Exploration par un automate fini de réseaux anonymes étiquetés

Reuven Cohen[1] Pierre Fraigniaud[2] David Ilcinkas[3]
Amos Korman[1] David Peleg[1]

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

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

[3]LRI, Université Paris-Sud, France

Journées Graphes et Algorithmes
3 novembre 2005

Exploration de graphes

But

Une entité mobile doit traverser chaque arête d'un graphe inconnu et anonyme.

Motivations (1)

Exploration par agents mobiles

- Robot physique : exploration d'endroits inaccessibles à l'homme
- Agent logiciel : maintenance d'un réseau informatique

Equivalence entre logique and automates (Engelfriet, Hoogeboom)

Par la caractérisation de langages de chaînes, d'arbres ou de graphes

- Automates munis de cailloux "imbriqués"
- Logique du premier ordre avec fermeture transitive

Motivations (1)

Exploration par agents mobiles

- Robot physique : exploration d'endroits inaccessibles à l'homme
- Agent logiciel : maintenance d'un réseau informatique

Equivalence entre logique and automates (Engelfriet, Hoogeboom)

Par la caractérisation de langages de chaînes, d'arbres ou de graphes

- Automates munis de cailloux "imbriqués"
- Logique du premier ordre avec fermeture transitive

Motivations (2)

USTCON (undirected st-connectivity)

- $G = \{V, E\}$ graphe non orienté
- $s, t \in V$ deux sommets de G

s et t sont-ils dans la même composante connexe ?

- 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)

- $G = \{V, E\}$ graphe non orienté
- $s, t \in V$ deux sommets de G

s et t sont-ils dans la même composante connexe ?

- 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-complet

- $G = \{V, E\}$ graphe non orienté
- $s, t \in V$ deux sommets de G

s et t sont-ils dans la même composante connexe ?

- 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-complet

- $G = \{V, E\}$ graphe non orienté
- $s, t \in V$ deux sommets de G

s et t sont-ils dans la même composante connexe ?

- 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

Inconnu, Anonyme

Inconnu

- topologie inconnue
- pas de bornes sur la taille

Anonyme

- sommets sans identifiant
- étiquetage local des arêtes

Inconnu, Anonyme

Inconnu

- topologie inconnue
- pas de bornes sur la taille

Anonyme

- sommets sans identifiant
- étiquetage local des arêtes

Inconnu, Anonyme

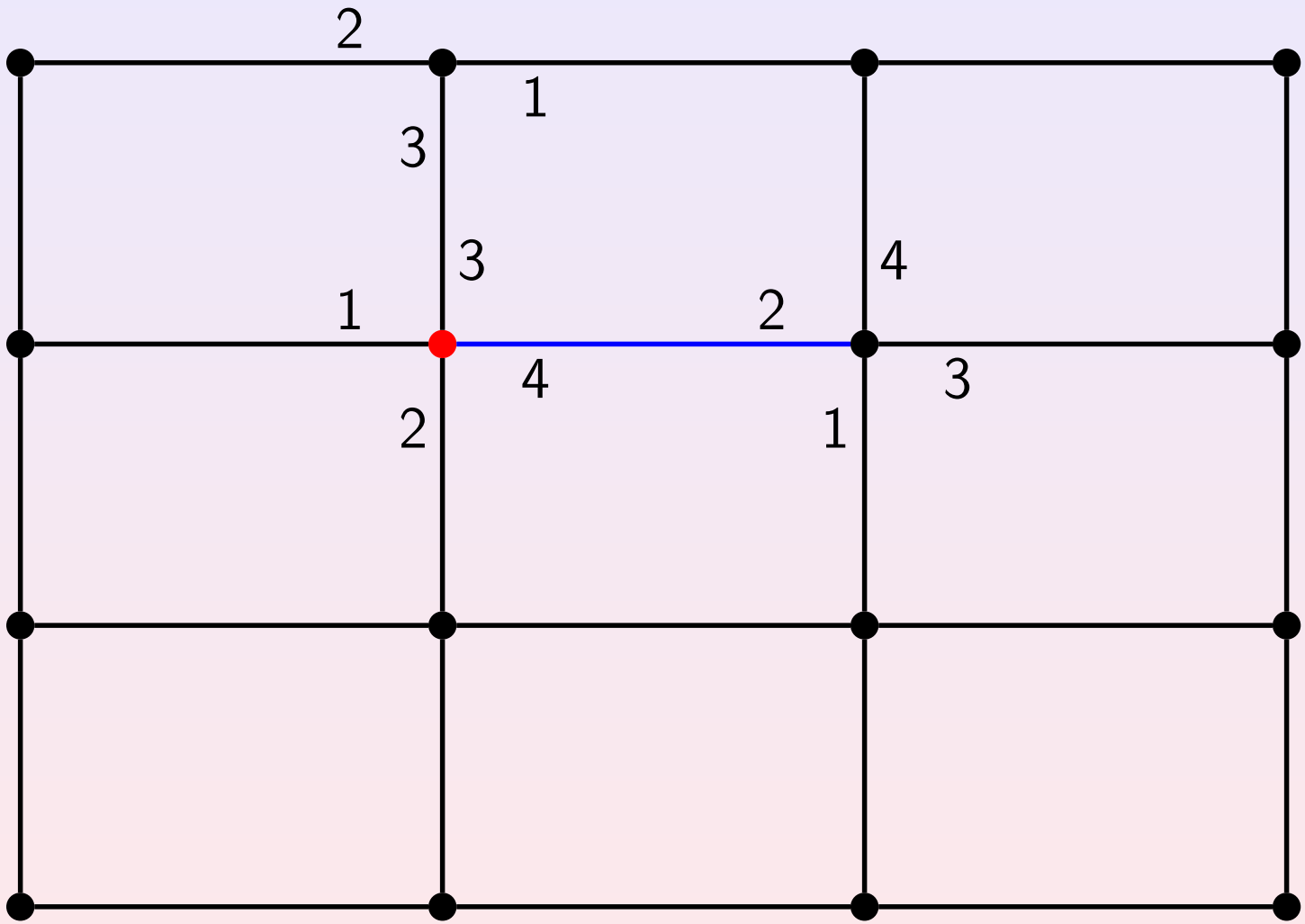
Inconnu

- topologie inconnue
- pas de bornes sur la taille

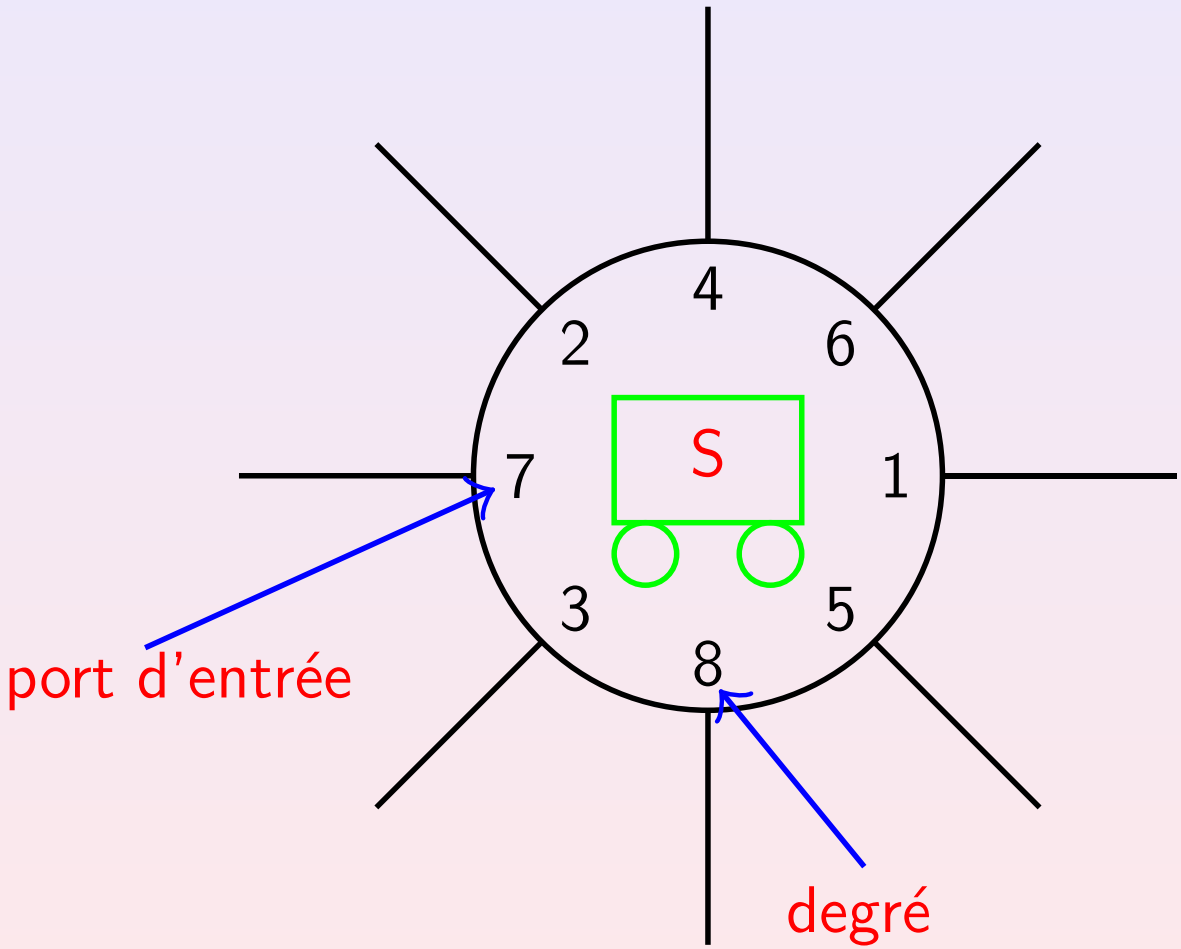
Anonyme

- sommets sans identifiant
- étiquetage local des arêtes

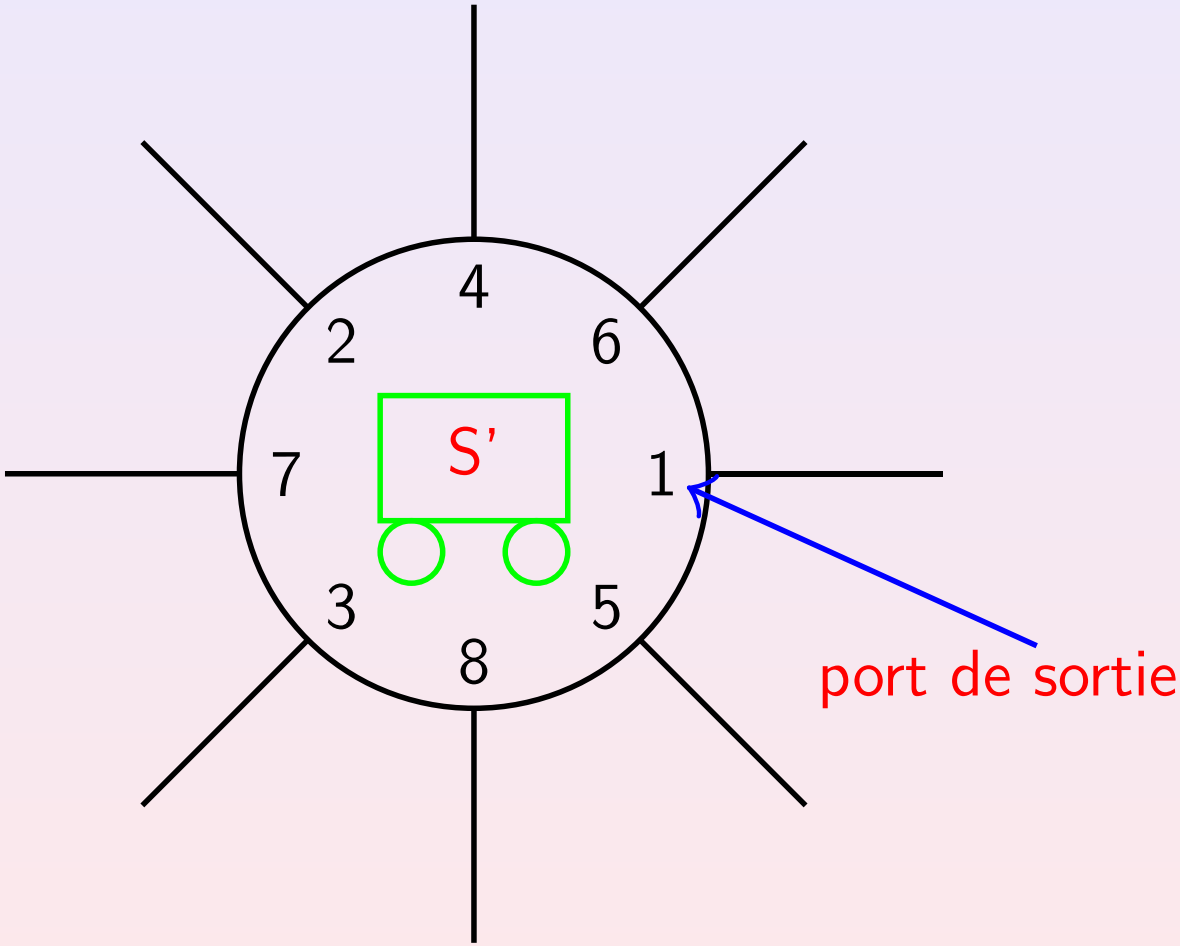
Exemple d'un graphe anonyme



Automate de Mealy (1)



Automate de Mealy (1)



Automate de Mealy (2)

Entrées

- S : état courant
- i : port d'entrée
- d : degré du nœud

Sorties

- S' : état courant
- j : port de sortie

Fonction de transition

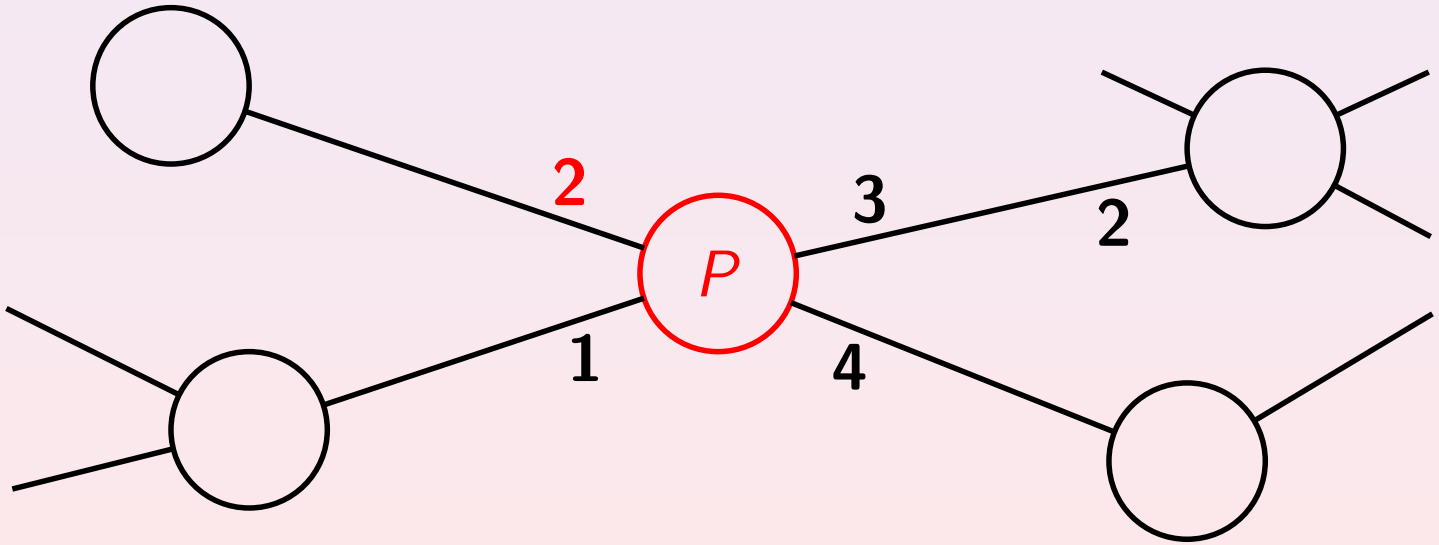
- $f(S, i, d) = (S', j)$

Exploration du voisinage

Étoile

L'étoile peut être explorée par un automate à deux états

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

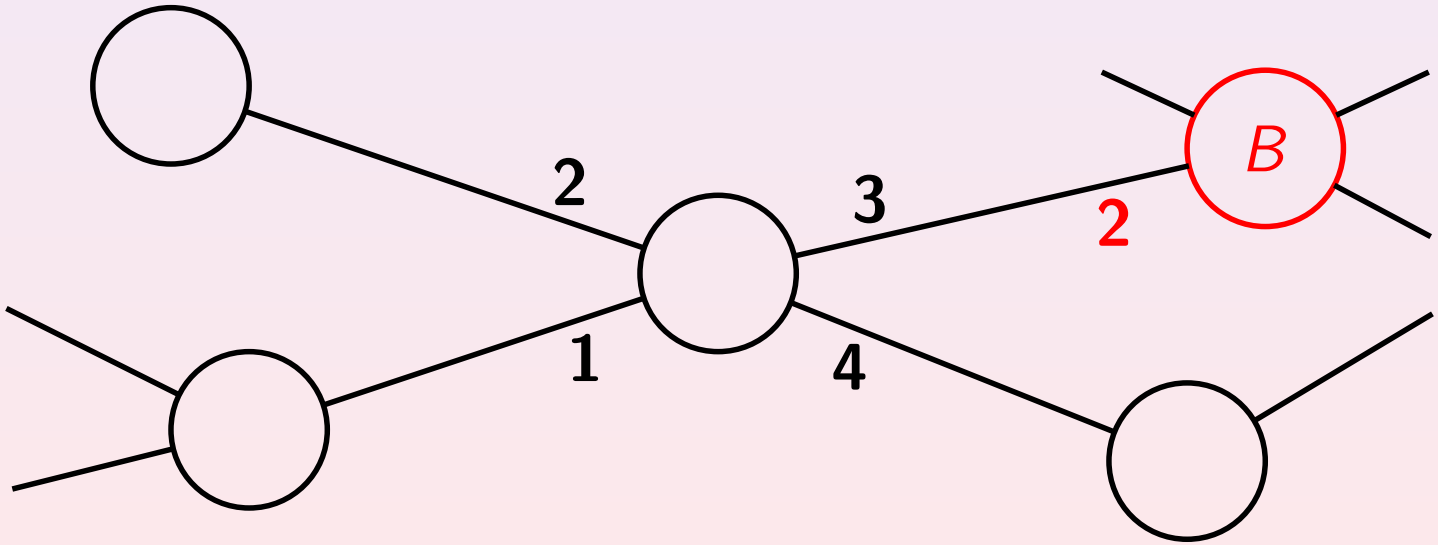


Exploration du voisinage

Étoile

L'étoile peut être explorée par un automate à deux états

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

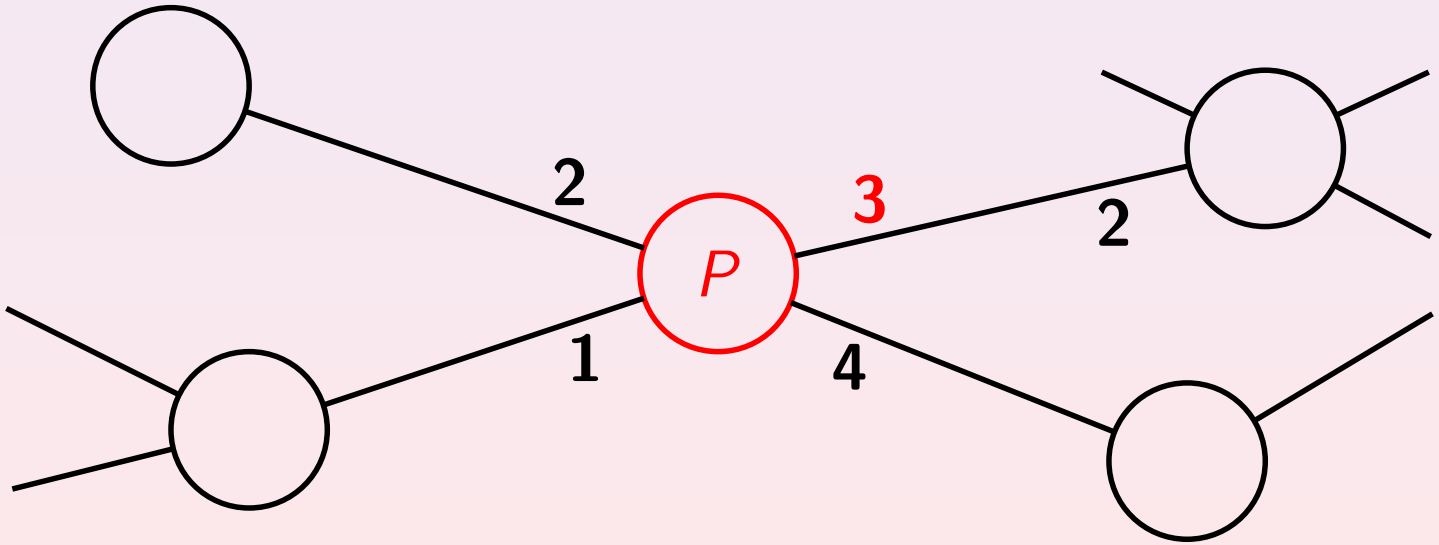


Exploration du voisinage

Étoile

L'étoile peut être explorée par un automate à deux états

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

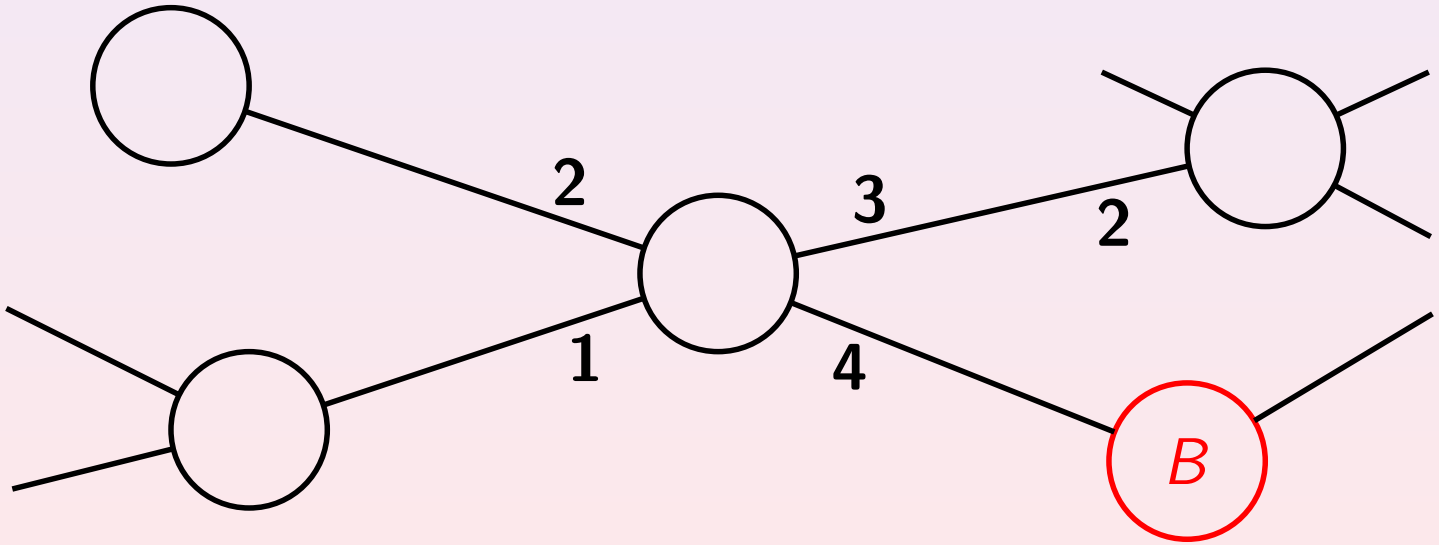


Exploration du voisinage

Étoile

L'étoile peut être explorée par un automate à deux états

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

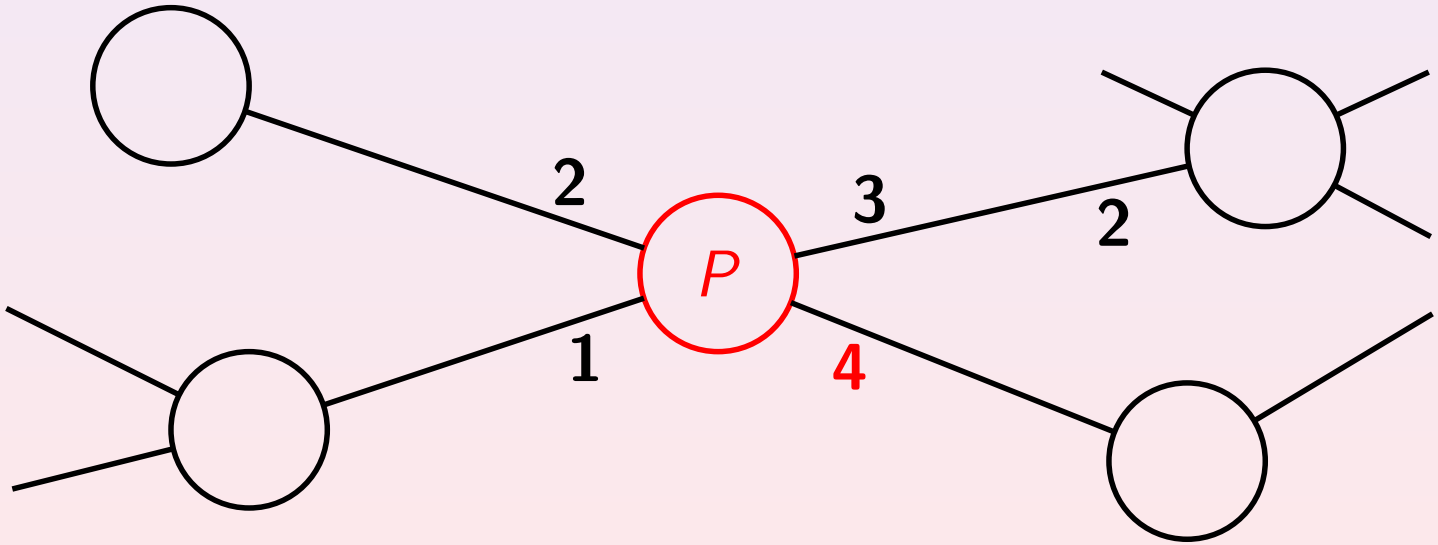


Exploration du voisinage

Étoile

L'étoile peut être explorée par un automate à deux états

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

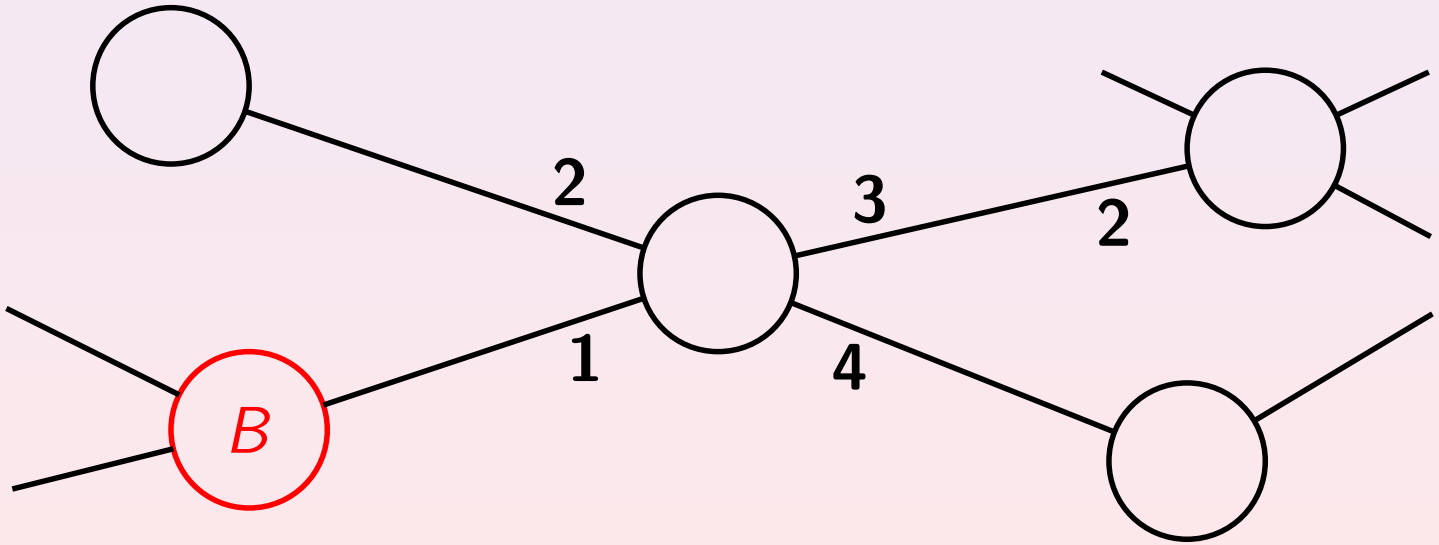


Exploration du voisinage

Étoile

L'étoile peut être explorée par un automate à deux états

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

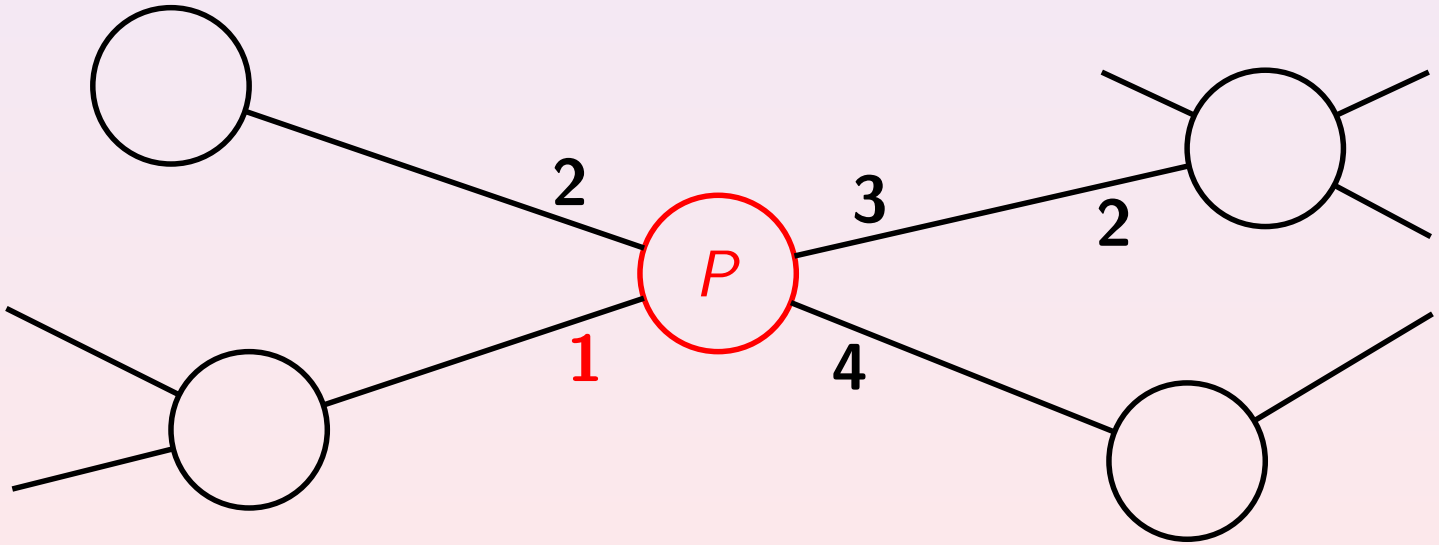


Exploration du voisinage

Étoile

L'étoile peut être explorée par un automate à deux états

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

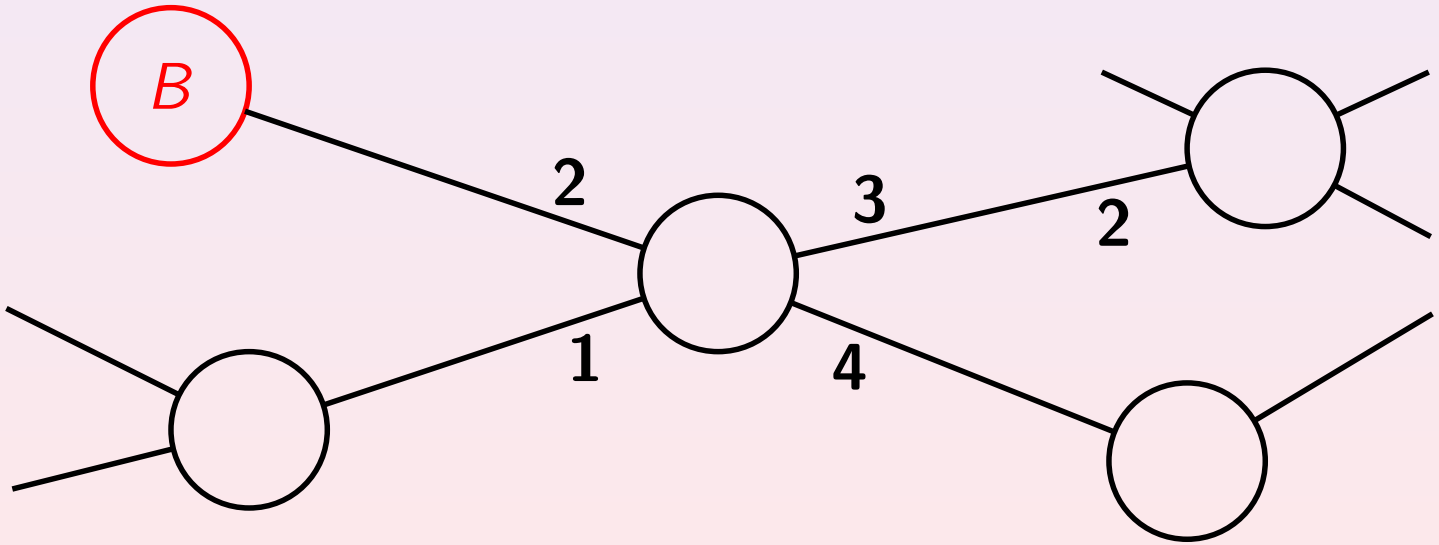


Exploration du voisinage

Étoile

L'étoile peut être explorée par un automate à deux états

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

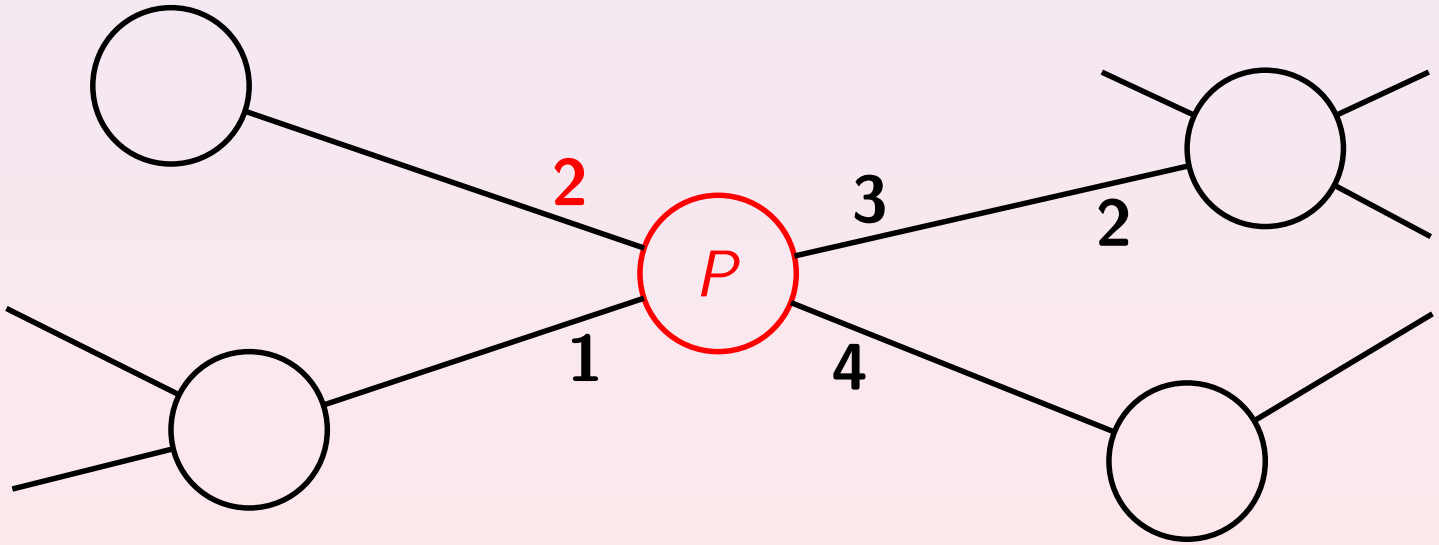


Exploration du voisinage

Étoile

L'étoile peut être explorée par un automate à deux états

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

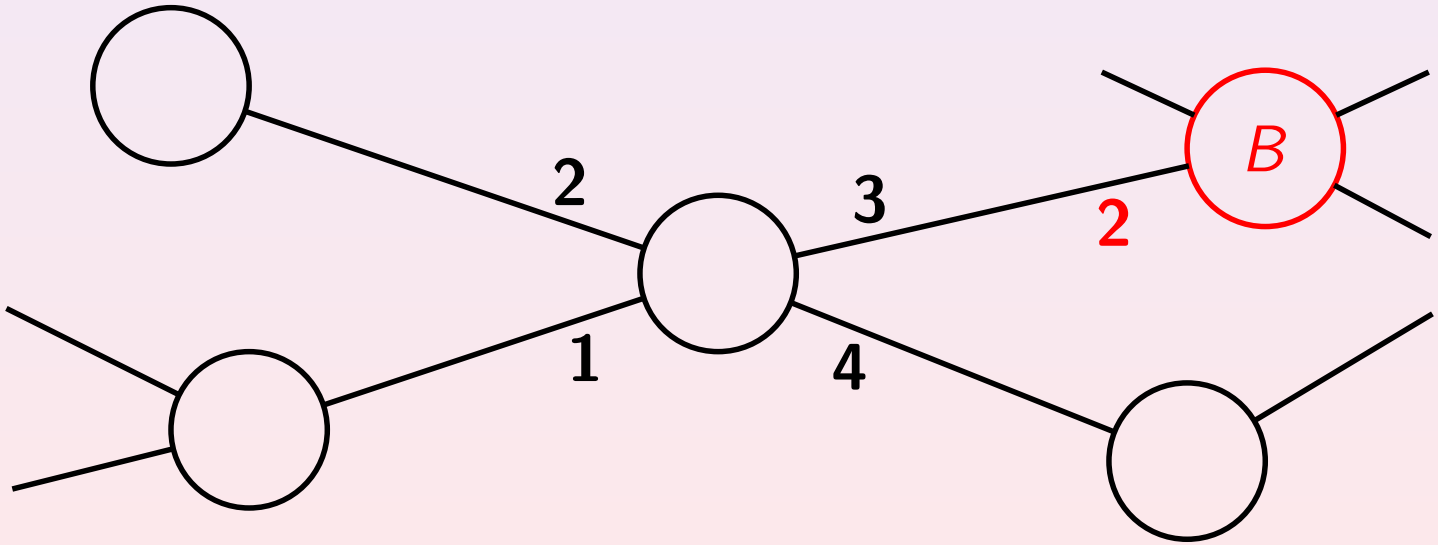


Exploration du voisinage

Étoile

L'étoile peut être explorée par un automate à deux états

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



Plan

Résultats d'impossibilité (1)

Budach, Math. Nachrichten, 1978
Automata and Labyrinths

Aucun automate fini n'explore tous les graphes.

Un caillou est un marqueur pouvant être déposé et repris sur les sommets.

Rabin, Seminar talk at Berkeley, 1967
Maze threading automata

Aucun automate fini avec un nombre fini de cailloux n'explore tous les graphes.

Résultats d'impossibilité (1)

Budach, Math. Nachrichten, 1978
Automata and Labyrinths

Aucun automate fini n'explore tous les graphes.

Un caillou est un marqueur pouvant être déposé et repris sur les sommets.

Rabin, Seminar talk at Berkeley, 1967
Maze threading automata

Aucun automate fini avec un nombre fini de cailloux n'explore tous les graphes.

Résultats d'impossibilité (1)

Budach, Math. Nachrichten, 1978
Automata and Labyrinths

Aucun automate fini n'explore tous les graphes.

Un caillou est un marqueur pouvant être déposé et repris sur les sommets.

Rabin, Seminar talk at Berkeley, 1967
Maze threading automata

Aucun automate fini avec un nombre fini de cailloux n'explore tous les graphes.

Résultats d'impossibilité (2)

Rollik, Acta Informatica, 1980
Automaten in planaren Graphen

Aucune équipe finie d'automates finis n'explore tous les graphes (mêmes planaires).

Un JAG (Jumping Automaton for Graphs) est une équipe d'automates finis coopérant en permanence. Un automate a la possibilité de se téléporter à côté d'un autre automate.

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

Aucun JAG n'explore tous les graphes.

Résultats d'impossibilité (2)

Rollik, Acta Informatica, 1980
Automaten in planaren Graphen

Aucune équipe finie d'automates finis n'explore tous les graphes (mêmes planaires).

Un JAG (Jumping Automaton for Graphs) est une équipe d'automates finis coopérant en permanence. Un automate a la possibilité de se téléporter à côté d'un autre automate.

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

Aucun JAG n'explore tous les graphes.

Résultats d'impossibilité (2)

Rollik, Acta Informatica, 1980
Automaten in planaren Graphen

Aucune équipe finie d'automates finis n'explore tous les graphes (mêmes planaires).

Un JAG (Jumping Automaton for Graphs) est une équipe d'automates finis coopérant en permanence. Un automate a la possibilité de se téléporter à côté d'un autre automate.

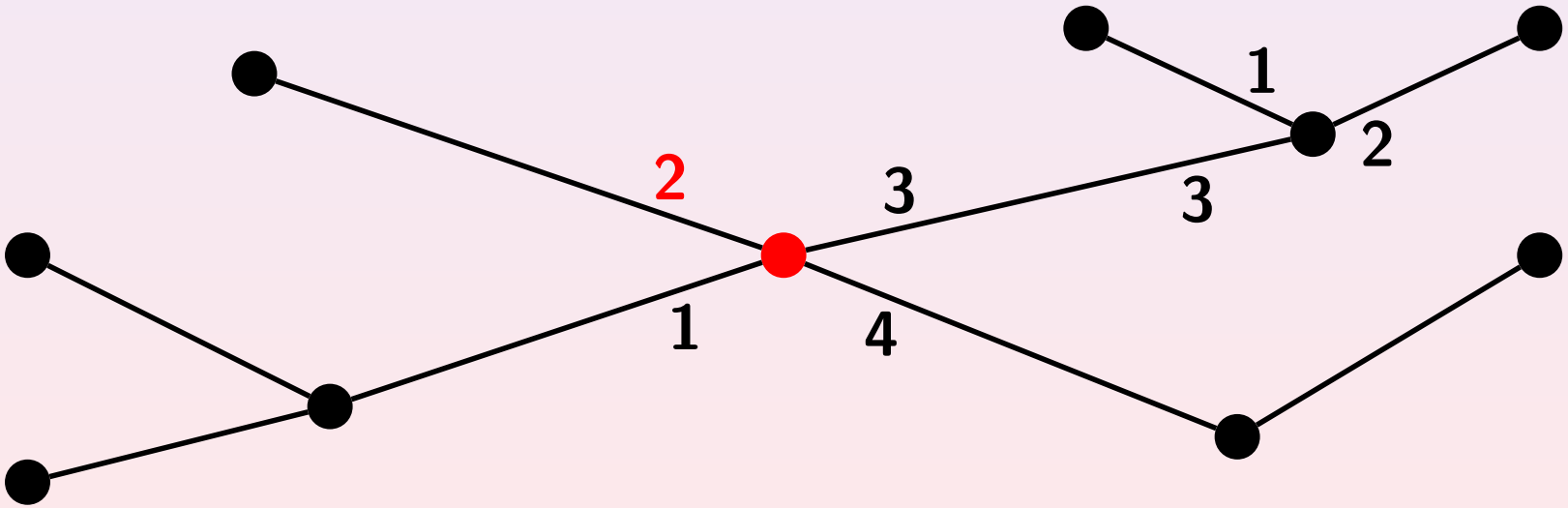
Cook, Rackoff, SIAMJC, 1980
Space lower bounds for maze threadability on restricted machines

Aucun JAG n'explore tous les graphes.

Universalité pour les arbres

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

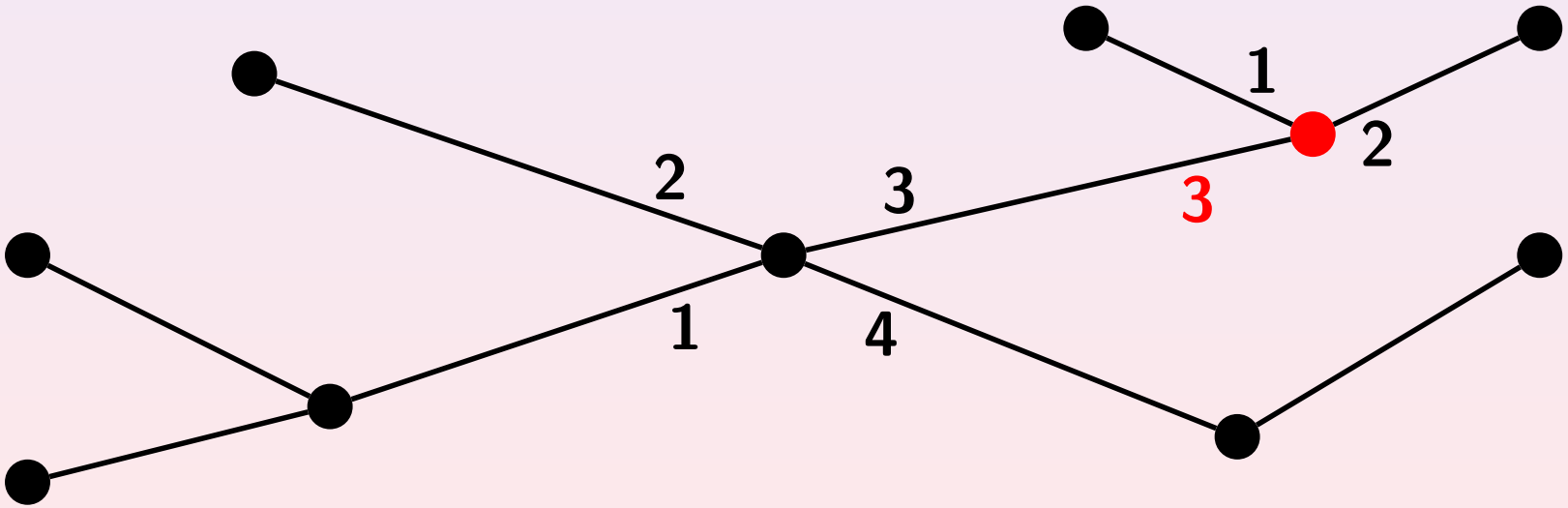
Un automate sans mémoire (un seul état) peut explorer tous les arbres.



Universalité pour les arbres

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

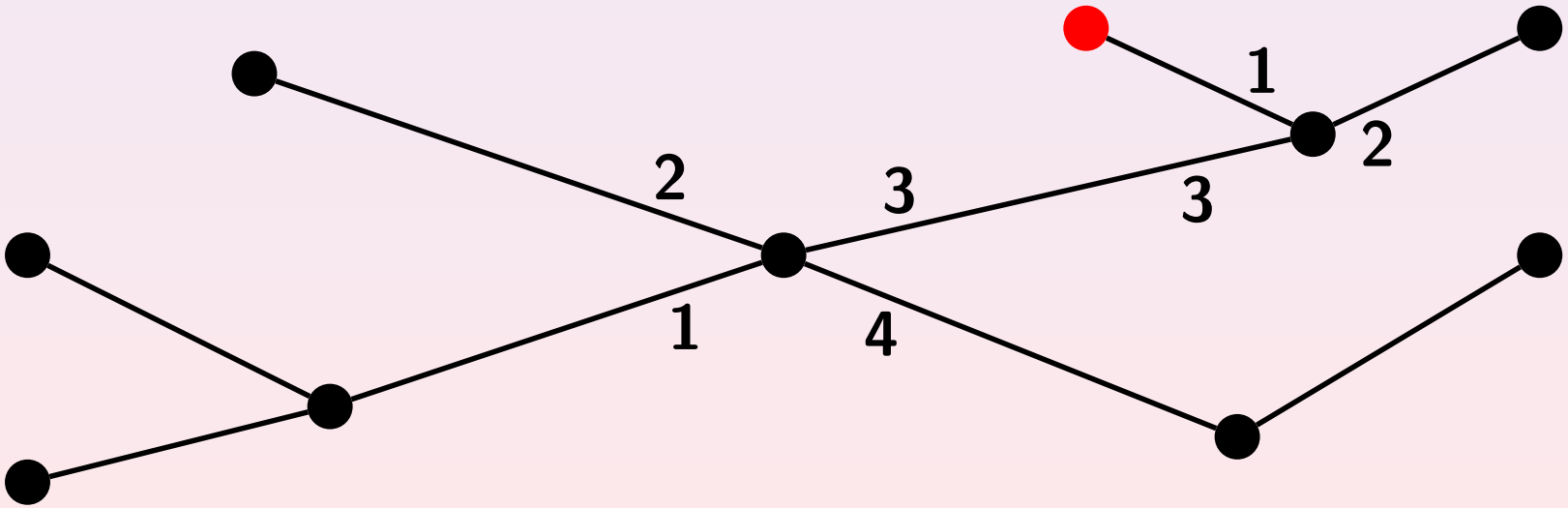
Un automate sans mémoire (un seul état) peut explorer tous les arbres.



Universalité pour les arbres

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

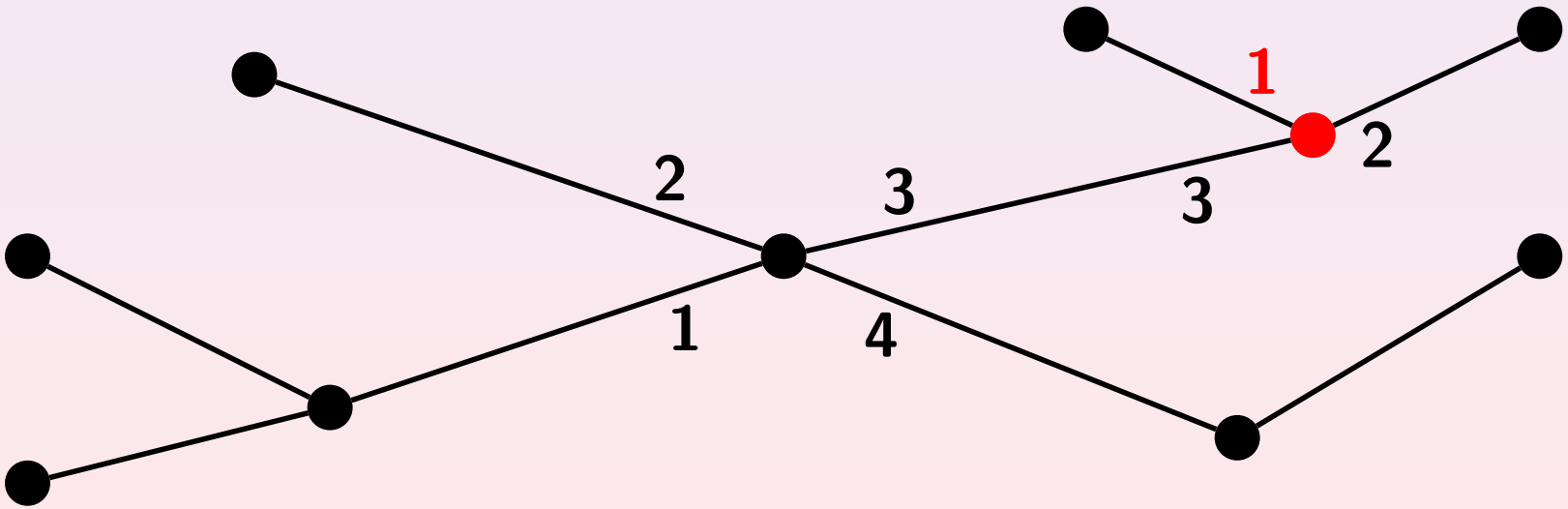
Un automate sans mémoire (un seul état) peut explorer tous les arbres.



Universalité pour les arbres

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

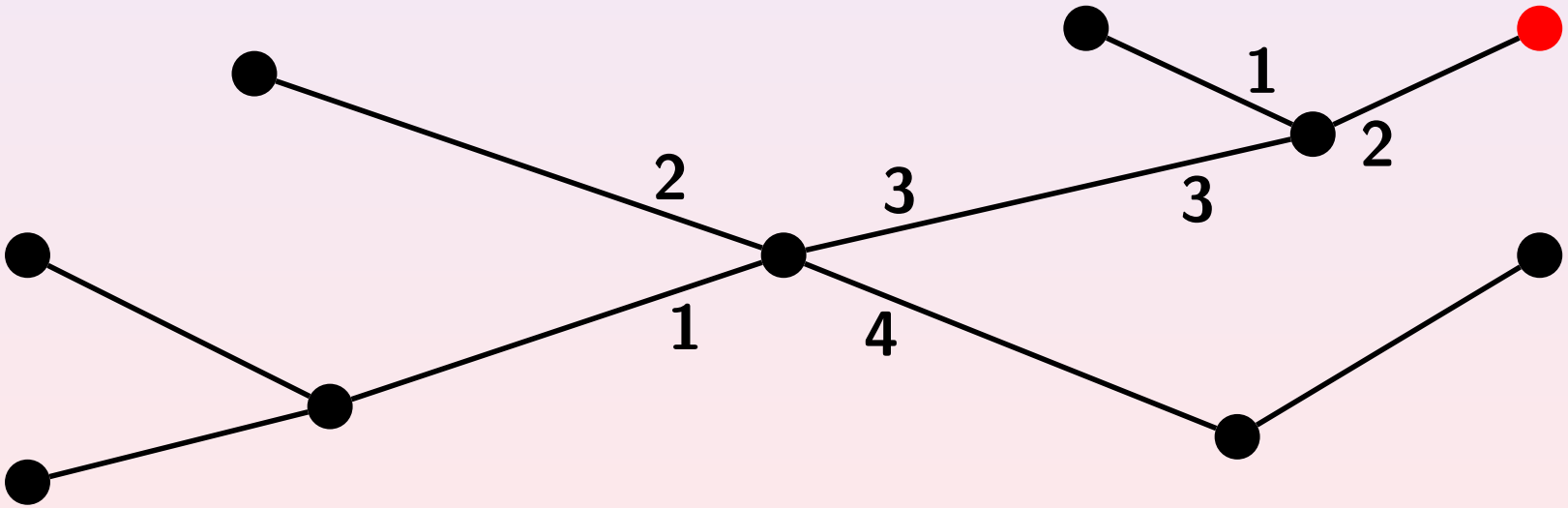
Un automate sans mémoire (un seul état) peut explorer tous les arbres.



Universalité pour les arbres

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

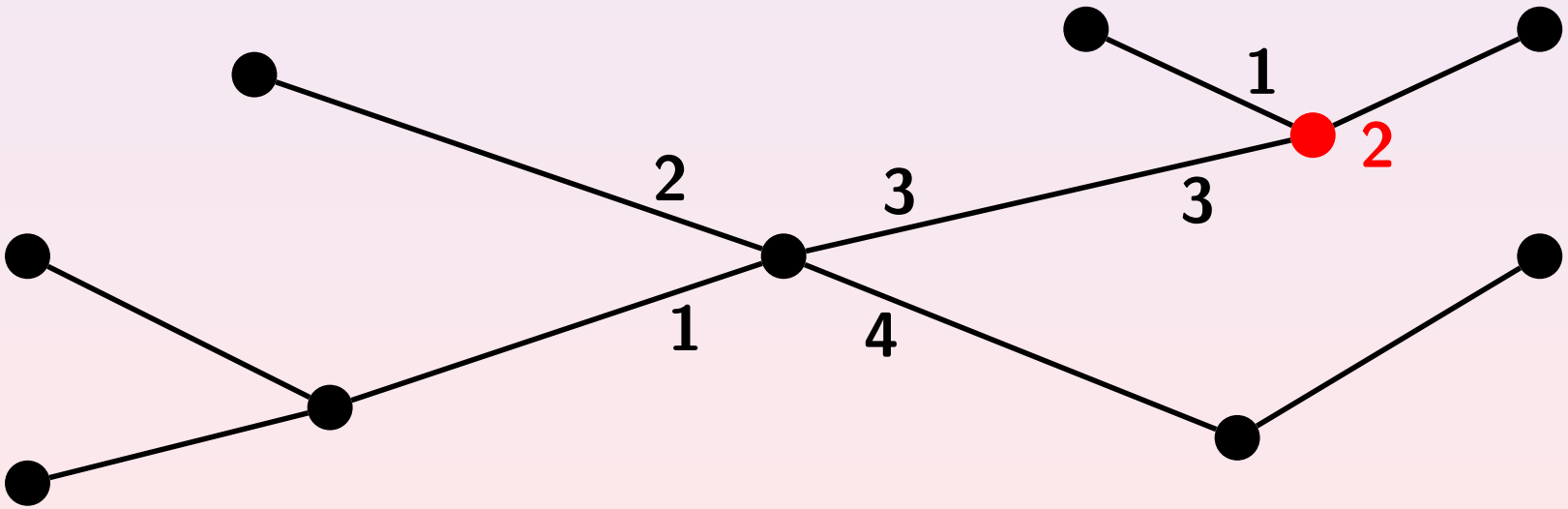
Un automate sans mémoire (un seul état) peut explorer tous les arbres.



Universalité pour les arbres

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

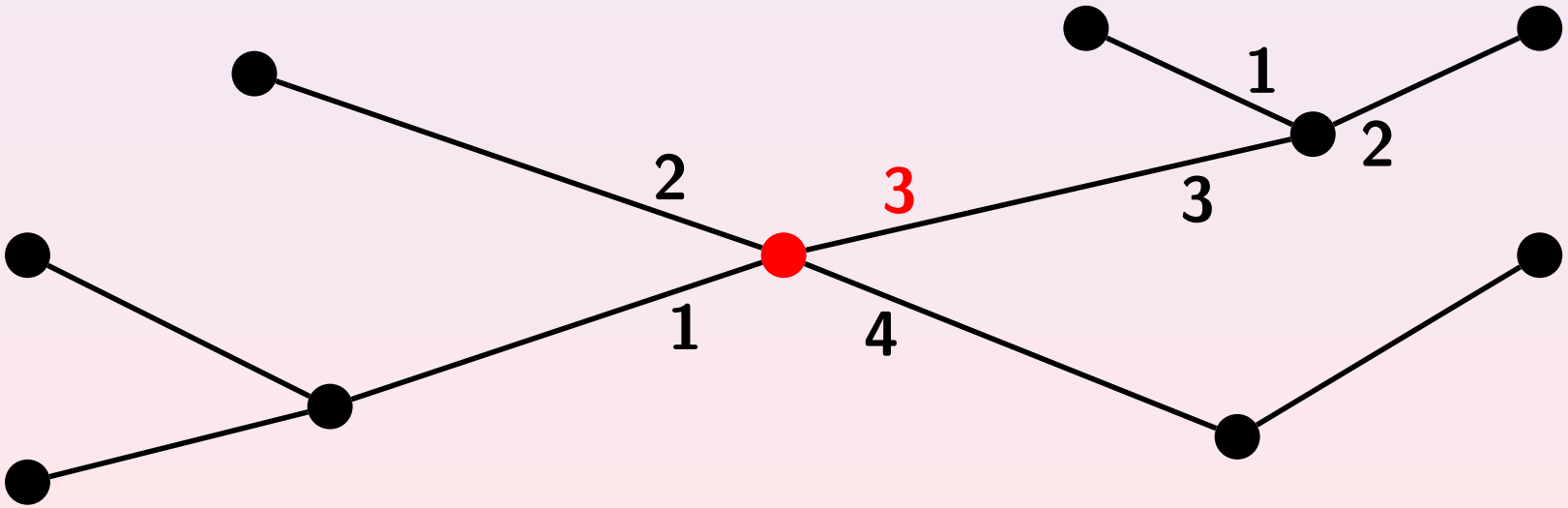
Un automate sans mémoire (un seul état) peut explorer tous les arbres.



Universalité pour les arbres

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

Un automate sans mémoire (un seul état) peut explorer tous les arbres.



Tableaux blancs

Un tableau blanc sur un nœud est un espace sur lequel un automate peut lire, écrire ou effacer des messages.

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

Un automate sans mémoire (un seul état) peut explorer tous les graphes de degré maximum Δ en n'utilisant que $O(log\,\Delta)$ bits de mémoire par nœud.

Algorithme : Au i-ème passage sur un nœud, prendre l'arête $i \mod d$ où d est le degré du nœud.

Tableaux blancs

Un tableau blanc sur un nœud est un espace sur lequel un automate peut lire, écrire ou effacer des messages.

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

Un automate sans mémoire (un seul état) peut explorer tous les graphes de degré maximum Δ en n'utilisant que $O(log\,\Delta)$ bits de mémoire par nœud.

Algorithme : Au i-ème passage sur un nœud, prendre l'arête $i \mod d$ où d est le degré du nœud.

Tableaux blancs

Un tableau blanc sur un nœud est un espace sur lequel un automate peut lire, écrire ou effacer des messages.

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

Un automate sans mémoire (un seul état) peut explorer tous les graphes de degré maximum Δ en n'utilisant que $O(log\,\Delta)$ bits de mémoire par nœud.

Algorithme : Au i-ème passage sur un nœud, prendre l'arête $i \mod d$ où d est le degré du nœud.

Plan

Modèle utilisé

Modèle

- Un oracle colorie (étiquette) le graphe pour aider l'automate.
- L'automate peut lire la couleur comme entrée de sa fonction de transition.

But

Utiliser le plus petit nombre de couleurs possible.

Nos Résultats

Trois couleurs

Il existe un automate fini et un algorithme de coloriage en seulement trois couleurs tels que l'automate peut explorer tous les graphes.

Degré constant en deux couleurs

Il existe un automate de $O(log\,\Delta)$ bits de mémoire et un algorithme de coloriage en deux couleurs tels que l'automate peut explorer tous les graphes de degré maximal Δ.

Nos Résultats

Trois couleurs

Il existe un automate fini et un algorithme de coloriage en seulement trois couleurs tels que l'automate peut explorer tous les graphes.

Degré constant en deux couleurs

Il existe un automate de $O(log\,\Delta)$ bits de mémoire et un algorithme de coloriage en deux couleurs tels que l'automate peut explorer tous les graphes de degré maximal Δ.

Plan

Arbre couvrant

Idée simple

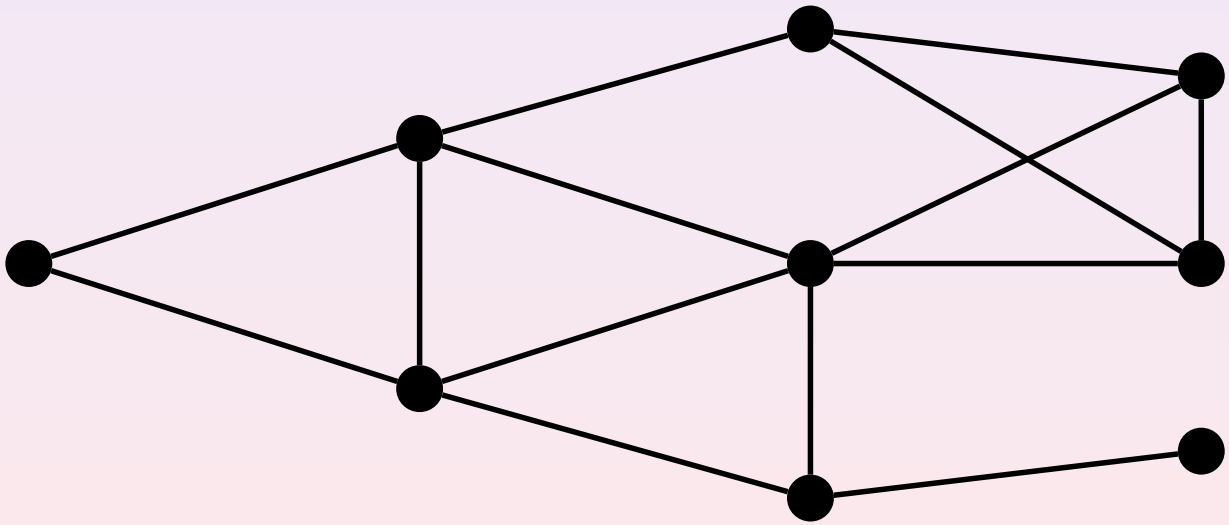
arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre

Utiliser l'algorithme d'exploration d'arbres

Arbre couvrant

Idée simple

arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre

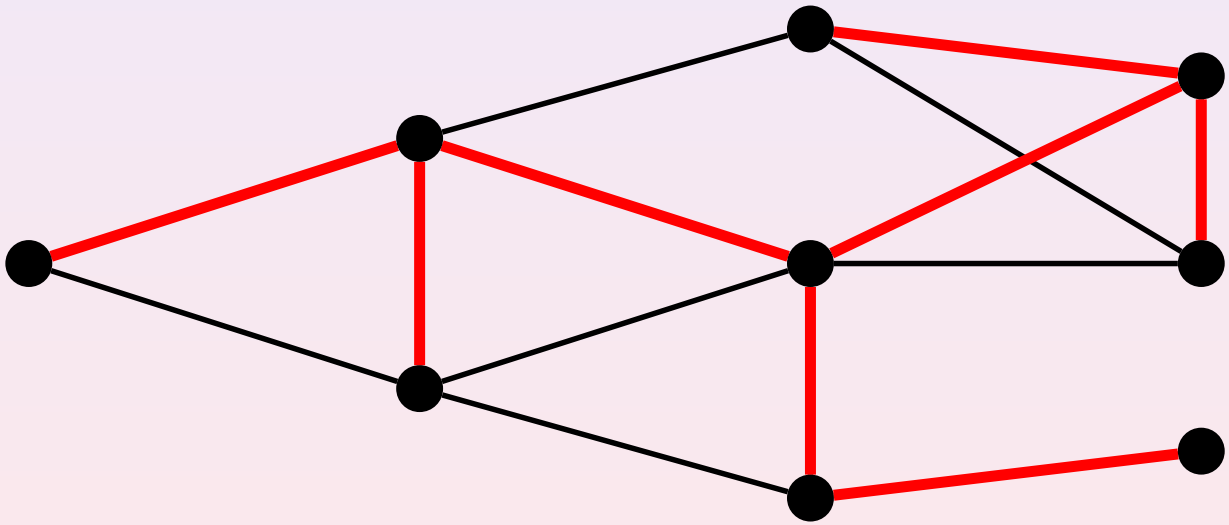


Utiliser l'algorithme d'exploration d'arbres

Arbre couvrant

Idée simple

arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre

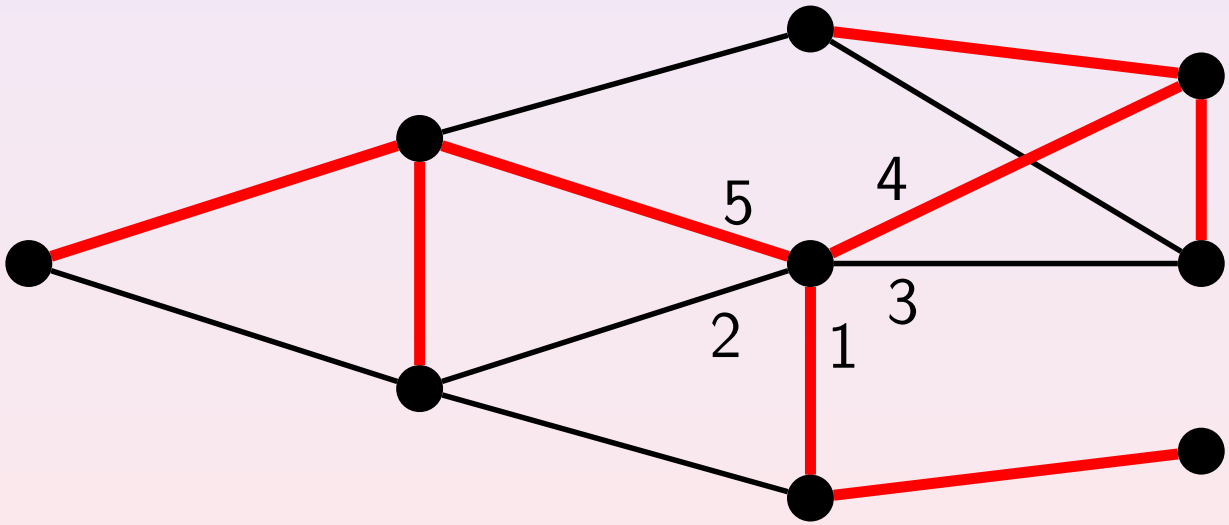


Utiliser l'algorithme d'exploration d'arbres

Arbre couvrant

Idée simple

arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre

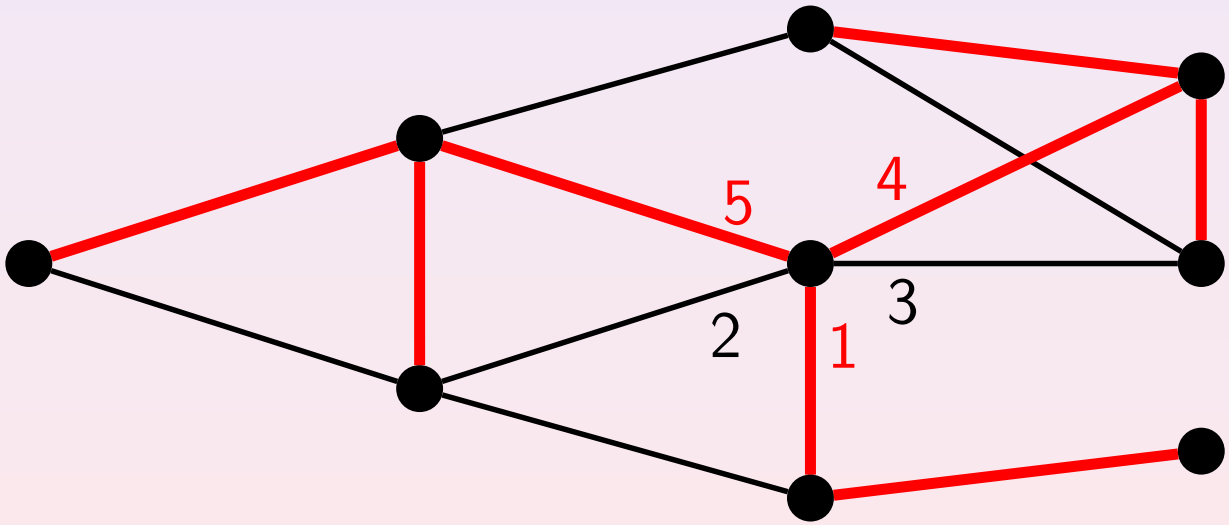


Utiliser l'algorithme d'exploration d'arbres

Arbre couvrant

Idée simple

arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre

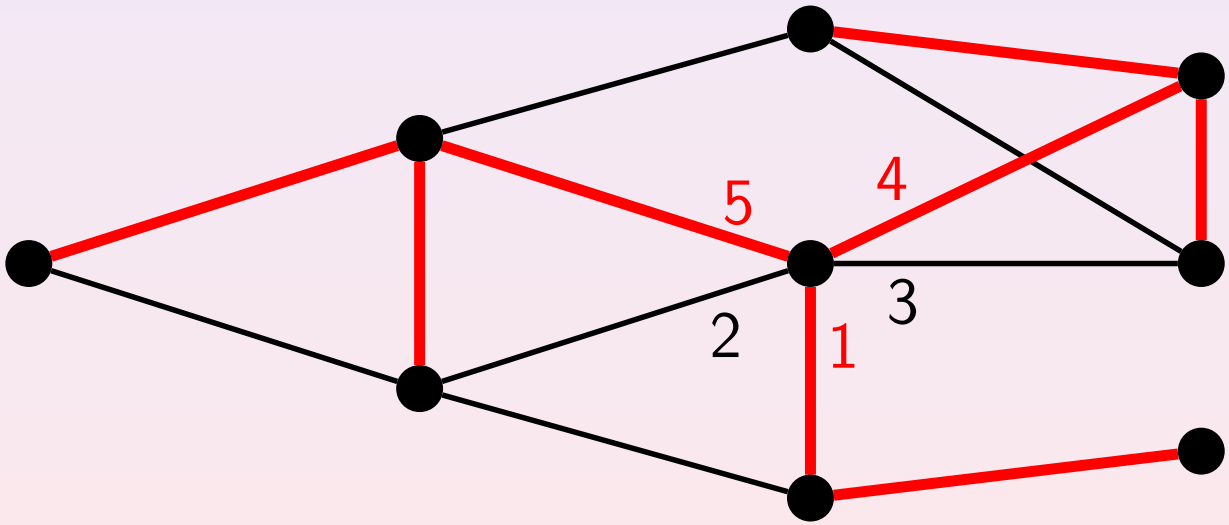


Utiliser l'algorithme d'exploration d'arbres

Arbre couvrant

Idée simple

arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre



Utiliser l'algorithme d'exploration d'arbres

Arbre enraciné

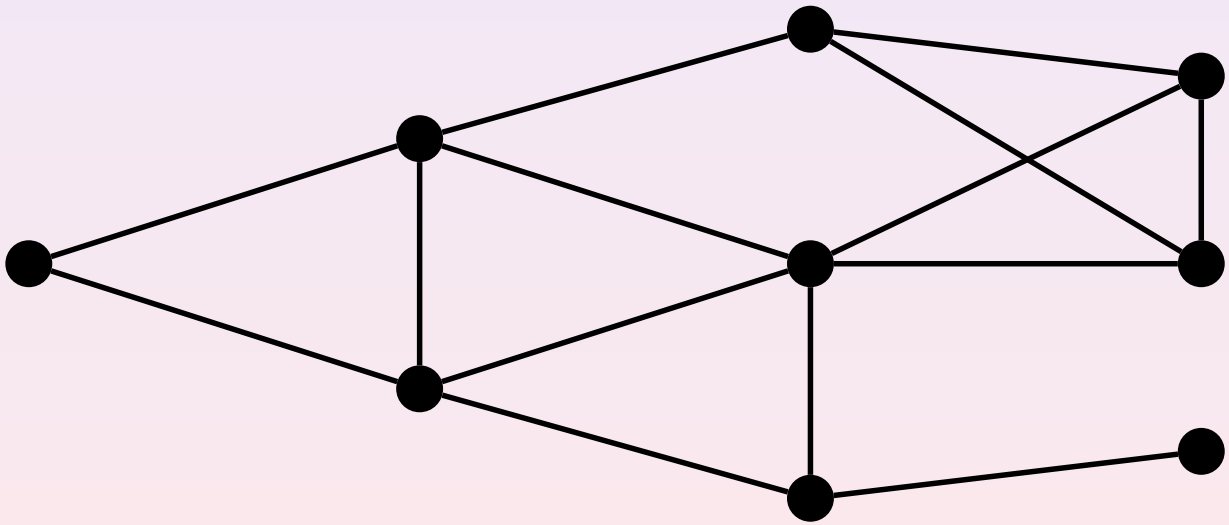
Etiquetage amélioré

seulement indiquer l'arête menant au père $\longrightarrow$ Δ couleurs

Arbre enraciné

Etiquetage amélioré

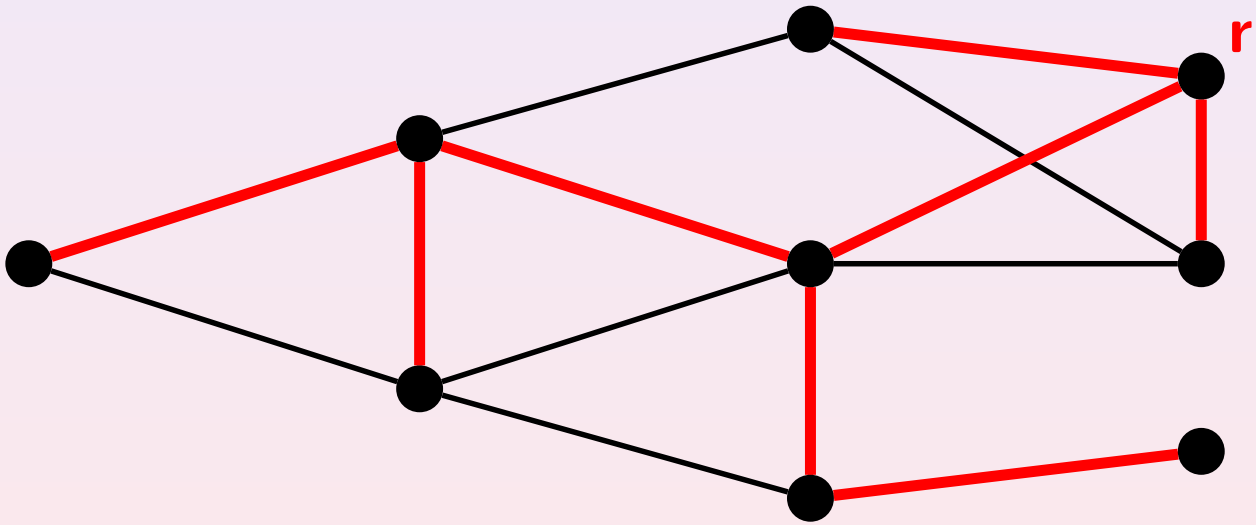
seulement indiquer l'arête menant au père $\longrightarrow$ Δ couleurs



Arbre enraciné

Etiquetage amélioré

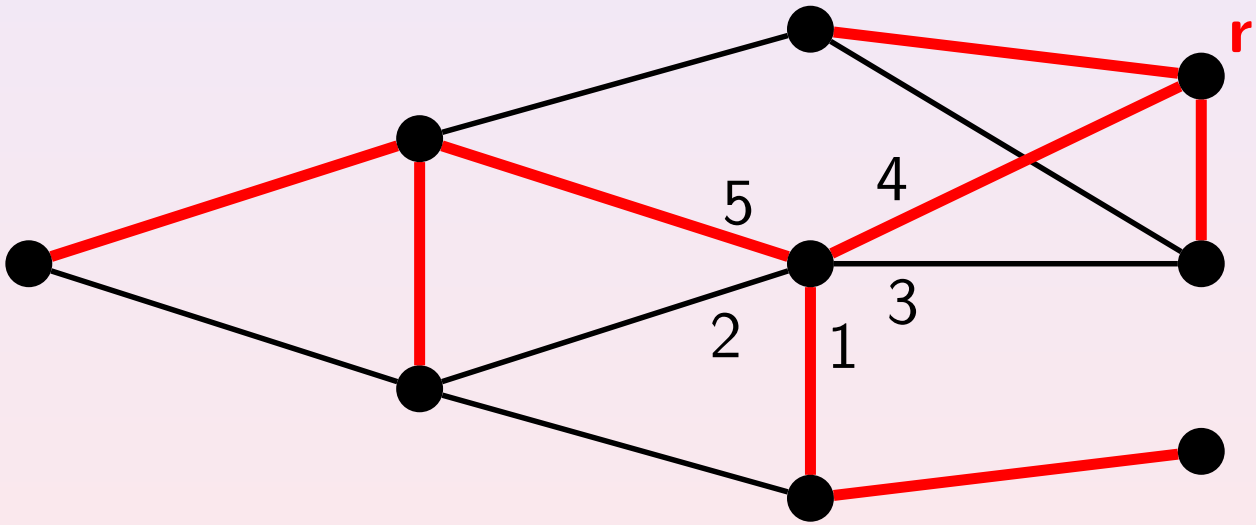
seulement indiquer l'arête menant au père $\longrightarrow$ Δ couleurs



Arbre enraciné

Etiquetage amélioré

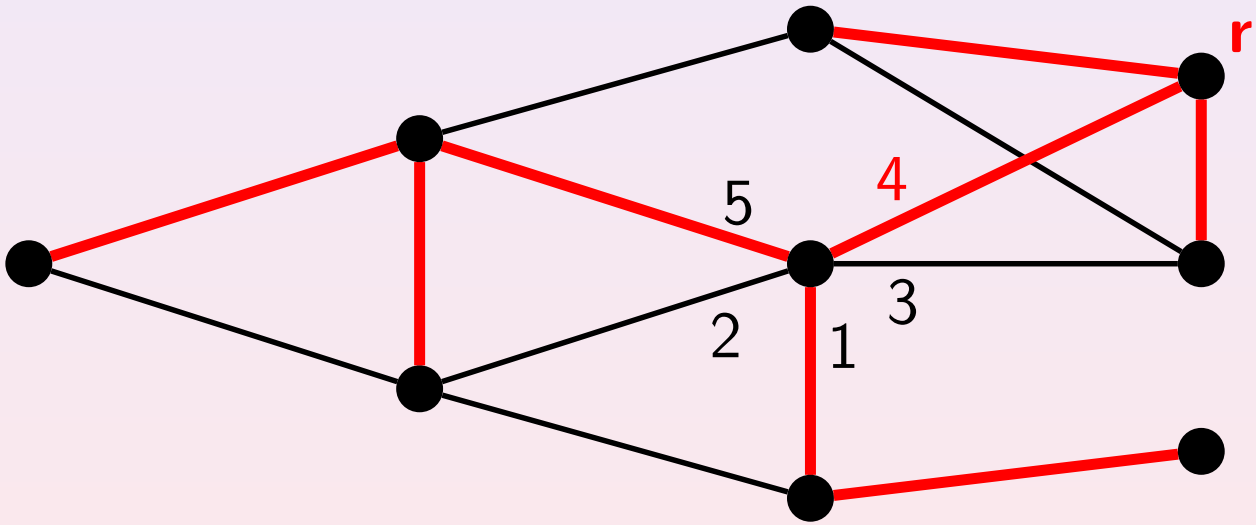
seulement indiquer l'arête menant au père $\longrightarrow$ Δ couleurs



Arbre enraciné

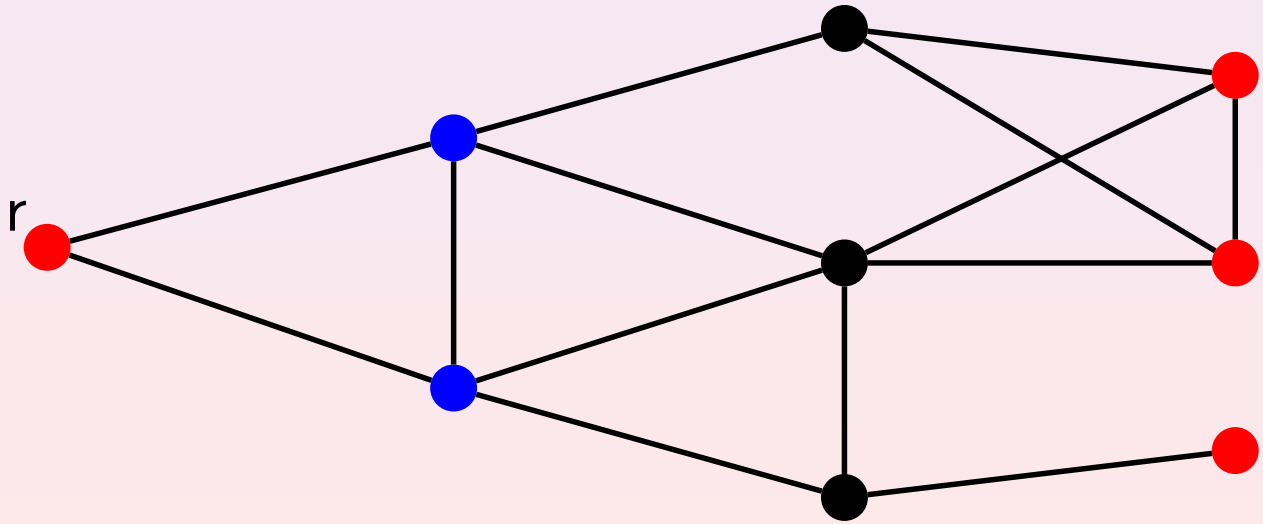
Etiquetage amélioré

seulement indiquer l'arête menant au père $\longrightarrow$ Δ couleurs



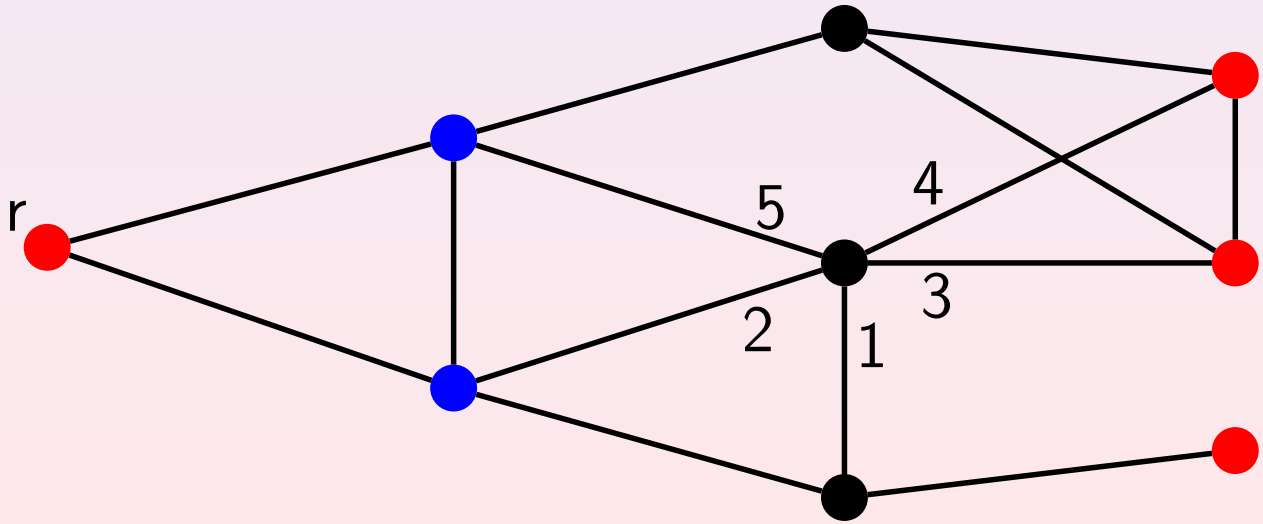
Trois couleurs suffisent

- choisir arbitrairement un nœud comme racine
- colorier tous les nœuds en fonction de leur distance d à la racine
  - distance $d \cong 0[3]$ rouge
  - distance $d \cong 1[3]$ bleu
  - distance $d \cong 2[3]$ noir



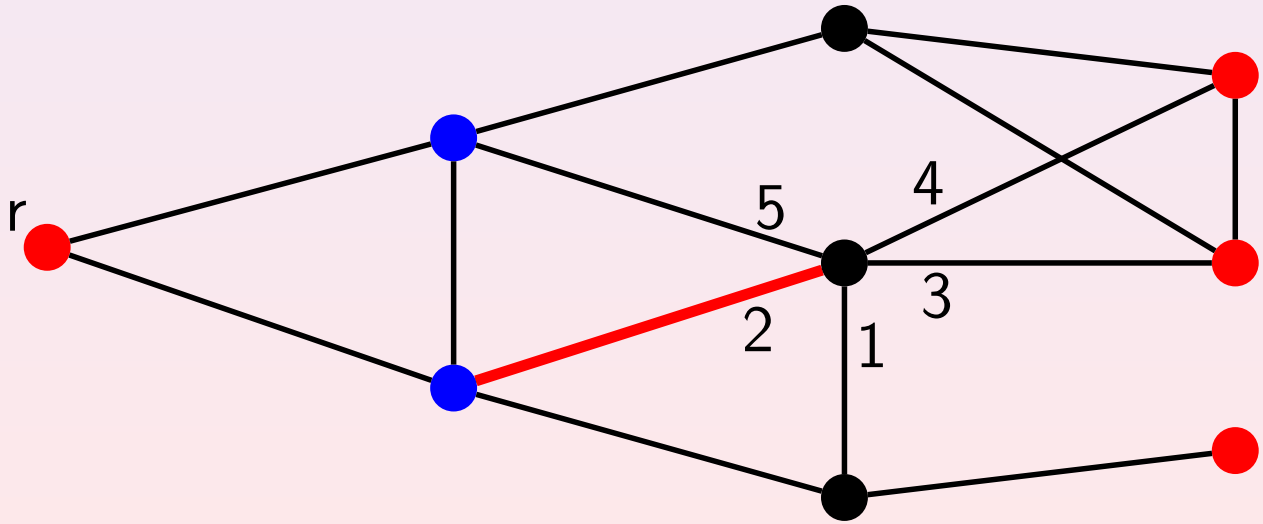
Trois couleurs suffisent

- choisir arbitrairement un nœud comme racine
- colorier tous les nœuds en fonction de leur distance d à la racine
  - distance $d \cong 0[3]$ rouge
  - distance $d \cong 1[3]$ bleu
  - distance $d \cong 2[3]$ noir

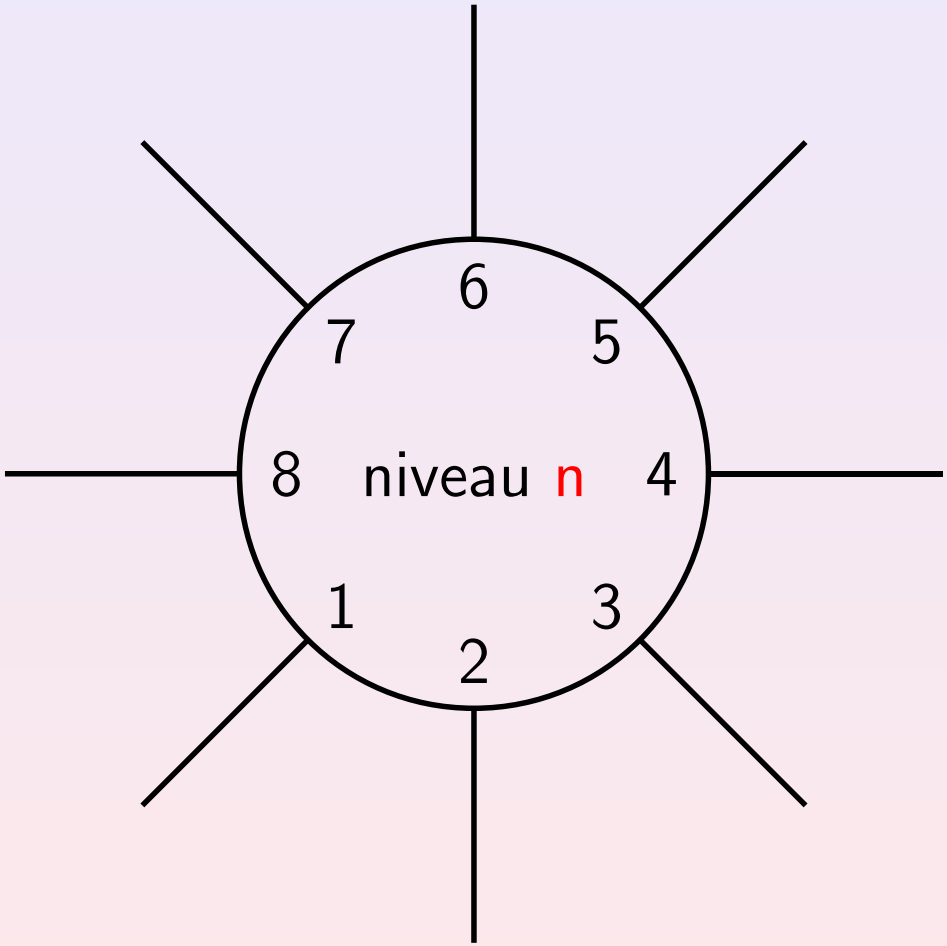


Trois couleurs suffisent

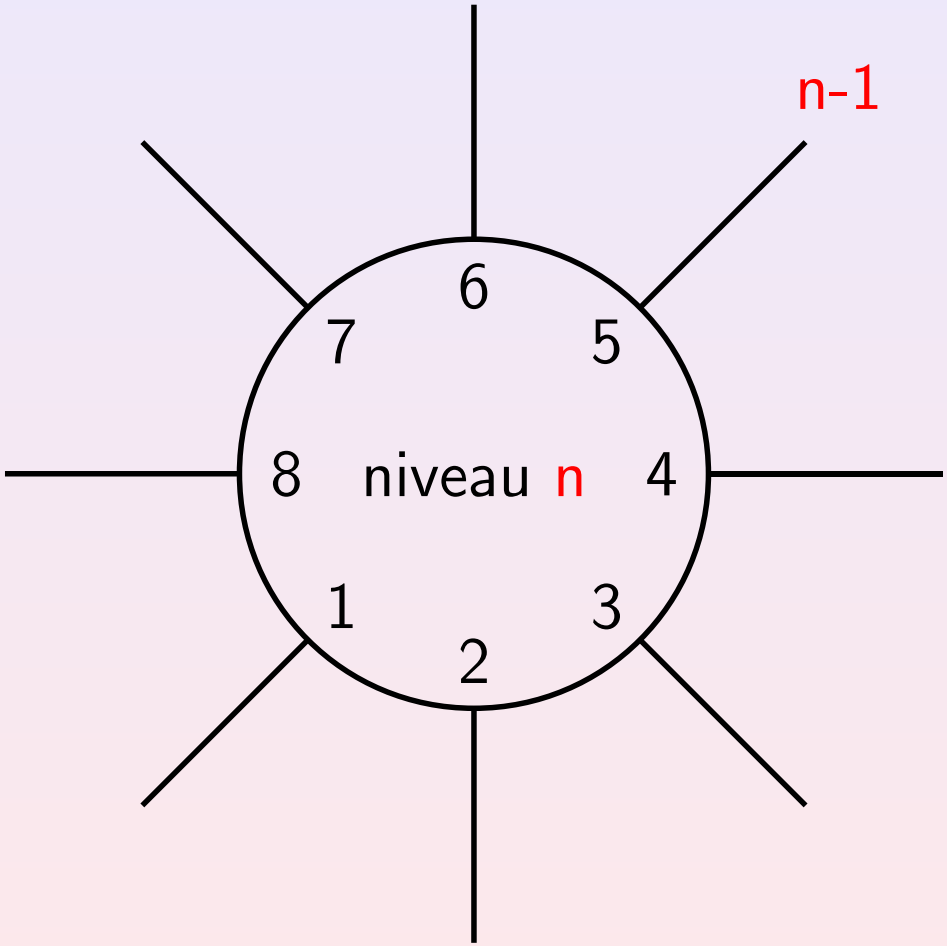
- choisir arbitrairement un nœud comme racine
- colorier tous les nœuds en fonction de leur distance d à la racine
  - distance $d \cong 0[3]$ rouge
  - distance $d \cong 1[3]$ bleu
  - distance $d \cong 2[3]$ noir



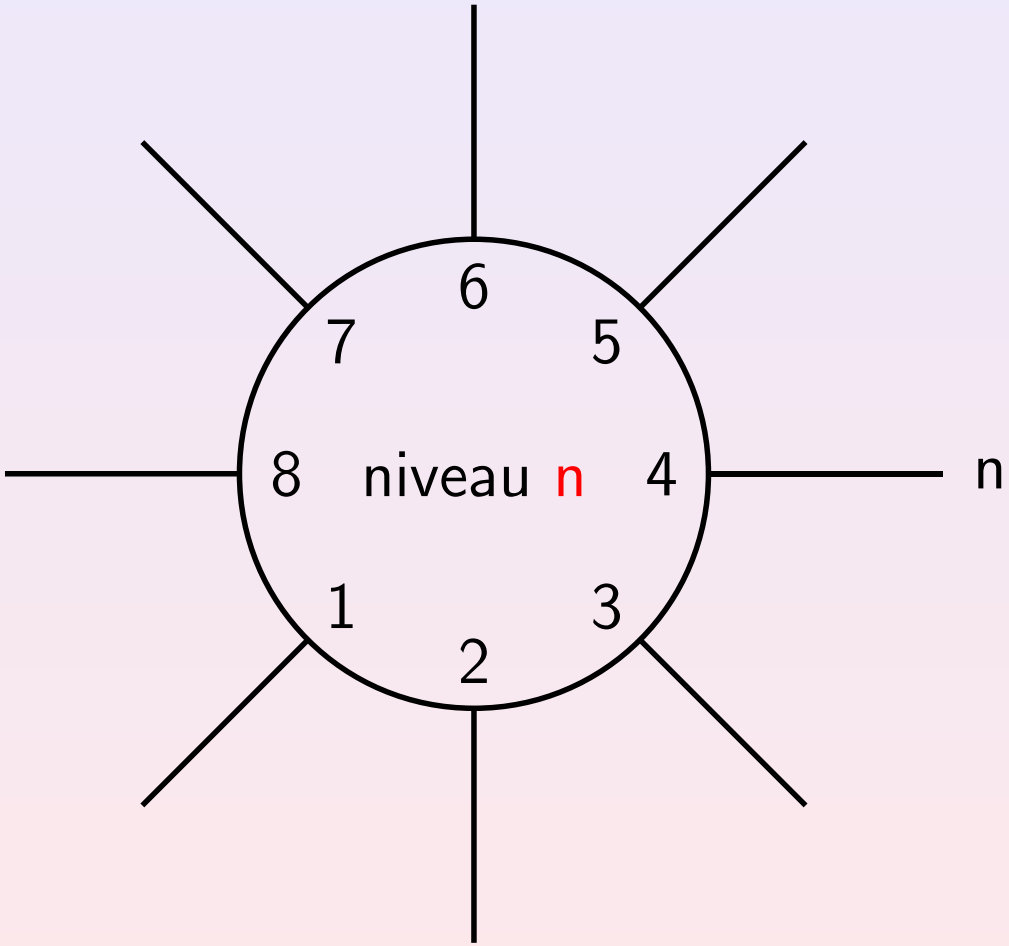
Reconnaissance du père



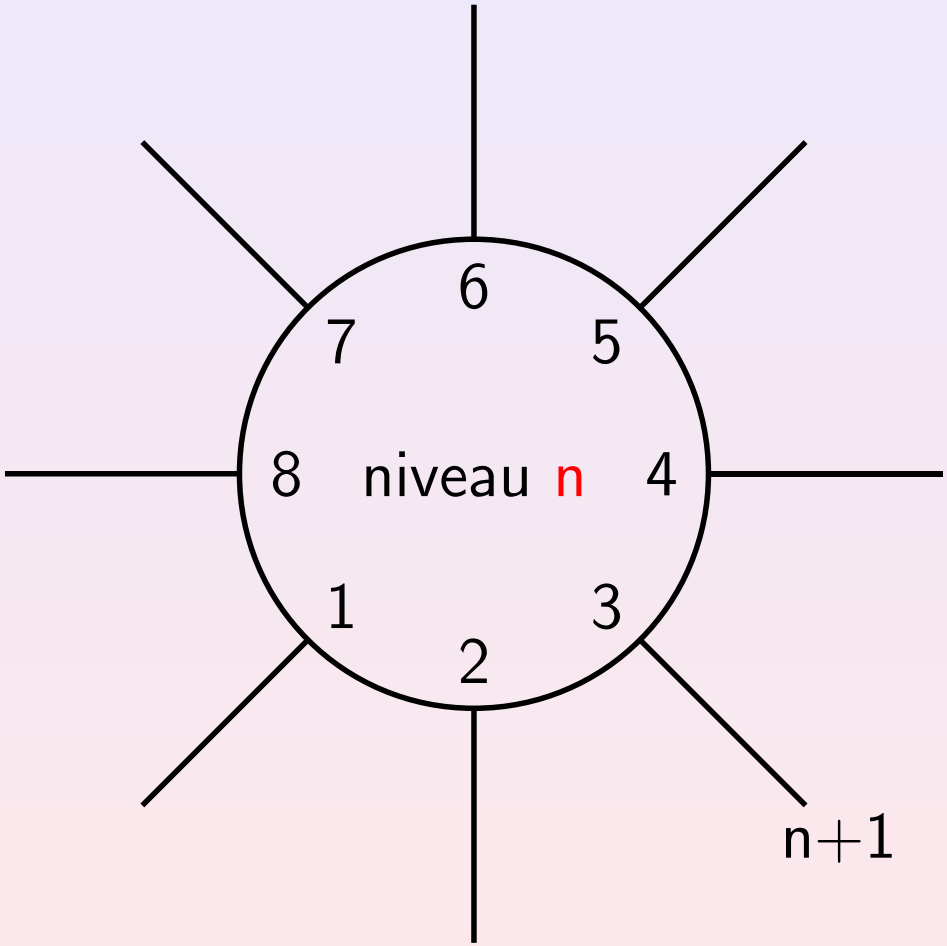
Reconnaissance du père



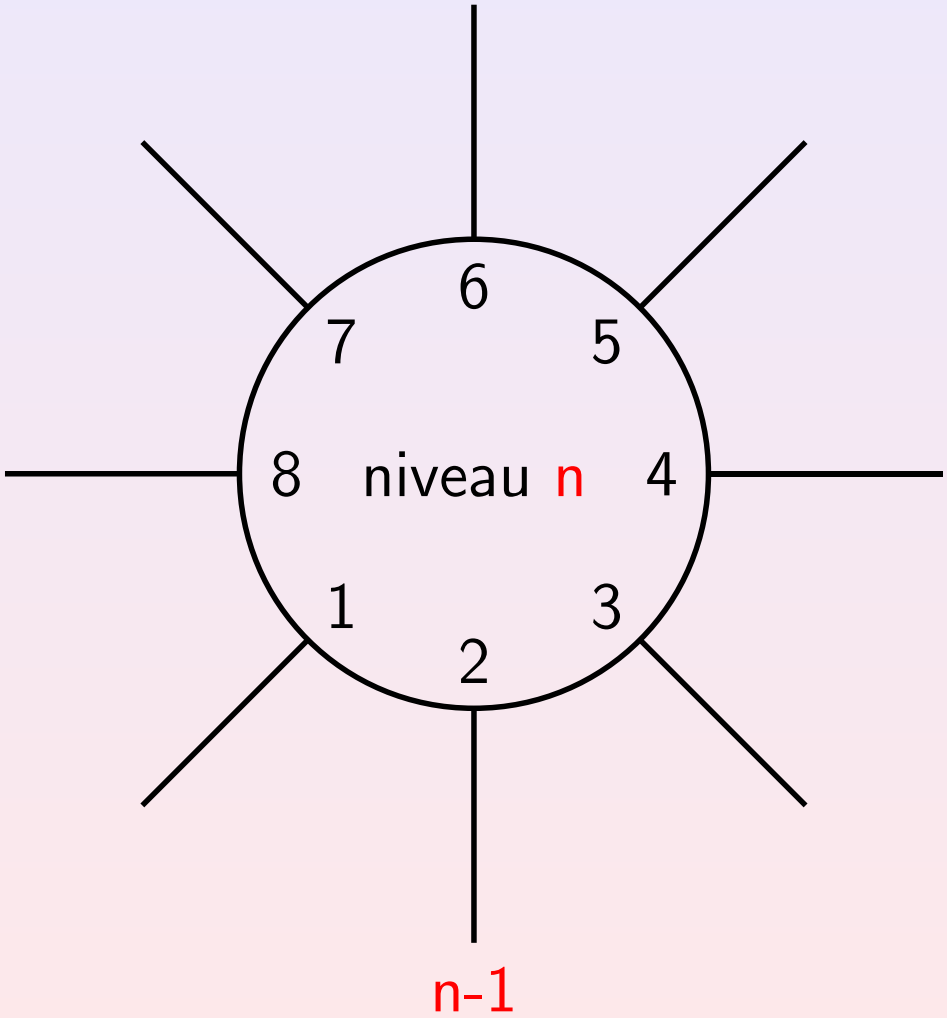
Reconnaissance du père



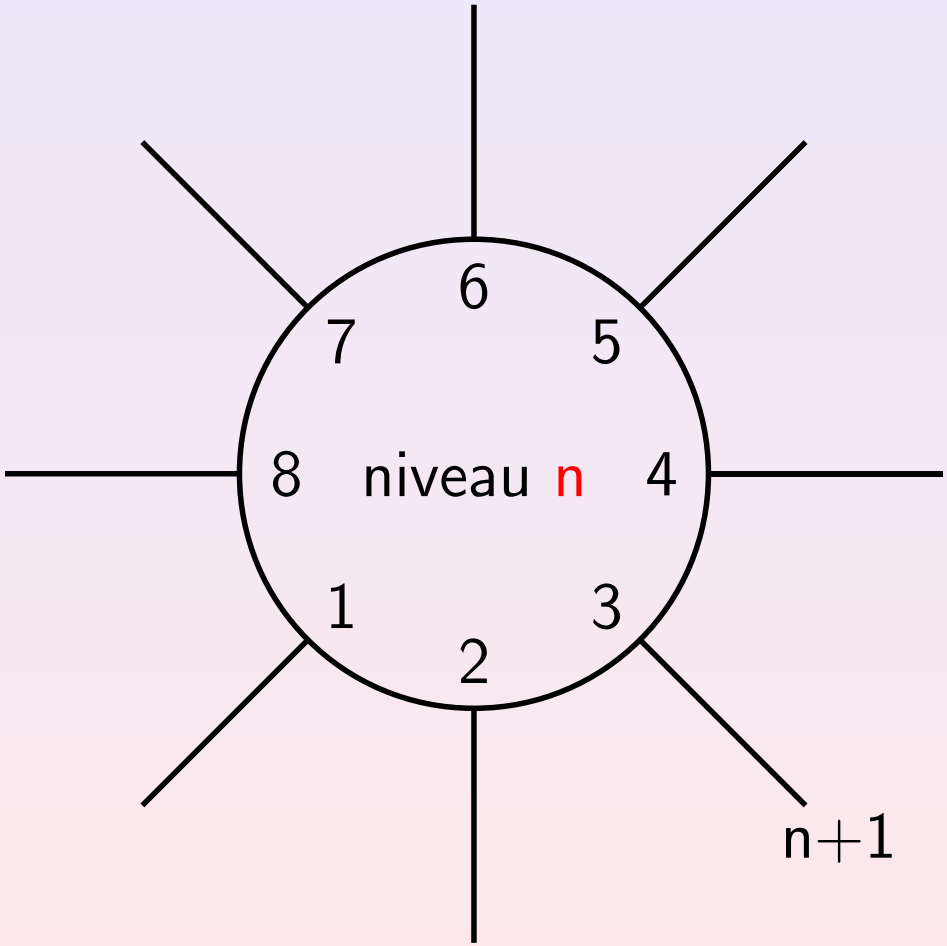
Reconnaissance du père



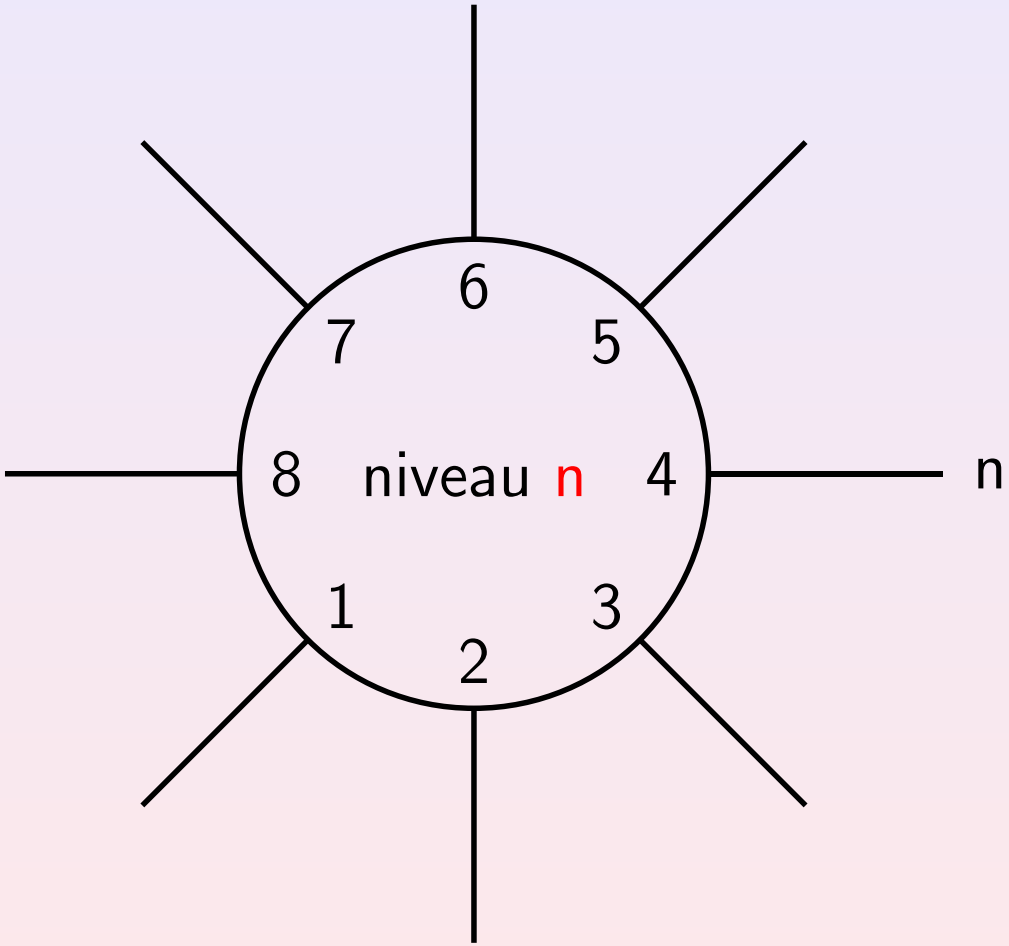
Reconnaissance du père



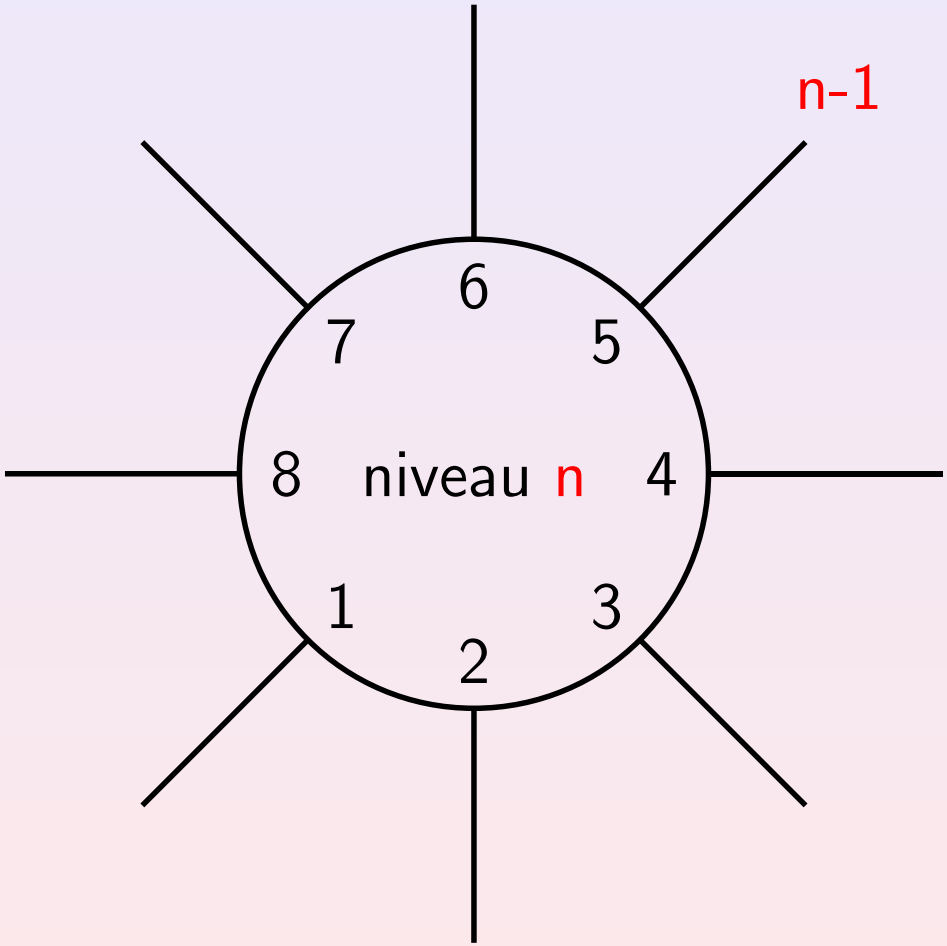
Reconnaissance du père



Reconnaissance du père



Reconnaissance du père



Seulement deux couleurs ?

- Couche 1 : rouge
- Couche 2 : bleu
- Couche 3 : rouge
- Couche 4 : rouge
- Couche 5 : rouge
- Couche 6 : bleu
- Couche 7 : bleu
- Couche 8 : bleu

Plan

Problème ouvert

Conclusion

- Trois couleurs suffisent pour les graphes arbitraires.
- Deux couleurs suffisent pour les graphes de degré constant.

Problème ouvert

Deux couleurs suffisent-elles pour les graphes arbitraires ?

Problème ouvert

Conclusion

- Trois couleurs suffisent pour les graphes arbitraires.
- Deux couleurs suffisent pour les graphes de degré constant.

Problème ouvert

Deux couleurs suffisent-elles pour les graphes arbitraires ?