Complexité des problèmes algorithmiques

Florent Foucaud - Pascal Lafourcade - Malika More
(malika.more@uca.fr)

3A - BUT INFO - UCA

Qualité algorithmique

vs. complexité des algorithmes

- Problème algorithmique : une entrée, une sortie
  - Trier une liste de n entiers
  - Déterminer si un graphe est 3-colorable
  - Déterminer si une formule en CNF est satisfaisable
- Algorithme
  - série d'instructions qui résoud un certain problème algorithmique
  - pour une machine $\approx$ programme informatique
- Complexité d'un algorithme : quantité de ressources nécessaires à l'algorithme, en fonction de la taille n de l'entrée
- Complexité d'un problème algorithmique : meilleure complexité d'un algorithme qui résoud ce problème

Plan du cours

Problème du tri

Problèmes de décision

Comparaison de problèmes algorithmiques

Plan du cours

Problème du tri

Problèmes de décision

Comparaison de problèmes algorithmiques

Complexité du problème du tri

Tri

- `Entrée` : un tableau de n nombres
- `Sortie` : un tableau des mêmes nombres rangés par ordre croissant

On a vu

- Des algorithmes de tri dont la complexité dans le pire des cas est en $\mathcal{O}(n^2)$ (tri par insertion, tri par sélection, tri à bulles)

  $\rightsquigarrow$ La complexité du problème du tri est en $\mathcal{O}(n^2)$

Oui mais...

- Le tri fusion, dont la complexité dans le pire des cas est en $\mathcal{O}(n \log n)$

  $\rightsquigarrow$ La complexité du problème du tri est en $\mathcal{O}(n \log n)$

Complexité du problème du tri

Tri

- `Entrée` : un tableau de n nombres
- `Sortie` : un tableau des mêmes nombres rangés par ordre croissant

Borne inférieure de complexité

- Pour trier n éléments, il est indispensable de les lire tous au moins une fois

  $\rightsquigarrow$ La complexité du problème du tri est en $\Omega(n)$

Peut-on faire mieux ?

- Des tris plus rapides que le tri fusion en $\mathcal{O}(n \log n)$?

  $\rightsquigarrow$ Existe-t-il des algorithmes de tri en $\mathcal{O}(n)$?

Tris par comparaisons

Définition

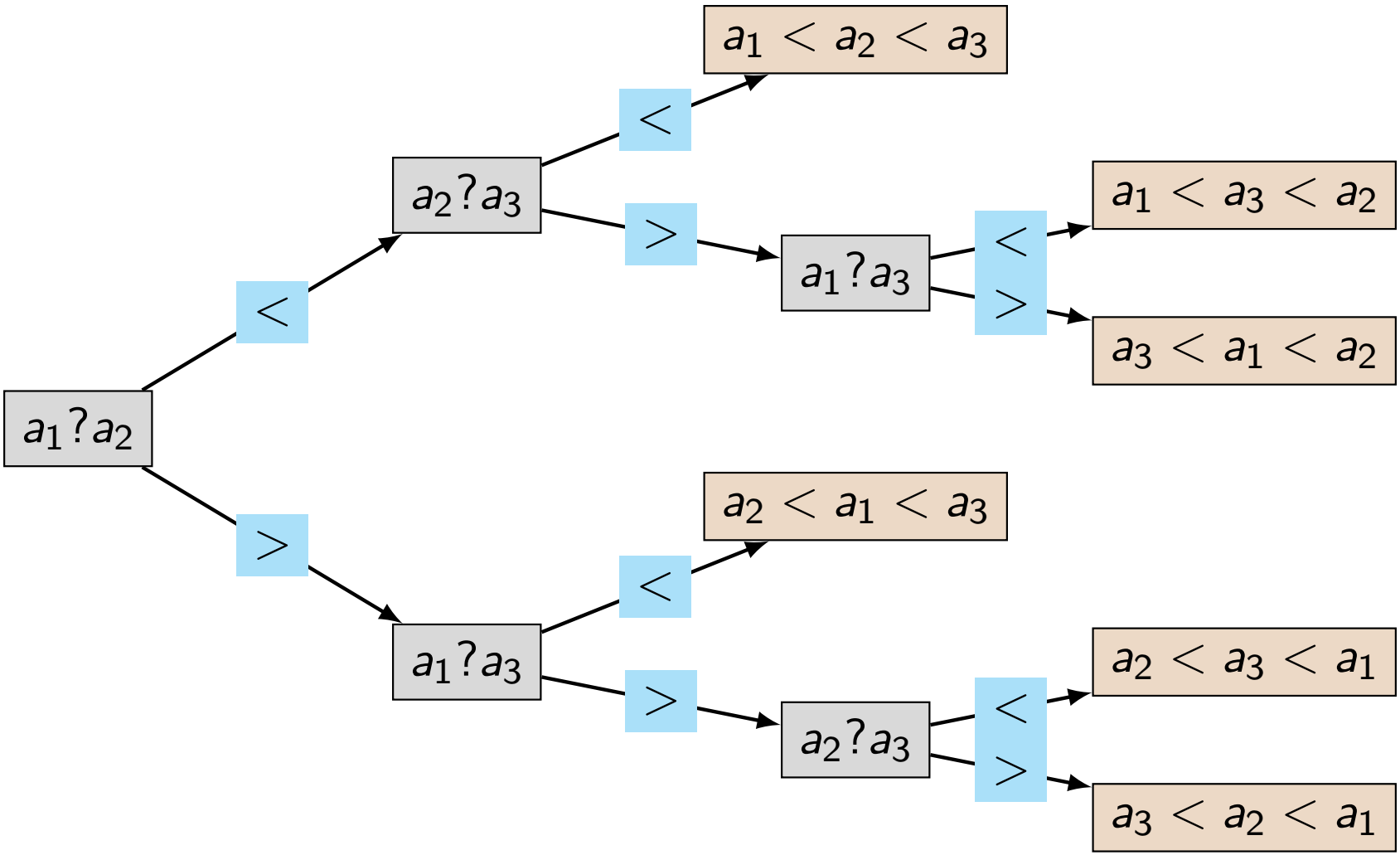
- Entrée $a_1, a_2, \ldots, a_n$
- Les informations sur les objets à trier sont obtenues uniquement par des comparaisons $a_i \stackrel{?}{<} a_j$
- (On suppose $a_1, a_2, \ldots, a_n$ tous distincts)

Tous les algorithmes de tri qu'on a vus sont des tris par comparaisons

Modèle abstrait

- Arbre de décision
- Représenter toutes les comparaisons effectuées par l'algorithme sur des entrées d'une taille donnée

Tri par insertion de 3 éléments a_1, a_2, a_3



Arbre de décision pour un tri sur n éléments

Structure

- Nœud interne : comparaison $a_i ? a_j$
- Arbre binaire : deux réponses possibles pour chaque comparaison
- Feuille : résultat du tri (étiquetée par une permutation de $1, \ldots, n$)

Exécution de l'algorithme

- Chemin de la racine à une feuille
- Nombre de comparaisons = nombre de nœuds internes

Algorithme correct

Toutes les $n!$ permutations de $1, \ldots, n$ sont des feuilles

Nombre de comparaisons

Pire des cas

- Plus long chemin de la racine à une feuille
- Hauteur de l'arbre de décision

Théorème

Un arbre de décision permettant de trier n éléments est de hauteur $\Omega(n \log n)$

Remarque (Notation Ω)

$f(n) = \underset{n \to +\infty}{\Omega}(g(n))$ ssi $\exists N, c$ t.q. $\forall n > N \; c\,|g(n)| \leq |f(n)|$

Preuve du théorème

Théorème

Un arbre de décision permettant de trier n éléments est de hauteur $\Omega(n \log n)$

- hauteur de l'arbre h
- un arbre binaire de hauteur h possède au plus 2^h feuilles
- cet arbre de décision permet de trier n éléments
- toutes les $n!$ permutations apparaissent comme feuilles de l'arbre
- donc $n! \leq 2^h$ ou encore $h \geq \log_2(n!)$
- formule de Stirling $n! > \left(\frac{n}{e}\right)^n$
- finalement
  $h \geq \log_2\left(\left(\frac{n}{e}\right)^n\right) = n\log_2(n) - n\log_2(e) = \Omega(n \log n)$

La réponse

Théorème

Un arbre de décision permettant de trier n éléments est de hauteur $\Omega(n \log n)$

Corollaire

- Aucun tri par comparaisons ne possède une complexité meilleure que $\mathcal{O}(n \log n)$
- Le tri fusion a une complexité optimale

Remarque

Ce résultat ne concerne que les tris par comparaisons, puisqu'il est basé sur le modèle des arbres de décision.

D'autres tris

Deux exemples qui ne sont pas des tris par comparaisons

- Tri par dénombrement
  - permet de trier des nombres bornés
- Tri par base
  - permet de trier des nombres de longueur bornée

Avant-goût

On va voir que ces tris ont une complexité linéaire

Remarque

Mais ce sont des tris spécialisés, qui ne permettent pas de trier n'importe quelles données

Tri par dénombrement

Hypothèse

Les entiers à trier appartiennent à $\{0, \ldots, k-1\}$

Principe

- Créer un tableau `Compte` de longueur k rempli de 0
- Parcourir le tableau `T` à trier
- Pour tout i, incrémenter `Compte[T[i]]`
- Ensuite, parcourir de nouveau le tableau `T` en le remplissant de `Compte[0]` fois le nombre 0, suivis de `Compte[1]` fois le nombre 1, etc.

Tri par dénombrement

`T` =

0	2	1	0	3	2	2	0	0	1	3	0	3	3	3

- Créer un tableau `Compte` de longueur k rempli de 0

  `Compte` =

0	0	0	0

- Parcourir le tableau `T` à trier
- Pour tout i, incrémenter `Compte[T[i]]`

  `Compte` =

5	2	3	5

- Ensuite, parcourir de nouveau le tableau `T` en le remplissant de `Compte[0]` fois le nombre 0, suivis de `Compte[1]` fois le nombre 1, etc.

`T` =

0	0	0	0	0	1	1	2	2	2	3	3	3	3	3

Tri par dénombrement

Complexité

Pourvu que k ne soit pas trop grand (c.-à-d. $k = \mathcal{O}(n)$), le tri par dénombrement est en $\mathcal{O}(n)$ car essentiellement on parcourt deux fois le tableau à trier.

Remarque

Une contradiction ?
Non, car ce n'est pas un tri par comparaisons !
Cet algorithme n'effectue d'ailleurs **aucune** comparaison...

Tri par base

Hypothèse

Les entiers à trier ont une longueur fixe

Exemple

$$
\begin{array}{c}
329 \\ 457 \\ 657 \\ 839 \\ 436 \\ 720 \\ 355
\end{array}
\Rightarrow
\begin{array}{c}
720 \\ 355 \\ 436 \\ 457 \\ 657 \\ 329 \\ 839 \\ \uparrow
\end{array}
\Rightarrow
\begin{array}{c}
720 \\ 329 \\ 436 \\ 839 \\ 355 \\ 457 \\ 657 \\ \uparrow
\end{array}
\Rightarrow
\begin{array}{c}
329 \\ 355 \\ 436 \\ 457 \\ 657 \\ 720 \\ 839 \\ \uparrow
\end{array}
$$

Tri par base

Principe

- On trie d'abord selon le chiffre des unités
- Puis selon le chiffre suivant, etc.
- On termine par le chiffre de plus grand poids
- Il est essentiel d'utiliser pour les étapes intermédiaires un tri stable

Remarque

Un tri stable est un tri qui conserve l'ordre relatif des indices des valeurs égales

Tri par base

Complexité

- Tout dépend de celle du tri stable utilisé...
- Il semble raisonnable d'utiliser un tri par dénombrement, puisque les chiffres dans une base donnée sont peu nombreux.
- La complexité du tri par base est alors en $\mathcal{O}(n)$, puisqu'on applique un nombre fixe de fois un algorithme linéaire.

Plan du cours

Problème du tri

Problèmes de décision

Comparaison de problèmes algorithmiques

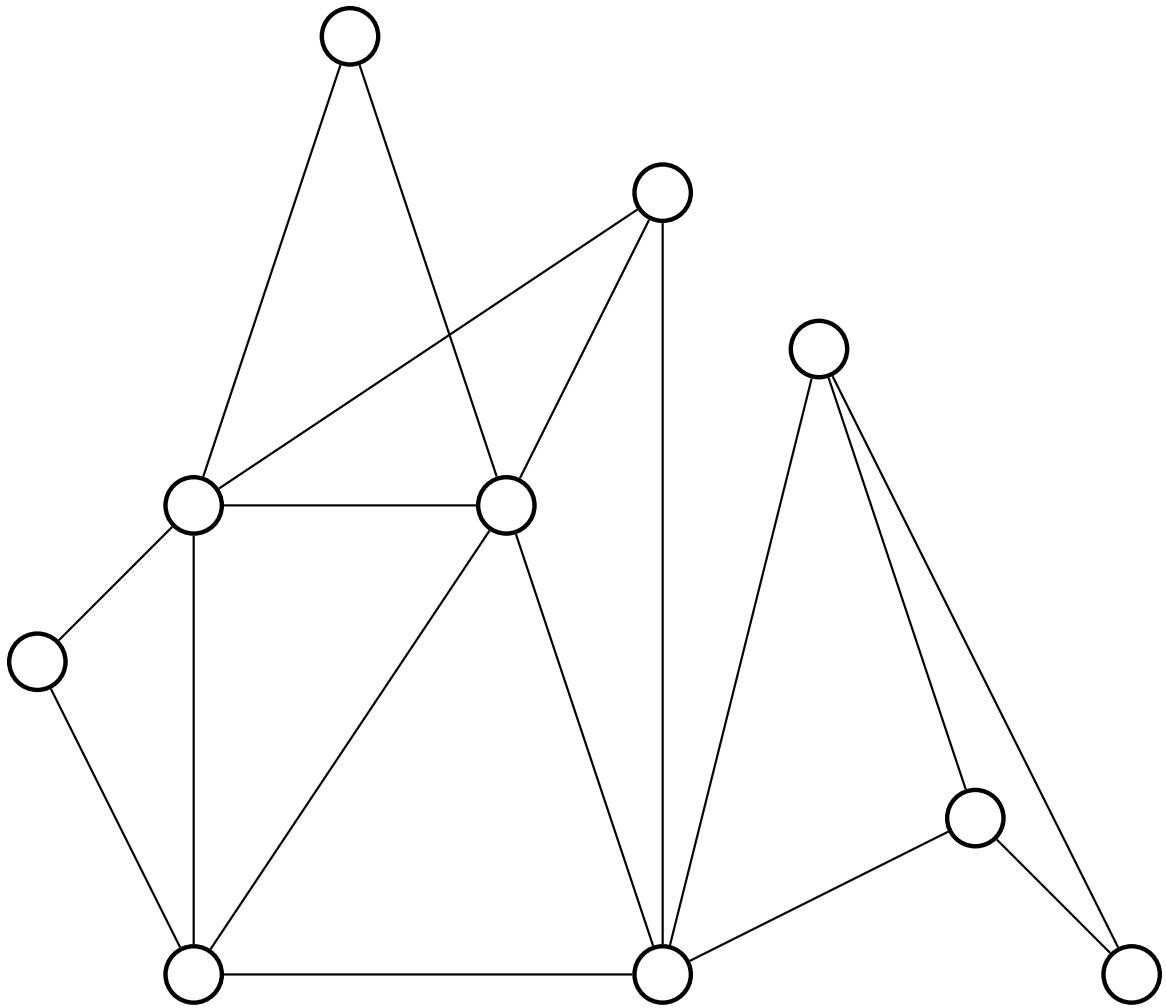
Vocabulaire

Problème de décision $\rightsquigarrow$ problème algorithmique à réponse oui/non

- **k-colorabilité** : déterminer si un graphe est k-colorable
- **SAT** : déterminer si une formule en CNF est satisfaisable

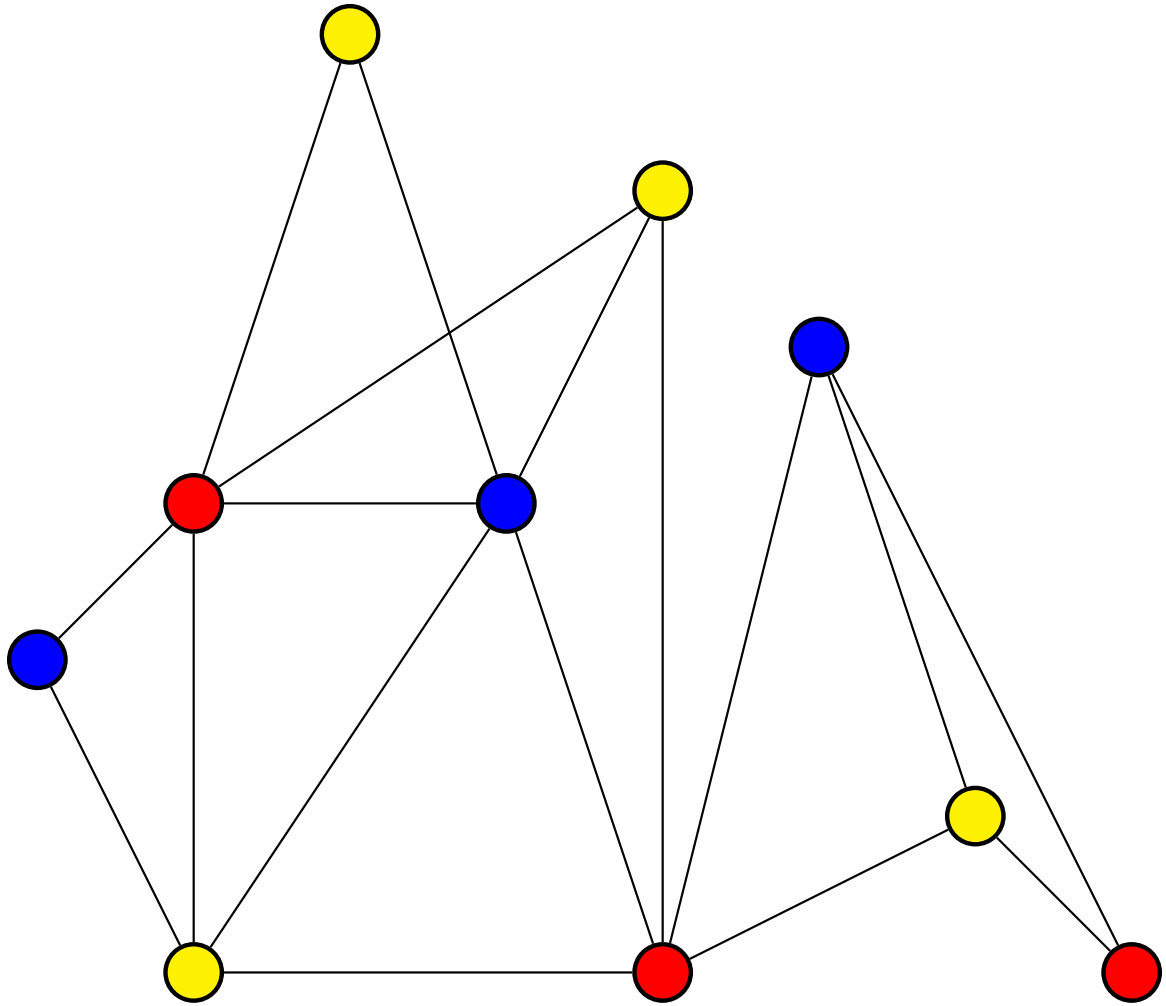
3-colorabilité

- Deux sommets reliés par une arête ne peuvent pas être de la même couleur
- Peut-on colorier tous les sommets avec 3 couleurs ?



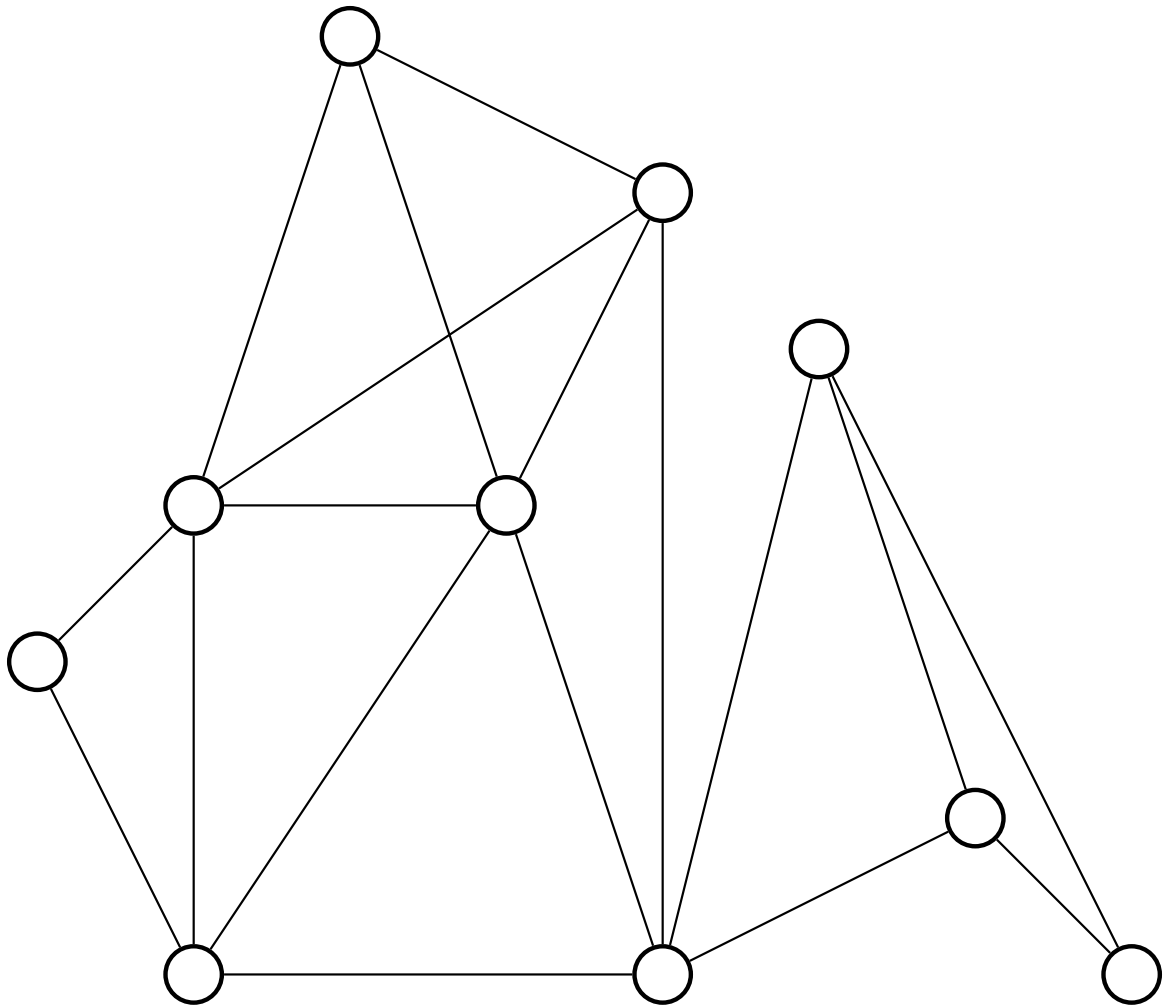
3-colorabilité

- Deux sommets reliés par une arête ne peuvent pas être de la même couleur
- Peut-on colorier tous les sommets avec 3 couleurs ?



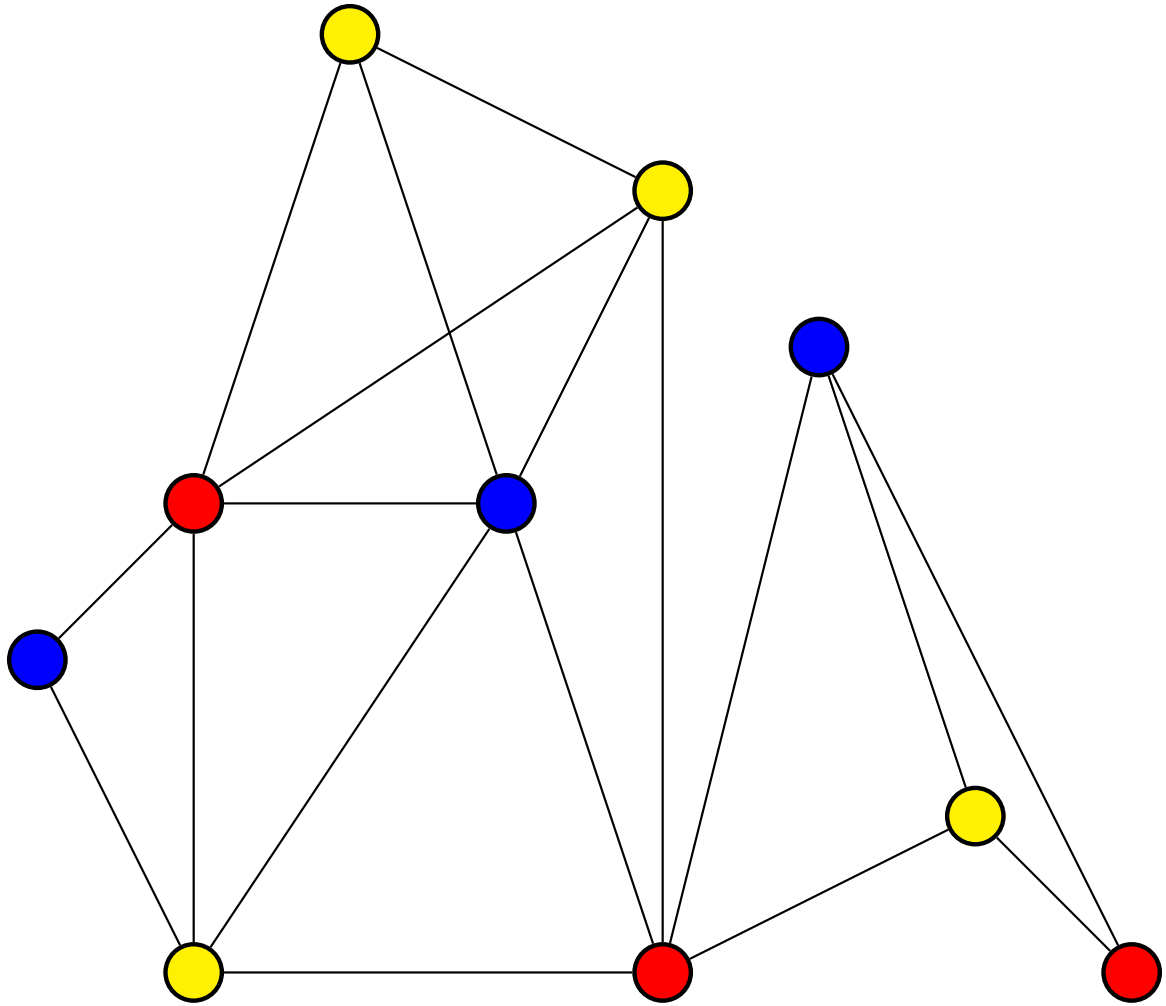
3-colorabilité

- Deux sommets reliés par une arête ne peuvent pas être de la même couleur
- Peut-on colorier tous les sommets avec 3 couleurs ?



3-colorabilité

- Deux sommets reliés par une arête ne peuvent pas être de la même couleur
- Peut-on colorier tous les sommets avec 3 couleurs ?



Un problème algorithmique célèbre

k-colorabilité

- `Entrée` : un graphe G
- `Question` : le graphe G peut-il être colorié avec k couleurs ?

(Deux sommets reliés par une arête ne doivent pas être de la même couleur)

Un problème algorithmique célèbre

k-colorabilité

- `Entrée` : un graphe G
- `Question` : le graphe G peut-il être colorié avec k couleurs ?

(Deux sommets reliés par une arête ne doivent pas être de la même couleur)

NE PAS CONFONDRE

vérifier une k-coloration

- `Entrée` : un graphe G colorié avec k couleurs
- `Question` : cette coloration est-elle valide ?

Complexité $\rightsquigarrow \mathcal{O}(n+m)$ (n sommets, m arêtes)

Un algorithme qui marche toujours, mais...

Recherche exhaustive

- Tester tous les coloriages possibles
- Vérifier la validité de chaque coloriage
- Décider si oui ou non G est k-coloriable

$$n \text{ sommets} \quad \rightsquigarrow \quad k^n \text{ coloriages}$$

Dans la vraie vie

- n est souvent grand
- le temps de calcul devient rapidement impraticable

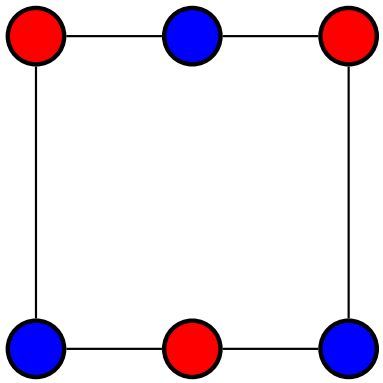
Complexité de cet algorithme

- $\mathcal{O}((n+m)k^n)$ $\quad \rightsquigarrow \quad$ exponentielle

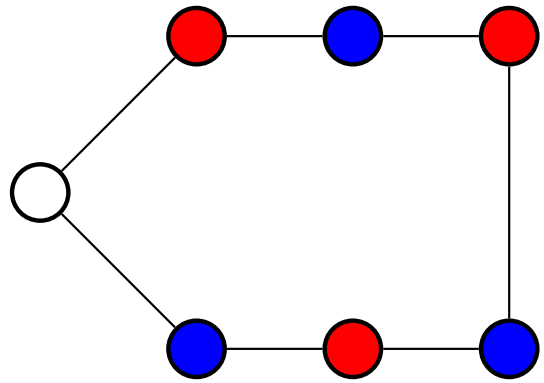
Une version simple

2-**colorabilité**

- `Entrée` : un graphe G
- `Question` : le graphe G peut-il être colorié avec 2 couleurs ?



OUI

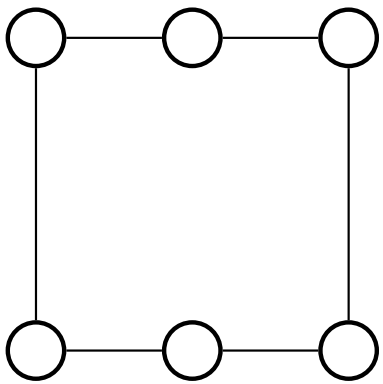


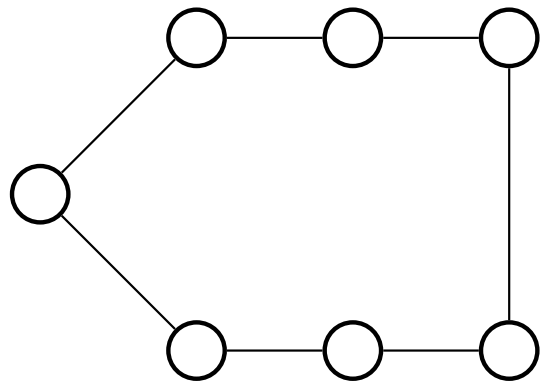
NON

Un algorithme rapide pour la 2-colorabilité

- assigner une couleur au hasard à un sommet au hasard
- assigner l'autre couleur à tous ses voisins
- propager la 2-coloration aux voisins des voisins, etc.

$\rightsquigarrow$ complexité $\mathcal{O}(n+m)$

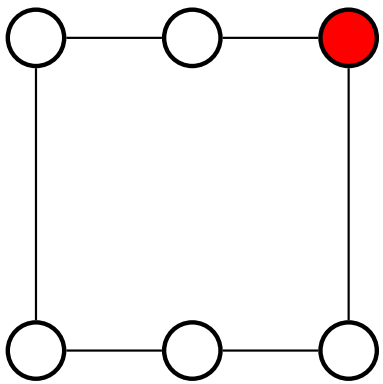


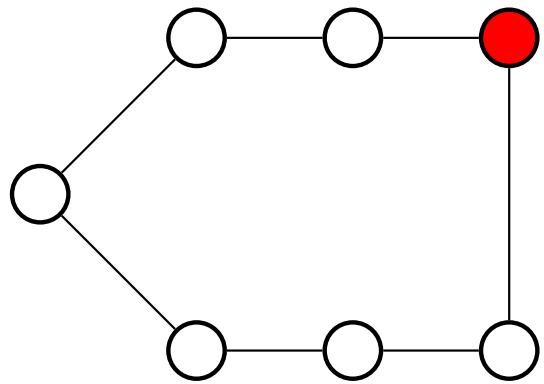


Un algorithme rapide pour la 2-colorabilité

- assigner une couleur au hasard à un sommet au hasard
- assigner l'autre couleur à tous ses voisins
- propager la 2-coloration aux voisins des voisins, etc.

$\rightsquigarrow$ complexité $\mathcal{O}(n+m)$

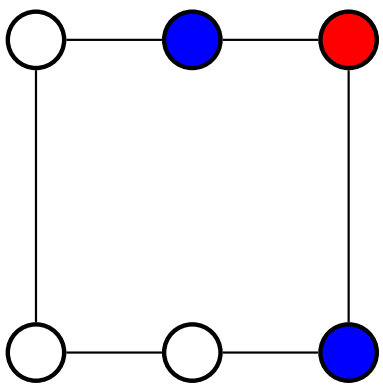


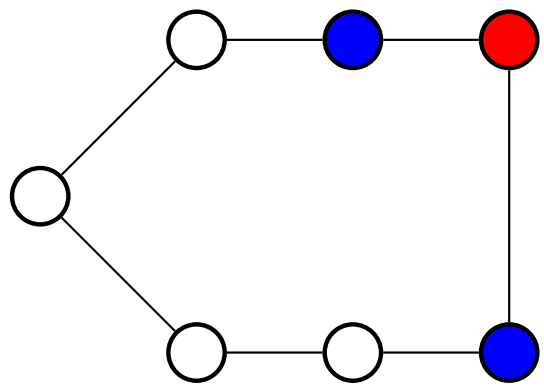


Un algorithme rapide pour la 2-colorabilité

- assigner une couleur au hasard à un sommet au hasard
- assigner l'autre couleur à tous ses voisins
- propager la 2-coloration aux voisins des voisins, etc.

$\rightsquigarrow$ complexité $\mathcal{O}(n+m)$

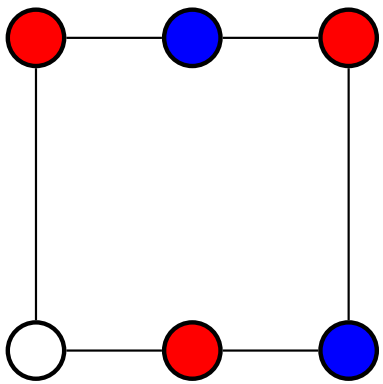


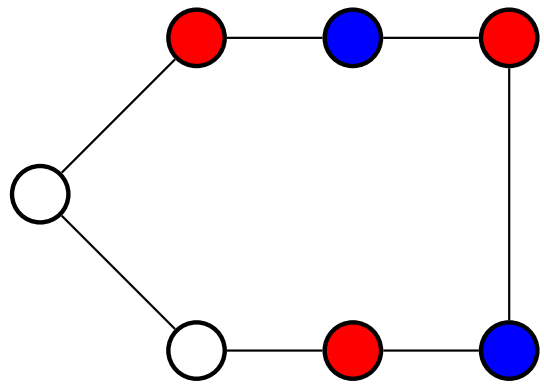


Un algorithme rapide pour la 2-colorabilité

- assigner une couleur au hasard à un sommet au hasard
- assigner l'autre couleur à tous ses voisins
- propager la 2-coloration aux voisins des voisins, etc.

$\rightsquigarrow$ complexité $\mathcal{O}(n+m)$

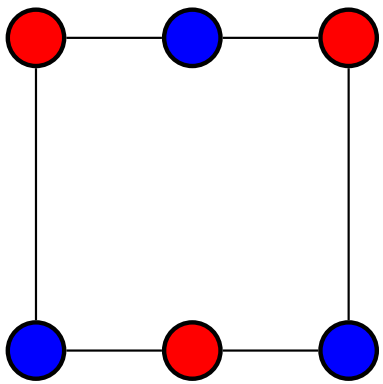


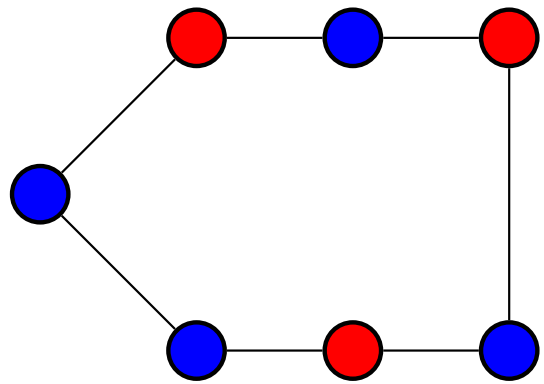


Un algorithme rapide pour la 2-colorabilité

- assigner une couleur au hasard à un sommet au hasard
- assigner l'autre couleur à tous ses voisins
- propager la 2-coloration aux voisins des voisins, etc.

$\rightsquigarrow$ complexité $\mathcal{O}(n+m)$

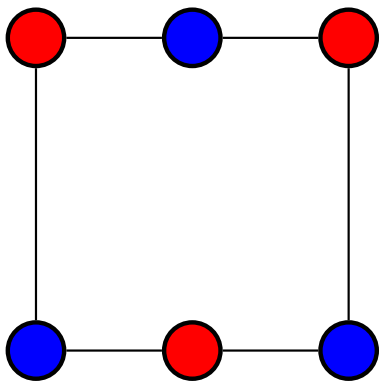


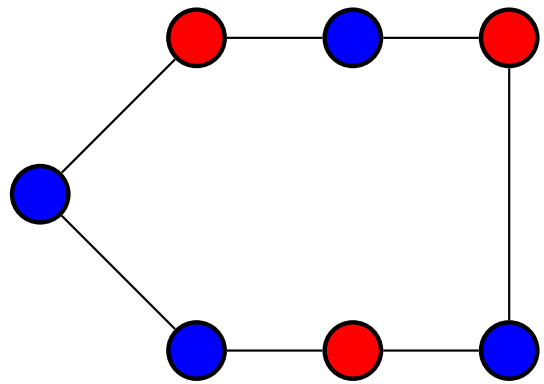


Un algorithme rapide pour la 2-colorabilité

- assigner une couleur au hasard à un sommet au hasard
- assigner l'autre couleur à tous ses voisins
- propager la 2-coloration aux voisins des voisins, etc.

$\rightsquigarrow$ complexité $\mathcal{O}(n+m)$



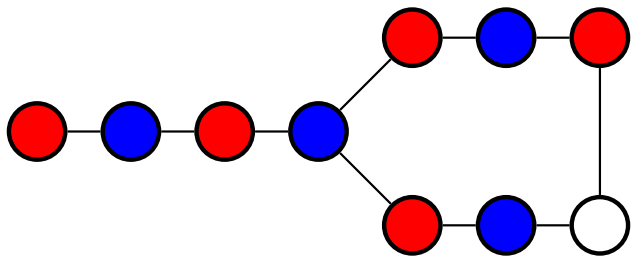


- *conflit* : assigner les deux couleurs à la fois à un sommet
- pas de conflit $\rightsquigarrow$ 2-coloration $\rightsquigarrow$ graphe 2-colorable
- conflit $\rightsquigarrow$ *graphe non 2-colorable* ??

Pourquoi ça marche ?

« conflit $\rightsquigarrow$ *graphe non 2-colorable* »

un autre choix de départ ne donne jamais un 2-coloriage valide ?



- sommet avec conflit $\rightsquigarrow$ deux chaînes « remontent » le coloriage
- les deux chaînes se rencontrent obligatoirement $\rightsquigarrow$ cycle
- couleurs différentes des deux voisins $\rightsquigarrow$ parités différentes des longueurs des deux chaînes
- les deux chaînes + sommet commun + sommet avec conflit $\rightsquigarrow$ cycle de longueur impaire $\rightsquigarrow$ graphe non 2-colorable

Complexité du problème

k-colorabilité

- `Entrée` : un graphe G
- `Question` : le graphe G peut-il être colorié avec k couleurs ?

- $k = 2$
  - algorithme linéaire
- $k \geq 3$
  - algorithme exponentiel (recherche exhaustive)
  - vérification polynomiale (même linéaire)
  - aucun algorithme polynomial connu
  - pas de borne inférieure non triviale connue

Vocabulaire de la logique

- valeurs de vérité 0, 1
- variables propositionnelles
- connecteurs logiques
  - conjonction $\cdot$
  - disjonction $+$
  - négation $\overline{}$
- littéral : une variable ou une négation de variable
  - $\overline{x}$
  - z
- clause : une disjonction de littéraux
  - $\overline{x} + z$
  - $\overline{x} + \overline{y} + z$
- forme normale conjonctive (CNF)
  - conjonction de clauses (= produits de sommes)
  - $(x + \overline{y} + \overline{z}) \cdot (\overline{x} + z)$

Assignation et valeur de vérité

- Assignation des variables :

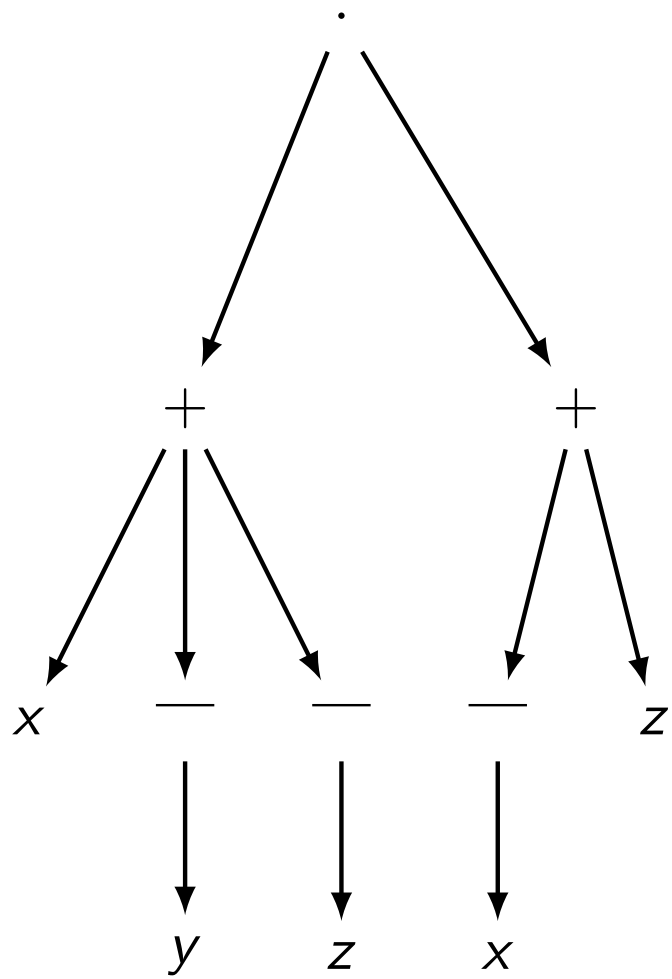
$$\begin{array}{ccc} \{x, y, z\} & \longrightarrow & \{0, 1\} \\ x & \mapsto & 1 \\ y & \mapsto & 1 \\ z & \mapsto & 0 \end{array}$$

- Valeur de vérité de la formule

$$(x + \overline{y} + \overline{z}) \cdot (\overline{x} + z)$$

Calcul des feuilles vers la racine

$$\rightsquigarrow \quad 0$$



Assignation et valeur de vérité

- Assignation des variables :

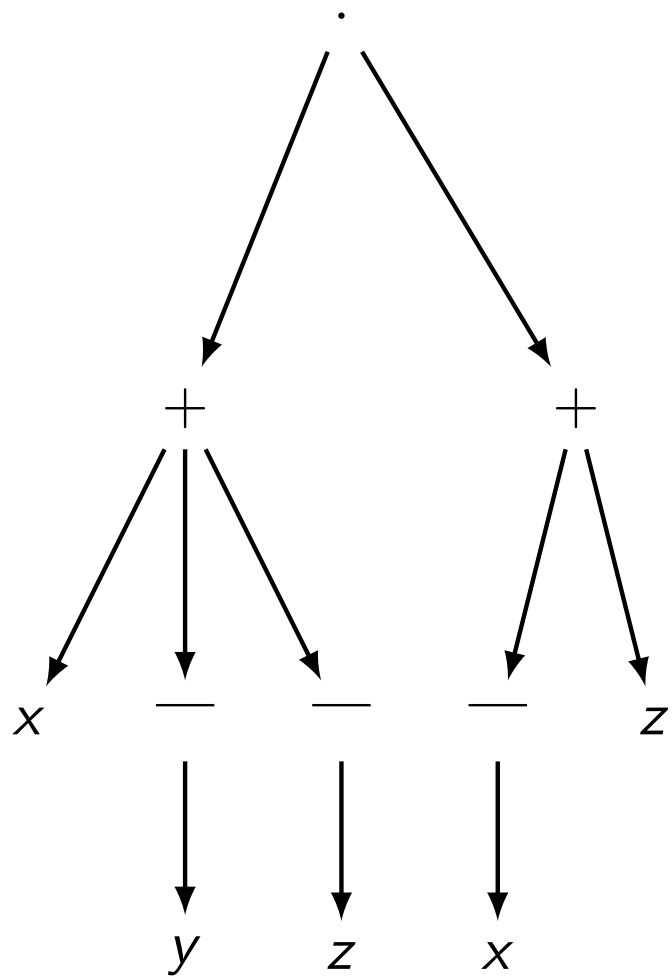
$$\begin{array}{ccc} \{x, y, z\} & \longrightarrow & \{0, 1\} \\ x & \mapsto & 0 \\ y & \mapsto & 0 \\ z & \mapsto & 1 \end{array}$$

- Valeur de vérité de la formule

$$(x + \overline{y} + \overline{z}) \cdot (\overline{x} + z)$$

Calcul des feuilles vers la racine

$$\rightsquigarrow \quad 1$$



Un (autre) problème algorithmique (encore plus) célèbre

SAT

- `Entrée` : une formule propositionnelle F en CNF
- `Question` : F est-elle satisfaisable ?

(F satisfaisable $\rightsquigarrow$ il existe une assignation lui donnant la valeur 1)

Un (autre) problème algorithmique (encore plus) célèbre

SAT

- `Entrée` : une formule propositionnelle F en CNF
- `Question` : F est-elle satisfaisable ?

(F satisfaisable $\rightsquigarrow$ il existe une assignation lui donnant la valeur 1)

- théorie de la complexité
- modélisation
- planification
- model-checking
- intelligence artificielle
- etc.

Un (autre) problème algorithmique (encore plus) célèbre

SAT

- `Entrée` : une formule propositionnelle F en CNF
- `Question` : F est-elle satisfaisable ?

(F satisfaisable $\rightsquigarrow$ il existe une assignation lui donnant la valeur 1)

NE PAS CONFONDRE

vérifier une assignation sur une formule

- `Entrée` : une formule F et une assignation A
- `Question` : F vaut-elle 1 pour A ?

Complexité $\rightsquigarrow$ polynomiale

Un algorithme qui marche toujours, mais...

Recherche exhaustive

- Tester toutes les assignations possibles
- Évaluer la valeur de vérité de F pour chaque assignation
- Décider si oui ou non F est satisfaisable

$$n \text{ variables} \quad \rightsquigarrow \quad 2^n \text{ assignations}$$

Dans la vraie vie

- n est souvent grand
- le temps de calcul devient rapidement impraticable

Complexité de cet algorithme

- exponentielle en n

Une version simple

« 2-clause » $\rightsquigarrow$ 2 littéraux (au plus)

$$a + b \qquad \overline{a} + b \qquad a + \overline{b} \qquad \overline{a} + \overline{b}$$

2-SAT

- `Entrée` : une formule propositionnelle F composée de 2-clauses
- `Question` : F est-elle satisfaisable ?

$$(x + \overline{y}) \cdot (\overline{x} + z) \cdot (\overline{y} + z)$$

$$\begin{array}{rcl} x & \mapsto & 1 \\ y & \mapsto & 1 \\ z & \mapsto & 1 \end{array}$$

algorithmes polynomiaux

Complexité du problème

SAT

- `Entrée` : un graphe G
- `Question` : le graphe G peut-il être colorié avec k couleurs ?

- **2-SAT**
  - algorithme polynomial
  - autre cas facile : **Horn-SAT**
- **SAT**
  - algorithmes exponentiels dans le pire des cas (résolution, DPLL, etc.)
  - vérification polynomiale
  - aucun algorithme polynomial connu
  - pas de borne inférieure non triviale connue

Les SAT-solveurs

- Outils informatiques qui donnent une réponse au problème SAT (pour des instances pas trop grandes...)
  - Lingeling
  - Glucose
  - Chaff
  - etc.
- Librairie pysat de python
  - picosat
  - minisat
  - z3
  - etc.
- Compétition annuelle des meilleur SAT-solveurs
  - http ://www.satcompetition.org/

Plan du cours

Problème du tri

Problèmes de décision

Comparaison de problèmes algorithmiques

Un problème plus facile qu'un autre ?

Min « plus facile » que **Tri**

- je sais trier un tableau $\rightsquigarrow$ je sais trouver le minimum de ce tableau
- un algorithme pour **Tri** permet de fabriquer un algorithme pour **Min**

k-**colorabilité** vs. **SAT** ?

- Sauf dans des cas particuliers, on ne connaît pas bien la complexité de l'un et de l'autre : peut-être polynomiale, et peut-être pas
- Un algorithme polynomial pour l'un permettrait-il de fabriquer un algorithme polynomial pour l'autre ?

Réduction (polynomiale)

$\mathcal{P}_1$

- `Entrée` : E_1
- `Question` : E_1 possède P_1 ?

$\mathcal{P}_2$

- `Entrée` : E_2
- `Question` : E_2 possède P_2 ?

Transformer E_1 en E_2 (en temps polynomial)
de telle sorte que
E_1 possède $P_1 \iff E_2$ possède P_2

$\mathcal{P}_1$ se réduit à $\mathcal{P}_2$ (en temps polynomial)

- Algo pour $\mathcal{P}_2$: A_2 (polynomial)
- Algo pour $\mathcal{P}_1$: $E_1 \rightsquigarrow E_2 \overset{A_2}{\rightsquigarrow}$ oui ou non (en temps polynomial)

$\mathcal{P}_1$ est « plus facile » que $\mathcal{P}_2$

k-**colorabilité** se réduit à **SAT**

- Entrée de k-**colorabilité** : graphe G à n sommets et m arêtes
- Entrée de **SAT** : formule F en CNF

Réduction polynomiale de k-**colorabilité** à **SAT**

- Écrire F en temps polynomial en $n + m$ de telle sorte que

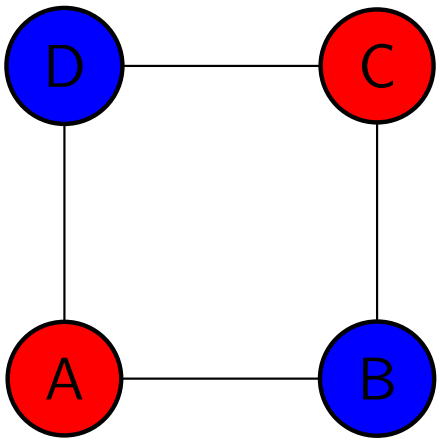
$$G \text{ est } k\text{-colorable} \iff F \text{ est satisfaisable}$$

Si jamais on inventait un algorithme en temps polynomial pour **SAT**, on aurait aussi un algorithme en temps polynomial pour k-**colorabilité**

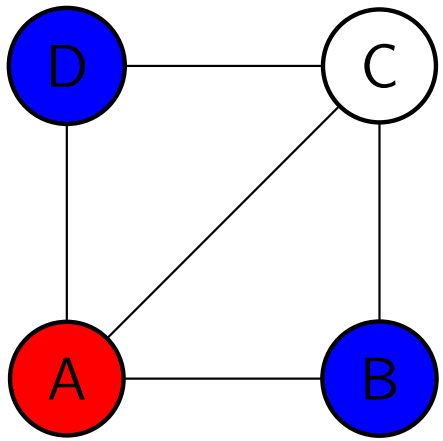
Un petit exemple

2-**colorabilité**

- `Entrée` : un graphe G
- `Question` : le graphe G peut-il être colorié avec 2 couleurs ?



OUI

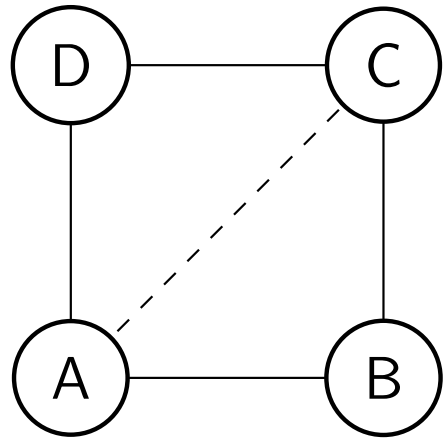


NON

Les variables du problème

Les couleurs des sommets

	Rouge	Bleu
A	$X1$	$X2$
B	$X3$	$X4$
C	$X5$	$X6$
D	$X7$	$X8$

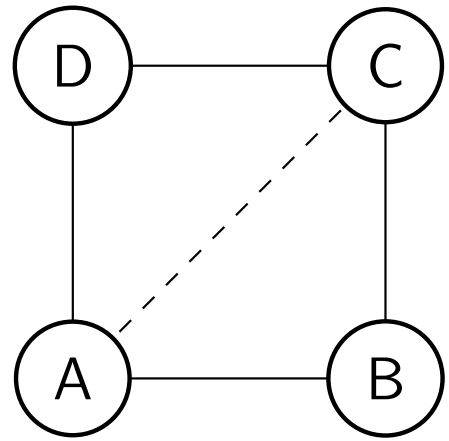


- $X1 = 1$: le sommet A est rouge
- $X1 = 0$: le sommet A n'est pas rouge
- $X2 = 1$: le sommet A est bleu
- $X2 = 0$: le sommet A n'est pas bleu
- etc.

Les contraintes portant sur les sommets

Chaque sommet est d'une couleur, mais pas des deux

	Rouge	Bleu
A	$X1$	$X2$
B	$X3$	$X4$
C	$X5$	$X6$
D	$X7$	$X8$



Pour A :

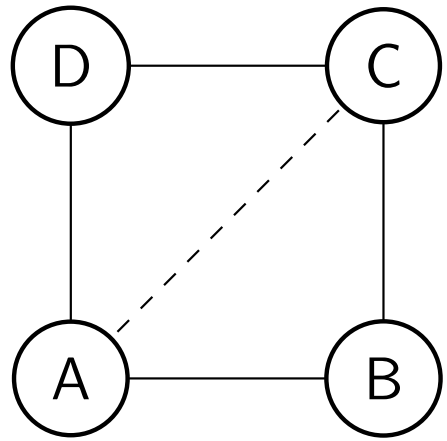
- $X1 + X2$
- $\overline{X1\,X2} = \overline{X1} + \overline{X2}$

Etc.

Les contraintes portant sur les arêtes

Deux sommets reliés ne peuvent pas être de la même couleur

	Rouge	Bleu
A	$X1$	$X2$
B	$X3$	$X4$
C	$X5$	$X6$
D	$X7$	$X8$



Pour AB :

- $\overline{X1\,X3} = \overline{X1} + \overline{X3}$
- $\overline{X2\,X4} = \overline{X1} + \overline{X4}$

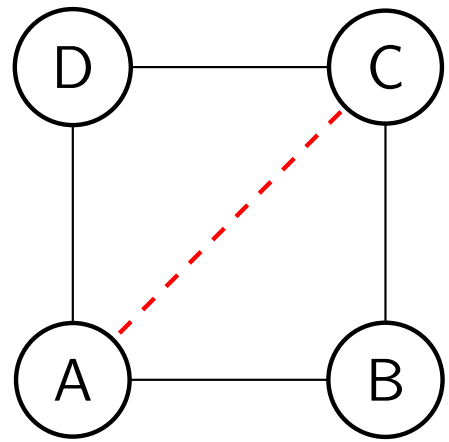
Etc.

La modélisation complète du problème

$$
\begin{array}{ll}
(X1 + X2)\,(\overline{X1} + \overline{X2}) & (\text{sommet } A) \\
(X3 + X4)\,(\overline{X3} + \overline{X4}) & (\text{sommet } B) \\
(X5 + X6)\,(\overline{X5} + \overline{X6}) & (\text{sommet } C) \\
(X7 + X8)\,(\overline{X7} + \overline{X8}) & (\text{sommet } D) \\
(\overline{X1} + \overline{X3})\,(\overline{X2} + \overline{X4}) & (\text{arête } AB) \\
(\overline{X1} + \overline{X7})\,(\overline{X2} + \overline{X8}) & (\text{arête } AD) \\
(\overline{X7} + \overline{X5})\,(\overline{X8} + \overline{X6}) & (\text{arête } DC) \\
(\overline{X3} + \overline{X5})\,(\overline{X4} + \overline{X6}) & (\text{arête } BC) \\
(\overline{X1} + \overline{X5})\,(\overline{X2} + \overline{X6}) & (\text{arête } AC)
\end{array}
$$

$\rightsquigarrow$ une formule en **CNF**

$\rightsquigarrow$ est-elle **satisfaisable** ?

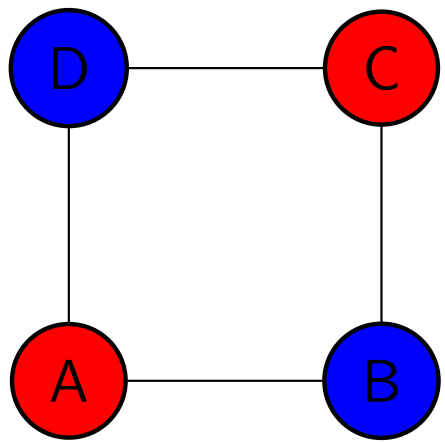


Formule satisfaisable = graphe 2-colorable

$$
\begin{array}{ll}
(X1 + X2)\,(\overline{X1} + \overline{X2}) & (\text{sommet } A) \\
(X3 + X4)\,(\overline{X3} + \overline{X4}) & (\text{sommet } B) \\
(X5 + X6)\,(\overline{X5} + \overline{X6}) & (\text{sommet } C) \\
(X7 + X8)\,(\overline{X7} + \overline{X8}) & (\text{sommet } D) \\
(\overline{X1} + \overline{X3})\,(\overline{X2} + \overline{X4}) & (\text{arête } AB) \\
(\overline{X1} + \overline{X7})\,(\overline{X2} + \overline{X8}) & (\text{arête } AD) \\
(\overline{X7} + \overline{X5})\,(\overline{X8} + \overline{X6}) & (\text{arête } DC) \\
(\overline{X3} + \overline{X5})\,(\overline{X4} + \overline{X6}) & (\text{arête } BC)
\end{array}
$$

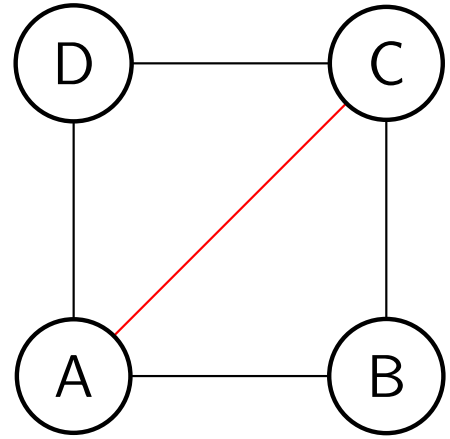
$$
\begin{array}{lcl}
X1 & \mapsto & 1 \\
X2 & \mapsto & 0 \\
X3 & \mapsto & 0 \\
X4 & \mapsto & 1 \\
X5 & \mapsto & 1 \\
X6 & \mapsto & 0 \\
X7 & \mapsto & 0 \\
X8 & \mapsto & 1
\end{array}
$$

	Rouge	Bleu
A	$X1 = 1$	$X2 = 0$
B	$X3 = 0$	$X4 = 1$
C	$X5 = 1$	$X6 = 0$
D	$X7 = 0$	$X8 = 1$



Formule non satisfaisable = graphe non 2-colorable

$(X1 + X2)\,(\overline{X1} + \overline{X2})$ (sommet A)
$(X3 + X4)\,(\overline{X3} + \overline{X4})$ (sommet B)
$(X5 + X6)\,(\overline{X5} + \overline{X6})$ (sommet C)
$(X7 + X8)\,(\overline{X7} + \overline{X8})$ (sommet D)
$(\overline{X1} + \overline{X3})\,(\overline{X2} + \overline{X4})$ (arête AB)
$(\overline{X1} + \overline{X7})\,(\overline{X2} + \overline{X8})$ (arête AD)
$(\overline{X7} + \overline{X5})\,(\overline{X8} + \overline{X6})$ (arête DC)
$(\overline{X3} + \overline{X5})\,(\overline{X4} + \overline{X6})$ (arête BC)
$(\overline{X1} + \overline{X5})\,(\overline{X2} + \overline{X6})$ (arête AC)

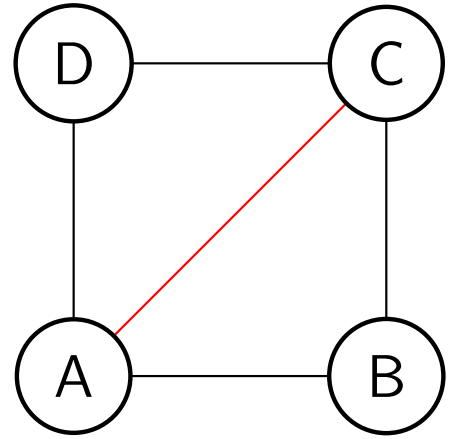


$(\overline{X1} + \overline{X3})(\overline{X3} + \overline{X5})(\overline{X1} + \overline{X5})$

$X1$	$X3$	$X5$	
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Formule non satisfaisable = graphe non 2-colorable

$(X1 + X2)\,(\overline{X1} + \overline{X2})$ (sommet A)
$(X3 + X4)\,(\overline{X3} + \overline{X4})$ (sommet B)
$(X5 + X6)\,(\overline{X5} + \overline{X6})$ (sommet C)
$(X7 + X8)\,(\overline{X7} + \overline{X8})$ (sommet D)
$(\overline{X1} + \overline{X3})\,(\overline{X2} + \overline{X4})$ (arête AB)
$(\overline{X1} + \overline{X7})\,(\overline{X2} + \overline{X8})$ (arête AD)
$(\overline{X7} + \overline{X5})\,(\overline{X8} + \overline{X6})$ (arête DC)
$(\overline{X3} + \overline{X5})\,(\overline{X4} + \overline{X6})$ (arête BC)
$(\overline{X1} + \overline{X5})\,(\overline{X2} + \overline{X6})$ (arête AC)

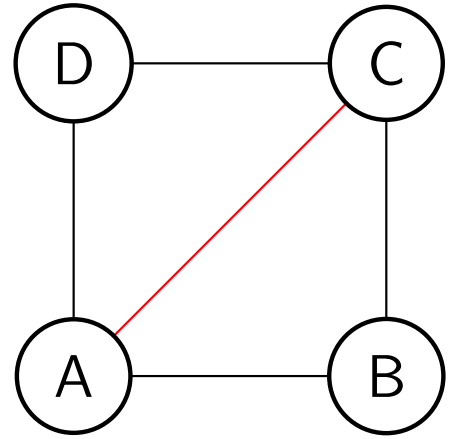


$(\overline{X2} + \overline{X4})(\overline{X4} + \overline{X6})(\overline{X2} + \overline{X6})$

$X2$	$X4$	$X6$	
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Formule non satisfaisable = graphe non 2-colorable

$(X1 + X2)\,(\overline{X1} + \overline{X2})$ (sommet A)
$(X3 + X4)\,(\overline{X3} + \overline{X4})$ (sommet B)
$(X5 + X6)\,(\overline{X5} + \overline{X6})$ (sommet C)
$(X7 + X8)\,(\overline{X7} + \overline{X8})$ (sommet D)
$(\overline{X1} + \overline{X3})\,(\overline{X2} + \overline{X4})$ (arête AB)
$(\overline{X1} + \overline{X7})\,(\overline{X2} + \overline{X8})$ (arête AD)
$(\overline{X7} + \overline{X5})\,(\overline{X8} + \overline{X6})$ (arête DC)
$(\overline{X3} + \overline{X5})\,(\overline{X4} + \overline{X6})$ (arête BC)
$(\overline{X1} + \overline{X5})\,(\overline{X2} + \overline{X6})$ (arête AC)



$X1$	$X3$	$X5$
0	0	0
0	0	1
0	1	0
1	0	0

$X2$	$X4$	$X6$
0	0	0
0	0	1
0	1	0
1	0	0

$X1$	$X2$
0	1
1	0

$X3$	$X4$
0	1
1	0

$X5$	$X6$
0	1
1	0

k-colorabilité se réduit à SAT

n sommets et m arêtes

- kn variables $X_{1,1}, \ldots, X_{1,k}, \ldots, X_{n,1}, \ldots, X_{n,k}$
  - $X_{i,j} = 1$
    « le sommet i est de la couleur j »
- contraintes sur les sommets : n clauses de k littéraux et $n\frac{k(k-1)}{2}$ clauses de 2 littéraux
  - $X_{i,1} + \ldots + X_{i,k}$
    « le sommet i est d'une des k couleurs »
  - $\overline{X_{i,j_1} X_{1,j_2}} = \overline{X_{1,j_1}} + \overline{X_{1,j_2}}$ pour $j_1 < j_2$
    « le sommet i n'est pas de deux couleurs à la fois »
- contraintes sur les arêtes : km clauses de 2 littéraux
  - $(\overline{X_{i_1,1}} + \overline{X_{i_2,1}}) \ldots (\overline{X_{i_1,k}} + \overline{X_{i_2,k}})$ pour chaque arête i_1, i_2
    « deux sommets reliés ne sont pas de la même couleur »

$\rightsquigarrow$ formule en CNF construite en temps polynomial en $n + m$

FIN