Setting Port Numbers for Fast Graph Exploration

David Ilcinkas

LRI, Université Paris-Sud, France

SIROCCO '06
July 3rd, 2006

Graph exploration

Graph exploration

A mobile entity has to visit every node of an unknown anonymous graph.

Periodic exploration by a finite automaton

A finite automaton has to visit every node infinitely often.

Performance criterion

To minimize the period

Graph exploration

Graph exploration

A mobile entity has to visit every node of an unknown anonymous graph.

Periodic exploration by a finite automaton

A finite automaton has to visit every node infinitely often.

Performance criterion

To minimize the period

Graph exploration

Graph exploration

A mobile entity has to visit every node of an unknown anonymous graph.

Periodic exploration by a finite automaton

A finite automaton has to visit every node infinitely often.

Performance criterion

To minimize the period

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, STACS 2006]

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, STACS 2006]

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

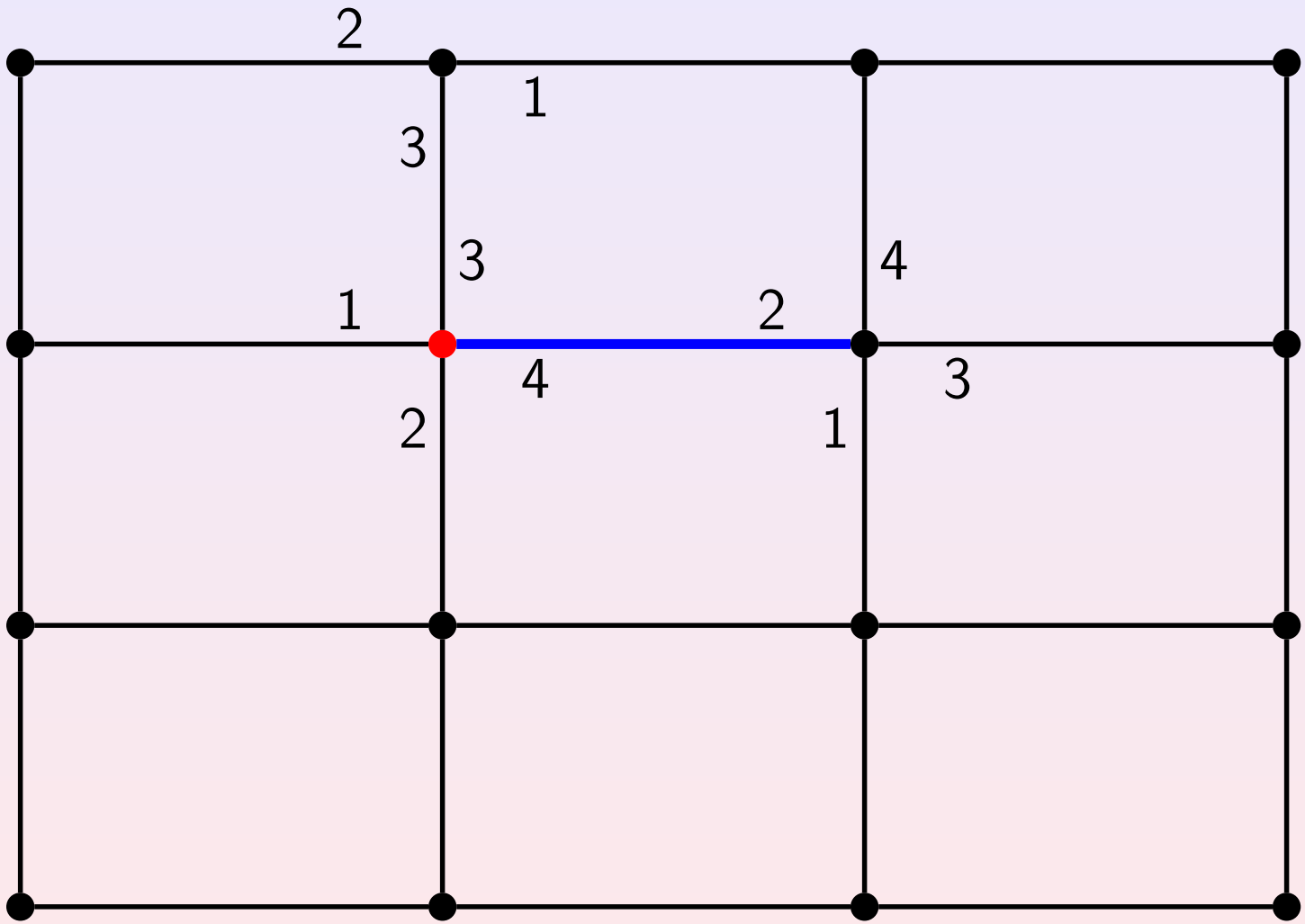
Unknown

- Unknown topology
- Unknown size (no upper bound)

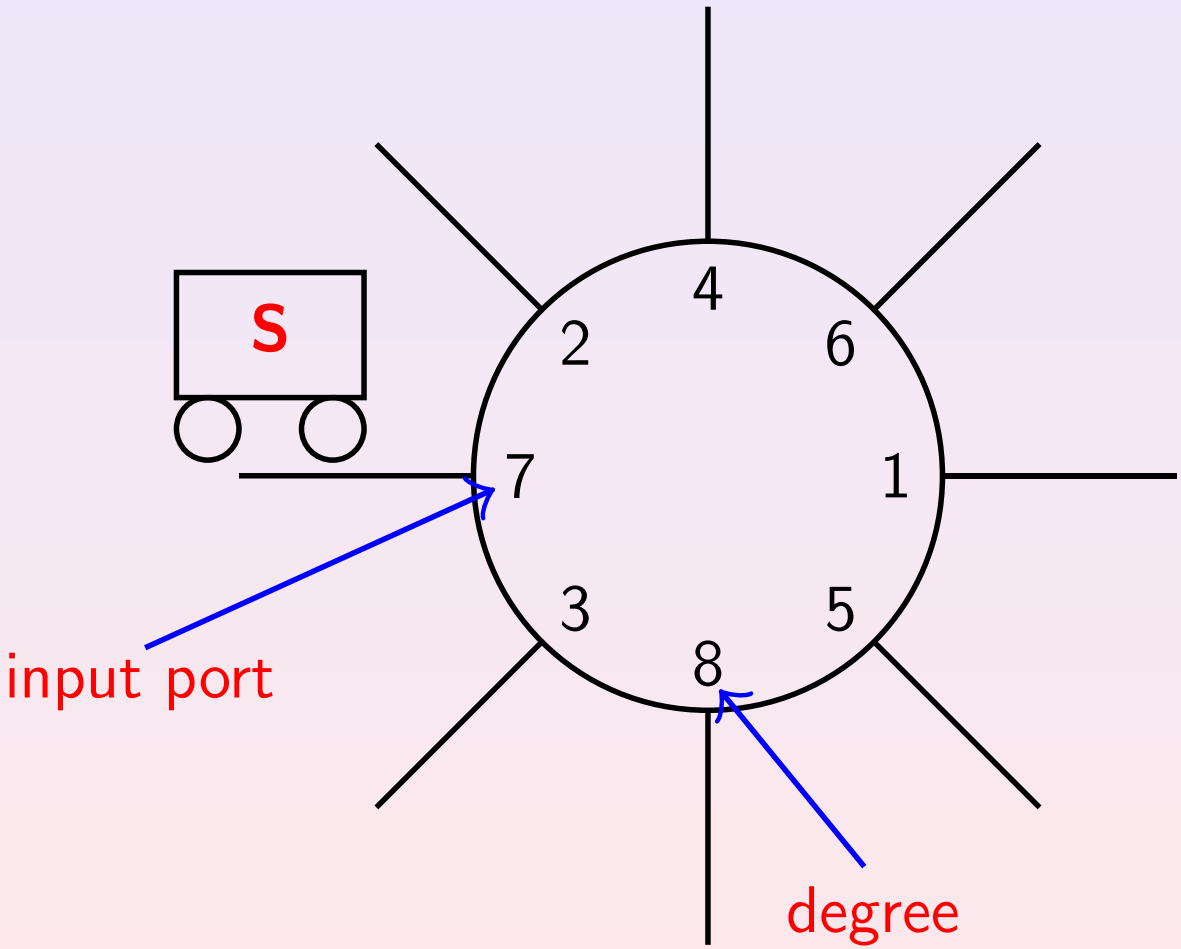
Anonymous

- No node labeling
- Local edge labeling

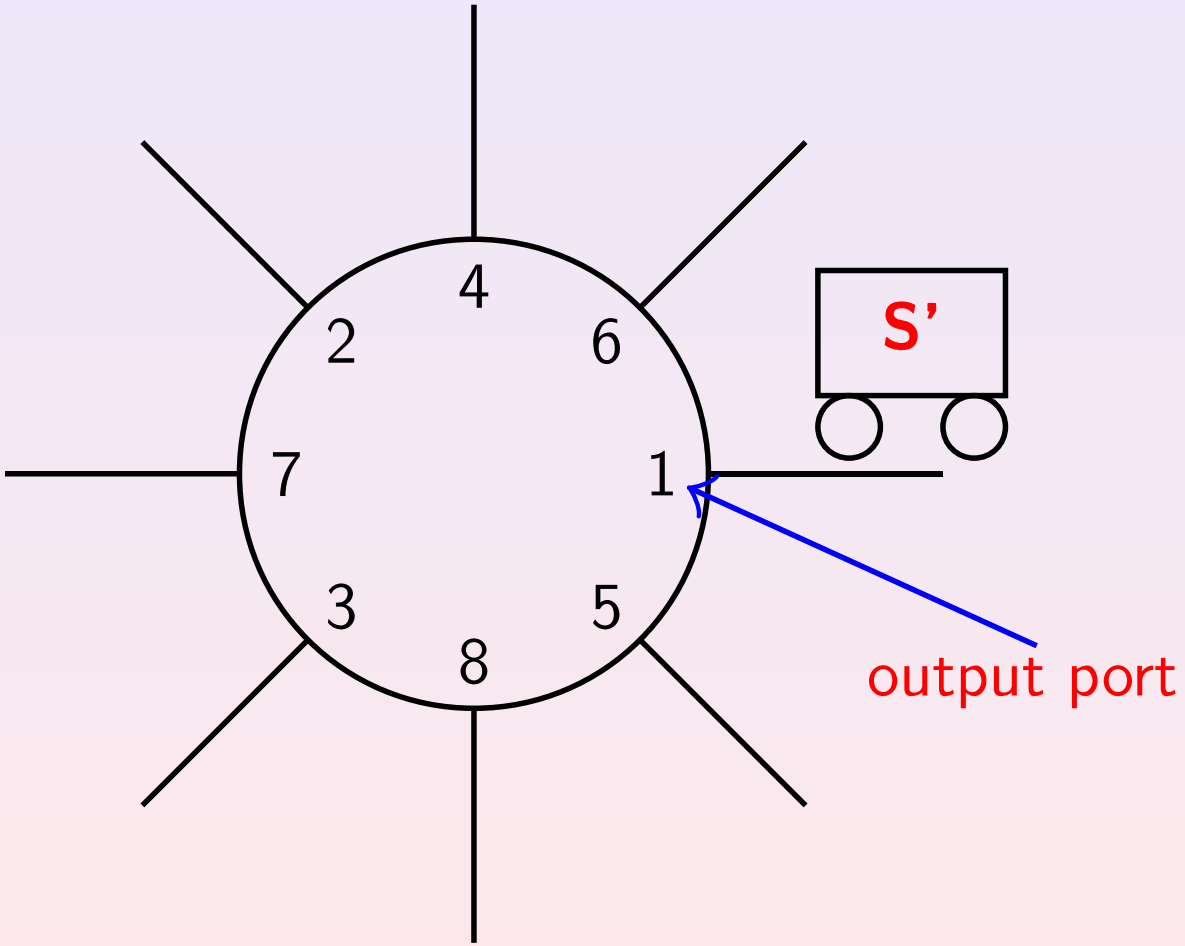
Example of an anonymous graph



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

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

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.

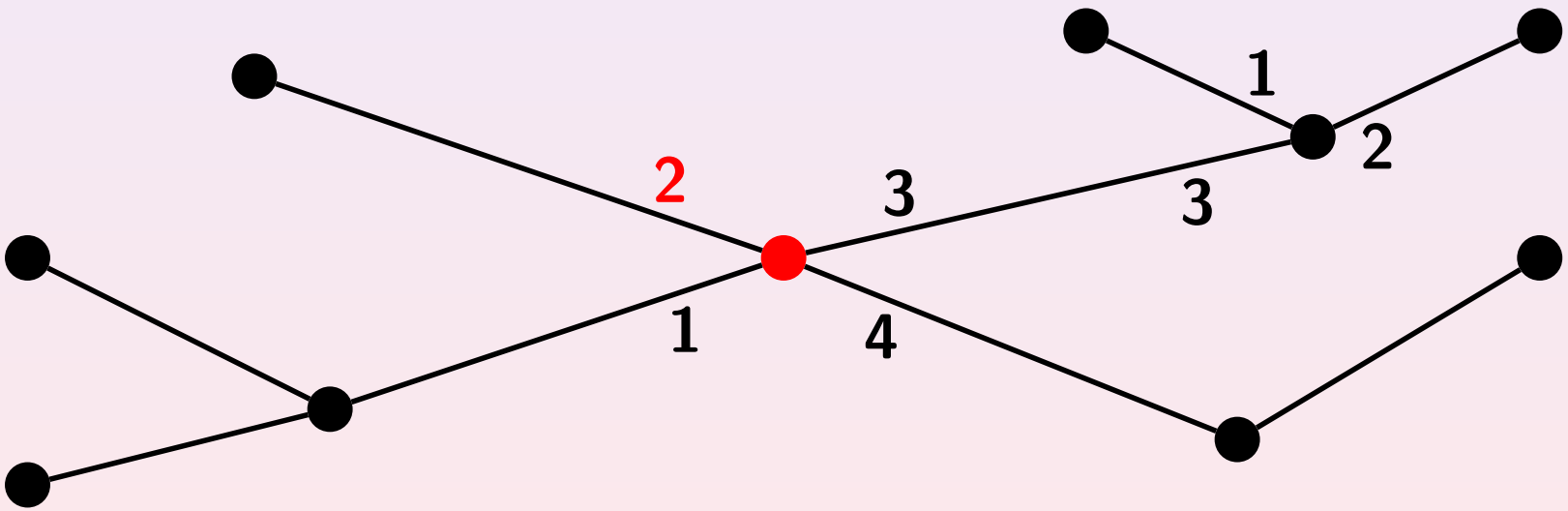
A JAG (Jumping Automaton for Graphs) is a team of finite automata that cooperate constantly. Moreover an automaton can jump to a vertex occupied by another automaton.

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

No JAG can explore all graphs.

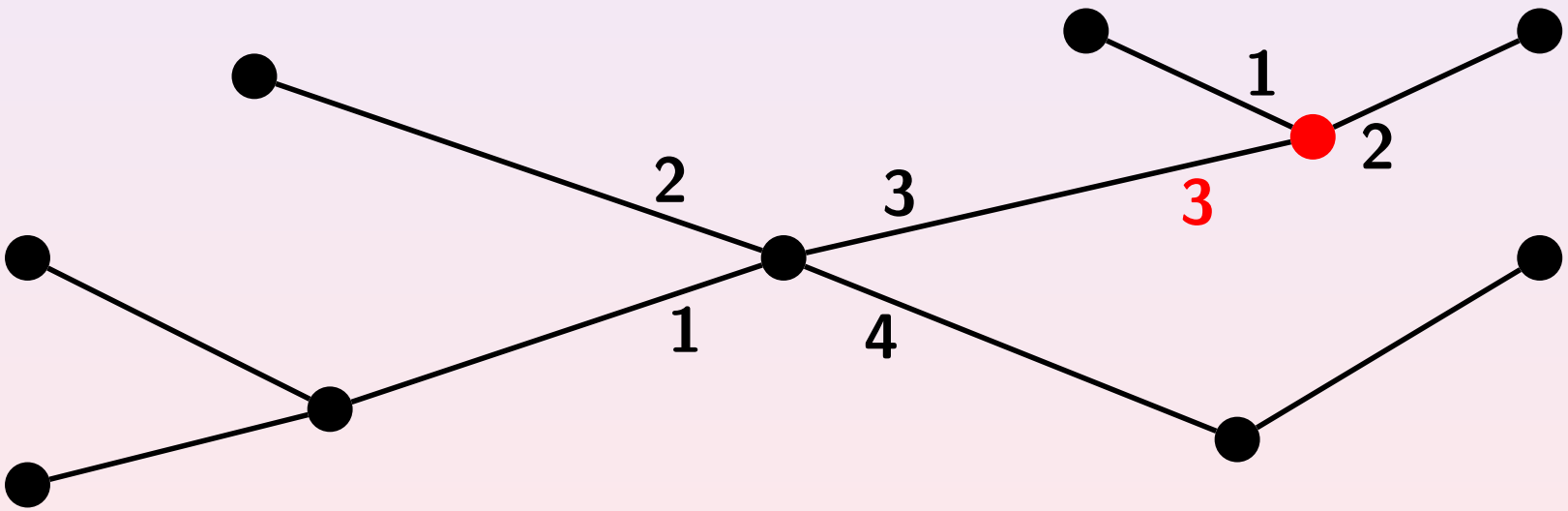
Universality for trees

An oblivious automaton (one single state) using the right-hand-on-the-wall rule ($i \mapsto i + 1$) explores all the trees.



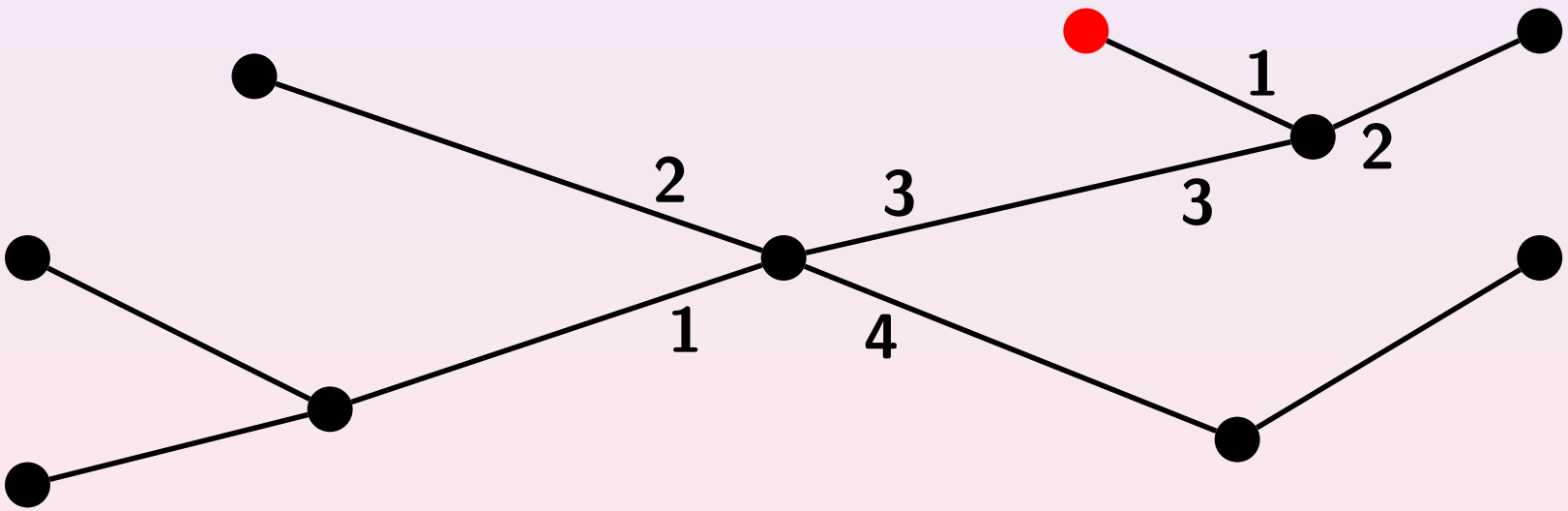
Universality for trees

An oblivious automaton (one single state) using the right-hand-on-the-wall rule ($i \mapsto i + 1$) explores all the trees.



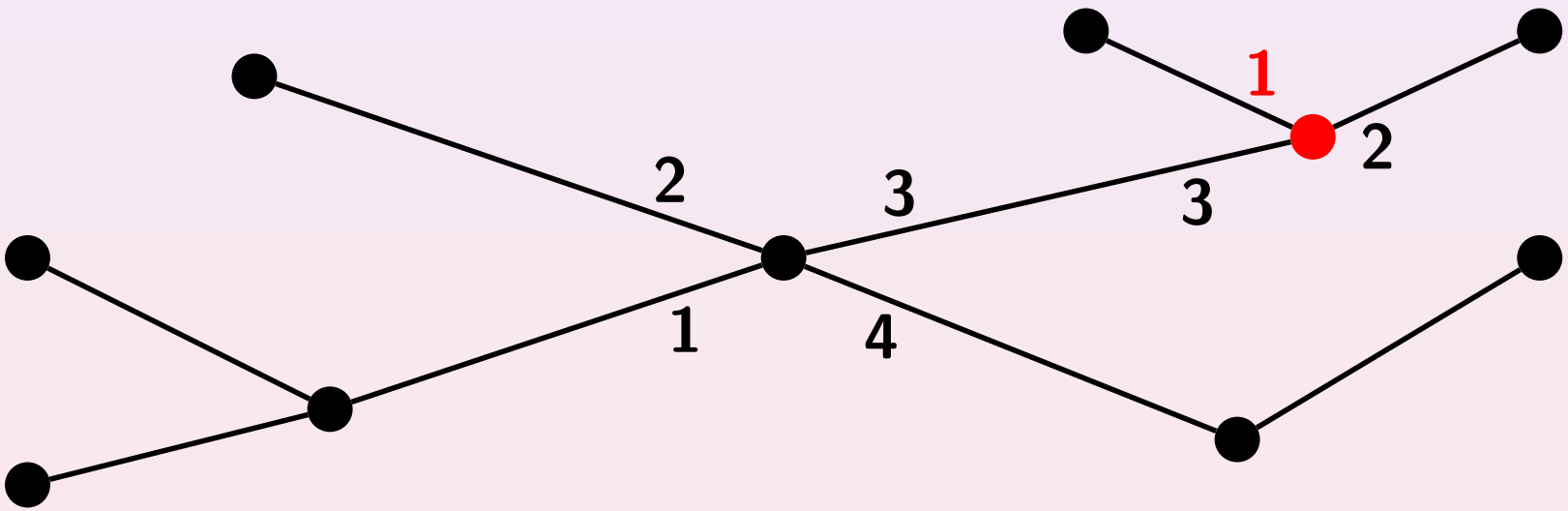
Universality for trees

An oblivious automaton (one single state) using the right-hand-on-the-wall rule ($i \mapsto i + 1$) explores all the trees.



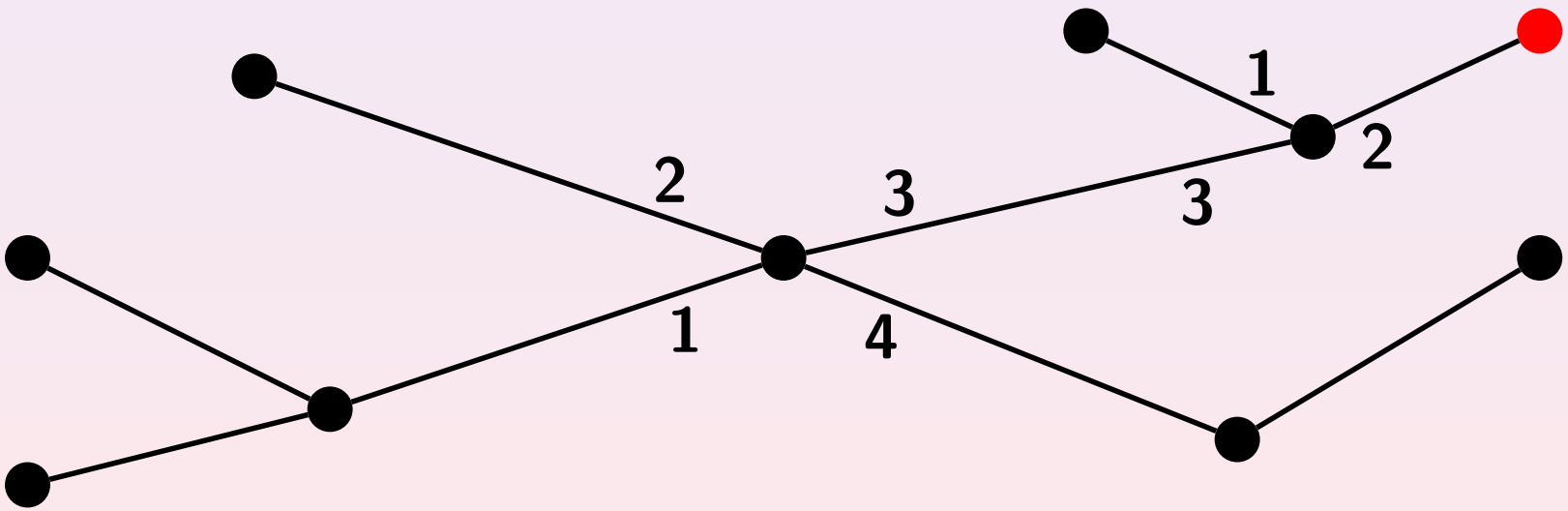
Universality for trees

An oblivious automaton (one single state) using the right-hand-on-the-wall rule ($i \mapsto i + 1$) explores all the trees.



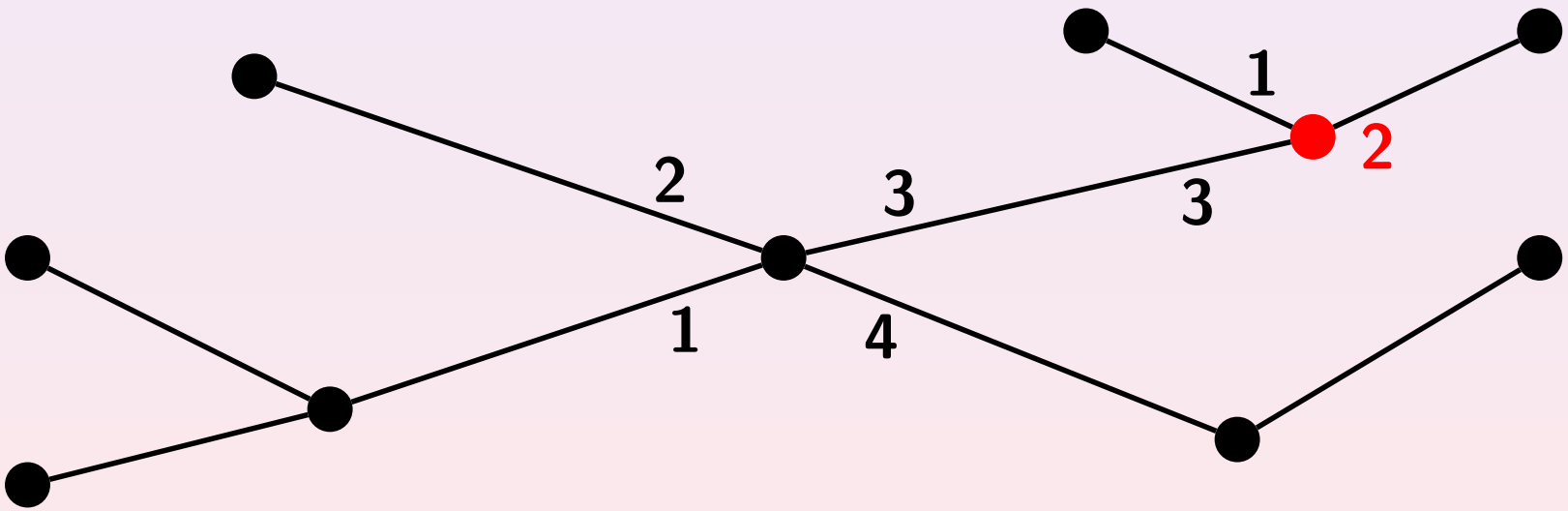
Universality for trees

An oblivious automaton (one single state) using the right-hand-on-the-wall rule ($i \mapsto i + 1$) explores all the trees.



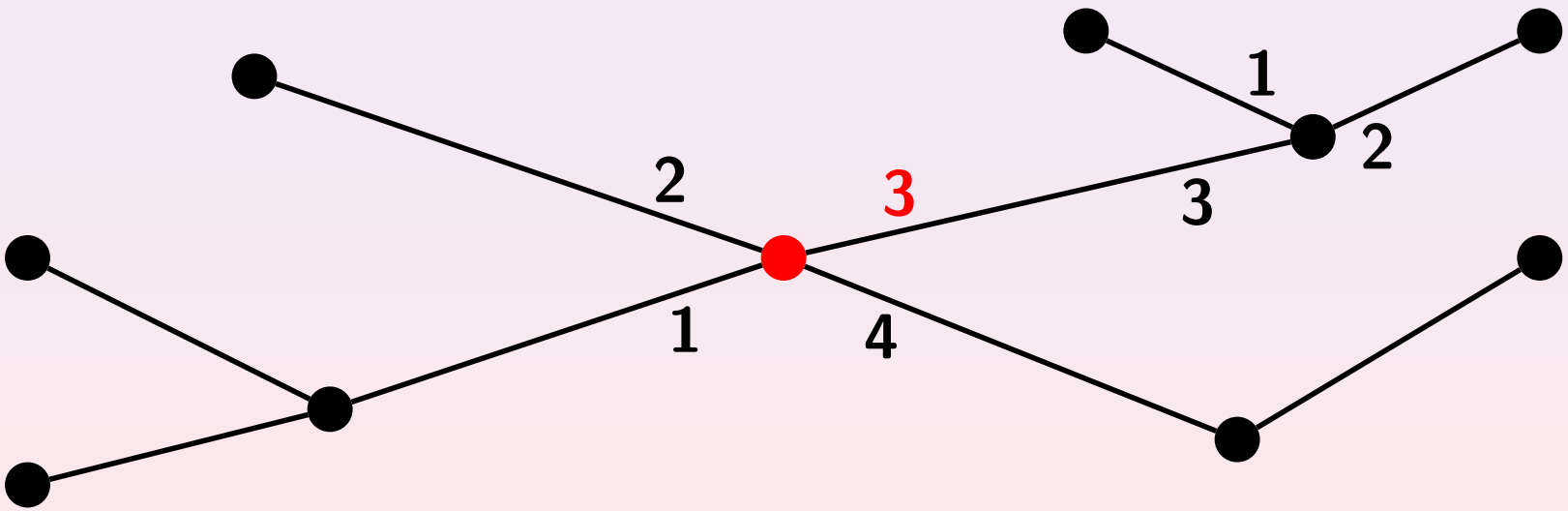
Universality for trees

An oblivious automaton (one single state) using the right-hand-on-the-wall rule ($i \mapsto i + 1$) explores all the trees.



Universality for trees

An oblivious automaton (one single state) using the right-hand-on-the-wall rule ($i \mapsto i + 1$) explores all the trees.



Coloring nodes

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.

Cohen, Fraigniaud, Ilcinkas, Korman, Peleg, ICALP, 2005
Label-Guided Graph Exploration by a Finite Automaton

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

Coloring nodes

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.

Cohen, Fraigniaud, Ilcinkas, Korman, Peleg, ICALP, 2005
Label-Guided Graph Exploration by a Finite Automaton

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

Setting port numbers

Model

- Port numbers are set to help the automaton.
- The automaton is ultimately simple : it is memoryless

Pelc, Panaite, Bui-Xuan, SIROCCO, 2005
Finding Short Right-Hand-on-the-Wall Walks in Graphs

There exist an algorithm for setting the port numbers, and an oblivious automaton using them, such that the automaton explores all graphs of size n within the period $10n$.

Setting port numbers

Model

- Port numbers are set to help the automaton.
- The automaton is ultimately simple : it is memoryless

Dobrev, Jansson, Sadakane, Sung, SIROCCO, 2005
Finding Short Right-Hand-on-the-Wall Walks in Graphs

There exist an algorithm for setting the port numbers, and an oblivious automaton using them, such that the automaton explores all graphs of size n within the period $10n$.

Outline

Our model

Model

- An algorithm sets the port numbers to help the automaton.
- The automaton is restricted to be finite (but not necessarily oblivious).

Question

What is the minimum function $\pi(n)$ such that there exist an algorithm for setting the local orientation, and a finite automaton using it, such that the automaton explores all graphs of size n within the period $\pi(n)$?

Dobrev et al.

$\pi(n) \leq 10n$

Our model

Model

- An algorithm sets the port numbers to help the automaton.
- The automaton is restricted to be finite (but not necessarily oblivious).

Question

What is the mimimum function $\pi(n)$ such that there exist an algorithm for setting the local orientation, and a finite automaton using it, such that the automaton explores all graphs of size n within the period $\pi(n)$?

Dobrev et al.

$\pi(n) \leq 10n$

Our model

Model

- An algorithm sets the port numbers to help the automaton.
- The automaton is restricted to be finite (but not necessarily oblivious).

Question

What is the mimimum function $\pi(n)$ such that there exist an algorithm for setting the local orientation, and a finite automaton using it, such that the automaton explores all graphs of size n within the period $\pi(n)$?

Dobrev et al.

$\pi(n) \leq 10n$

Our results

Theorem

$\pi(n) \leq 4n - 2$

More precisely

- Very simple algorithm based on a spanning tree
- Three-state automaton
- Performance independent from the initial state and initial position of the automaton

Additional properties

- Distributed algorithm
- Dynamic environment

Our results

Theorem

$\pi(n) \leq 4n - 2$

More precisely

- Very simple algorithm based on a spanning tree
- Three-state automaton
- Performance independent from the initial state and initial position of the automaton

Additional properties

- Distributed algorithm
- Dynamic environment

Our results

Theorem

$\pi(n) \leq 4n - 2$

More precisely

- Very simple algorithm based on a spanning tree
- Three-state automaton
- Performance independent from the initial state and initial position of the automaton

Additional properties

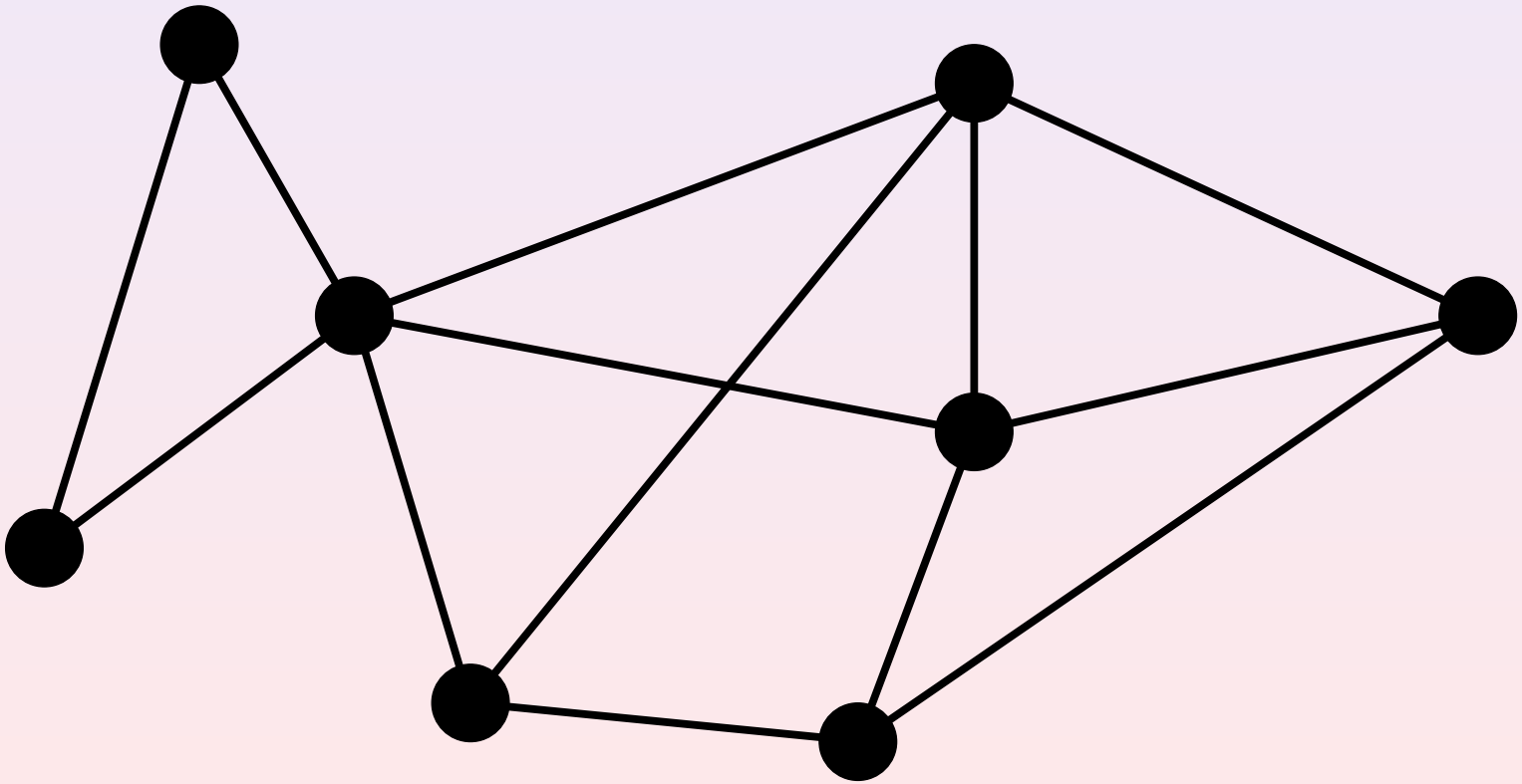
- Distributed algorithm
- Dynamic environment

Outline

Algorithm

Port numbers compatible with a spanning tree T

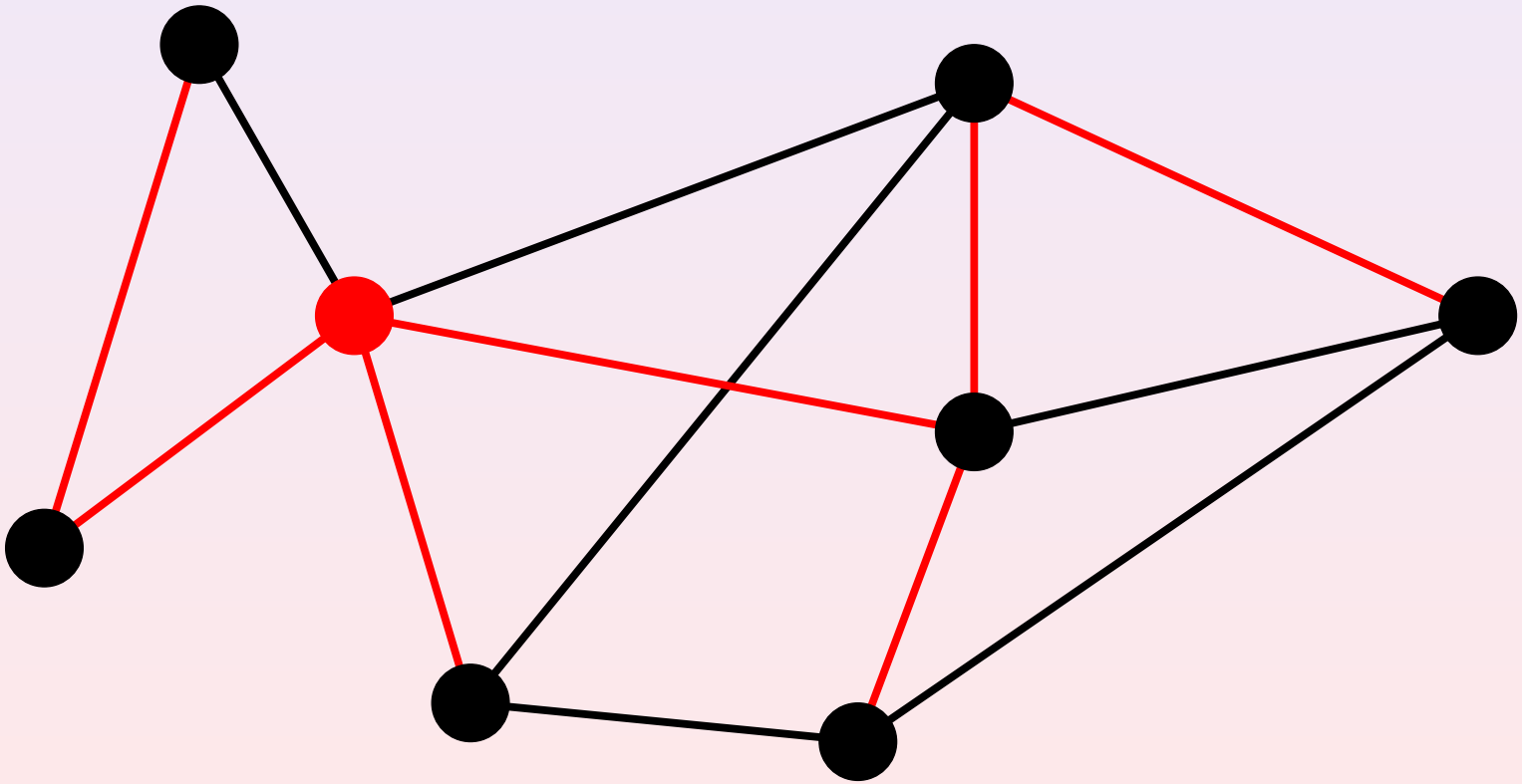
- edge e is in T $\iff$ at least one of its port numbers is 1;
- the edges belonging to T have the smallest port numbers.



Algorithm

Port numbers compatible with a spanning tree T

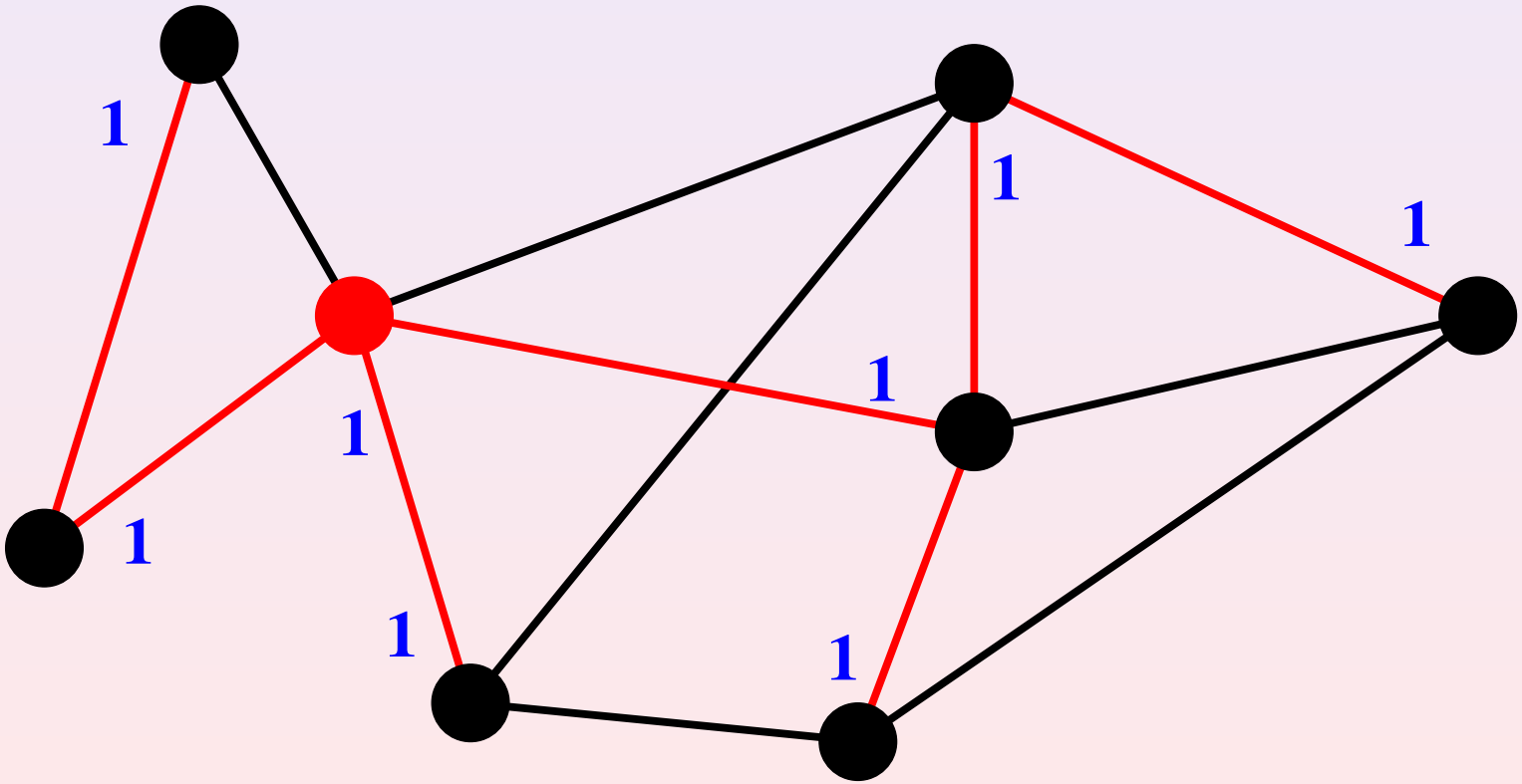
- edge e is in T $\iff$ at least one of its port numbers is 1;
- the edges belonging to T have the smallest port numbers.



Algorithm

Port numbers compatible with a spanning tree T

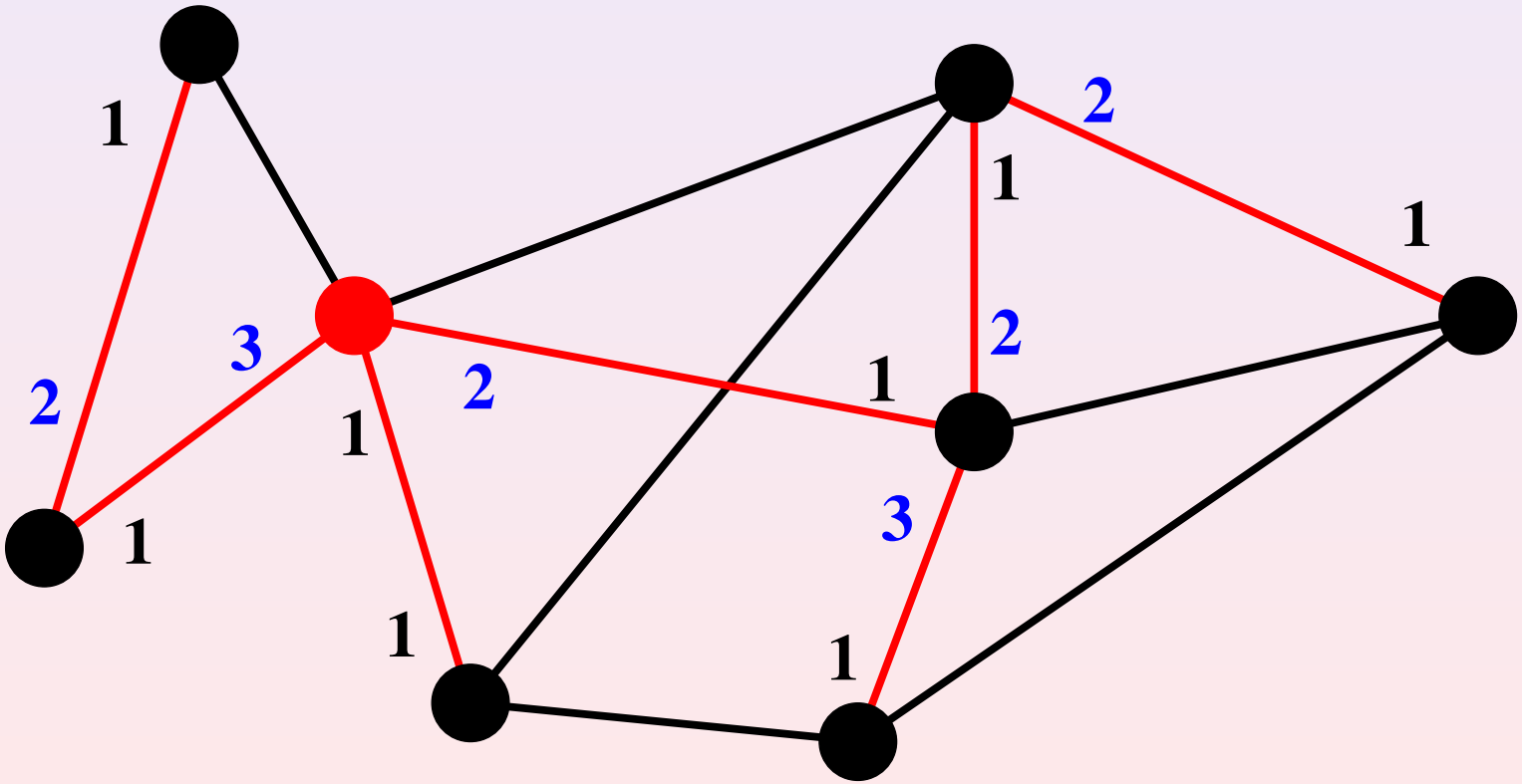
- edge e is in T $\iff$ at least one of its port numbers is 1;
- the edges belonging to T have the smallest port numbers.



Algorithm

Port numbers compatible with a spanning tree T

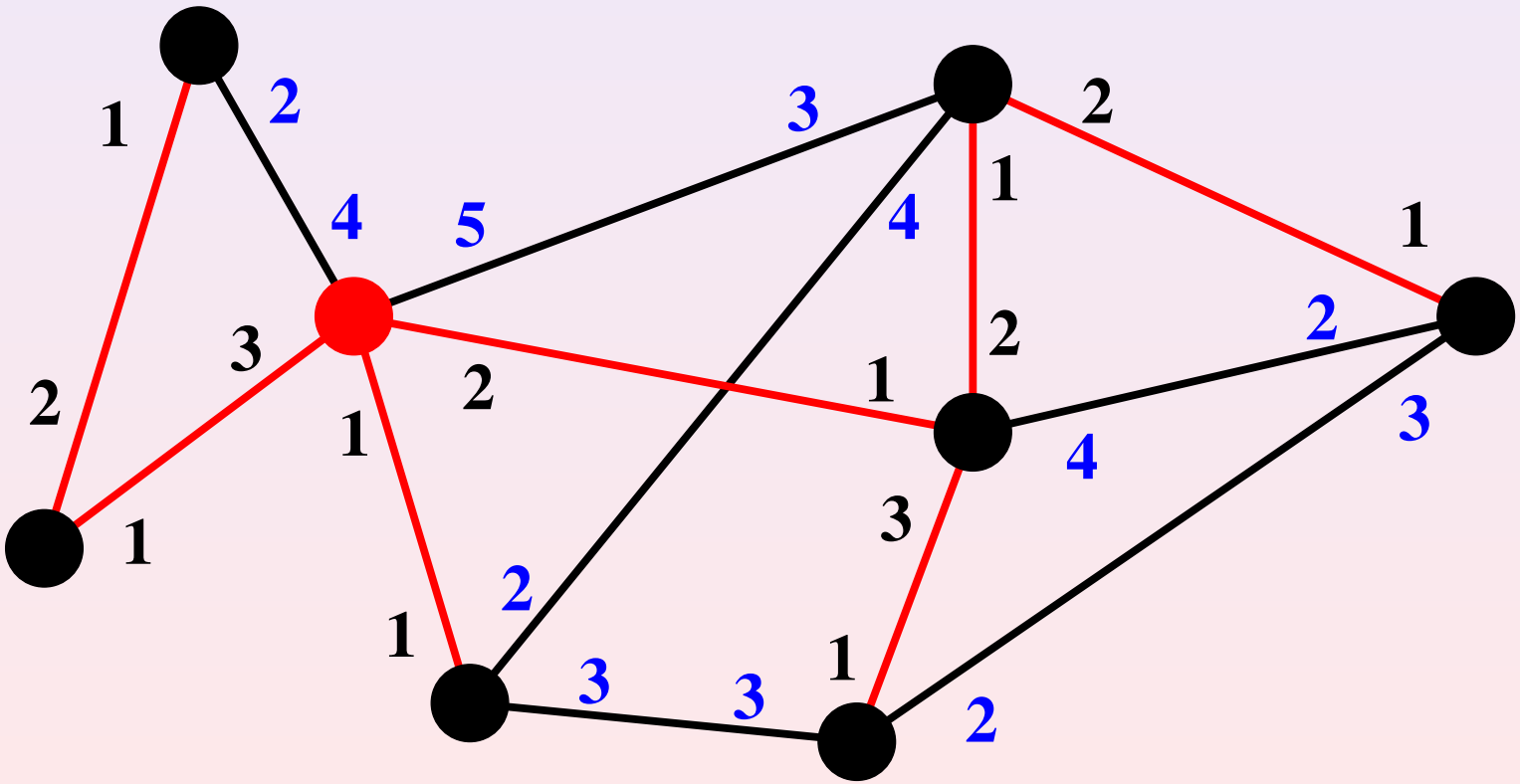
- edge e is in T $\iff$ at least one of its port numbers is 1;
- the edges belonging to T have the smallest port numbers.



Algorithm

Port numbers compatible with a spanning tree T

- edge e is in T $\iff$ at least one of its port numbers is 1;
- the edges belonging to T have the smallest port numbers.



Automaton

States

- N: Normal
- T: Test
- B: Backtrack

Transition function

$$f(N,i,d) = \begin{cases} (N,1) & \text{if } i = d \\ (T,i+1) & \text{if } i \neq d \end{cases}$$

$$f(T,i,d) = \begin{cases} (N,1) & \text{if } i = 1 \text{ and } d = 1 \\ (T,i+1) & \text{if } i = 1 \text{ and } d \neq 1 \\ (B,i) & \text{if } i \neq 1 \end{cases}$$

$$f(B,i,d) = (N,1)$$

Automaton

States

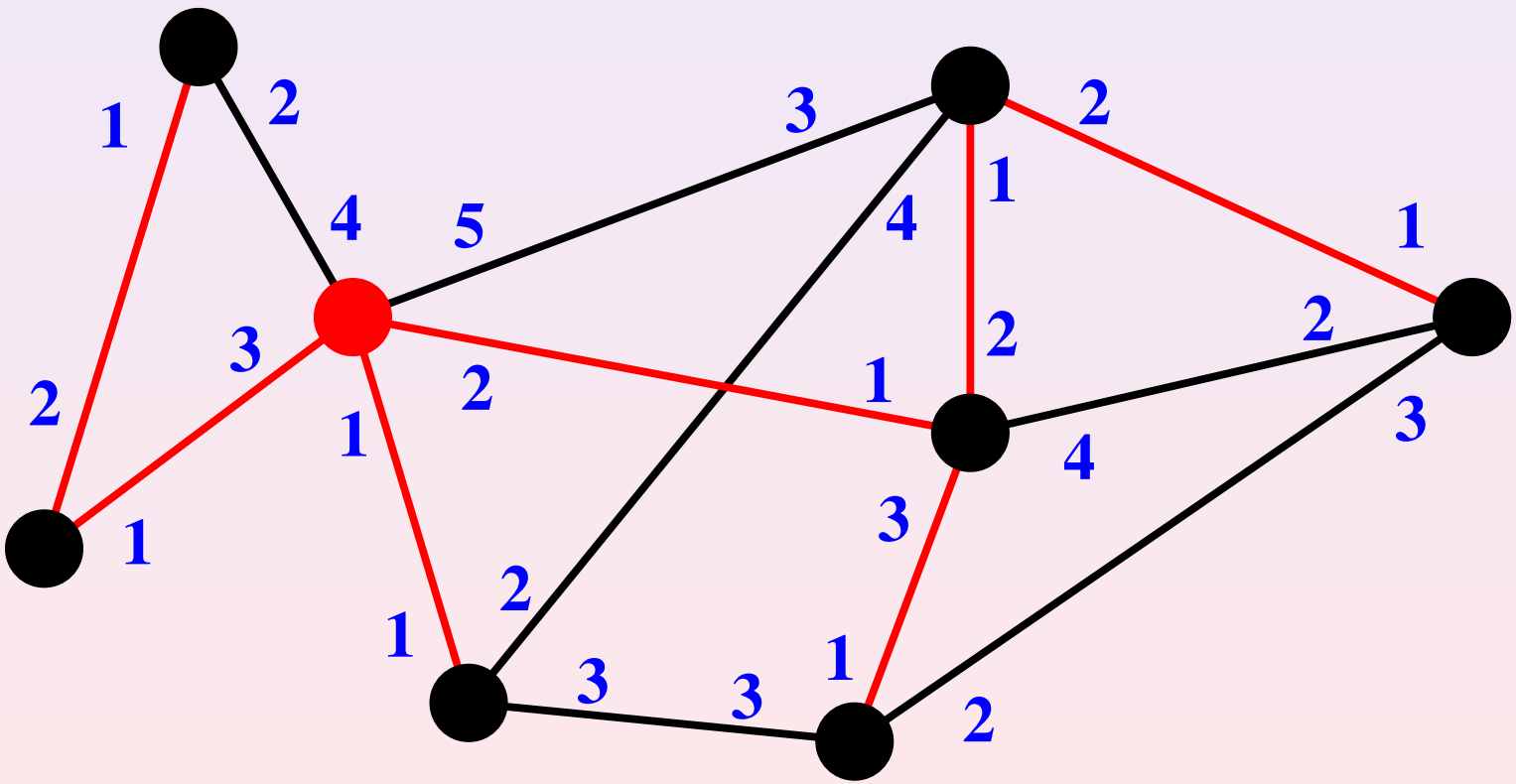
- N: Normal
- T: Test
- B: Backtrack

Transition function

$$
\begin{aligned}
f(N,i,d) &= \begin{cases} (N,1) & \text{if } i = d \\ (T,i+1) & \text{if } i \neq d \end{cases} \\
f(T,i,d) &= \begin{cases} (N,1) & \text{if } i = 1 \text{ and } d = 1 \\ (T,i+1) & \text{if } i = 1 \text{ and } d \neq 1 \\ (B,i) & \text{if } i \neq 1 \end{cases} \\
f(B,i,d) &= (N,1)
\end{aligned}
$$

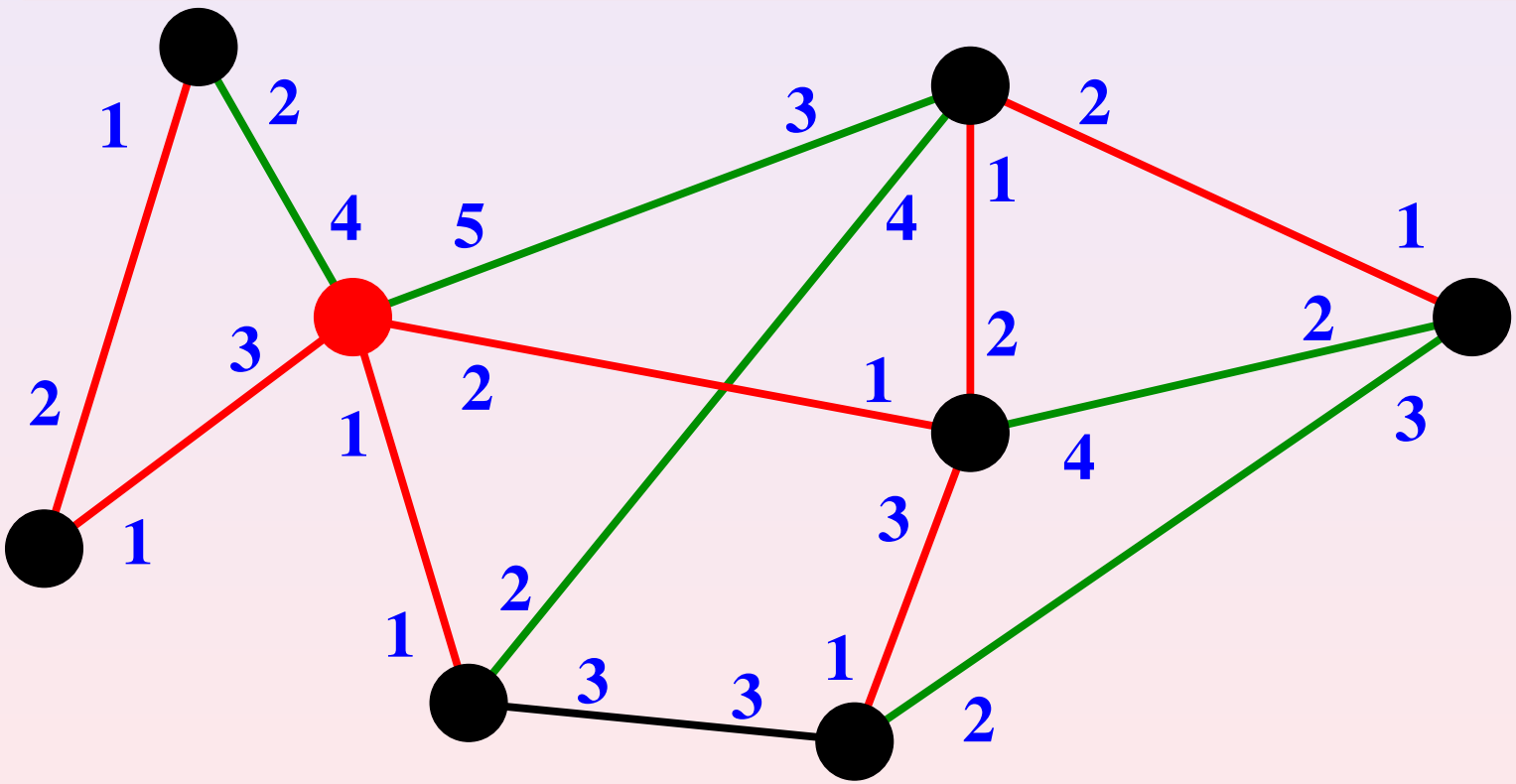
Example

- If on a tree-edge: right-hand-on-the-wall
- If not on a tree-edge: backtrack



Example

- If on a tree-edge: right-hand-on-the-wall
- If not on a tree-edge: backtrack



Outline

Open problem

Conclusion

$\pi(n) \leq 4n - 2$

Conjecture

$\pi(n) = 4n - O(1)$

Open problems

- $\pi(n)$ when the automaton is restricted to be oblivious?
- Find a fully self-stabilizing pair algorithm/automaton

Open problem

Conclusion

$\pi(n) \leq 4n - 2$

Conjecture

$\pi(n) = 4n - O(1)$

Open problems

- $\pi(n)$ when the automaton is restricted to be oblivious?
- Find a fully self-stabilizing pair algorithm/automaton

Open problem

Conclusion

$\pi(n) \leq 4n - 2$

Conjecture

$\pi(n) = 4n - O(1)$

Open problems

- $\pi(n)$ when the automaton is restricted to be oblivious?
- Find a fully self-stabilizing pair algorithm/automaton