

Exploration par un automate fini de réseaux anonymes étiquetés

Reuven Cohen¹ Pierre Fraigniaud² David Ilcinkas³
Amos Korman¹ David Peleg¹

¹Dept. of Computer Science, Weizmann Institute, Israel

²CNRS, LRI, Université Paris-Sud, France

³LRI, Université Paris-Sud, France

AlgoTel '05, 11 mai 2005

Exploration de graphes

But

Un automate fini (ou robot) doit **traverser** chaque arête d'un graphe **inconnu** et **anonyme**.

Motivation

- exploration d'endroits inaccessibles à l'homme
- maintenance d'un réseau informatique
- cartographie

Exploration de graphes

But

Un automate fini (ou robot) doit **traverser** chaque arête d'un graphe **inconnu** et **anonyme**.

Motivation

- exploration d'endroits inaccessibles à l'homme
- maintenance d'un réseau informatique
- cartographie

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

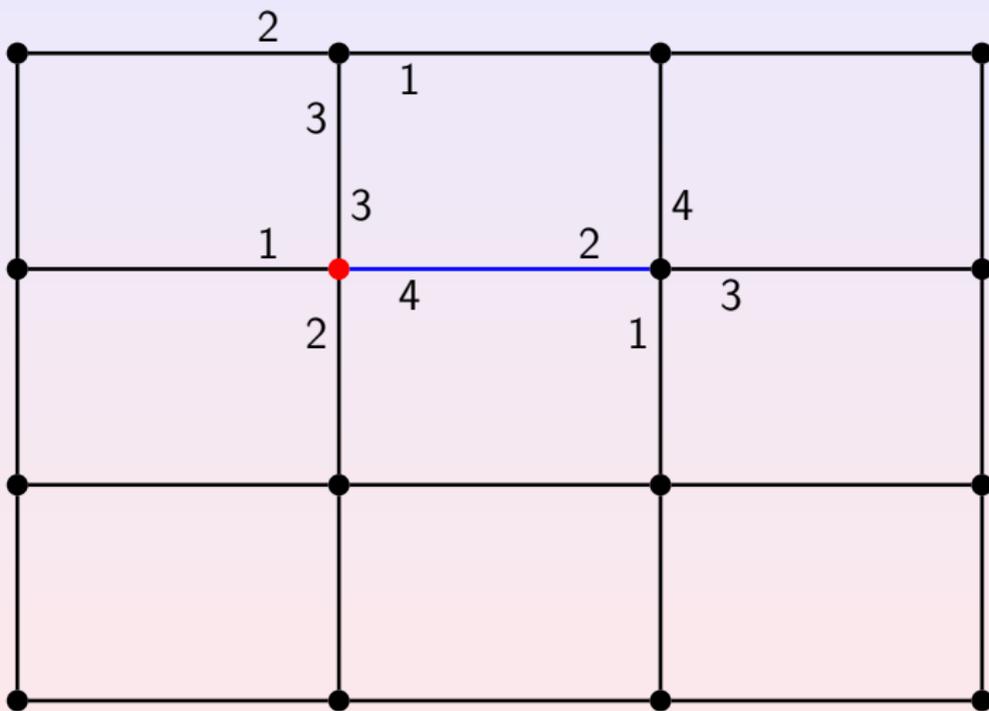
Inconnu

- topologie inconnue
- pas de bornes sur la taille

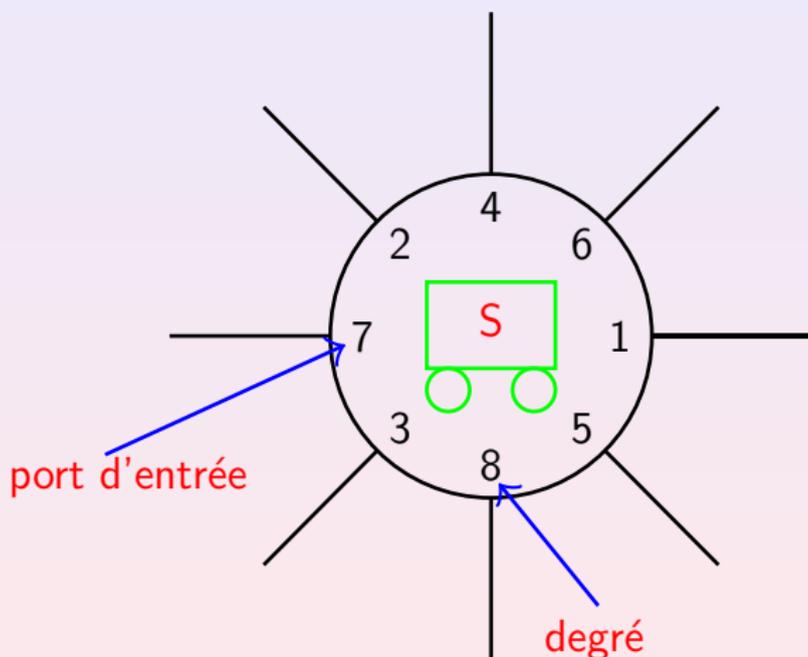
Anonyme

- sommets sans identifiant
- étiquetage local des arêtes

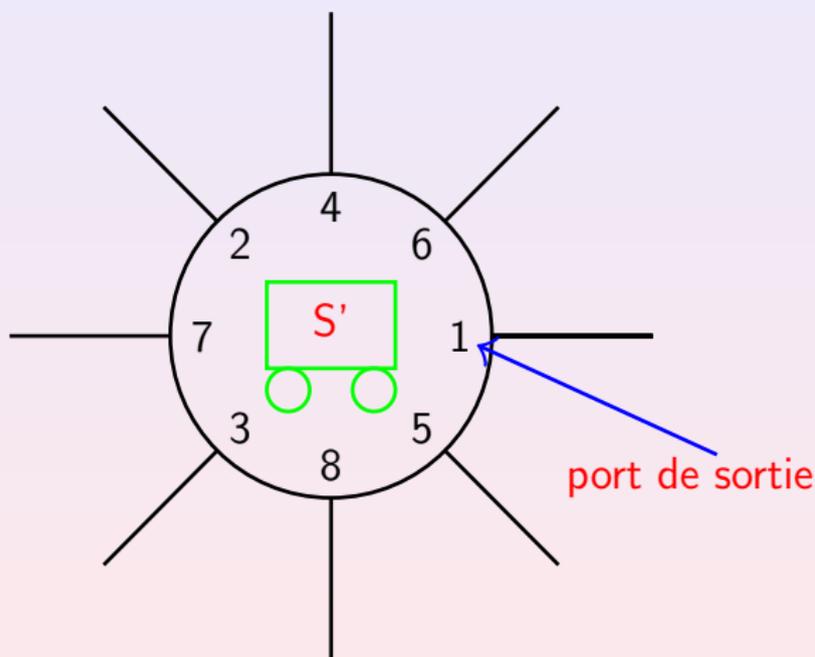
Exemple d'un graphe anonyme



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

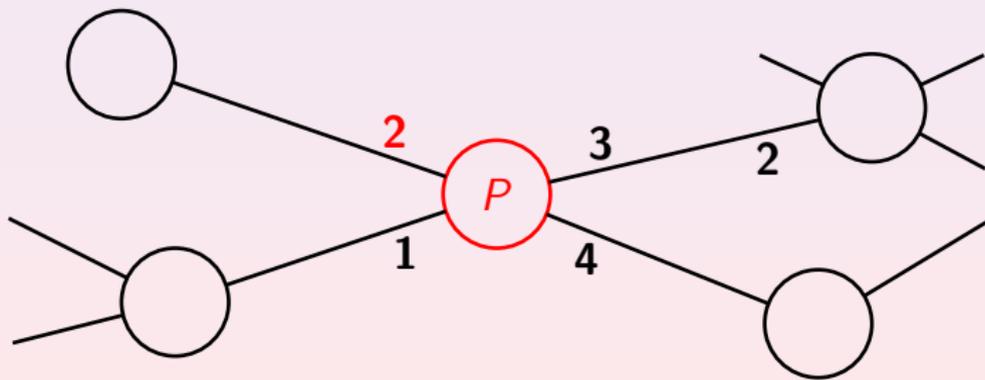
- $f(S, i, d) = (S', j)$

Exploration du voisinage

Étoile

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

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

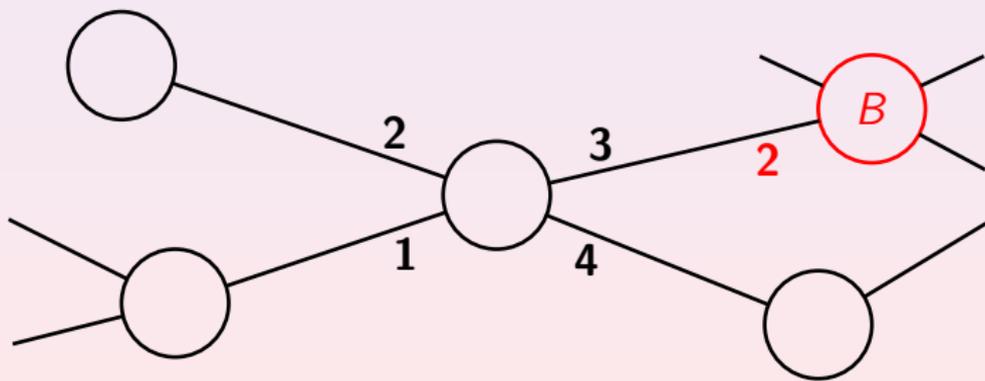


Exploration du voisinage

Étoile

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

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

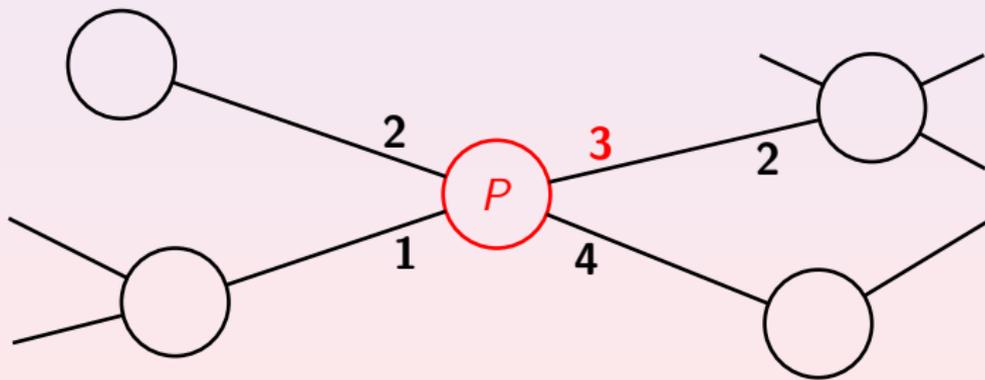


Exploration du voisinage

Étoile

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

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

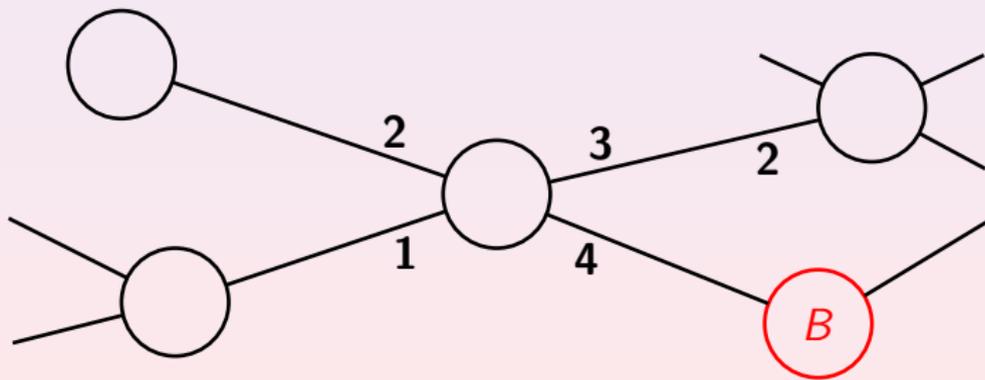


Exploration du voisinage

Étoile

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

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

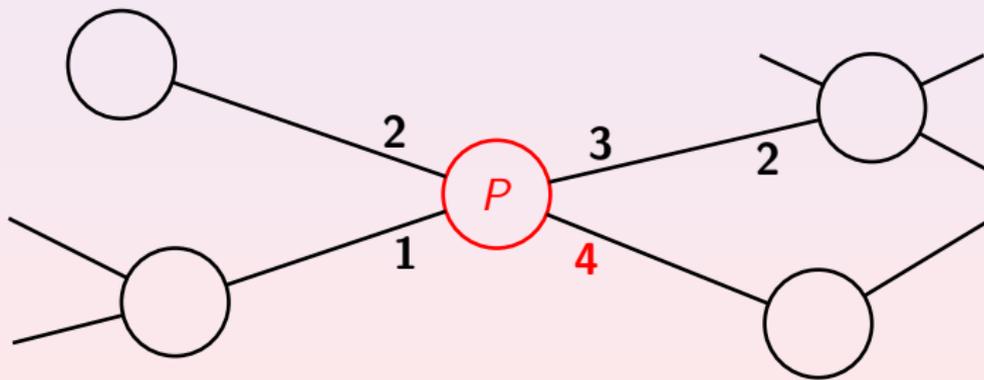


Exploration du voisinage

Étoile

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

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

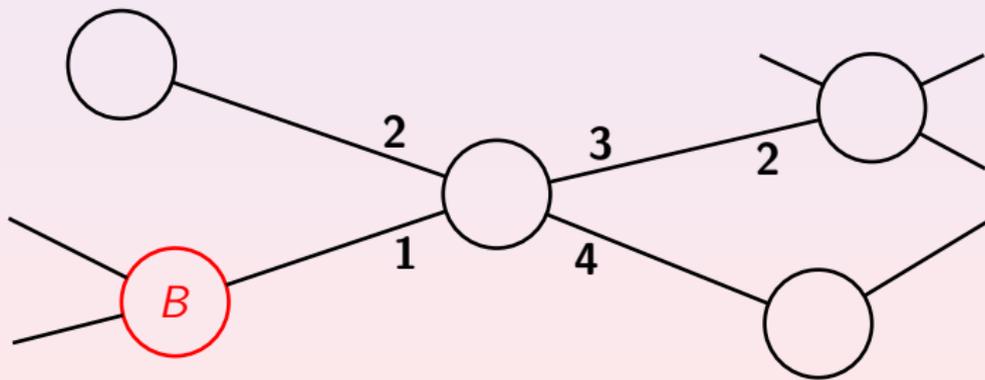


Exploration du voisinage

Étoile

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

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

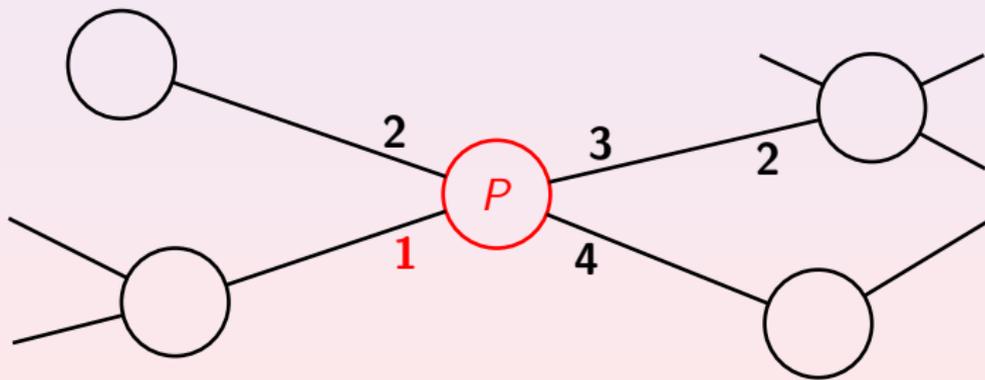


Exploration du voisinage

Étoile

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

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

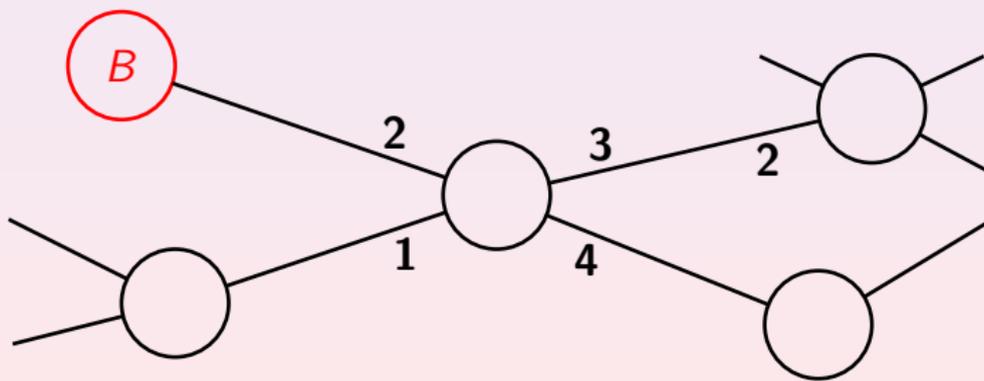


Exploration du voisinage

Étoile

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

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

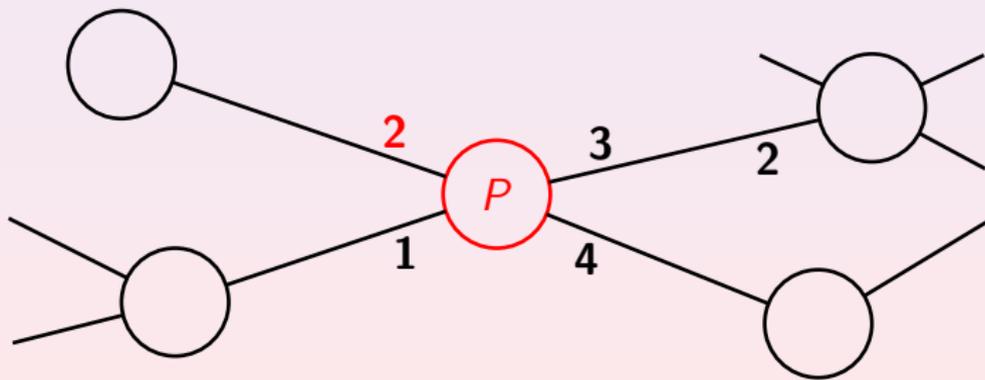


Exploration du voisinage

Étoile

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

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

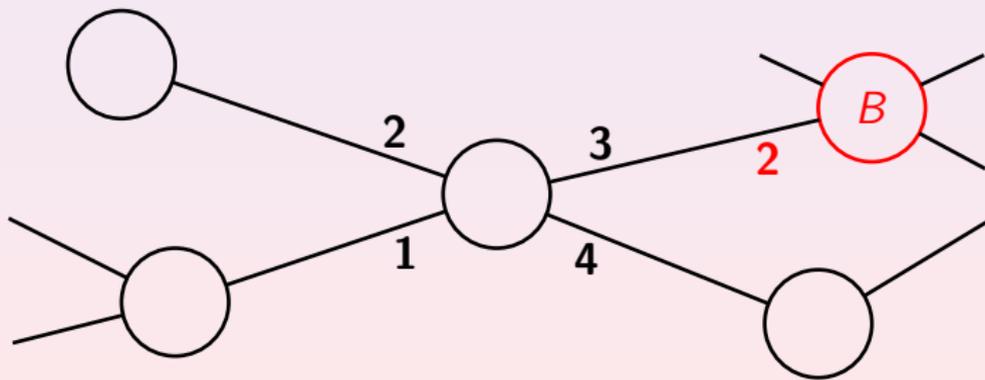


Exploration du voisinage

Étoile

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

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



Modèle utilisé

Modèle

- On autorise un oracle à colorier (étiqueter) 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.

Plan

- 1 Introduction
- 2 Historique**
 - Résultats d'impossibilité
 - Exploration d'arbres
 - Tableaux blancs
 - Résultats
- 3 Algorithmes de coloriage
- 4 Conclusion

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.

Sebastian Edelkamp, 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.

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 (même 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.

Rollik, SIAMJIC, 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 (même 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.

SIAMJOC, 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 (même 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

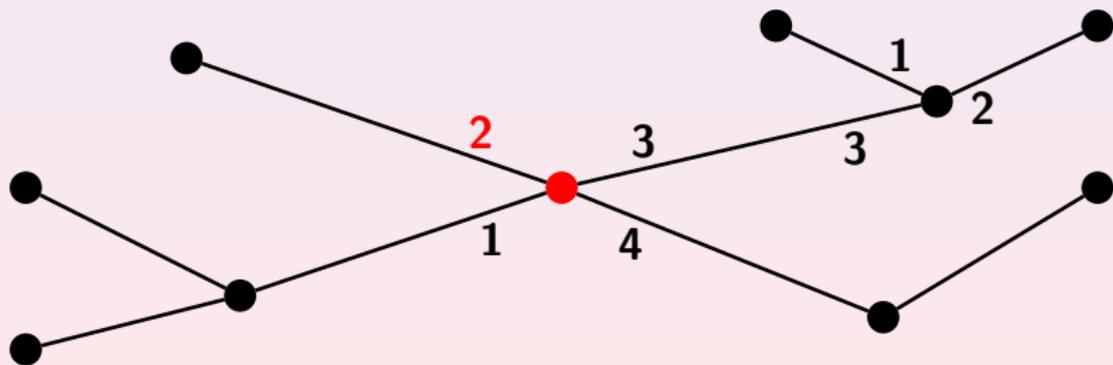
Aucun JAG n'explore tous les graphes.

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.

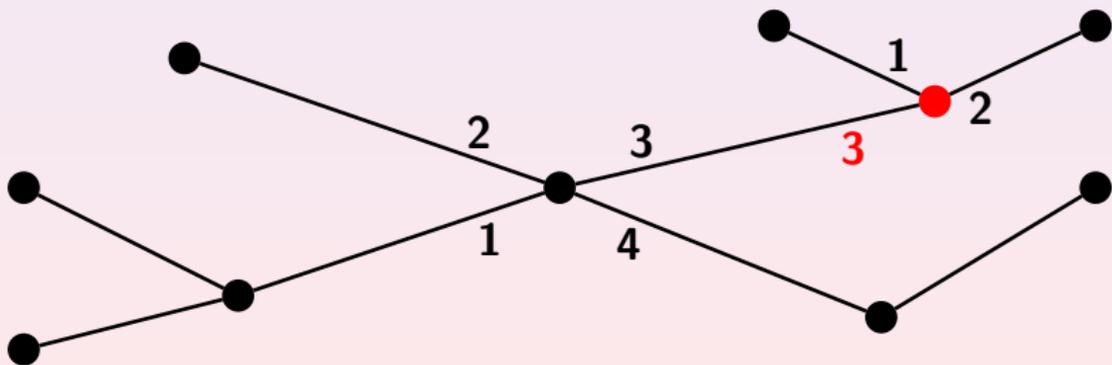


Universalité pour les arbres

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002

Tree exploration with little memory

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

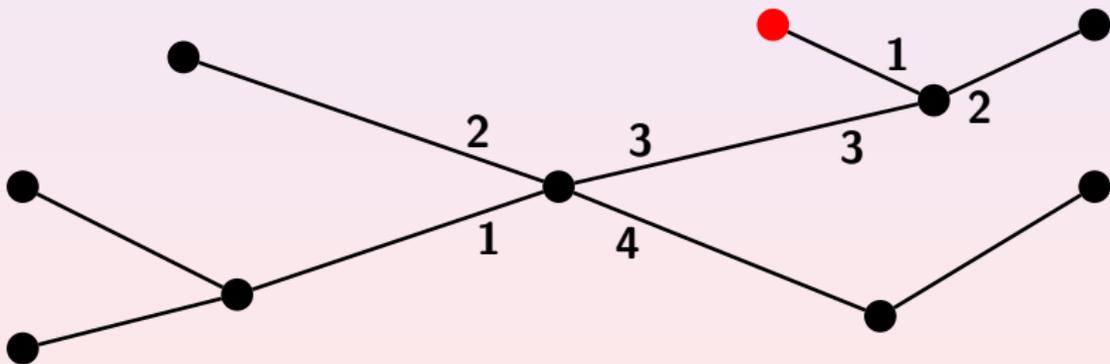


Universalité pour les arbres

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002

Tree exploration with little memory

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

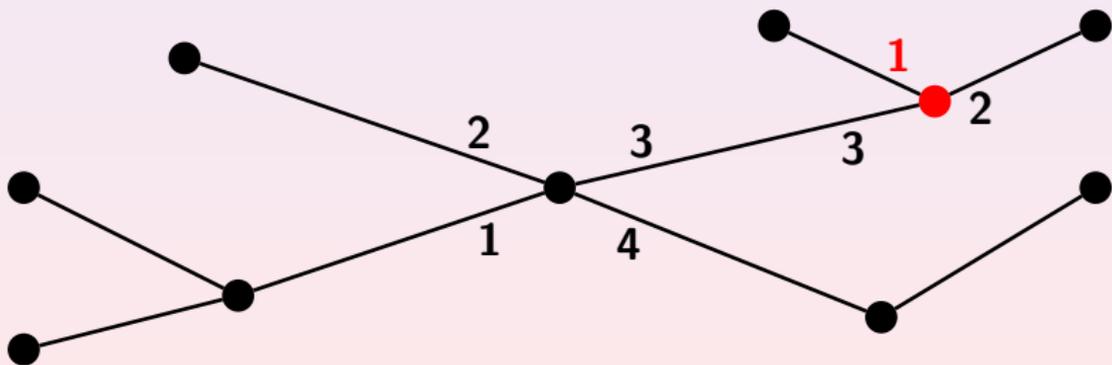


Universalité pour les arbres

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002

Tree exploration with little memory

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

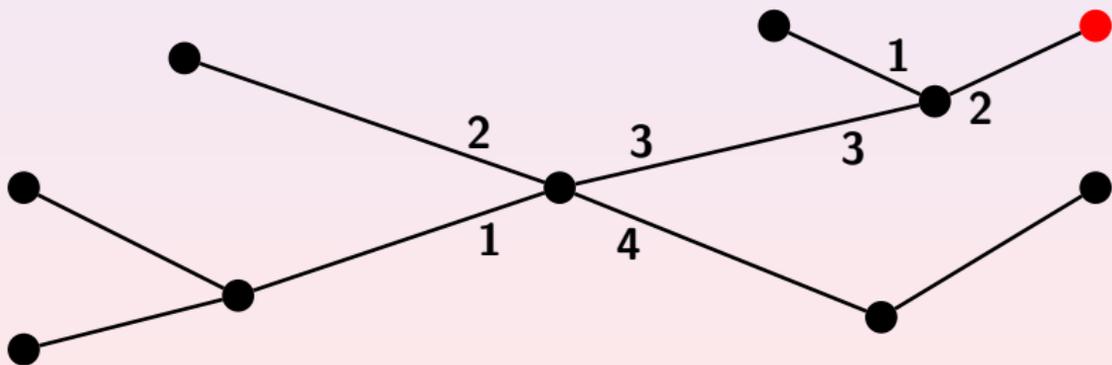


Universalité pour les arbres

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002

Tree exploration with little memory

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

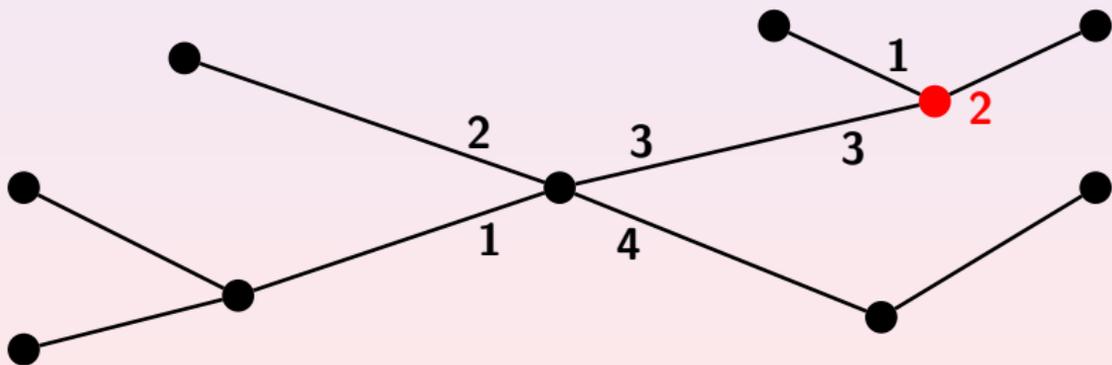


Universalité pour les arbres

Diks, Fraigniaud, Kranakis, Pelc, SODA 2002

Tree exploration with little memory

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

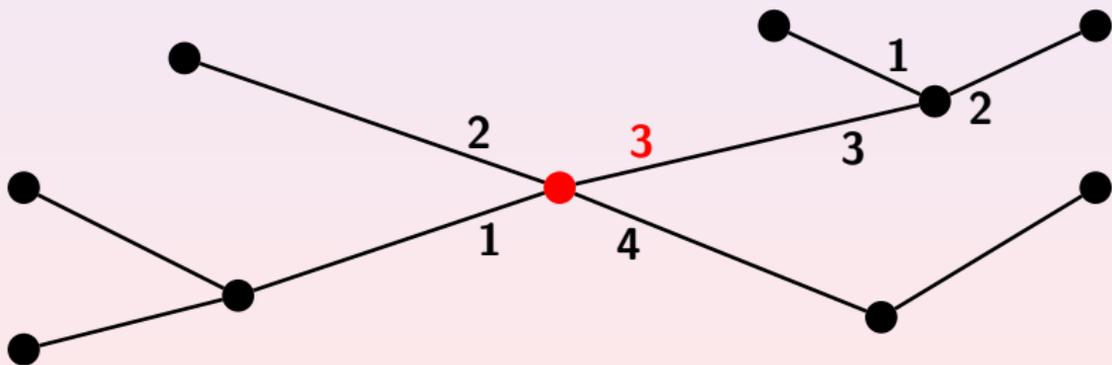


Universalité pour les arbres

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 écrire ou effacer des messages.

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 $i \bmod d$ où d est le degré du nœud.

Tableaux blancs

Un **tableau blanc** sur un nœud est un espace sur lequel un automate peut 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 $i \bmod d$ où d est le degré du nœud.

Tableaux blancs

Un **tableau blanc** sur un nœud est un espace sur lequel un automate peut 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 $i \bmod d$ où d est le degré du nœud.

Rappel du modèle

Modèle

- On autorise un oracle à colorier (étiqueter) 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

- 1 Introduction
- 2 Historique
- 3 Algorithmes de coloriage**
 - Arbre couvrant
 - Trois couleurs
 - Deux couleurs
- 4 Conclusion

Arbre couvrant

Idée simple

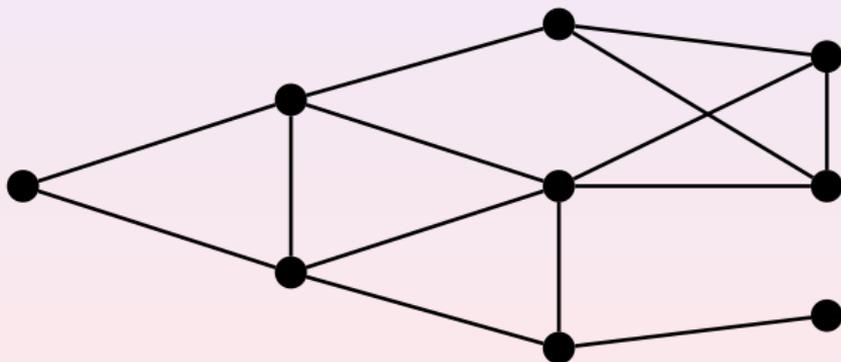
arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre



Arbre couvrant

Idée simple

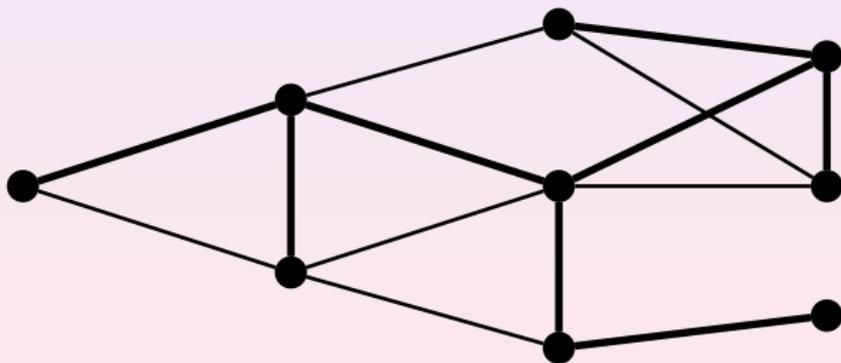
arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre



Arbre couvrant

Idée simple

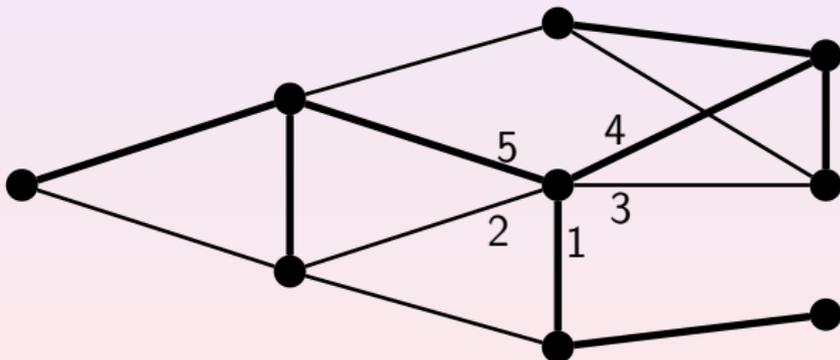
arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre



Arbre couvrant

Idée simple

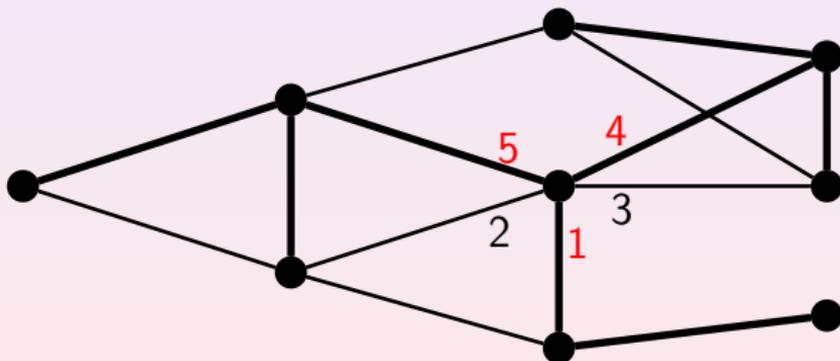
arbre couvrant : l'étiquette indique quelles arêtes appartiennent à l'arbre



Arbre couvrant

Idée simple

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



Arbre enraciné

Etiquetage amélioré

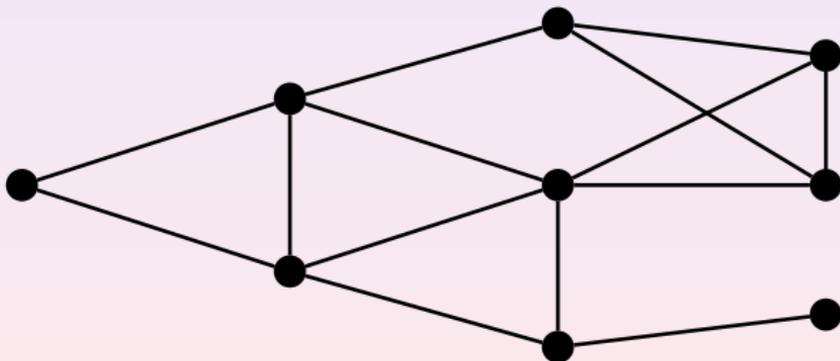
seulement indiquer l'arête menant au père \rightarrow Δ couleurs



Arbre enraciné

Etiquetage amélioré

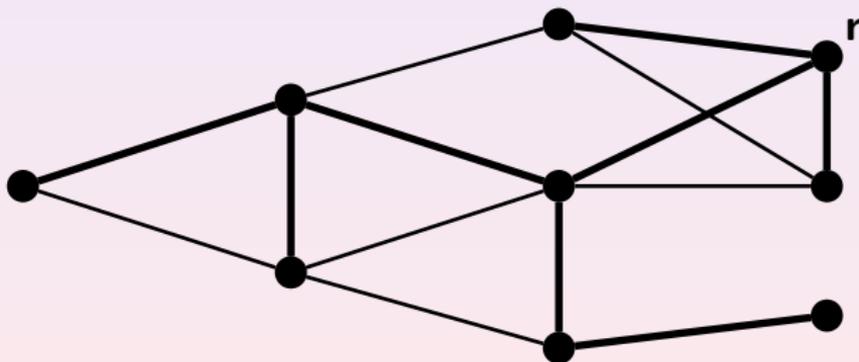
seulement indiquer l'arête menant au père \rightarrow Δ couleurs



Arbre enraciné

Etiquetage amélioré

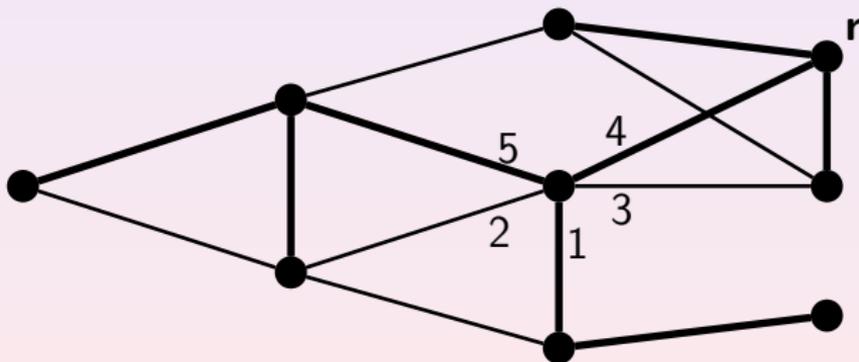
seulement indiquer l'arête menant au père \rightarrow Δ couleurs



Arbre enraciné

Etiquetage amélioré

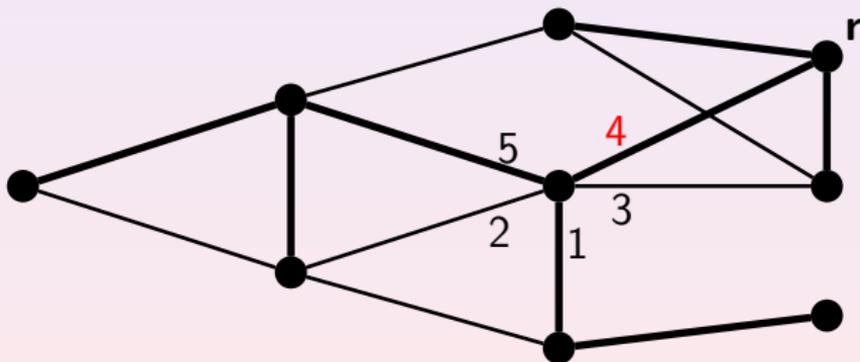
seulement indiquer l'arête menant au père \rightarrow Δ couleurs



Arbre enraciné

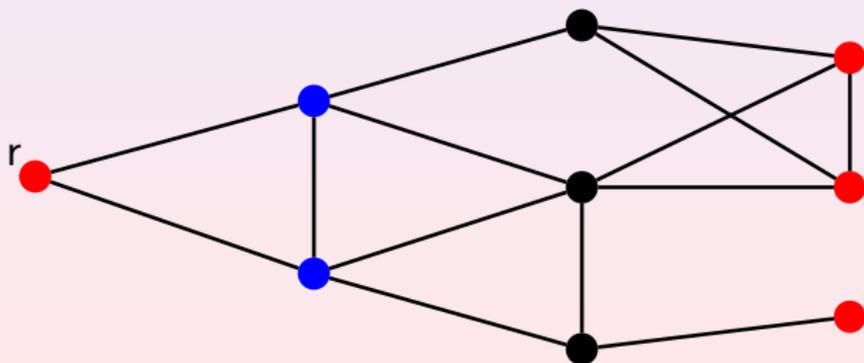
Etiquetage amélioré

seulement indiquer l'arête menant au père \rightarrow Δ couleurs



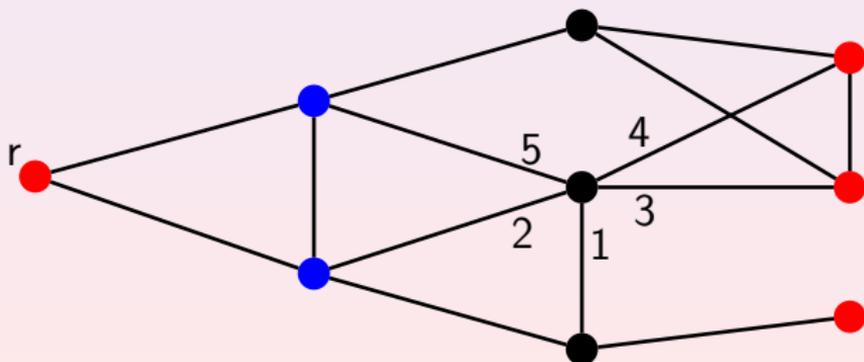
Trois couleurs suffisent

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



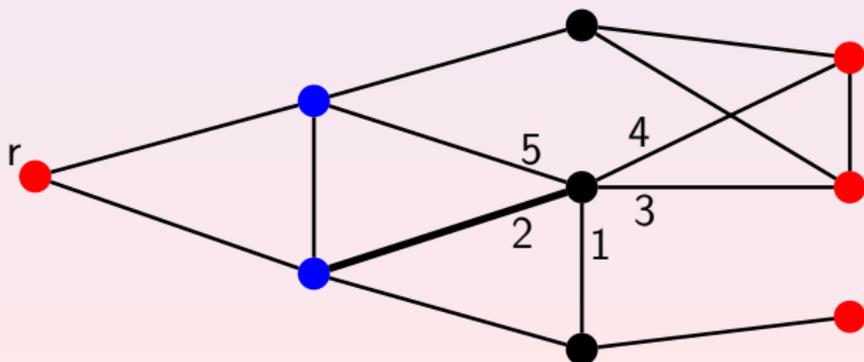
Trois couleurs suffisent

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

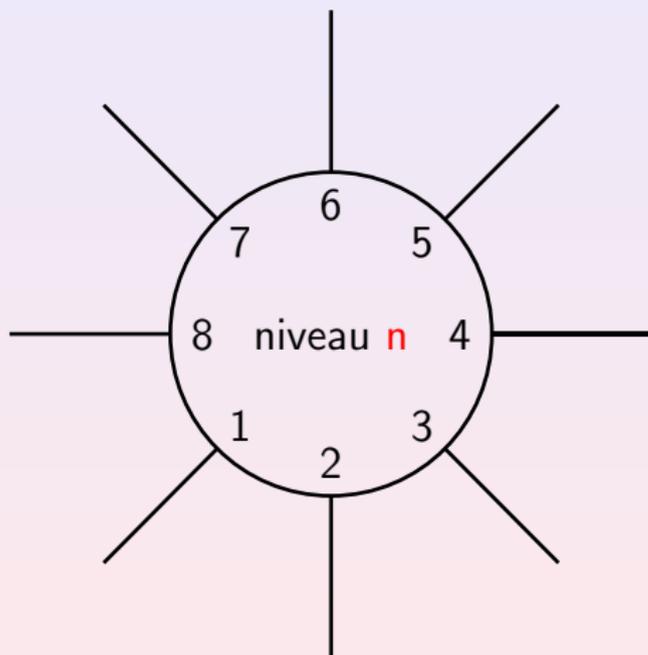


Trois couleurs suffisent

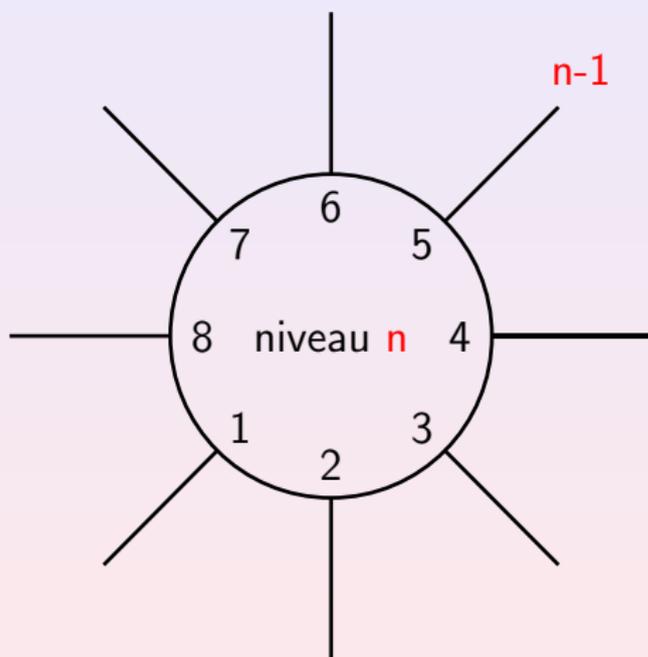
- choisir arbitrairement un nœud comme racine
- colorier tous les nœuds en fonction de leur distance d à la racine
 - distance $d \cong 0[3]$ rouge
 - distance $d \cong 1[3]$ bleu
 - distance $d \cong 2[3]$ noir



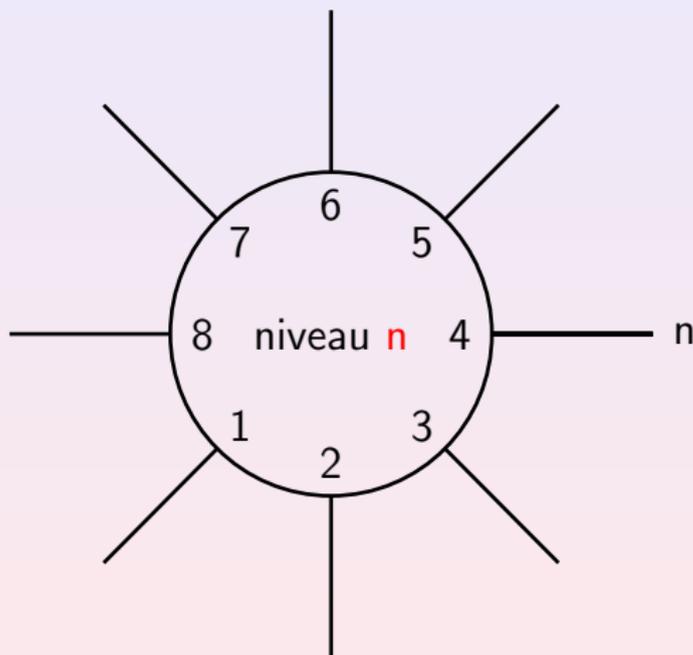
Reconnaissance du père



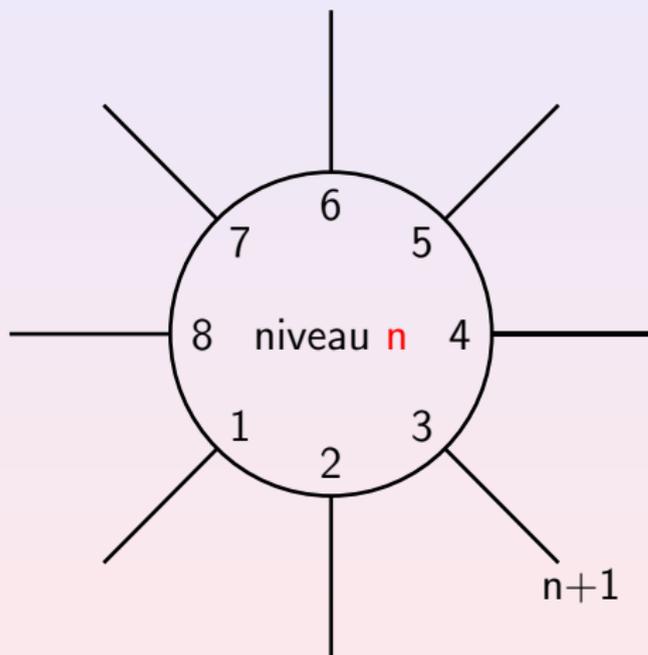
Reconnaissance du père



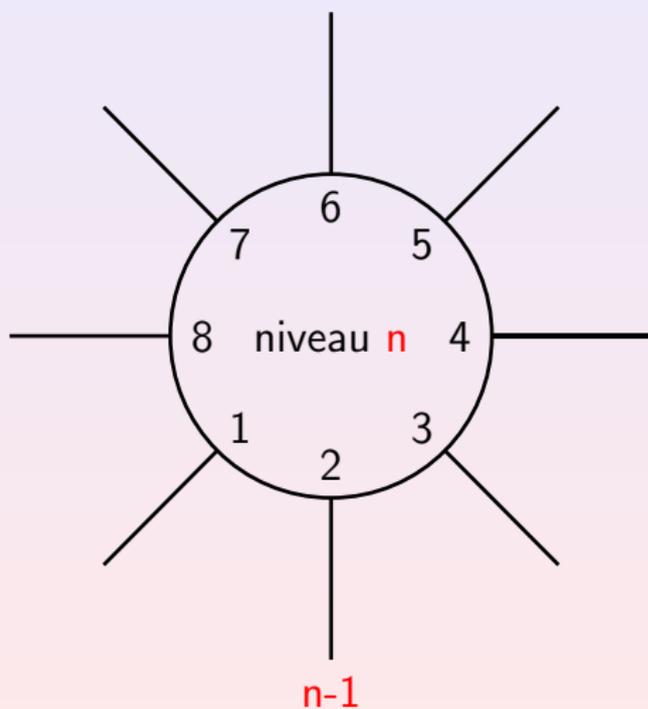
Reconnaissance du père



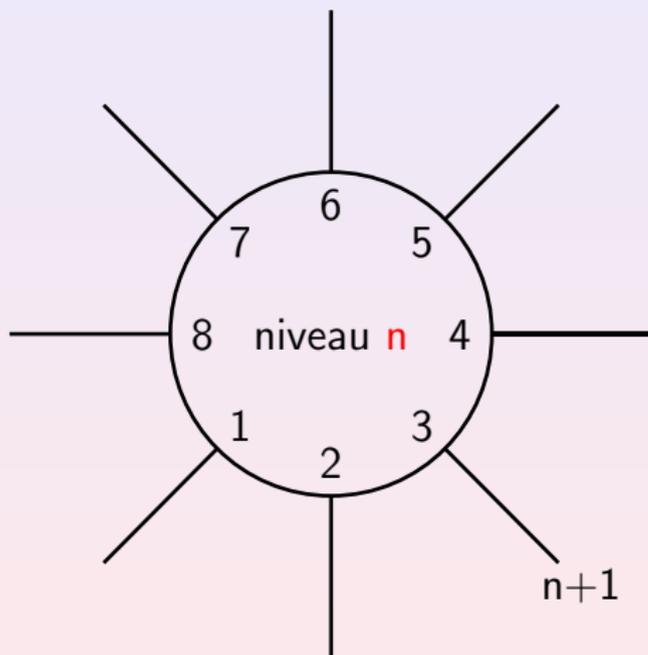
Reconnaissance du père



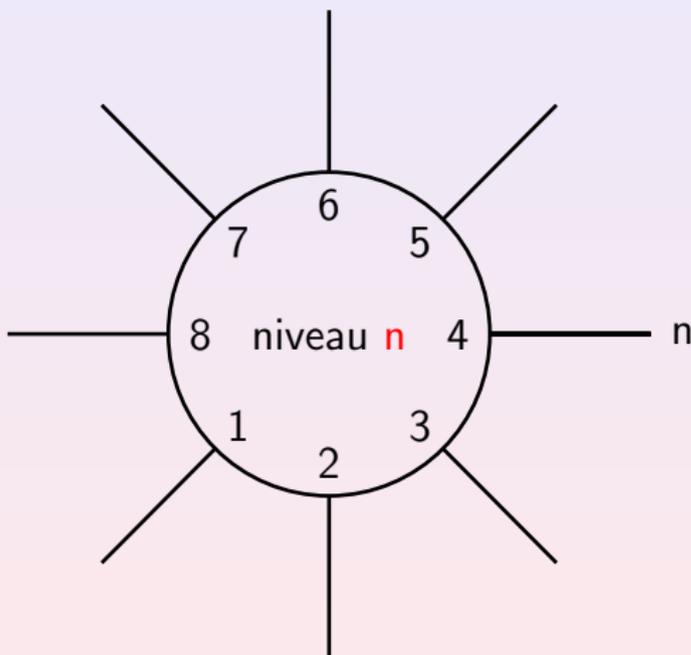
Reconnaissance du père



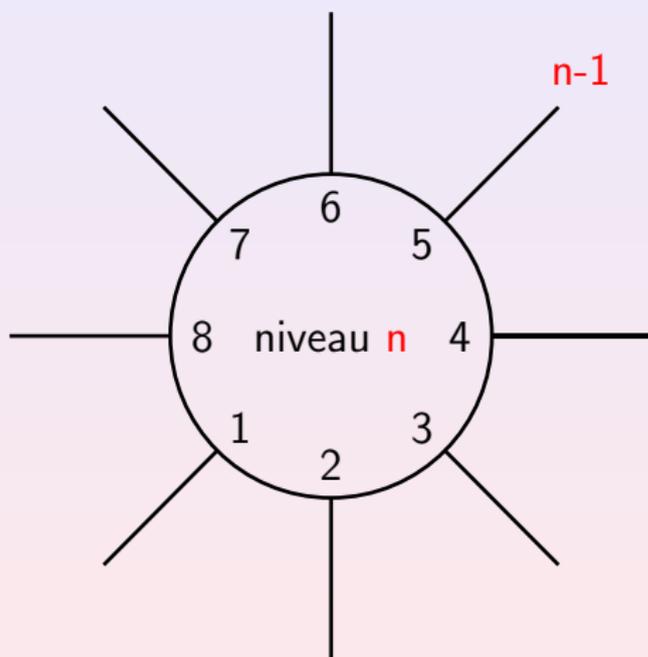
Reconnaissance du père



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

- 1 Introduction
- 2 Historique
- 3 Algorithmes de coloriage
- 4 Conclusion**

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** ?