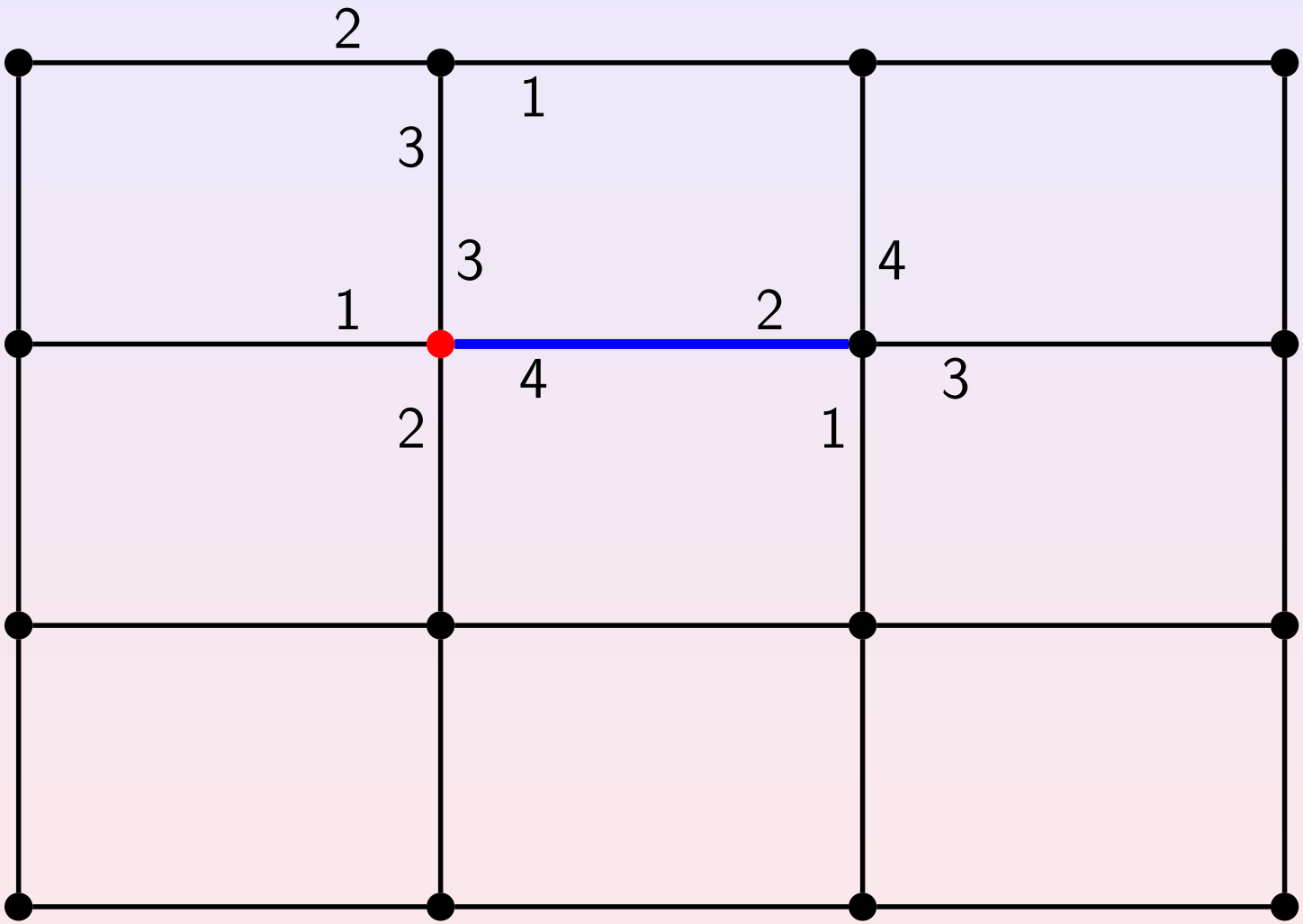
Unknown

- Unknown topology
- Unknown size (no upper bound)

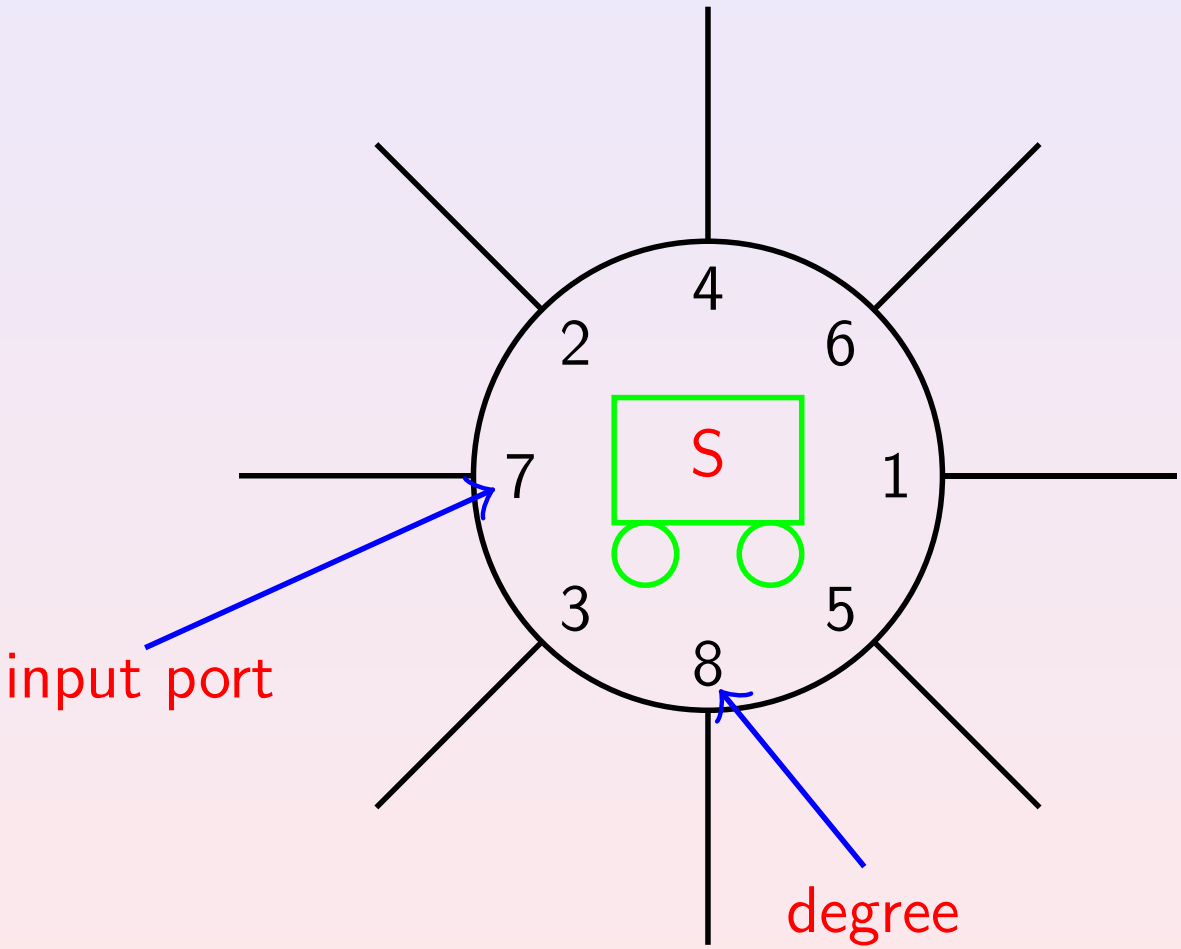
Anonymous

- No node labeling
- Local edge labeling

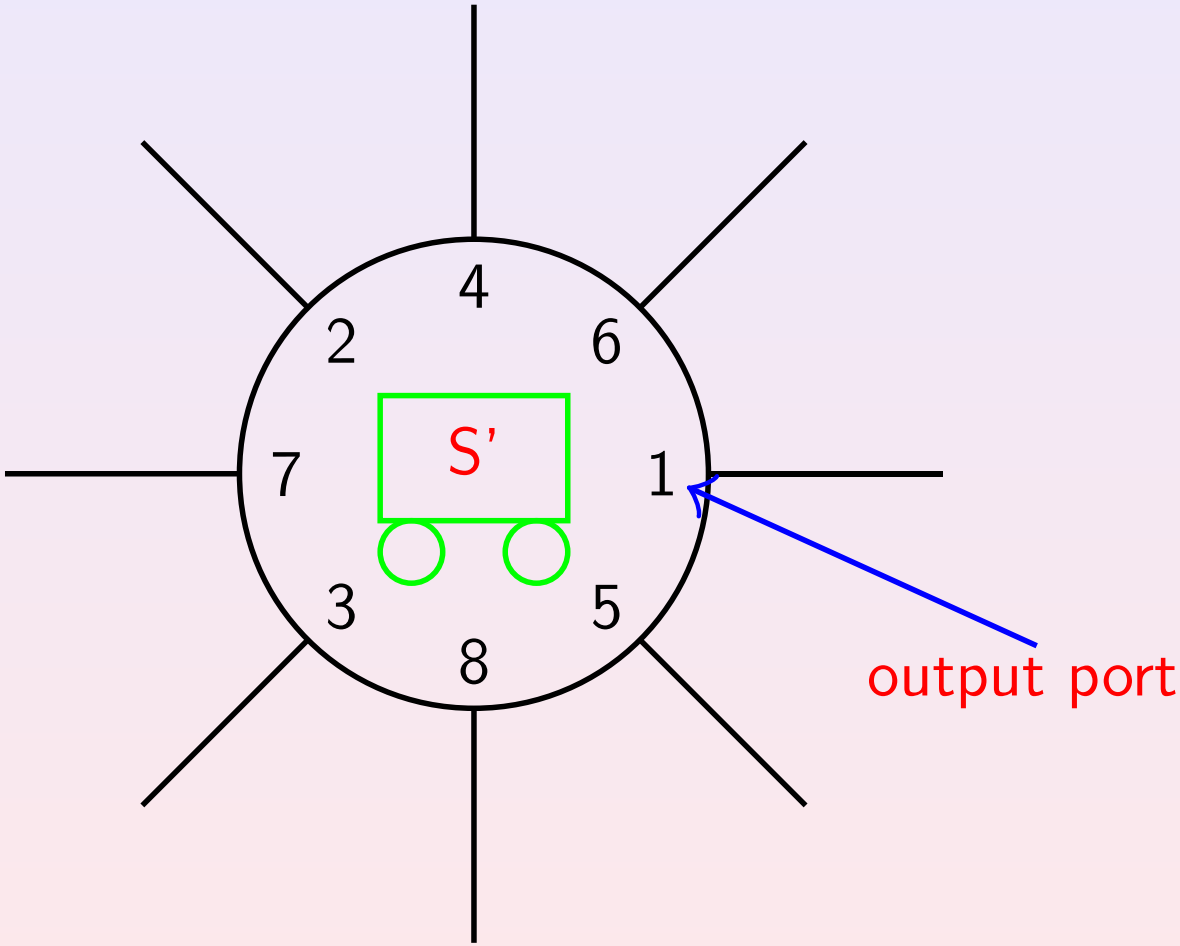
Example of an anonymous graph



Mealy automaton (1)



Mealy automaton (1)



Mealy automaton (2)

Input

- S : current state
- i : input port number
- d : node's degree

Output

- S' : new state
- j : output port number

Transition function

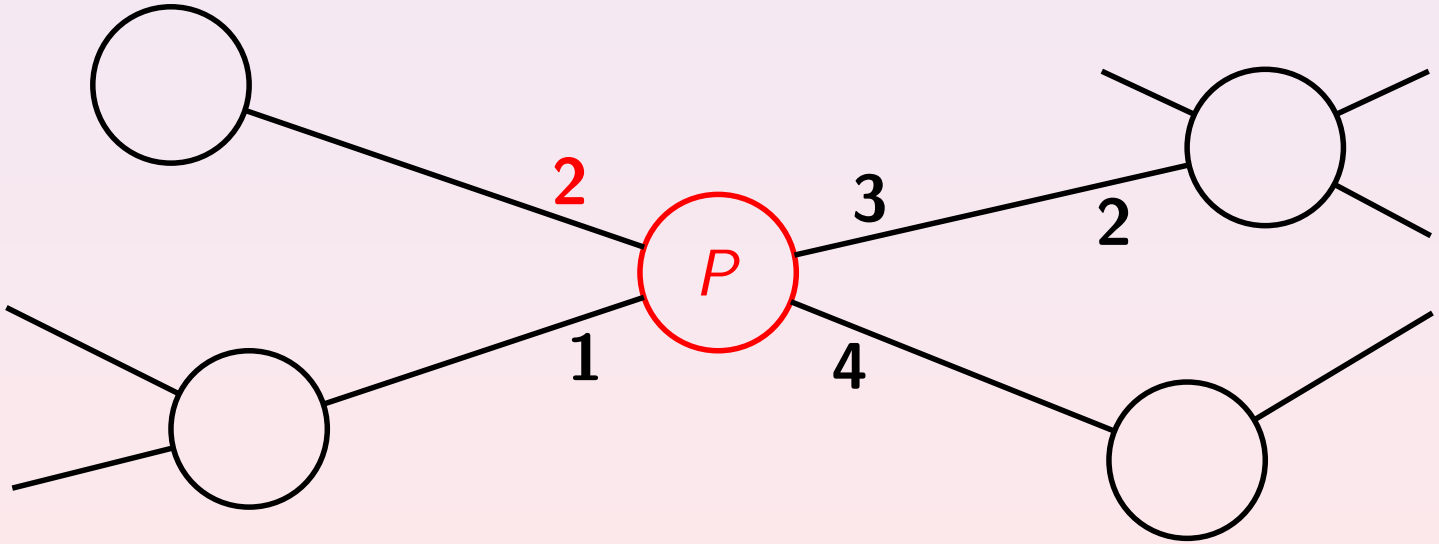
- $f(S, i, d) = (S', j)$

Exploration of the neighborhood

Star

The star can be explored by a 2-state automaton

- PlusOne P : $f(i, d, P) = (i + 1 \mod d, B)$
- Backtrack B : $f(i, d, B) = (i, P)$

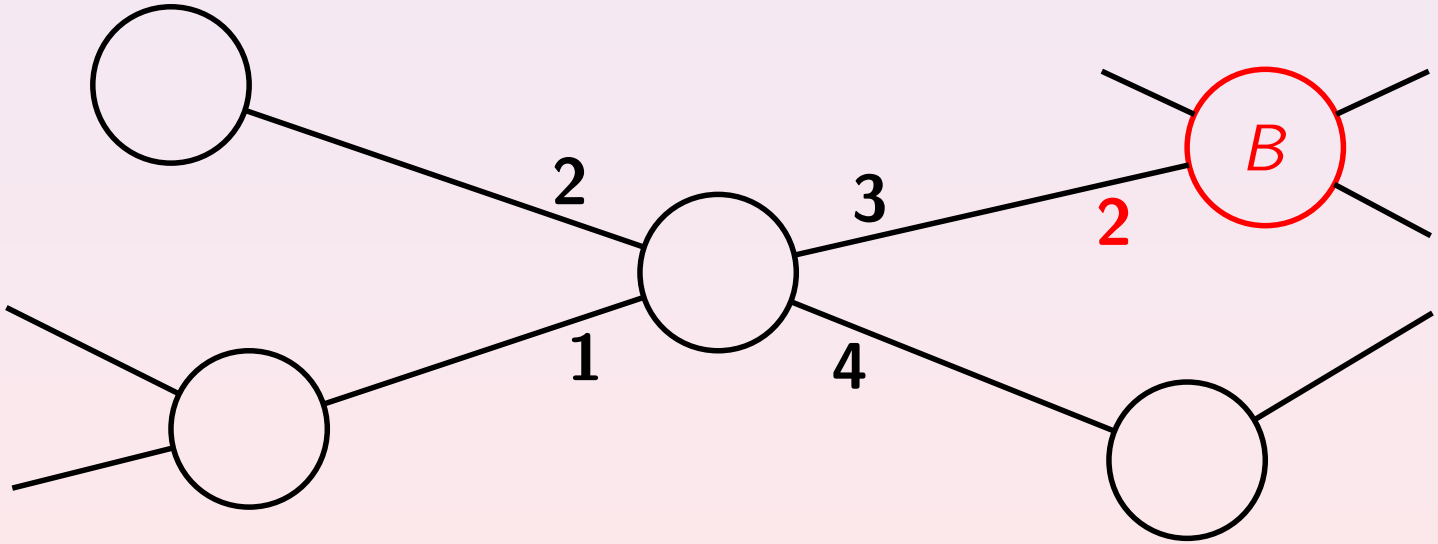


Exploration of the neighborhood

Star

The star can be explored by a 2-state automaton

- PlusOne P : $f(i, d, P) = (i + 1 \mod d, B)$
- Backtrack B : $f(i, d, B) = (i, P)$

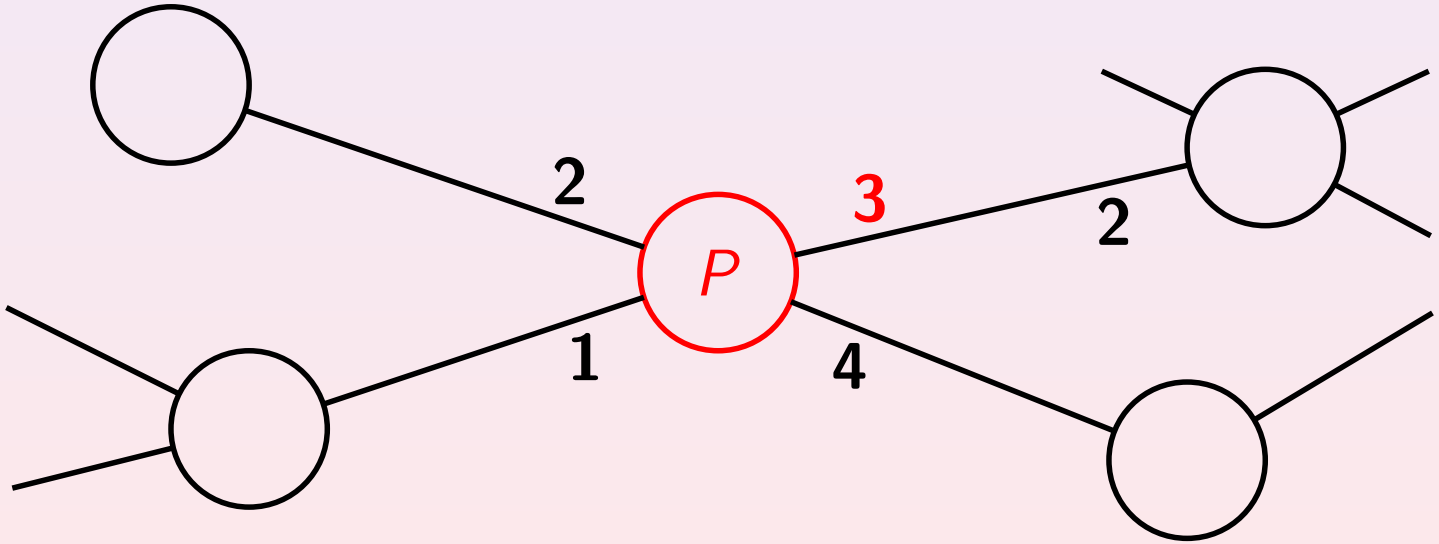


Exploration of the neighborhood

Star

The star can be explored by a 2-state automaton

- PlusOne P : $f(i, d, P) = (i + 1 \mod d, B)$
- Backtrack B : $f(i, d, B) = (i, P)$

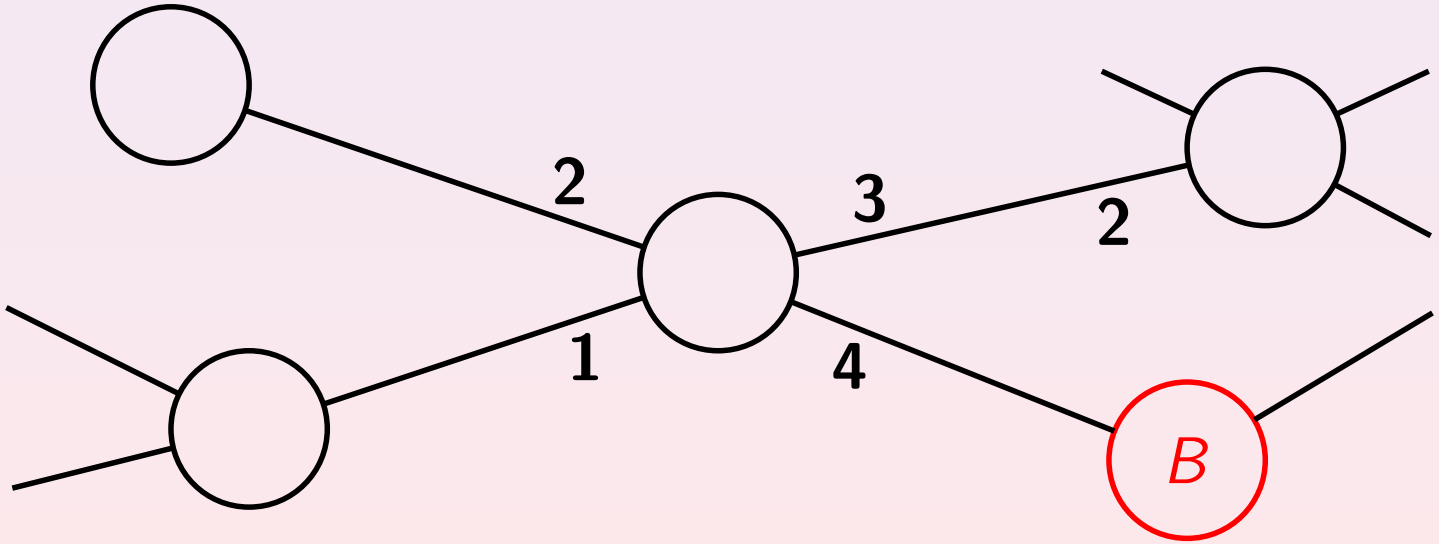


Exploration of the neighborhood

Star

The star can be explored by a 2-state automaton

- PlusOne P : $f(i, d, P) = (i + 1 \mod d, B)$
- Backtrack B : $f(i, d, B) = (i, P)$

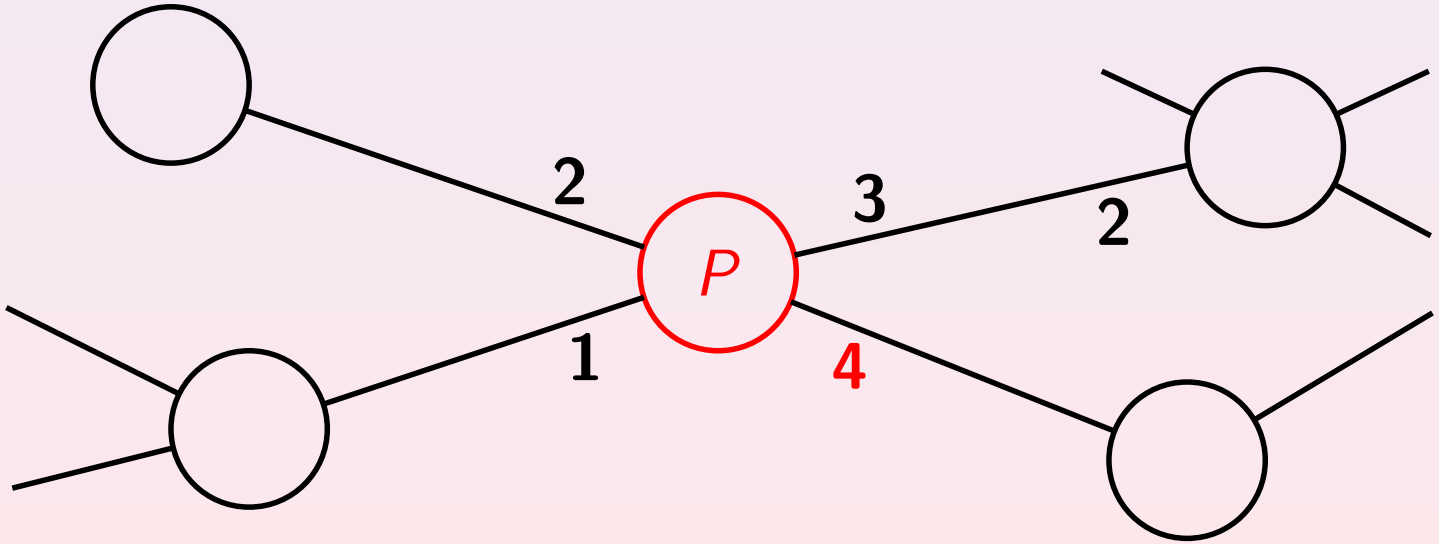


Exploration of the neighborhood

Star

The star can be explored by a 2-state automaton

- PlusOne P : $f(i, d, P) = (i + 1 \mod d, B)$
- Backtrack B : $f(i, d, B) = (i, P)$

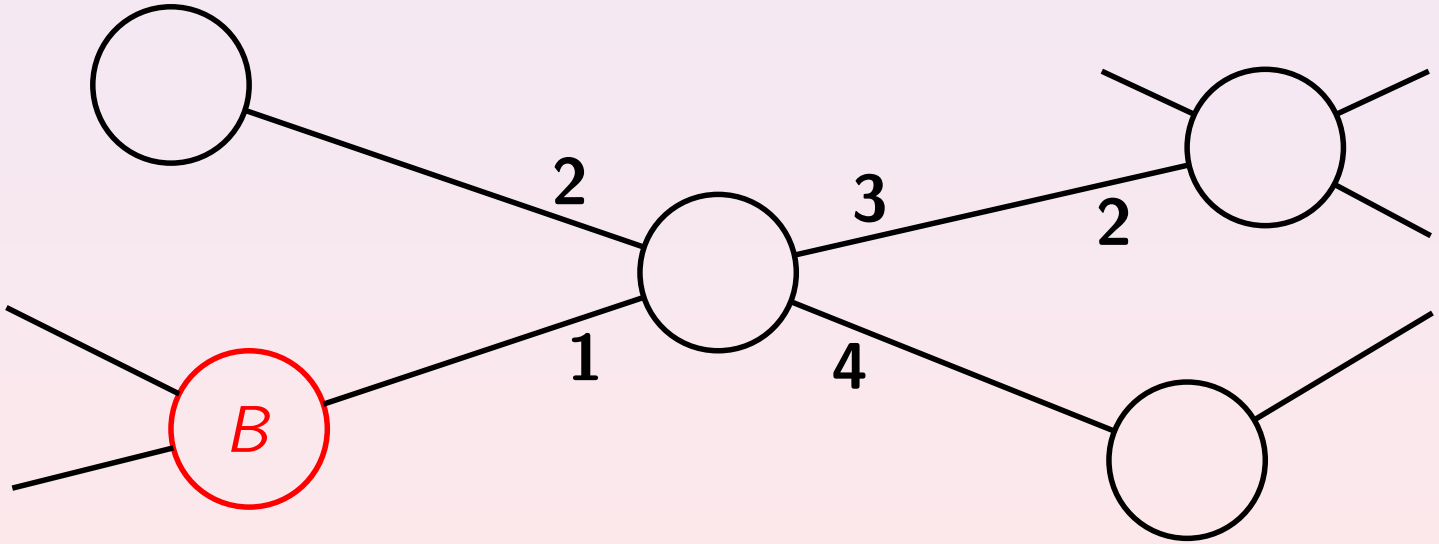


Exploration of the neighborhood

Star

The star can be explored by a 2-state automaton

- PlusOne P : $f(i, d, P) = (i + 1 \mod d, B)$
- Backtrack B : $f(i, d, B) = (i, P)$

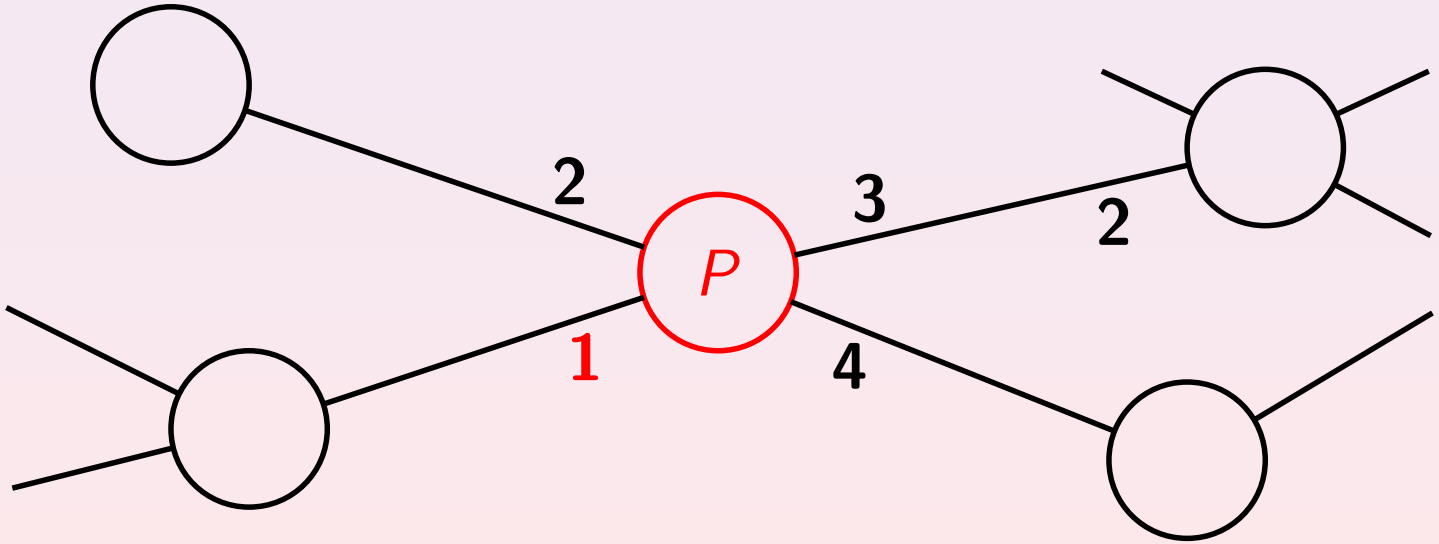


Exploration of the neighborhood

Star

The star can be explored by a 2-state automaton

- PlusOne P : $f(i, d, P) = (i + 1 \mod d, B)$
- Backtrack B : $f(i, d, B) = (i, P)$

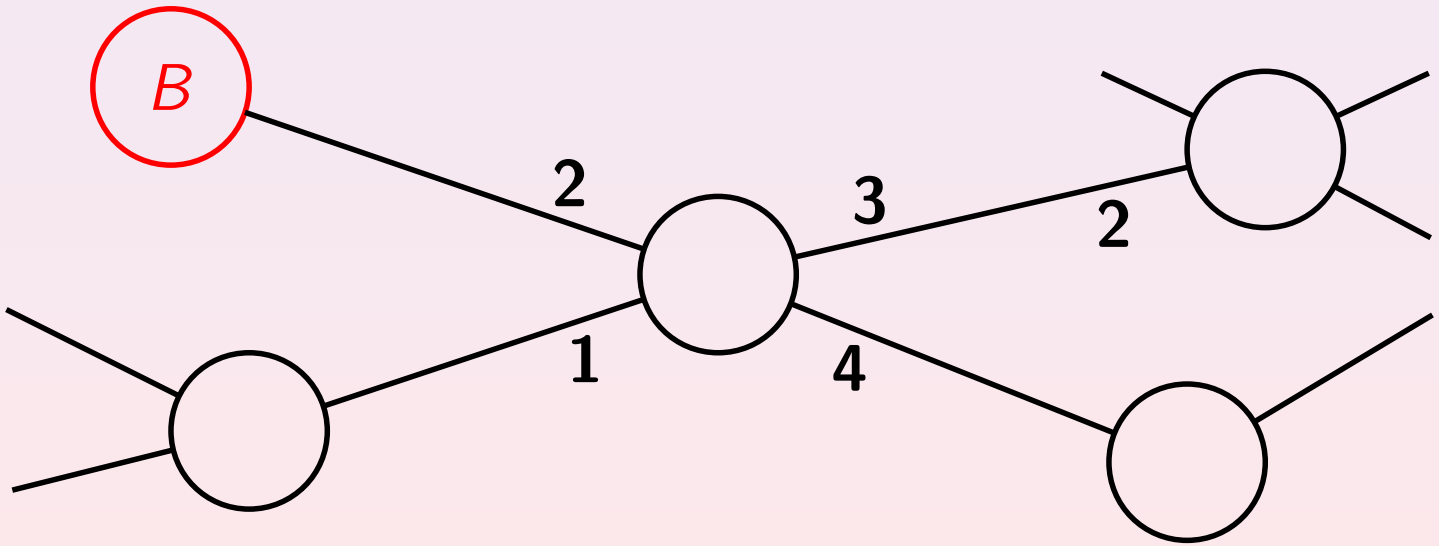


Exploration of the neighborhood

Star

The star can be explored by a 2-state automaton

- PlusOne P : $f(i, d, P) = (i + 1 \mod d, B)$
- Backtrack B : $f(i, d, B) = (i, P)$

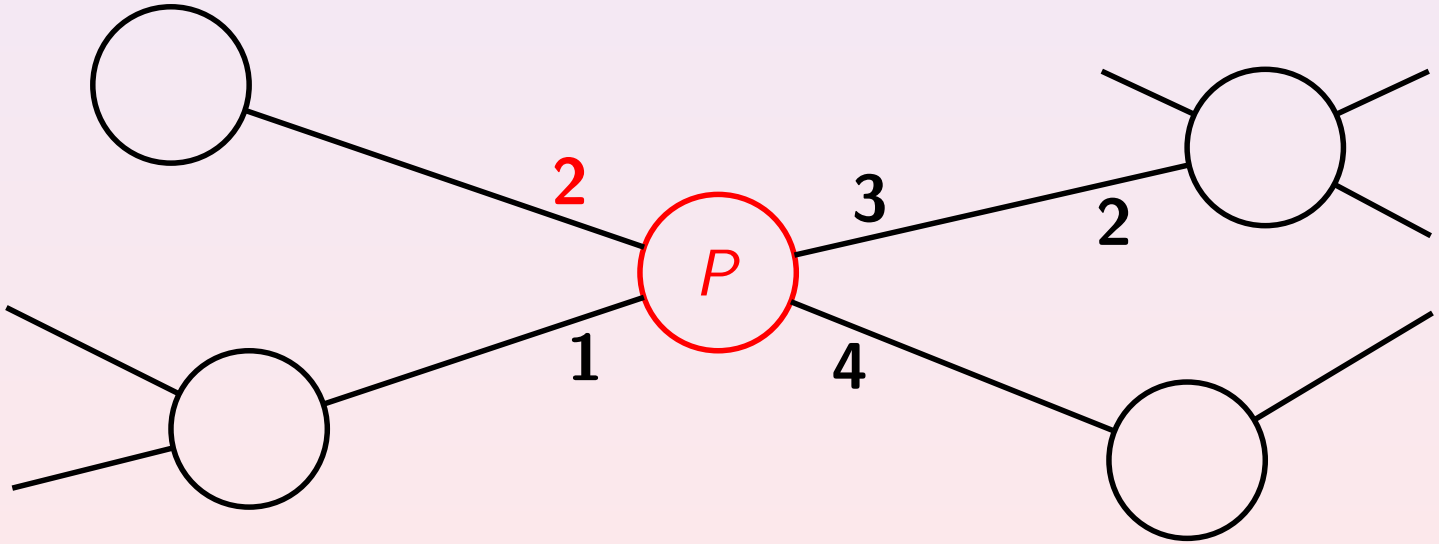


Exploration of the neighborhood

Star

The star can be explored by a 2-state automaton

- PlusOne P : $f(i, d, P) = (i + 1 \mod d, B)$
- Backtrack B : $f(i, d, B) = (i, P)$

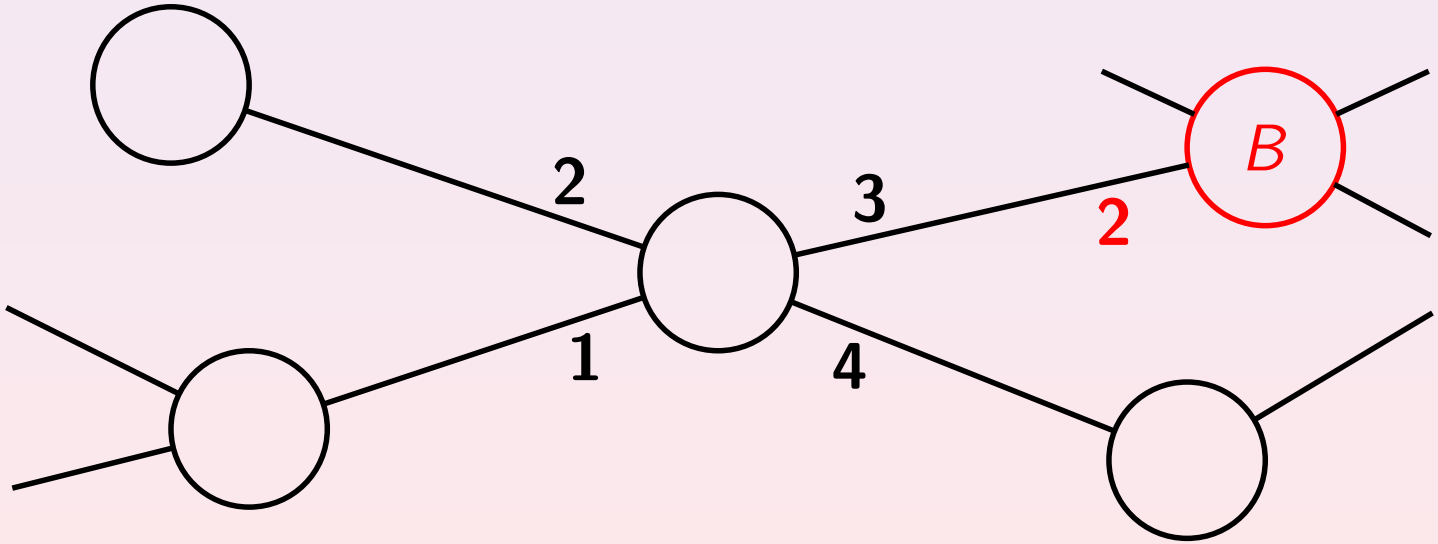


Exploration of the neighborhood

Star

The star can be explored by a 2-state automaton

- PlusOne P : $f(i, d, P) = (i + 1 \mod d, B)$
- Backtrack B : $f(i, d, B) = (i, P)$



Outline

Impossibility results (1)

Budach, Math. Nachrichten, 1978
Automata and Labyrinths

No finite automaton can explore all graphs.

A pebble is a node-marker that can be dropped at and removed from nodes.

Rabin, Seminar talk at Berkeley, 1967
Maze threading automata

No finite automaton with a finite number of pebbles can explore all graphs.

Impossibility results (1)

Budach, Math. Nachrichten, 1978
Automata and Labyrinths

No finite automaton can explore all graphs.

A pebble is a node-marker that can be dropped at and removed from nodes.

Rabin, Seminar talk at Berkeley, 1967
Maze threading automata

No finite automaton with a finite number of pebbles can explore all graphs.

Impossibility results (1)

Budach, Math. Nachrichten, 1978
Automata and Labyrinths

No finite automaton can explore all graphs.

A pebble is a node-marker that can be dropped at and removed from nodes.

Rabin, Seminar talk at Berkeley, 1967
Maze threading automata

No finite automaton with a finite number of pebbles can explore all graphs.

Impossibility results (2)

Rollik, Acta Informatica, 1980
Automaten in planaren Graphen

No finite team of finite cooperative automata can explore all (cubic planar) graphs.

A JAG (Jumping Automaton for Graphs) is a team of finite automata that cooperate constantly. Moreover an automaton can jump to a vertex occupied by another automaton.

Cook, Rackoff, SIAMJC, 1980
Space lower bounds for maze threadability on restricted machines

No JAG can explore all graphs.

Impossibility results (2)

Rollik, Acta Informatica, 1980
Automaten in planaren Graphen

No finite team of finite cooperative automata can explore all (cubic planar) graphs.

A JAG (Jumping Automaton for Graphs) is a team of finite automata that cooperate constantly. Moreover an automaton can jump to a vertex occupied by another automaton.

Cook, Rackoff, SIAMJC, 1980
Space lower bounds for maze threadability on restricted machines

No JAG can explore all graphs.

Impossibility results (2)

Rollik, Acta Informatica, 1980
Automaten in planaren Graphen

No finite team of finite cooperative automata can explore all (cubic planar) graphs.

A JAG (Jumping Automaton for Graphs) is a team of finite automata that cooperate constantly. Moreover an automaton can jump to a vertex occupied by another automaton.

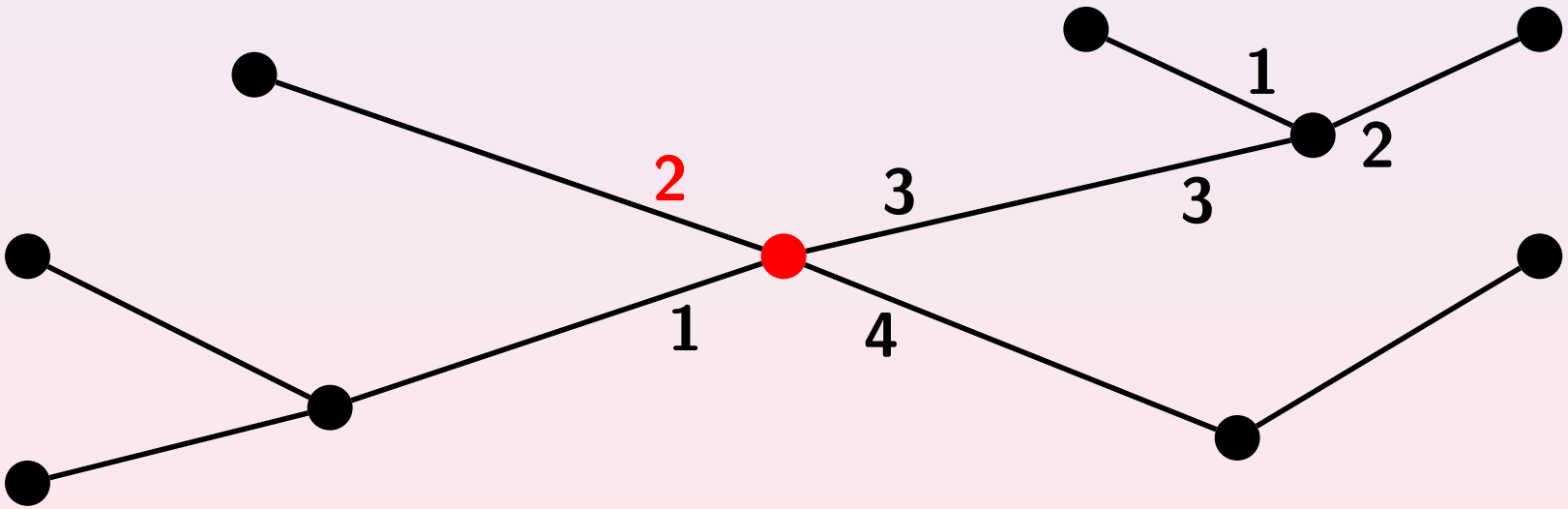
Cook, Rackoff, SIAMJC, 1980
Space lower bounds for maze threadability on restricted machines

No JAG can explore all graphs.

Universality for trees

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002
Tree exploration with little memory

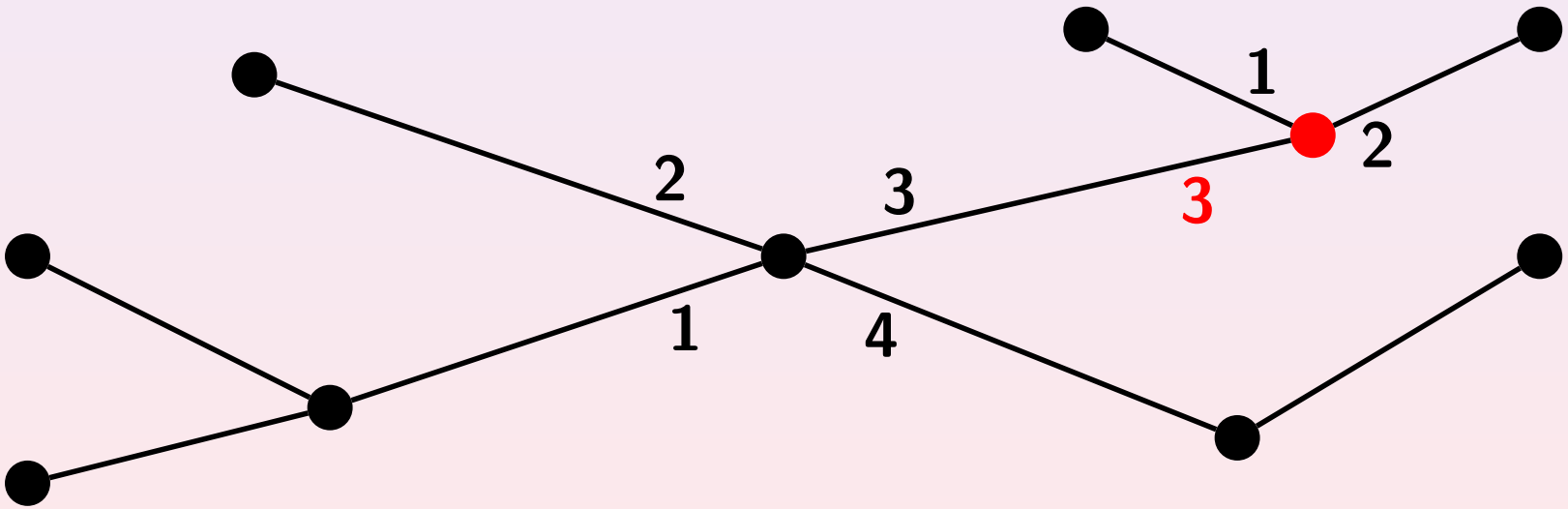
An oblivious automaton (one single state) can explore all trees.



Universality for trees

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002
Tree exploration with little memory

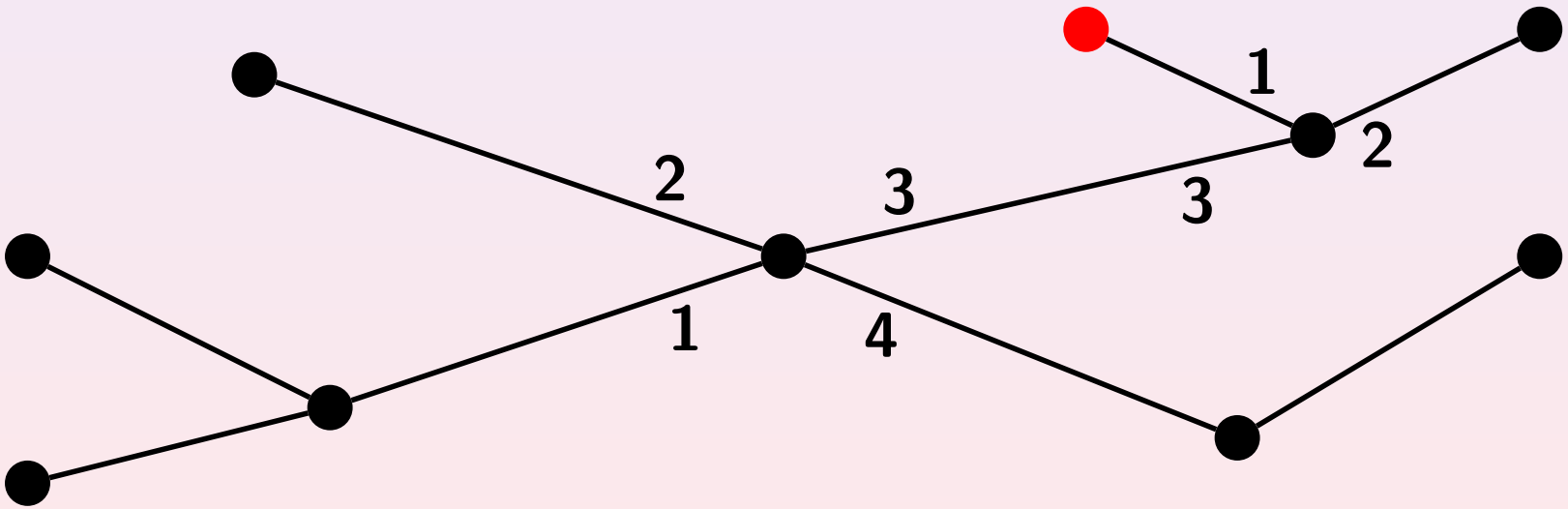
An oblivious automaton (one single state) can explore all trees.



Universality for trees

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002
Tree exploration with little memory

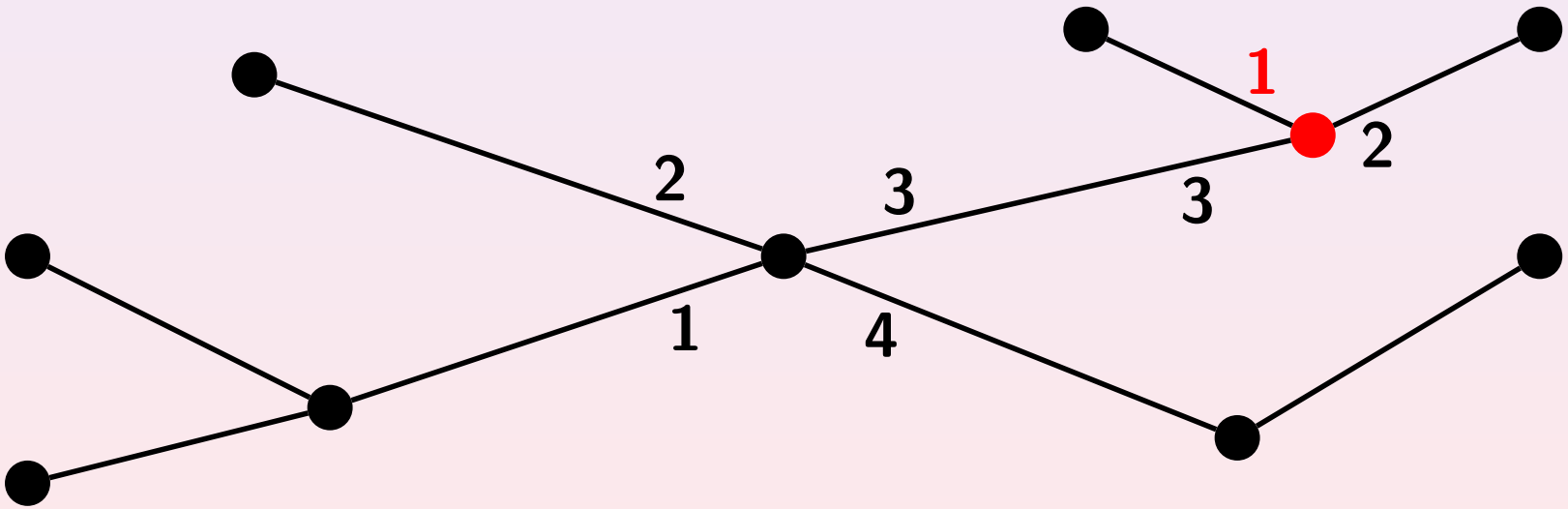
An oblivious automaton (one single state) can explore all trees.



Universality for trees

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002
Tree exploration with little memory

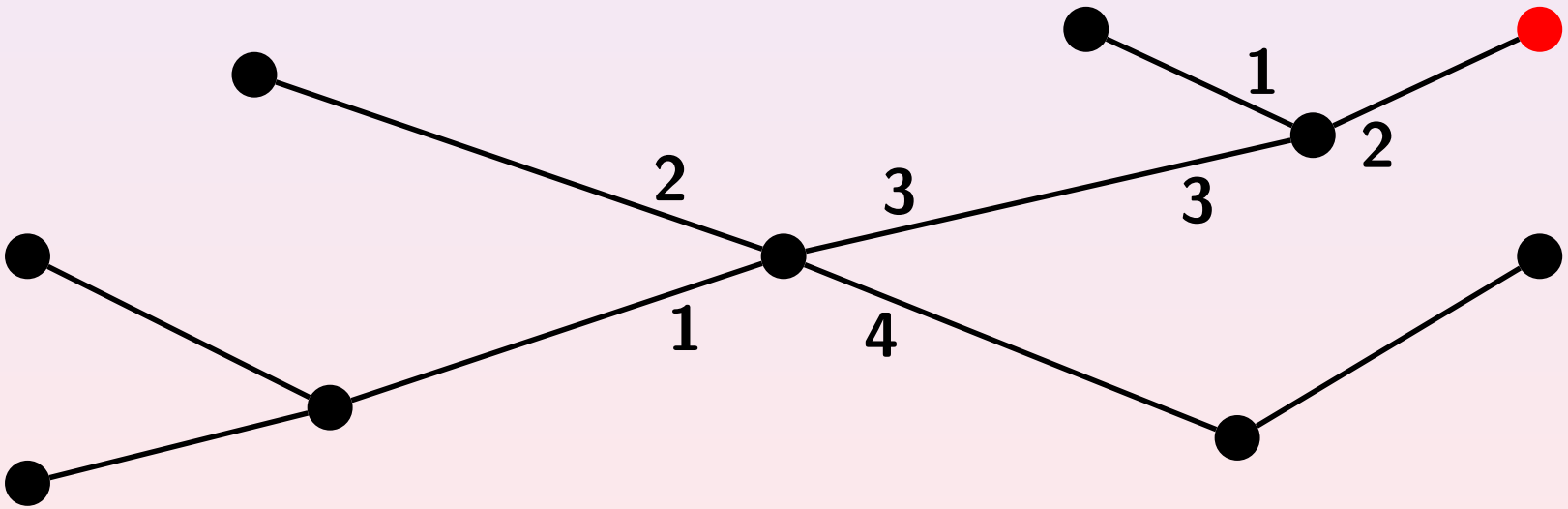
An oblivious automaton (one single state) can explore all trees.



Universality for trees

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002
Tree exploration with little memory

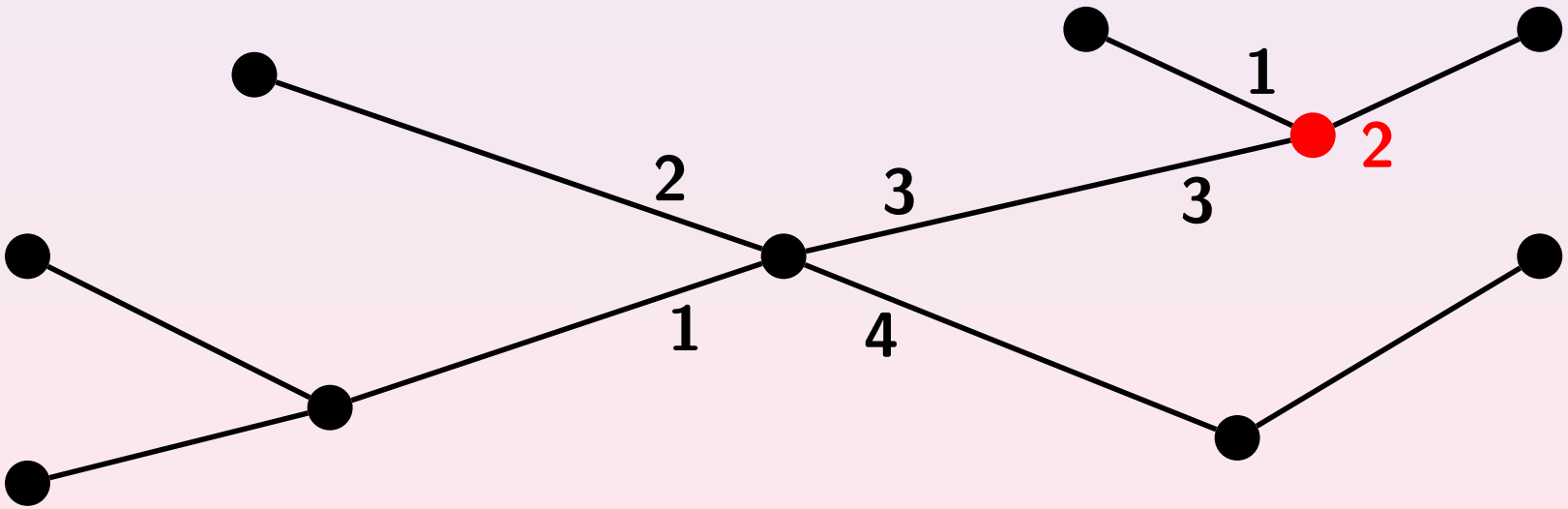
An oblivious automaton (one single state) can explore all trees.



Universality for trees

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002
Tree exploration with little memory

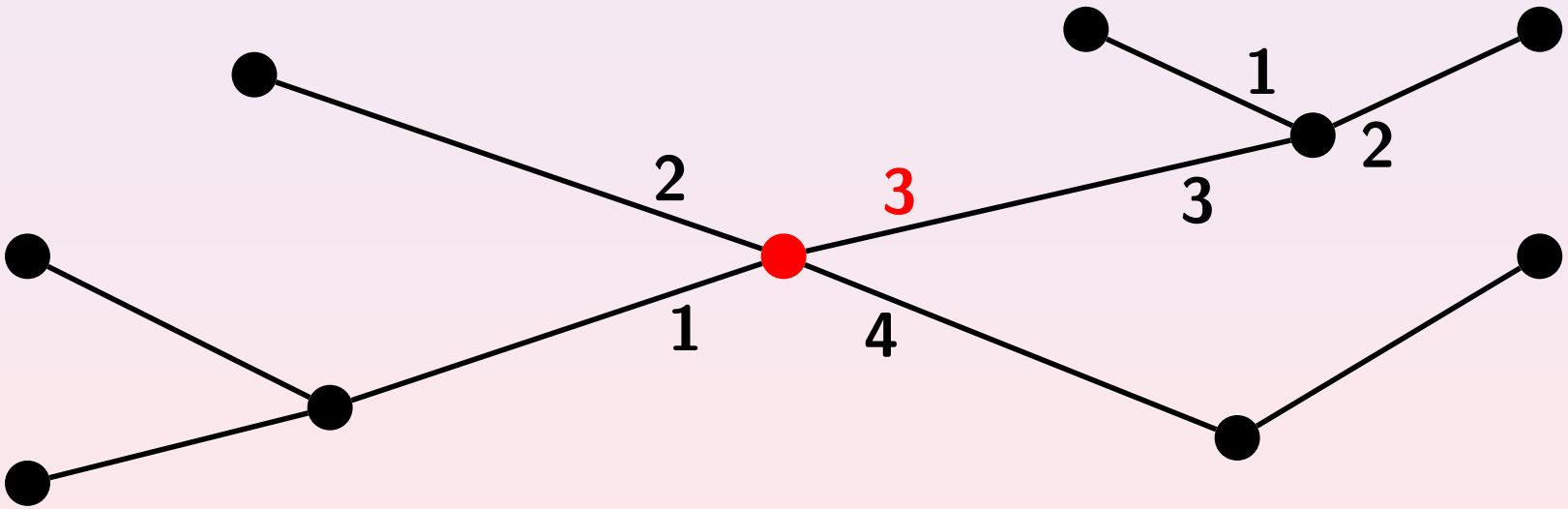
An oblivious automaton (one single state) can explore all trees.



Universality for trees

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002
Tree exploration with little memory

An oblivious automaton (one single state) can explore all trees.



Whiteboards

A whiteboard is a local storage where an automaton can write, read and erase information.

Fraigniaud, WSS 2001
On a Space-optimal Distributed Traversal Algorithm

An oblivious automaton (one single state) can explore all graphs of maximum degree Δ using only $O(log\,\Delta)$ memory bits per node.

Algorithm: When visiting a node for the ith times, take the edge $i \mod d$ where d is the degree of the node.

Whiteboards

A whiteboard is a local storage where an automaton can write, read and erase information.

Tixeuil, WSS 2001
On a Space-optimal Distributed Traversal Algorithm

An oblivious automaton (one single state) can explore all graphs of maximum degree Δ using only $O(log\,\Delta)$ memory bits per node.

Algorithm: When visiting a node for the ith times, take the edge $i \mod d$ where d is the degree of the node.

Whiteboards

A whiteboard is a local storage where an automaton can write, read and erase information.

Tixeuil, WSS 2001
On a Space-optimal Distributed Traversal Algorithm

An oblivious automaton (one single state) can explore all graphs of maximum degree Δ using only $O(log\,\Delta)$ memory bits per node.

Algorithm: When visiting a node for the ith times, take the edge $i \mod d$ where d is the degree of the node.

Outline

Our model

Model

- An oracle colors (labels) the graph to help the automaton.
- The finite automaton can read the color of the node as an input of its transition function.

Goal

Use the smallest possible number of colors.

Our results

Theorem 1: Three colors

There exist a finite automaton and an algorithm coloring in three colors such that the automaton can explore all graphs.

Theorem 2: Two colors

There exist a automaton of $O(log\Delta)$ memory bits and an algorithm coloring in only two colors such that the automaton can explore all graphs of maximum degree Δ.

Our results

Theorem 1: Three colors

There exist a finite automaton and an algorithm coloring in three colors such that the automaton can explore all graphs.

Theorem 2: Two colors

There exist a automaton of $O(log\,\Delta)$ memory bits and an algorithm coloring in only two colors such that the automaton can explore all graphs of maximum degree Δ.

Outline

Spanning tree

Basic idea

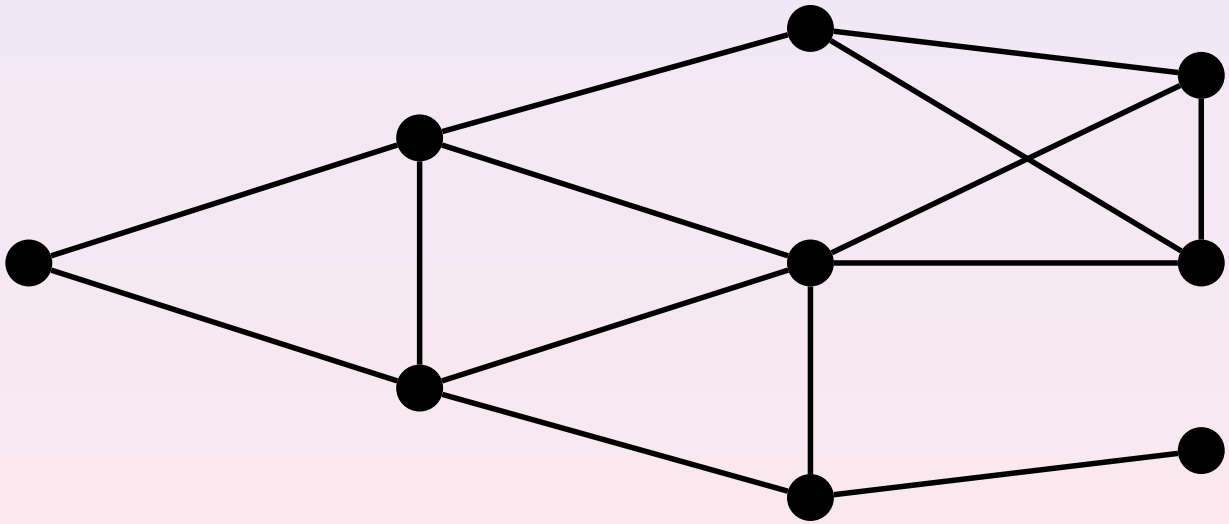
spanning tree: the label tells which edges are in the tree

Use the tree exploration algorithm

Spanning tree

Basic idea

spanning tree: the label tells which edges are in the tree

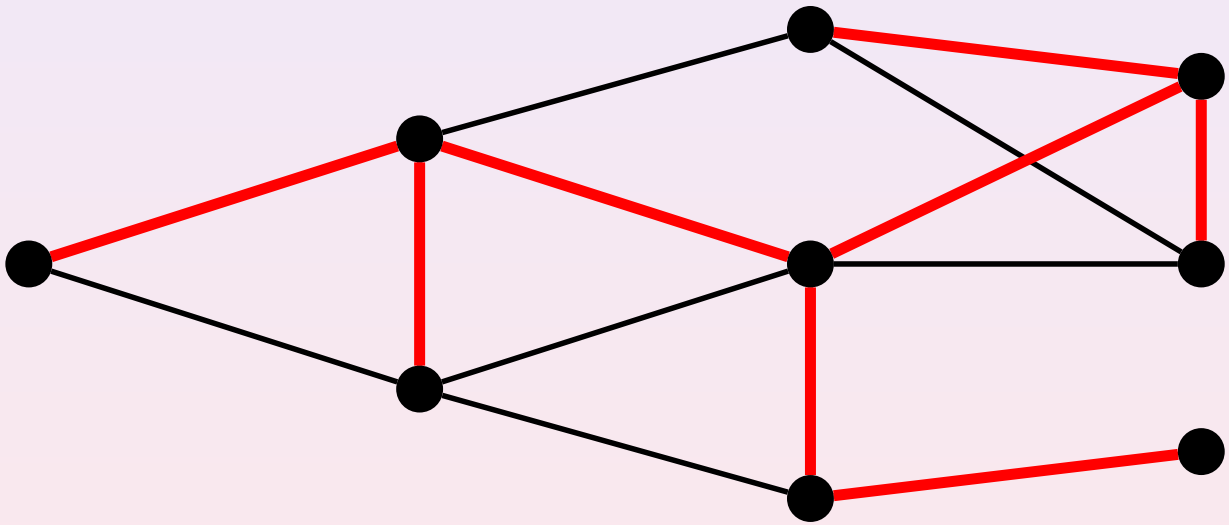


Use the tree exploration algorithm

Spanning tree

Basic idea

spanning tree: the label tells which edges are in the tree

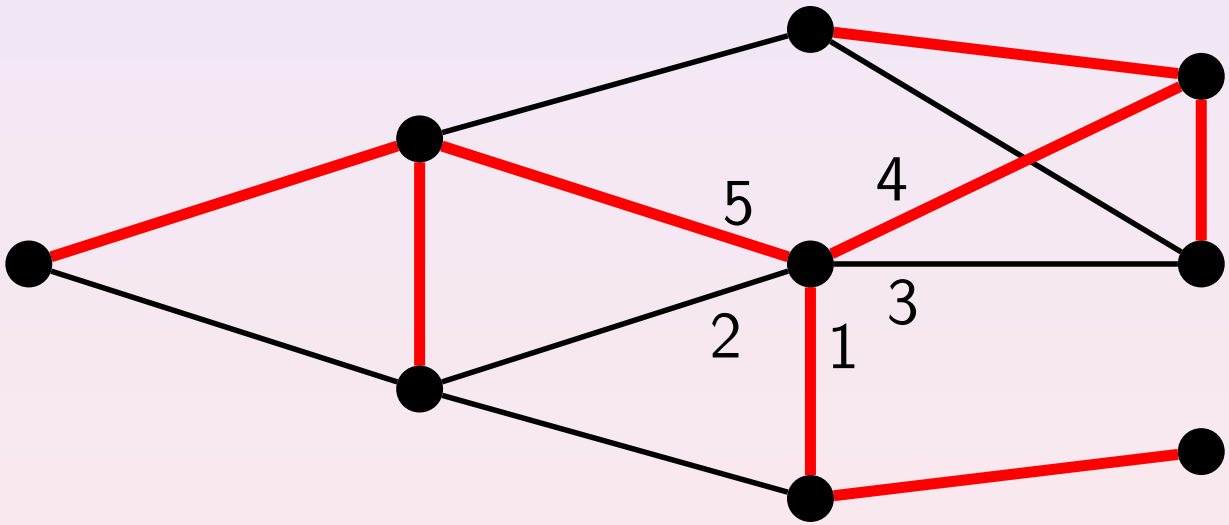


Use the tree exploration algorithm

Spanning tree

Basic idea

spanning tree: the label tells which edges are in the tree

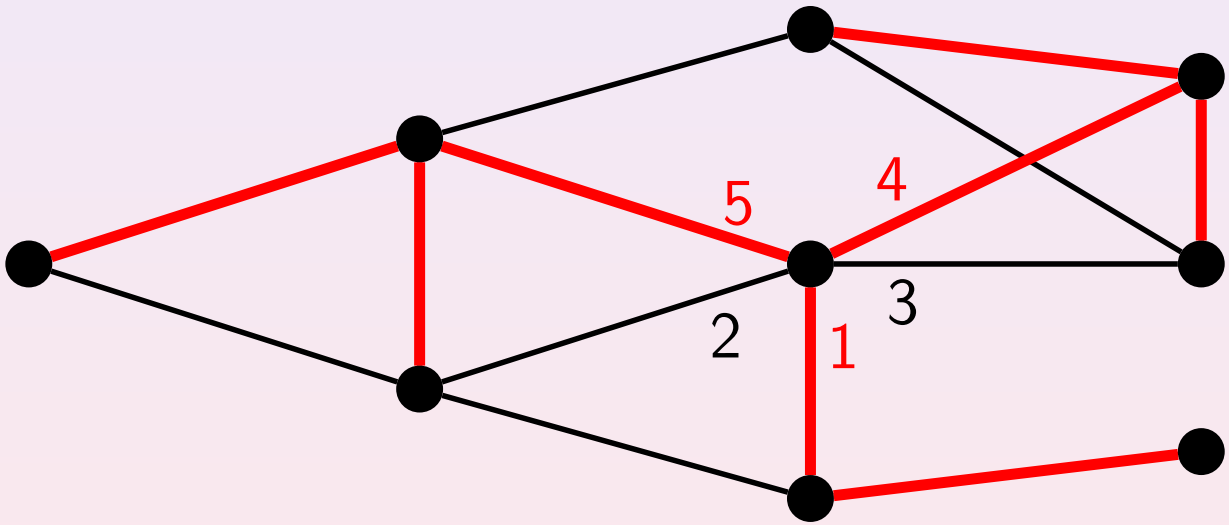


Use the tree exploration algorithm

Spanning tree

Basic idea

spanning tree: the label tells which edges are in the tree

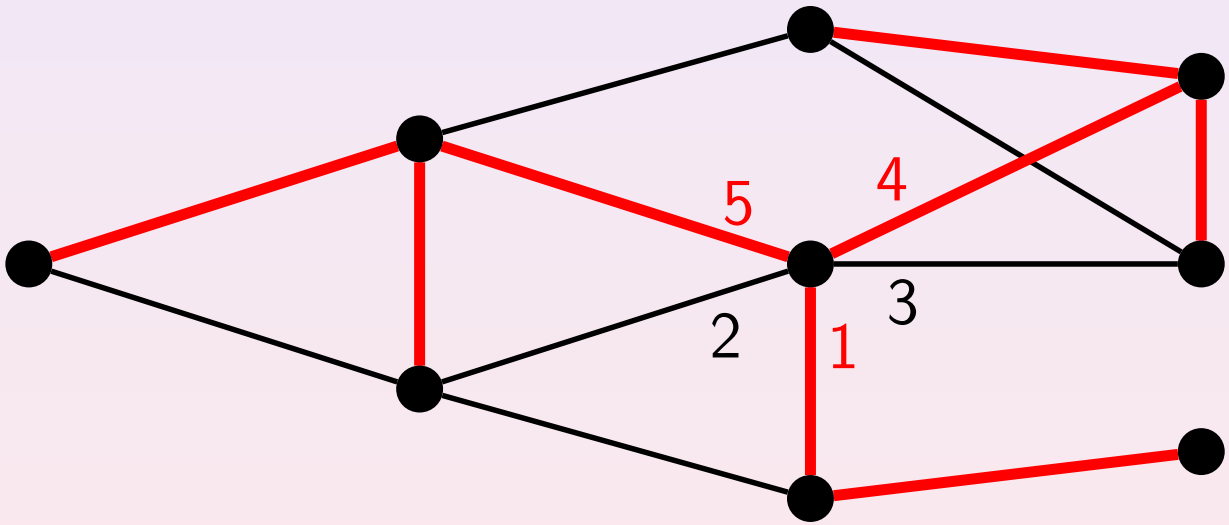


Use the tree exploration algorithm

Spanning tree

Basic idea

spanning tree: the label tells which edges are in the tree



Use the tree exploration algorithm

Rooted tree

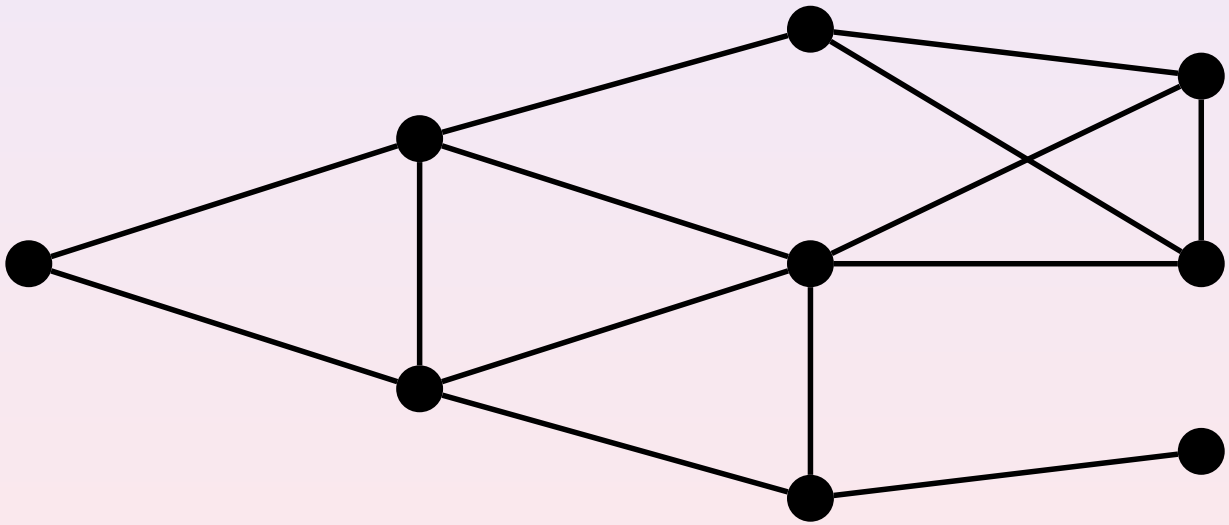
Enhanced labeling

only indicate edge leading to the parent $\longrightarrow$ Δ colors

Rooted tree

Enhanced labeling

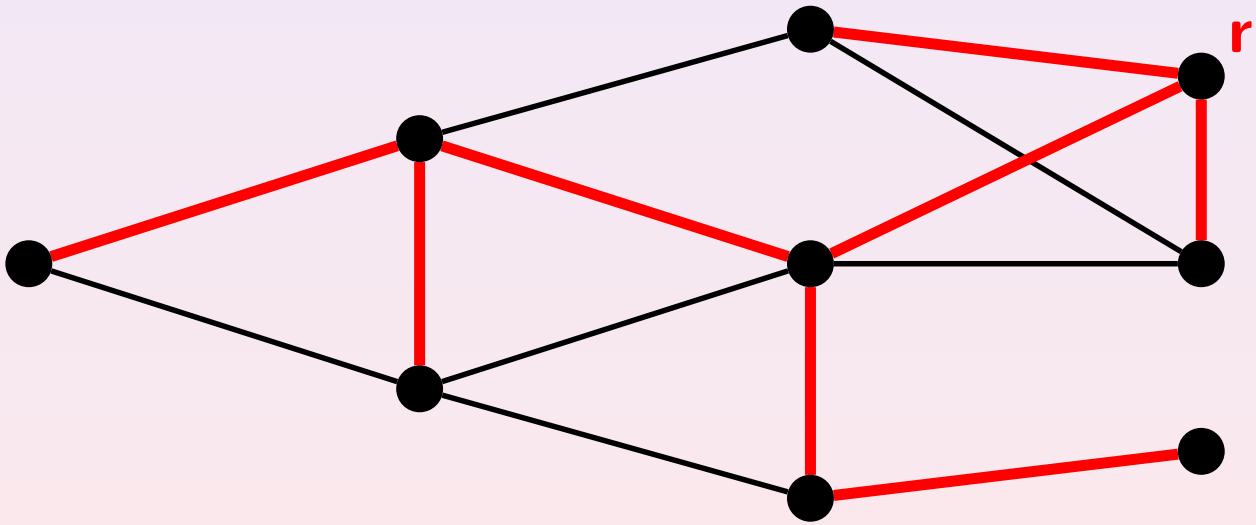
only indicate edge leading to the parent $\longrightarrow$ Δ colors



Rooted tree

Enhanced labeling

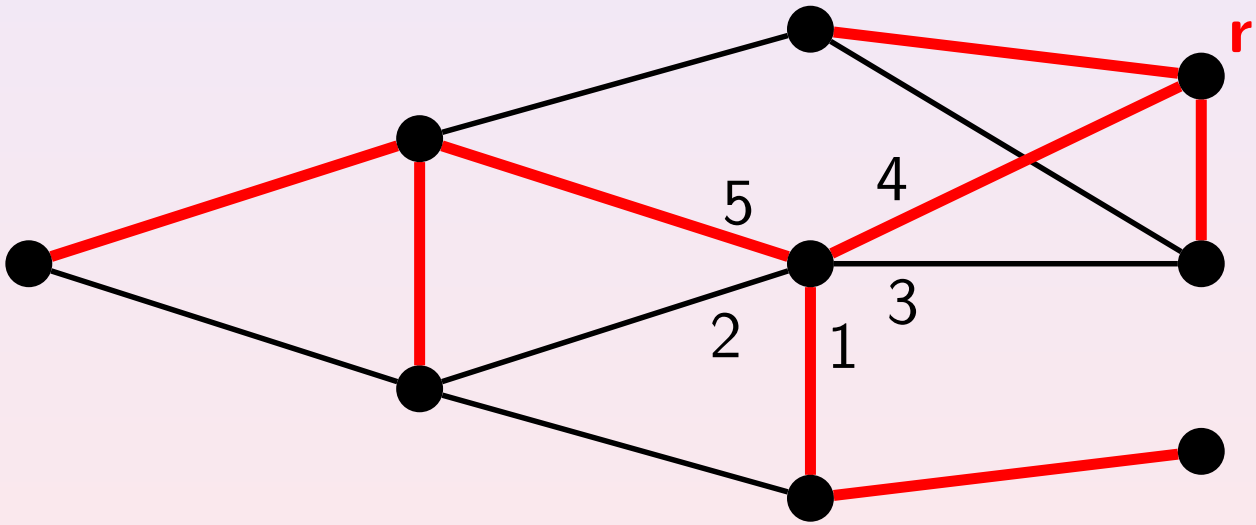
only indicate edge leading to the parent $\longrightarrow$ Δ colors



Rooted tree

Enhanced labeling

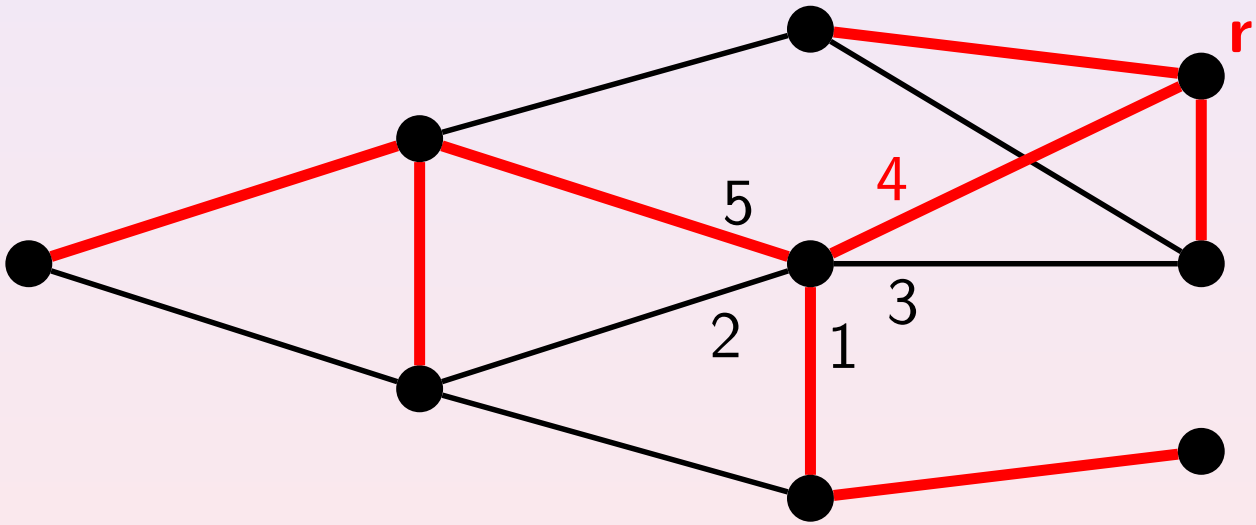
only indicate edge leading to the parent $\longrightarrow$ Δ colors



Rooted tree

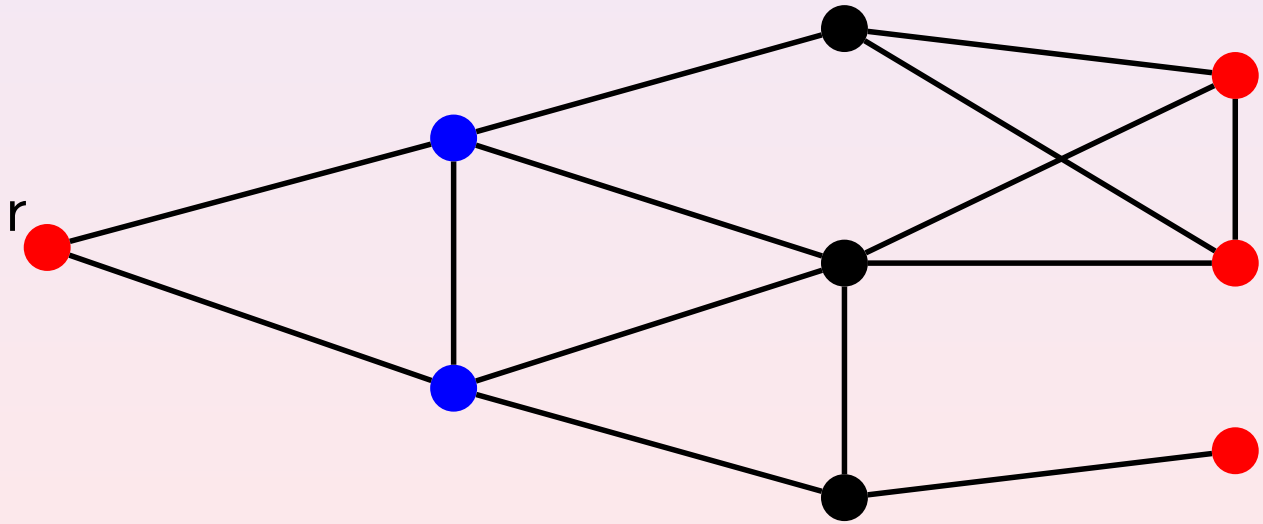
Enhanced labeling

only indicate edge leading to the parent $\longrightarrow$ Δ colors



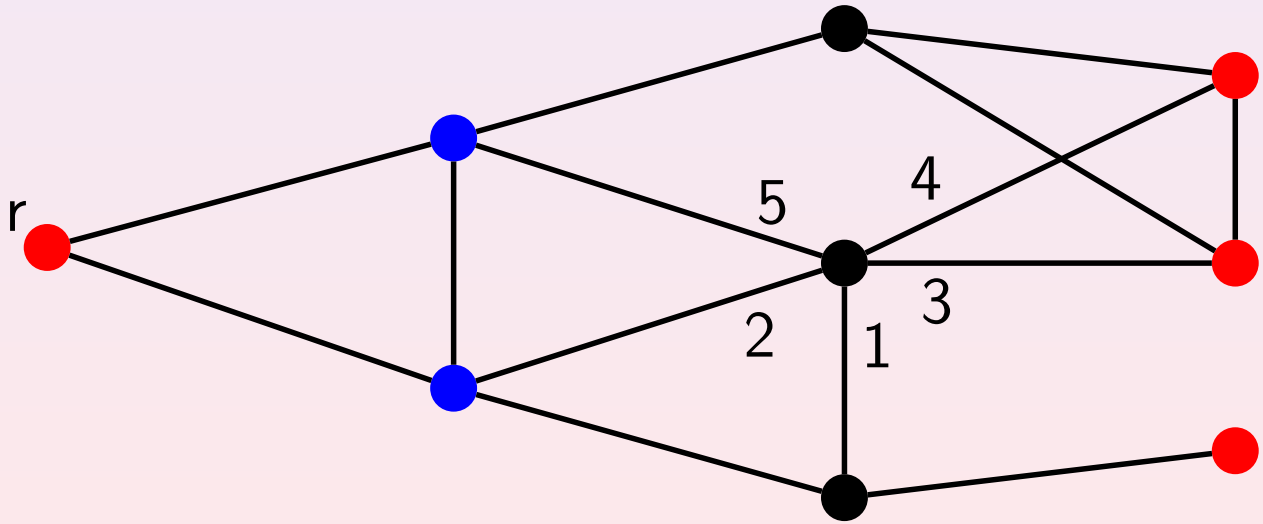
Three colors are enough

- choose arbitrarily a node as the root
- colors all nodes according to their distance d to the root
  - distance $d \cong 0[n]$ red
  - distance $d \cong 1[n]$ blue
  - distance $d \cong 2[n]$ black



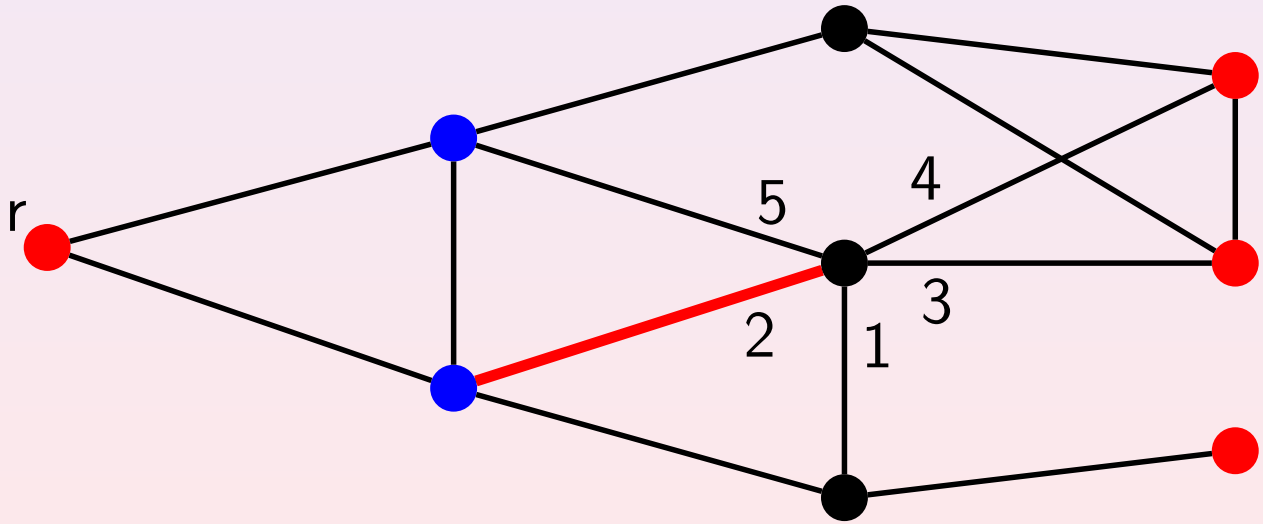
Three colors are enough

- choose arbitrarily a node as the root
- colors all nodes according to their distance d to the root
  - distance $d \cong 0[n]$ red
  - distance $d \cong 1[n]$ blue
  - distance $d \cong 2[n]$ black

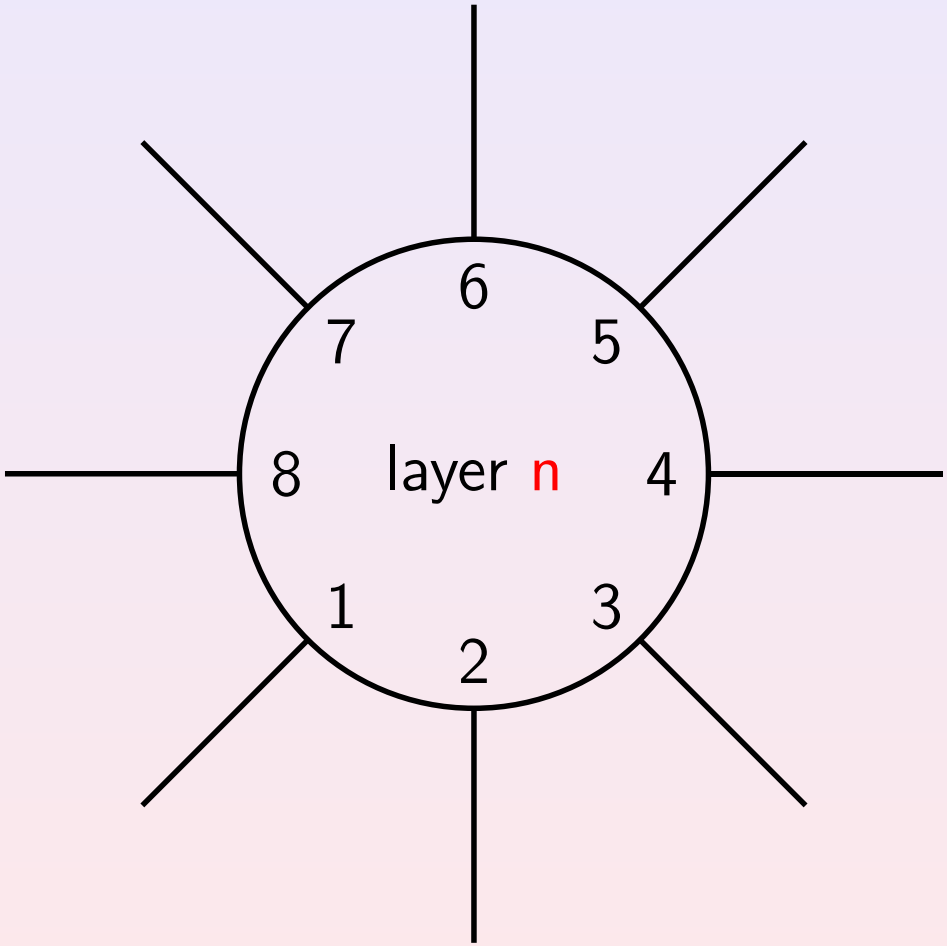


Three colors are enough

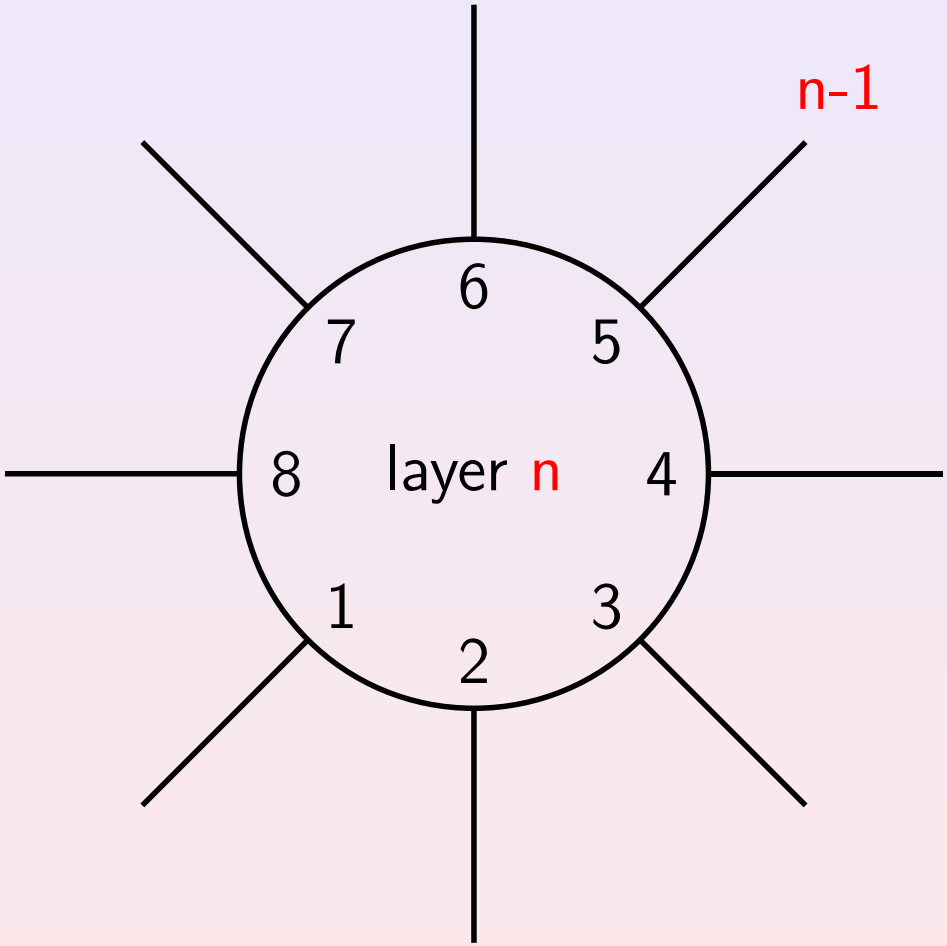
- choose arbitrarily a node as the root
- colors all nodes according to their distance d to the root
  - distance $d \cong 0[n]$ red
  - distance $d \cong 1[n]$ blue
  - distance $d \cong 2[n]$ black



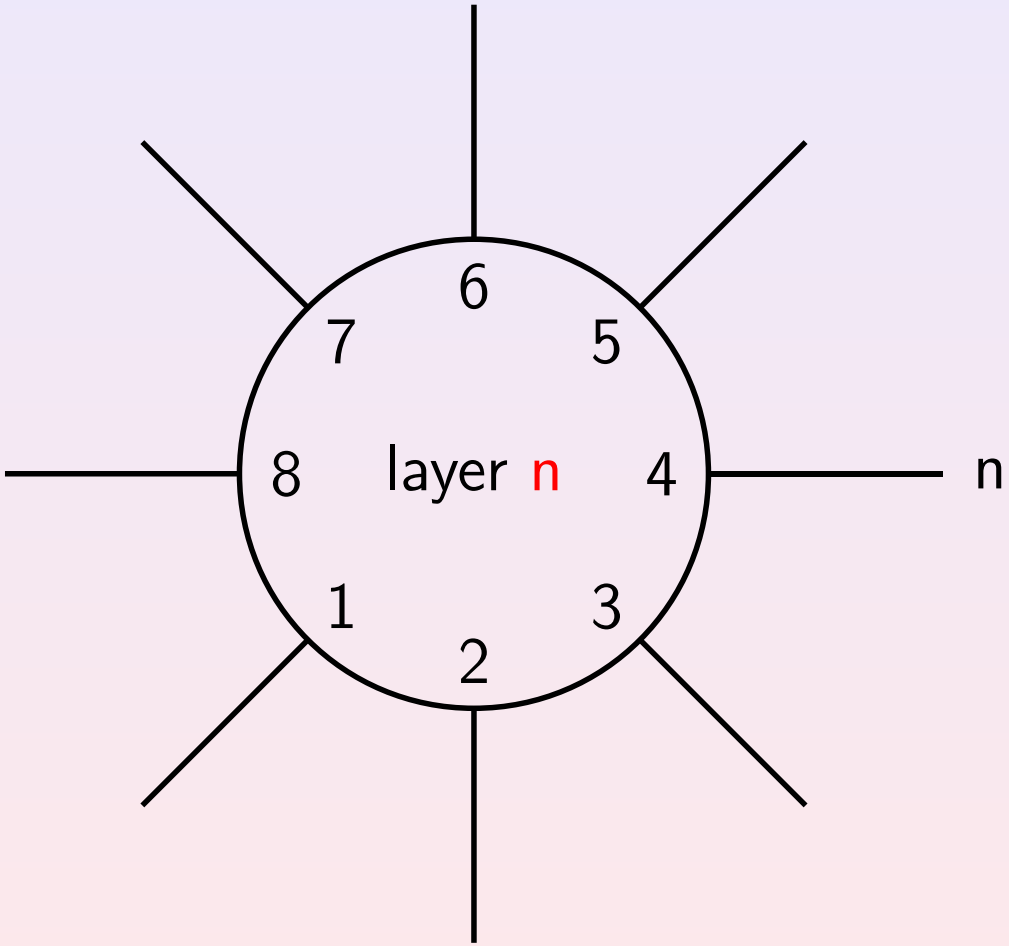
How to find the parent



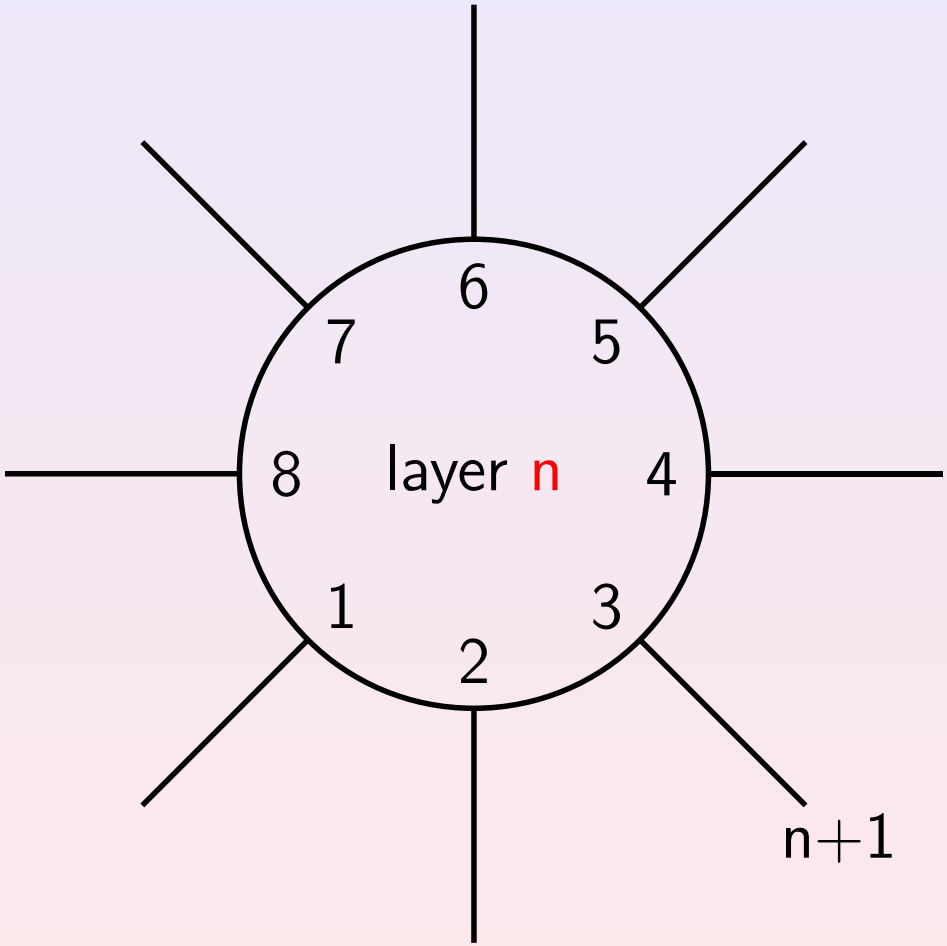
How to find the parent



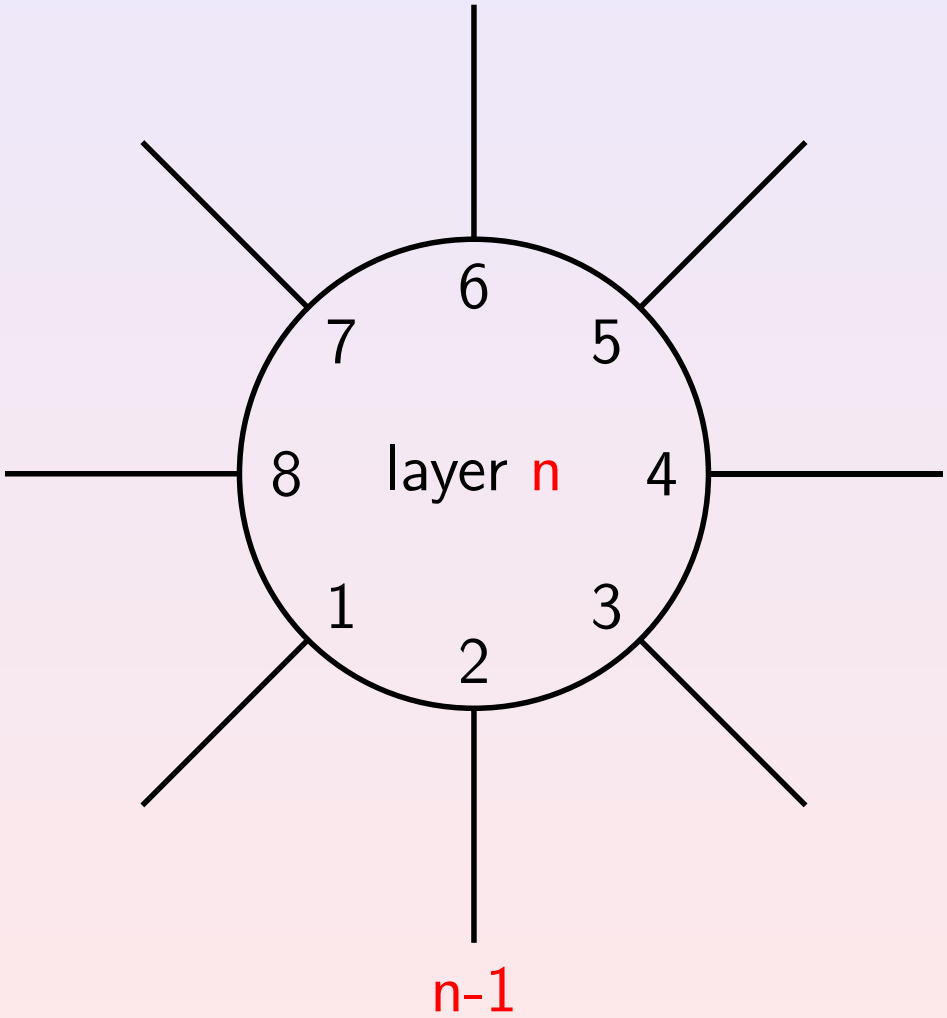
How to find the parent



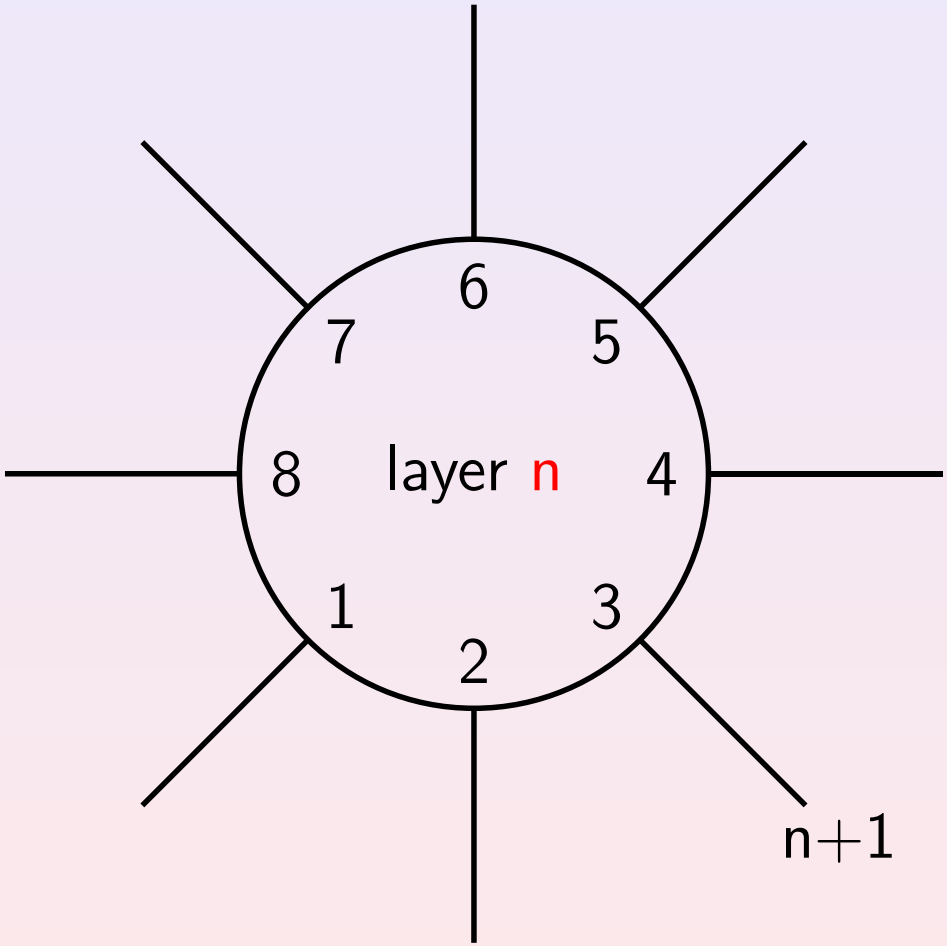
How to find the parent



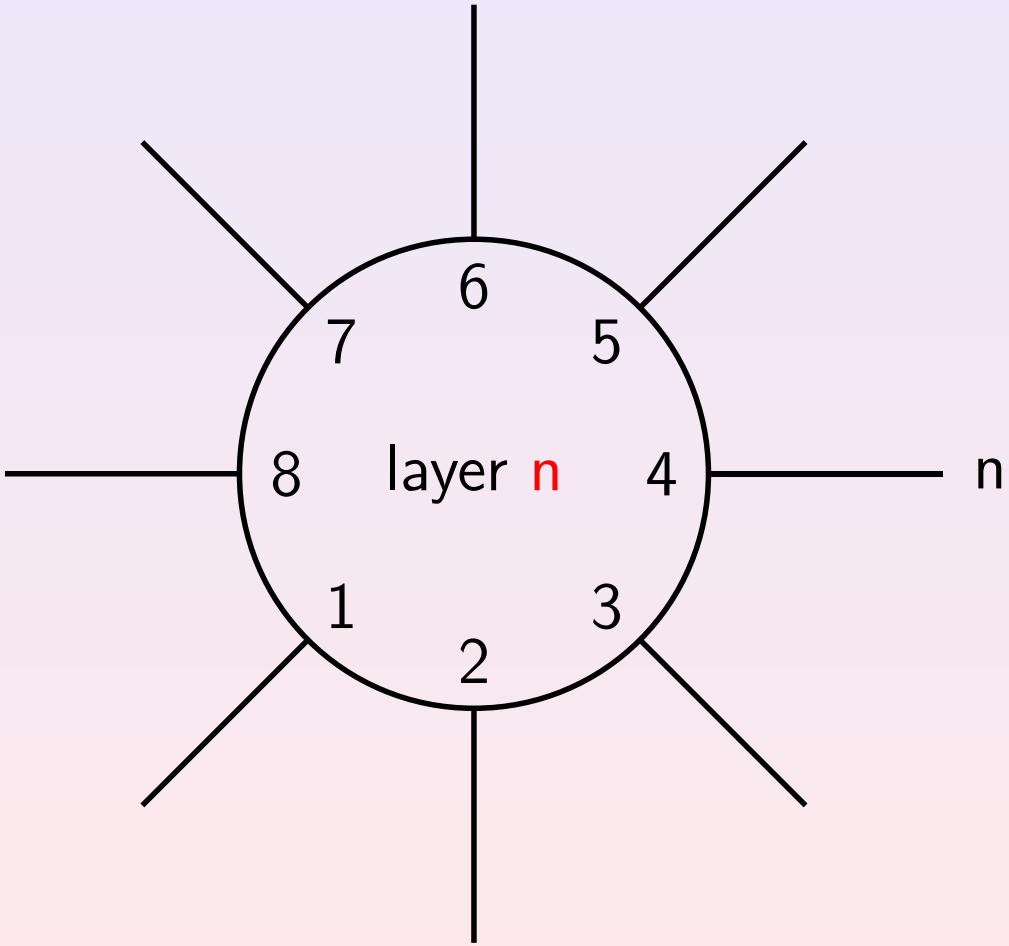
How to find the parent



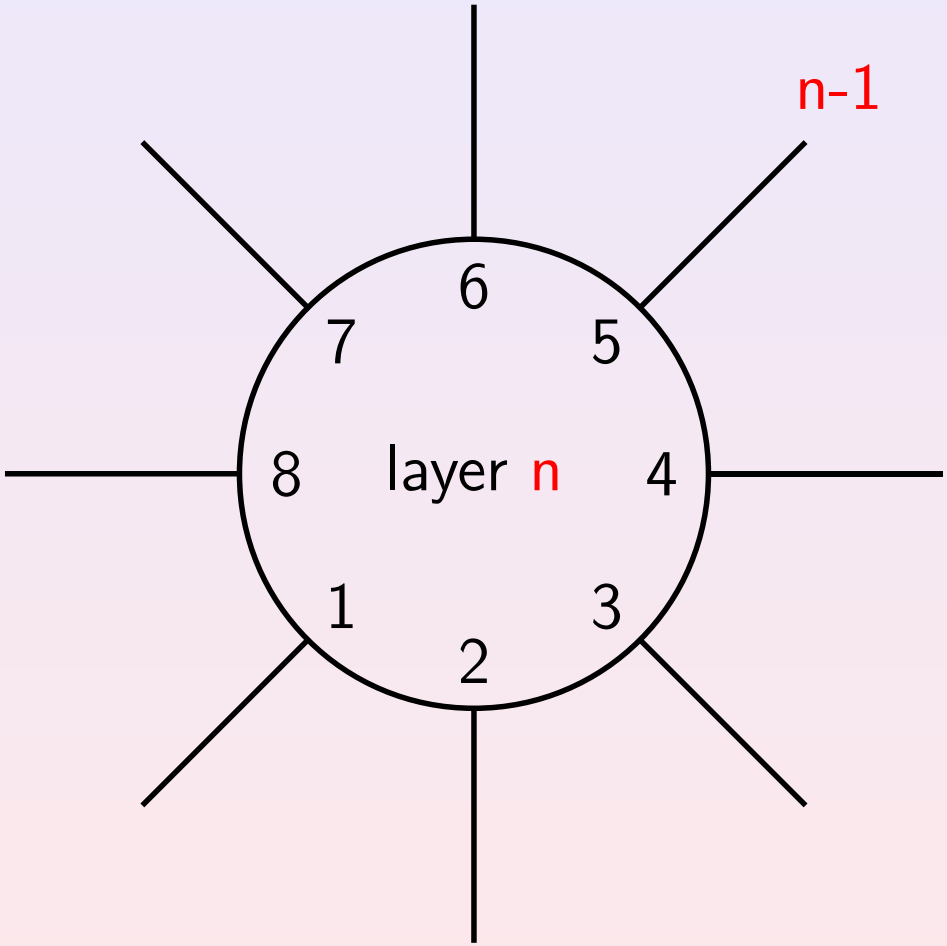
How to find the parent



How to find the parent



How to find the parent



Only two colors?

- Layer 1: red
- Layer 2: blue
- Layer 3: red
- Layer 4: red
- Layer 5: red
- Layer 6: blue
- Layer 7: blue
- Layer 8: blue

Outline

Open problem

Conclusion

- Three colors are sufficient for arbitrary graphs.
- Two colors are necessary and sufficient for graphs of constant degree.

Open problem

Are two colors sufficient for arbitrary graphs?

Open problem

Conclusion

- Three colors are sufficient for arbitrary graphs.
- Two colors are necessary and sufficient for graphs of constant degree.

Open problem

Are two colors sufficient for arbitrary graphs?