Le problème **P** = **NP**

Guide à l'usage du promeneur égaré dans le paradis (ou l'enfer) de la complexité des algorithmes

Richard Lassaigne
avec la collaboration d'Arnaud Durand

Equipe de Logique mathématique
CNRS/Université Paris Diderot

21/04/2014

Problèmes du millénium

Depuis 2000, le Clay Mathematics Institute propose un prix (1 Million dollars) pour la résolution de chacun des 7 problèmes mathématiques suivants :

- La conjecture de Birch et Swinnerton-Dyer
- La conjecture de Hodge
- Les équations de Navier-Stokes
- Le problème **P = NP**
- La conjecture de Poincaré (résolue par G. Perelman, 2003)
- L'hypothèse de Riemann
- La théorie de Yang-Mills

Problème **P** = **NP**

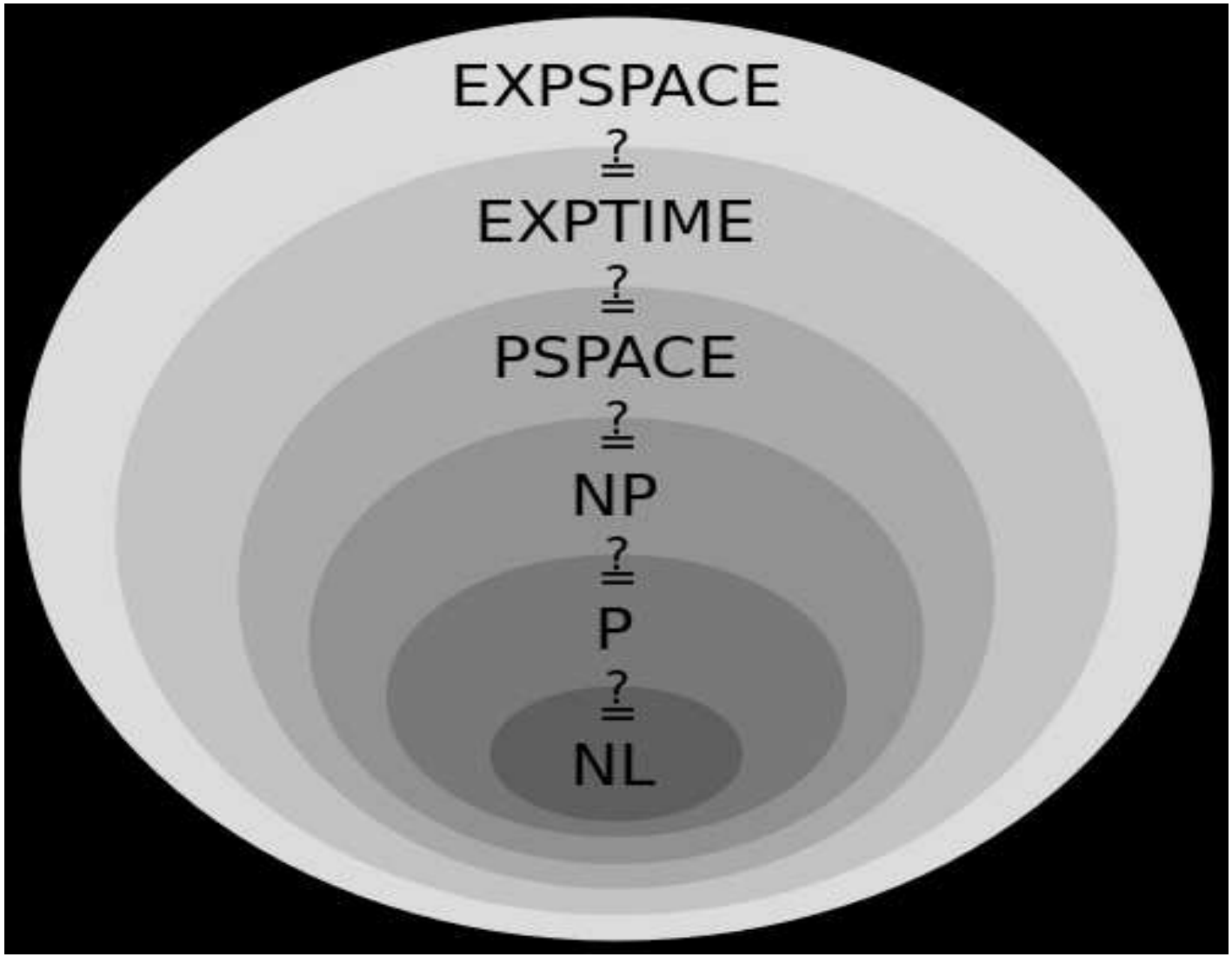
Lance Fortnow (Communications of the ACM, sept. 2009) :

"The **P** vs **NP** problem has gone from an interesting problem related to logic to perhaps the most fundamental and important mathematical question of our time, whose importance only grows as computers become more powerful and widespread."

Le problème **P** = **NP** concerne la classe des problèmes "faciles", c'est-à-dire pour lesquels il existe des algorithmes de résolution "efficaces".

En 40 ans, les théoriciens de la complexité ont imaginé plus de 400 classes pour mesurer la complexité des problèmes.

Quelques classes de complexité



Motivation pour le problème **P** = **NP**

- Comment modéliser la notion de problème "facile" ?
- Quelles sont les bonnes mesures de la complexité d'un problème ?
- Existe-t-il des problèmes intrinsèquement "difficilles" ?
- En quoi l'existence de tels problèmes pourrait-elle être un atout ?
- Existe-t-il des méthodes pour distinguer les différents problèmes réputés difficilles ?

P vs NP

- Question de mathématique et d'informatique.
- Question ouverte depuis environ 40 ans mais aux origines bien plus lointaines...

Questions (plus anciennes)

- Qu'est-ce qu'une fonction mathématique calculable ?
- Qu'est-ce qu'une propriété ou un problème décidable ?
- Qu'est-ce qu'un algorithme ?

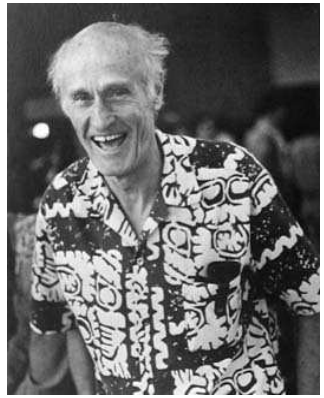
Modèles de calcul et fonction calculable

Début des années 1930 :

- formalisation de la notion de fonction calculable et d'algorithme. Approches équivalentes (machines de Turing, fonctions récursives, lambda calcul) et actuelles...

Alan Turing

Alonzo Church



Stephen C. Kleene

- mise en évidence des limites : il existe des fonctions non calculables (et des problèmes indécidables) !

Problèmes indécidables

Entrée : un programme P, une entrée x

Question : l'exécution de P sur l'entrée x s'arrête-elle ?

Indécidabilité et Incomplétude

Toute axiomatisation cohérente et "convenable" de l'arithmétique est indécidable et incomplète.
K. Godel (1931) Réponse négative au 2e problème du programme de D. Hilbert (1904)

Algorithme et complexité

Petit à petit, le débat se déplace. La question :
"Qu'est-ce qu'une fonction calculable ?"
devient :
"Si une fonction est calculable, peut-on mesurer le "coût" (la complexité) nécessaire pour la calculer ?"

Coût / complexité :

Nombre d'étapes, d'opérations élémentaires, de l'algorithme ("temps de calcul abstrait").
Rabin, Cobham, Blum, Edmonds, Hartmanis, Stearns (milieu des années 60)

Multiplication : $m \times n$ "petites" multiplications et additions pour des entiers de tailles m et n.

Algorithme et complexité

Mesure de complexité : nombre d'opérations "élémentaires" nécessaires pour réaliser un calcul

Entrée : deux matrices A et B carrées d'ordre n

Sortie : matrice produit $C = A \times B$

Nombre d'opérations "élémentaires" :

- pour chaque élément de la matrice produit, n multiplications et $n-1$ additions,
- au total, $n^2 \times (2n-1) = O(n^3)$ opérations "élémentaires"

P vs *NP*

Le problème **P** vs **NP** porte sur deux classes de problèmes :

- **P** : les problèmes “faciles” à décider, c’est à dire de complexité raisonnable.
- **NP** : les problèmes “faciles” à vérifier.

Tout cela reste à définir...

Problèmes de calcul et problèmes de décision

Entrée : un entier n

Question : n est-il premier ?

Entrée : un graphe G et 2 sommets s, t

Question : existe-t-il dans G un chemin de s à t ?

Entrée : un entier $n > 2$

Question : existe-t-il des entiers non nuls x, y et z tels que $x^n + y^n = z^n$?

Problèmes de calcul et problèmes de décision

Entrée : une liste de mots $l_1, l_2, \ldots, l_n$

Sortie : la liste triée par ordre croissant

Entrée : une relation binaire R

Sortie : la clôture transitive de la relation R

Entrée : un graphe valué G et 2 sommets s, t

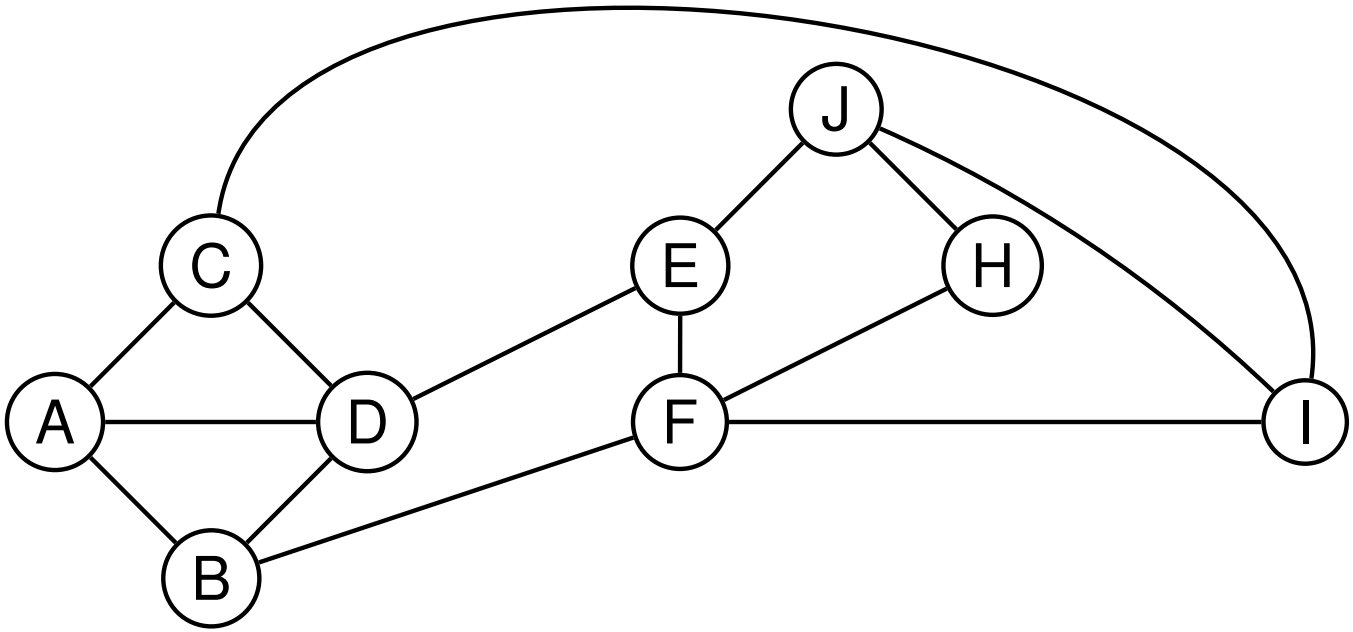
Sortie : un plus court chemin de s à t

Un exemple : le circuit hamiltonien

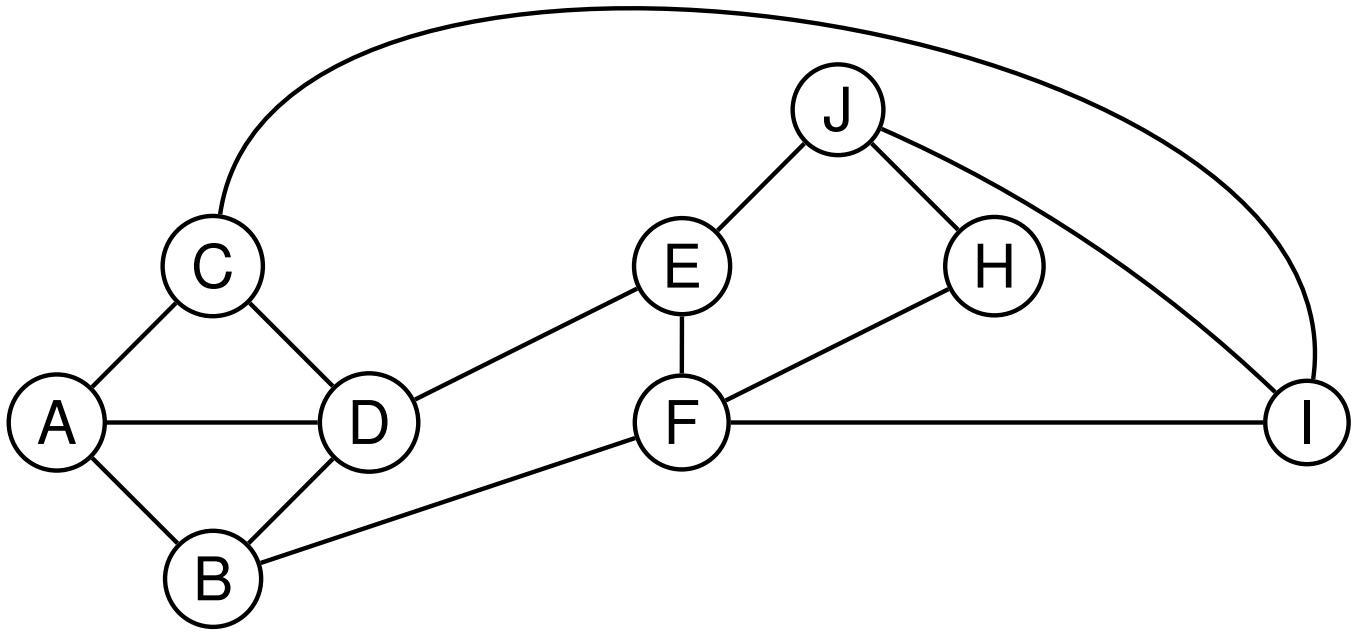
Circuit hamiltonien

Étant donné un graphe $G = (V, E)$ et un sommet $v \in V$, peut-on trouver un circuit empruntant des arêtes du graphe, commençant et finissant en v et passant une seule fois par tout autre sommet de G ?

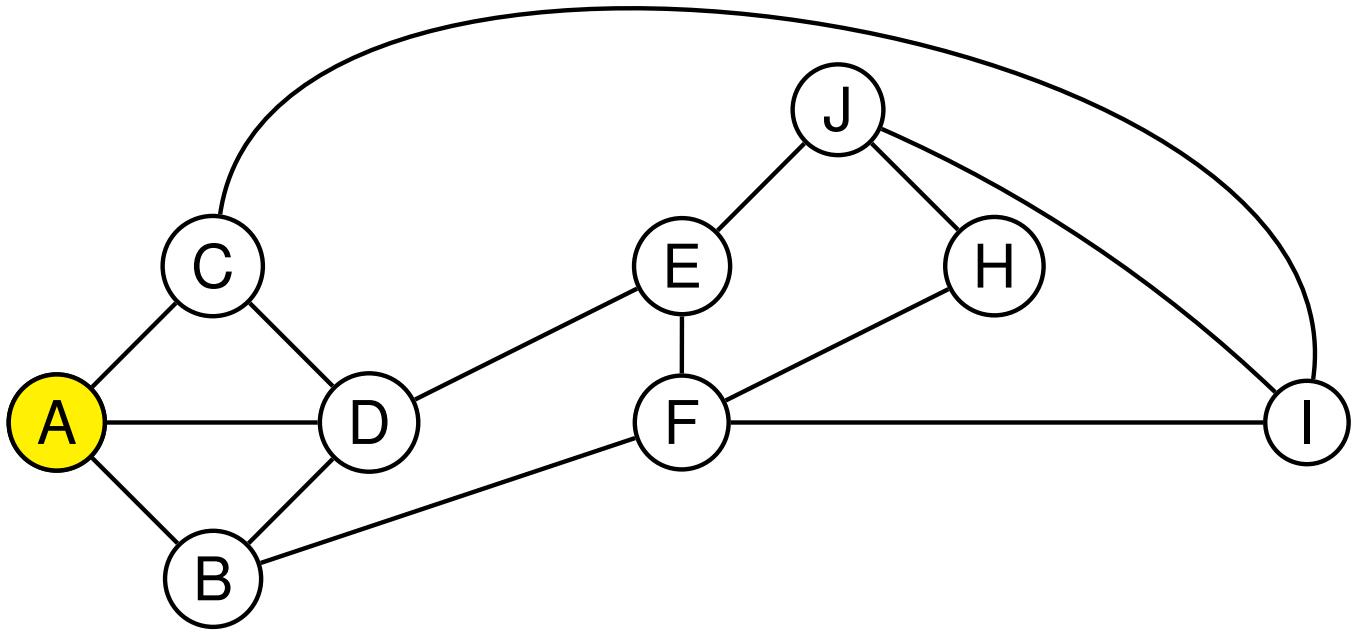
Aussi : voyageur de commerce (graphe valué).



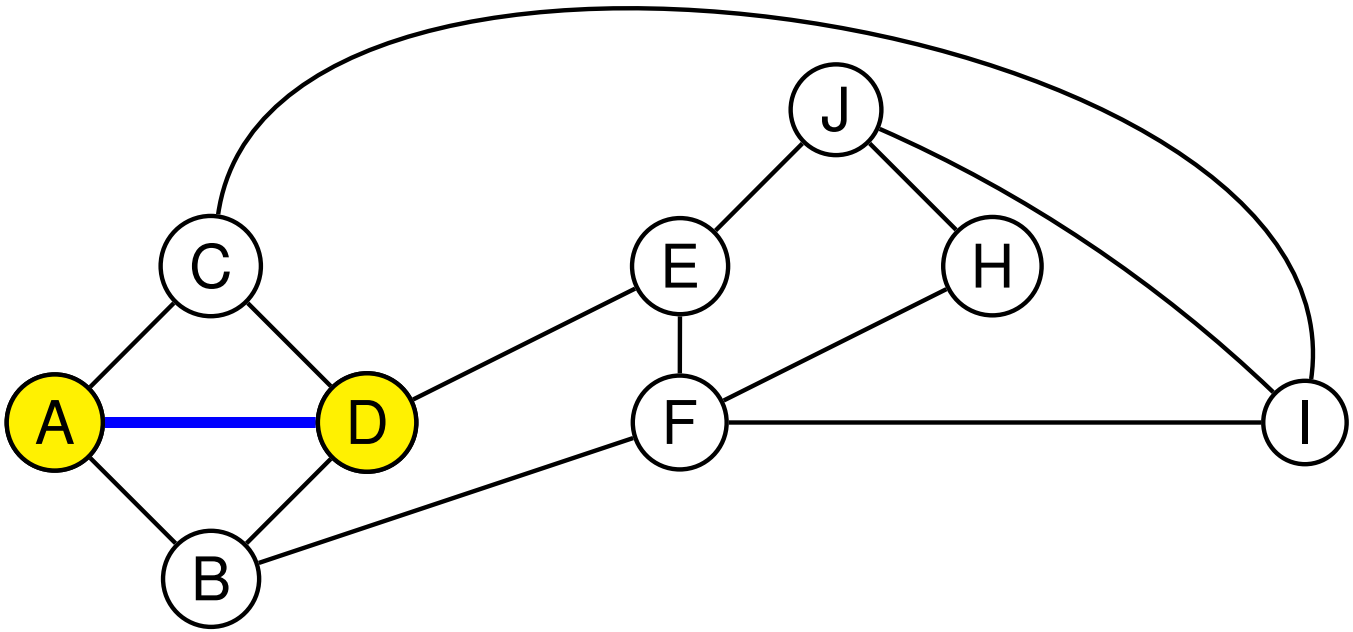
Un exemple : le circuit hamiltonien



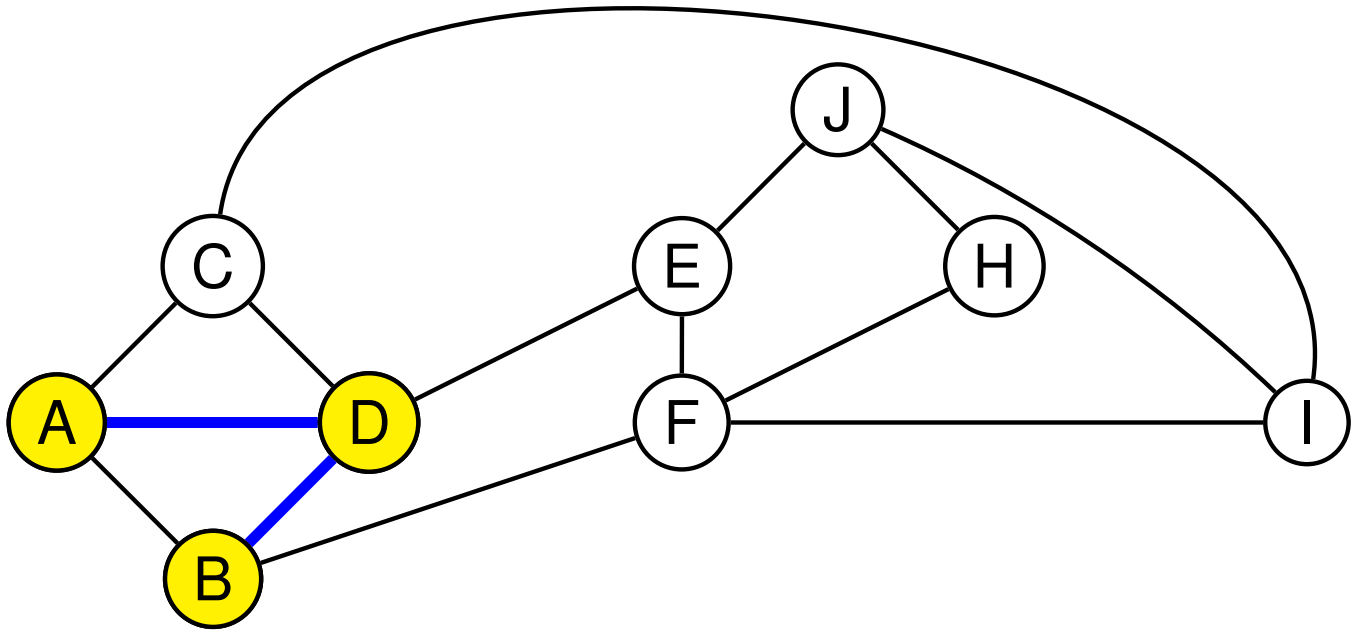
Un exemple : le circuit hamiltonien



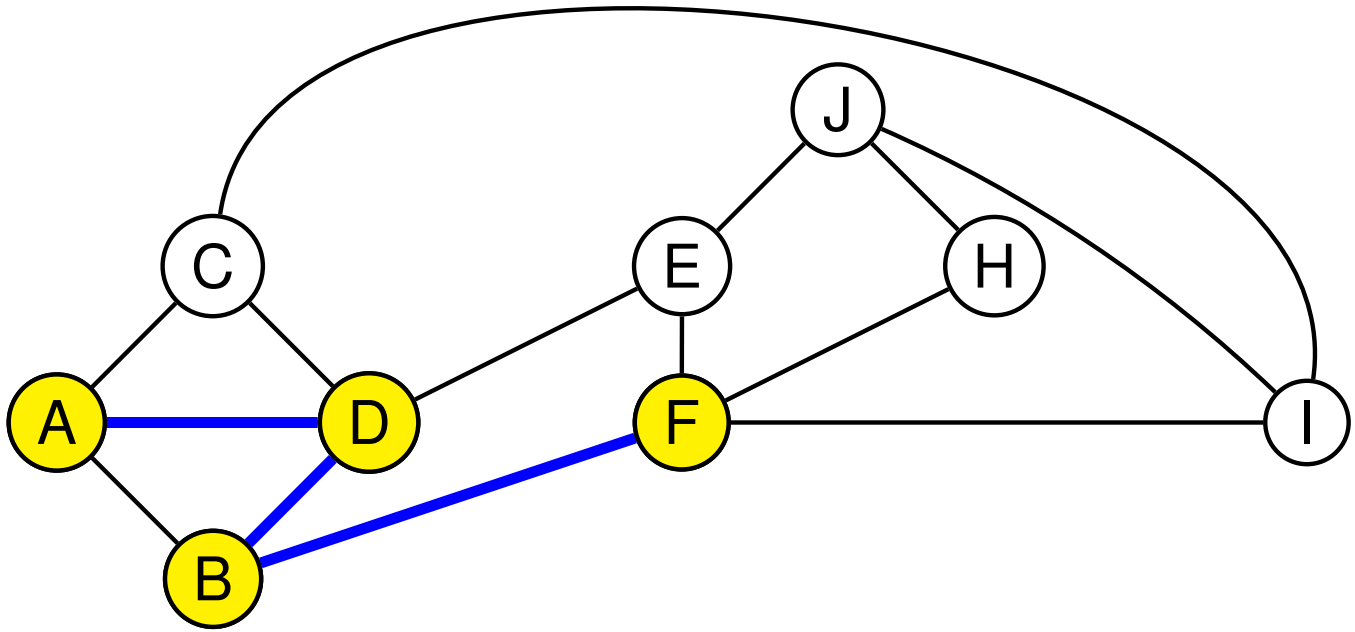
Un exemple : le circuit hamiltonien



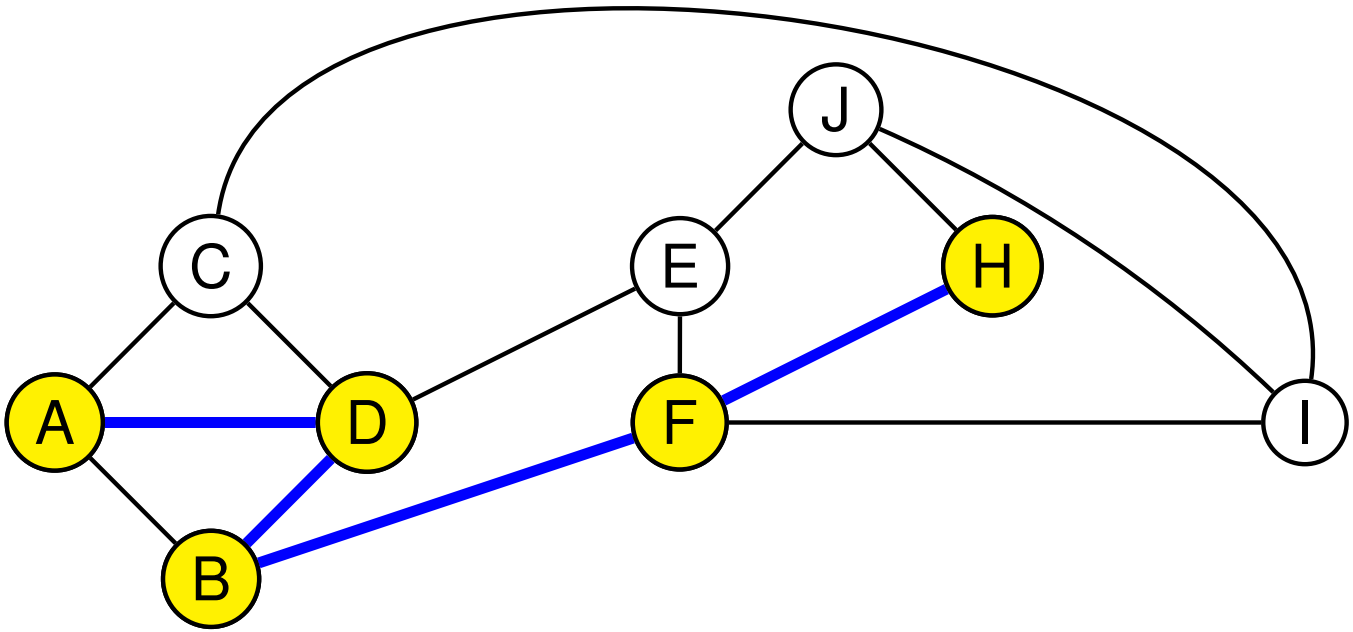
Un exemple : le circuit hamiltonien



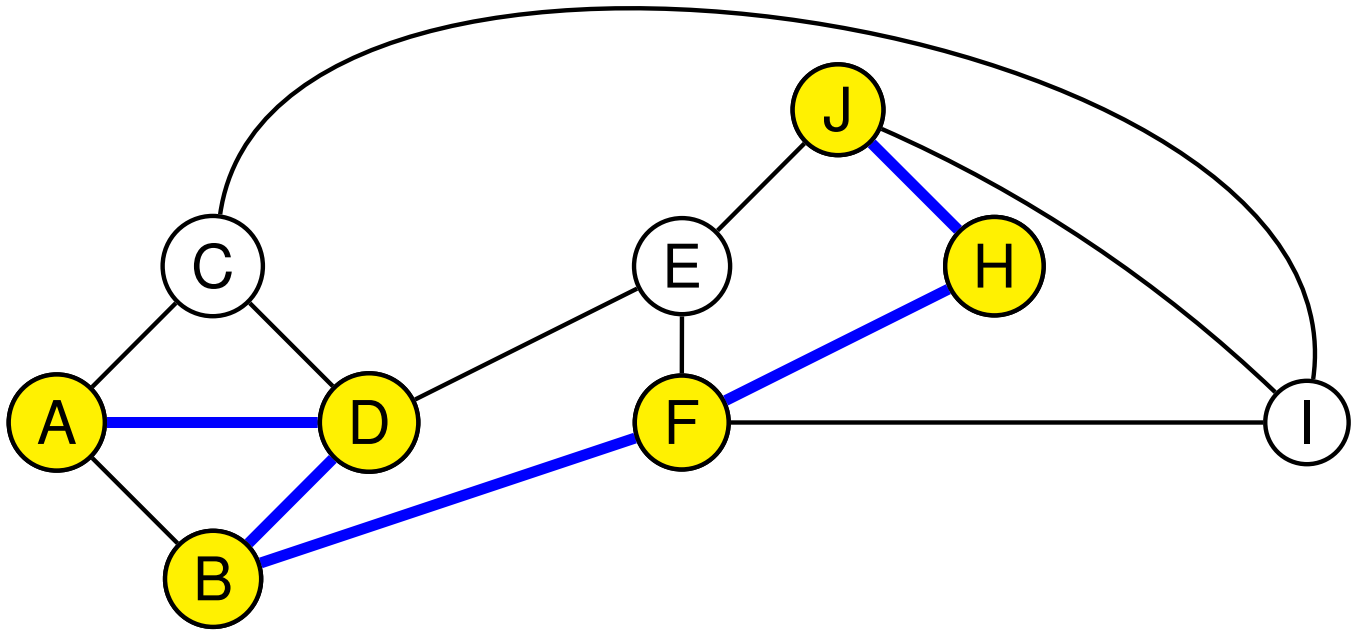
Un exemple : le circuit hamiltonien



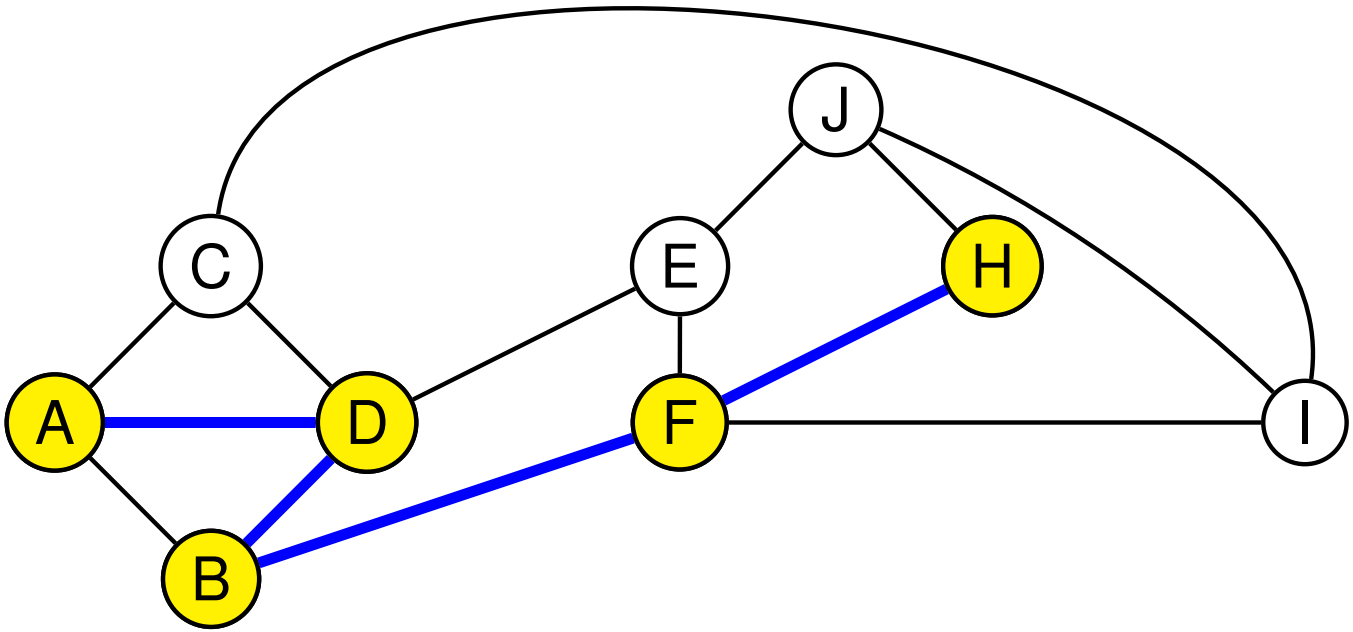
Un exemple : le circuit hamiltonien



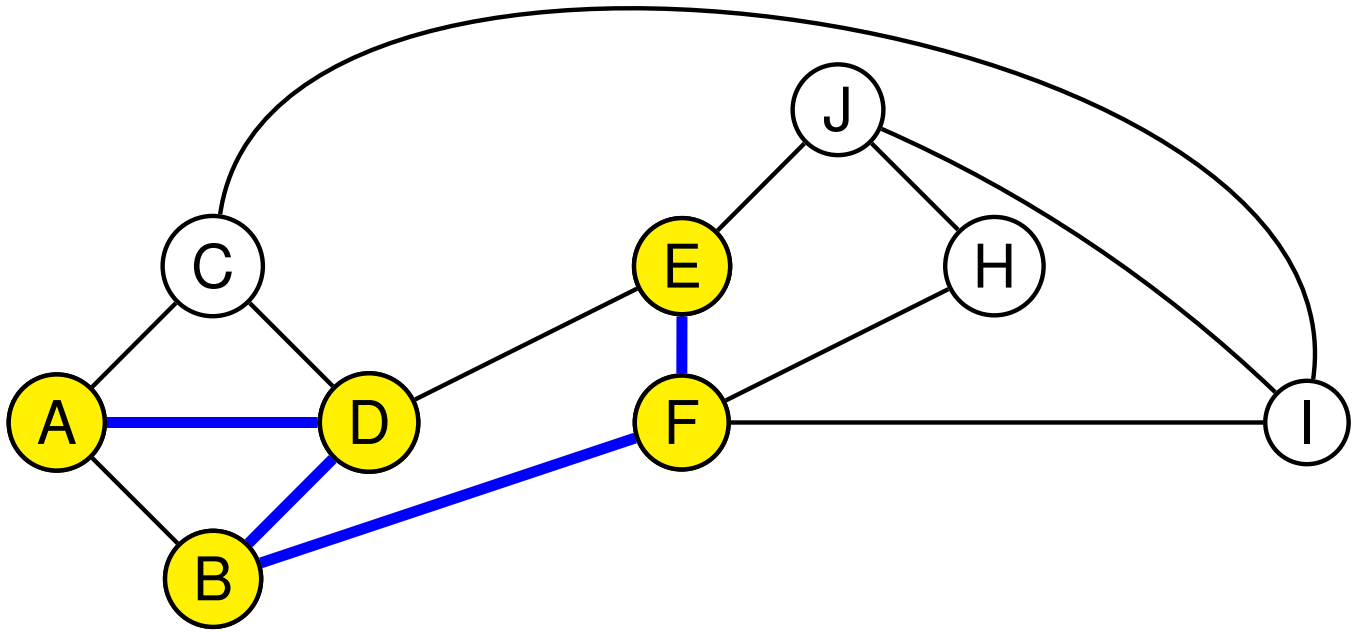
Un exemple : le circuit hamiltonien



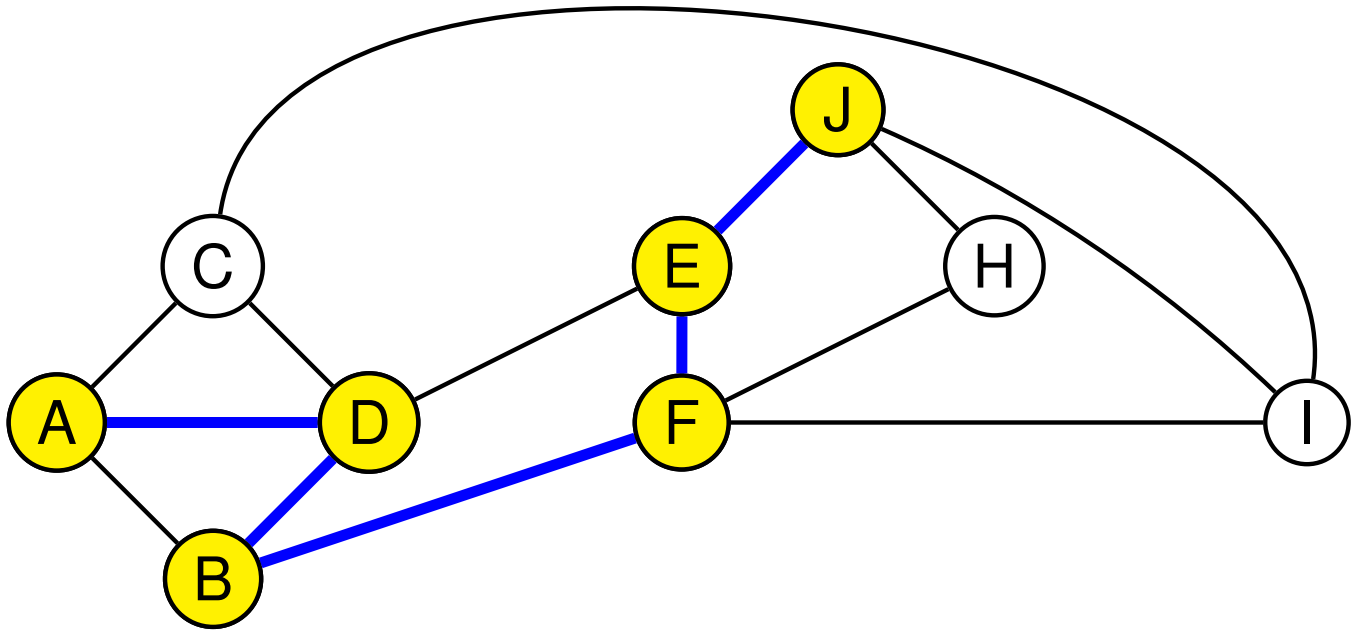
Un exemple : le circuit hamiltonien



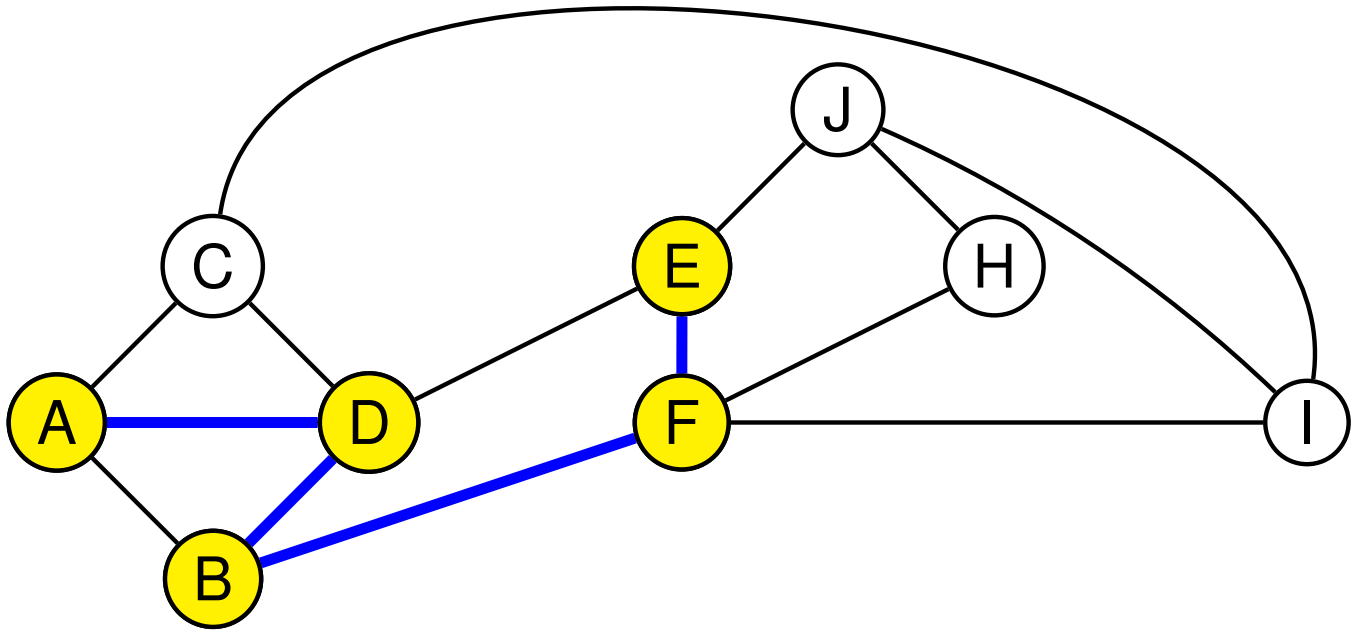
Un exemple : le circuit hamiltonien



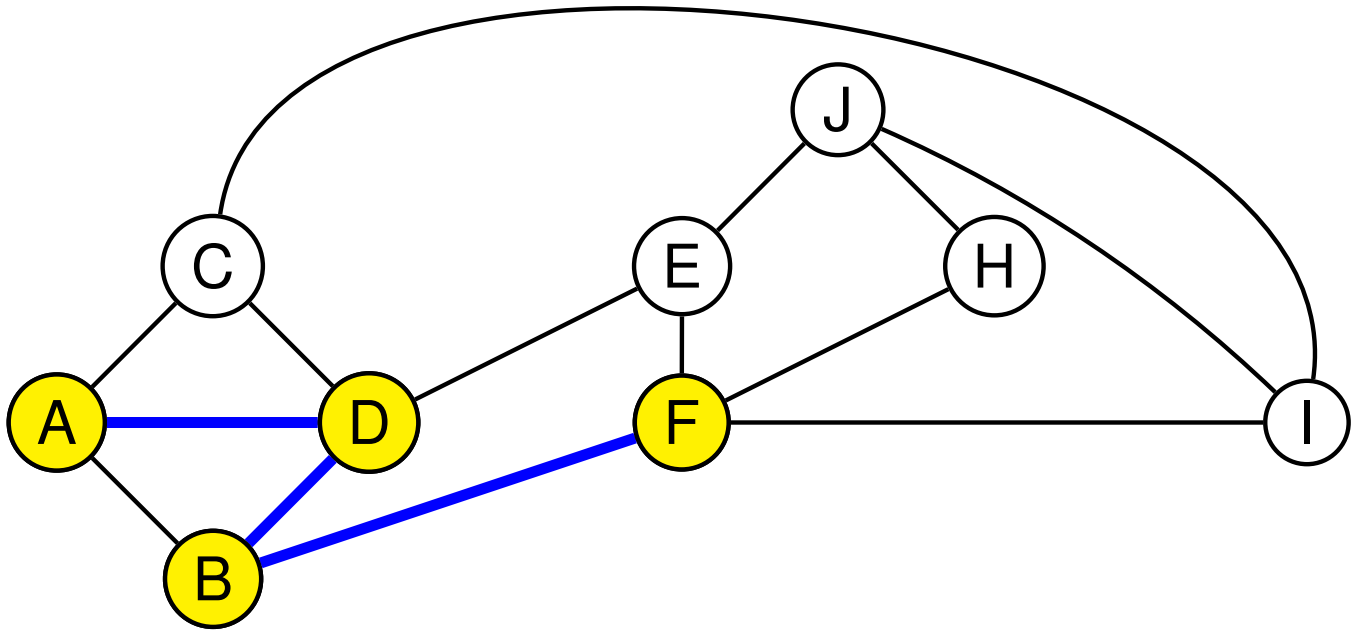
Un exemple : le circuit hamiltonien



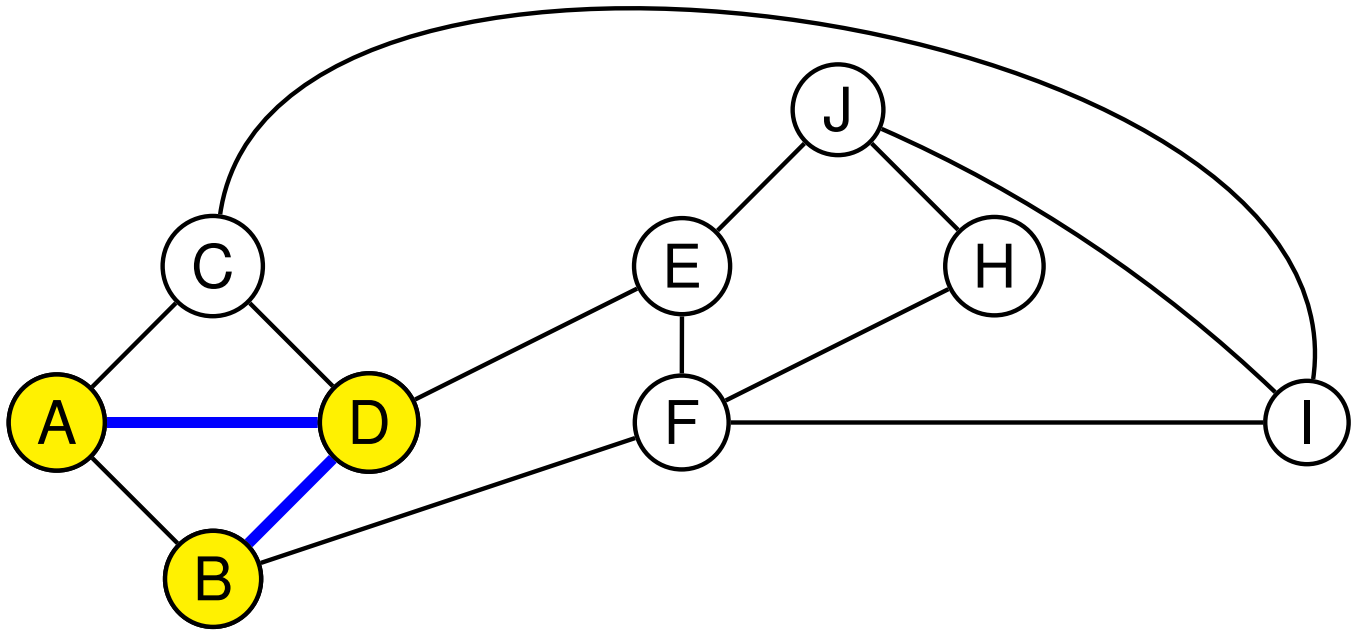
Un exemple : le circuit hamiltonien



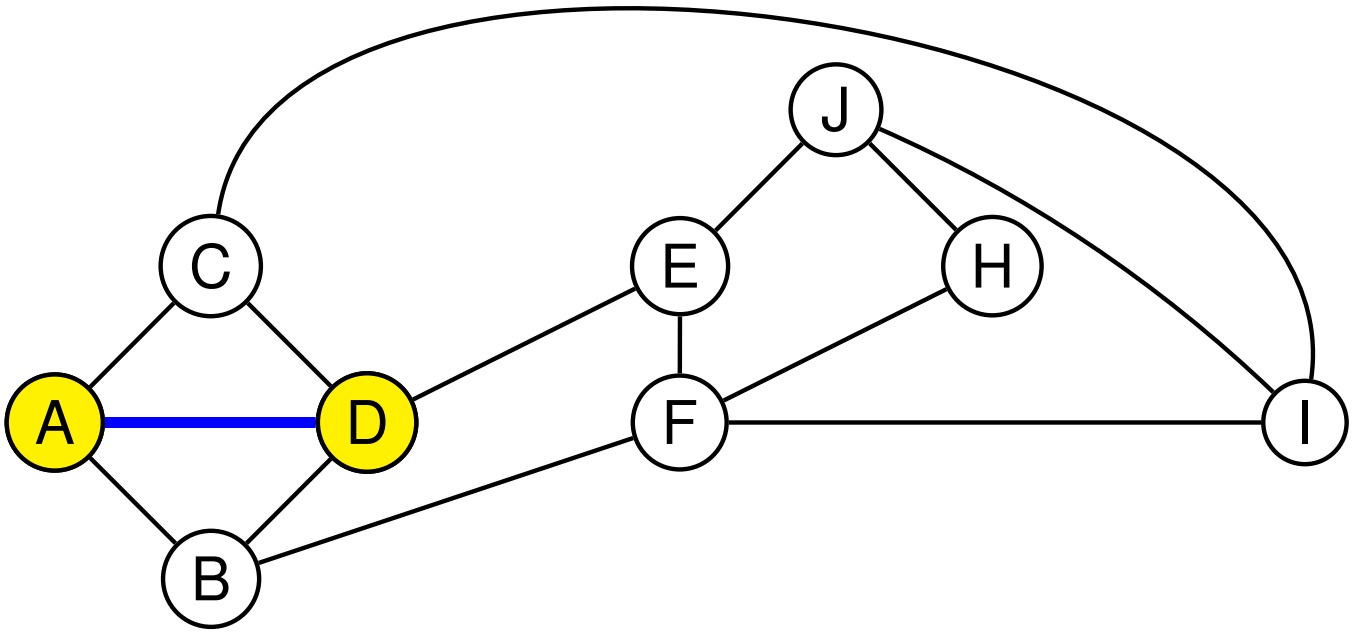
Un exemple : le circuit hamiltonien



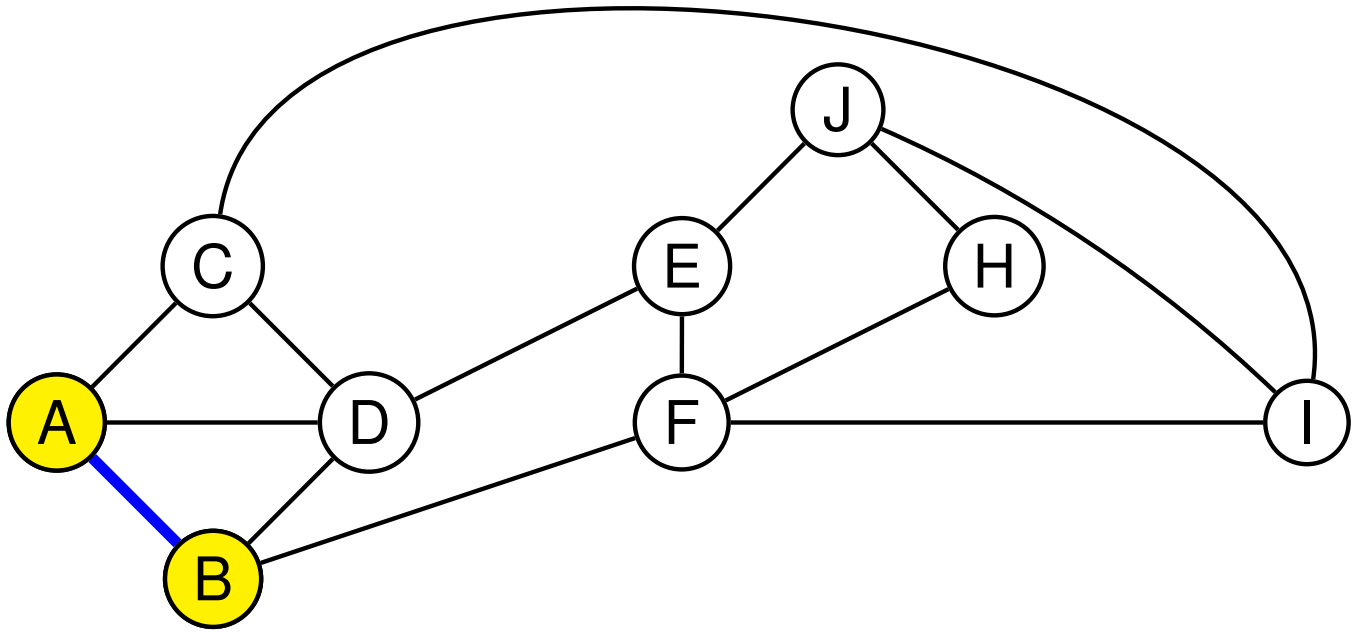
Un exemple : le circuit hamiltonien



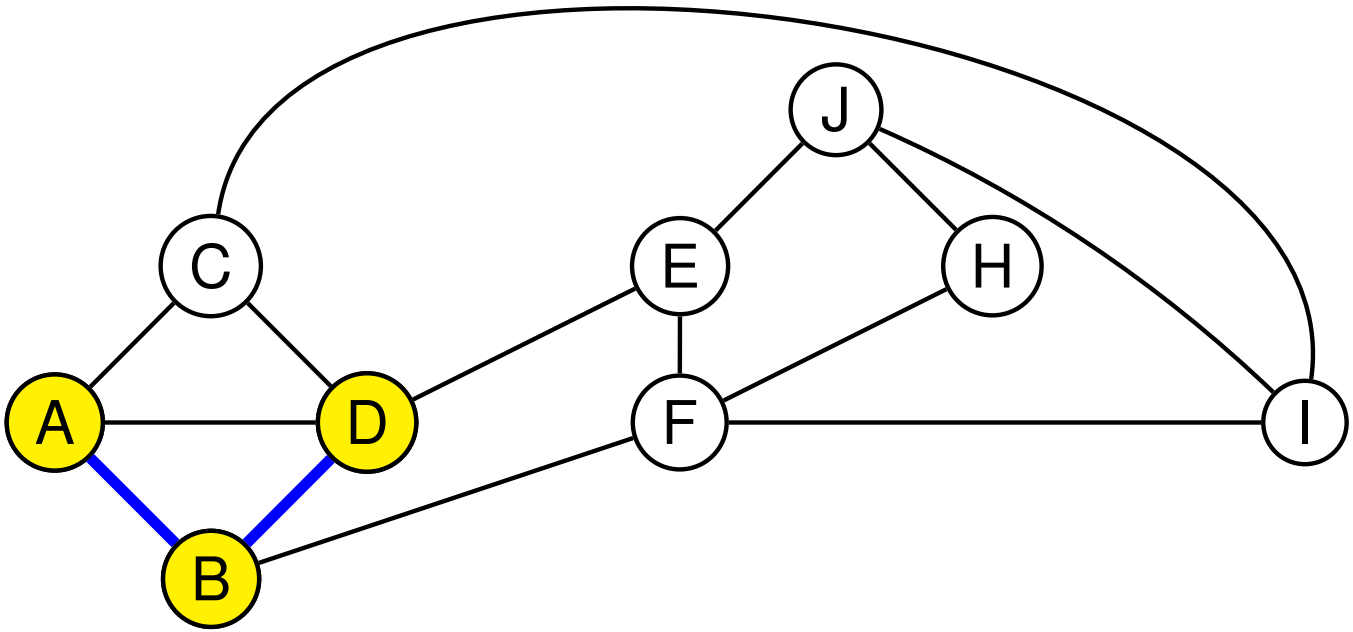
Un exemple : le circuit hamiltonien



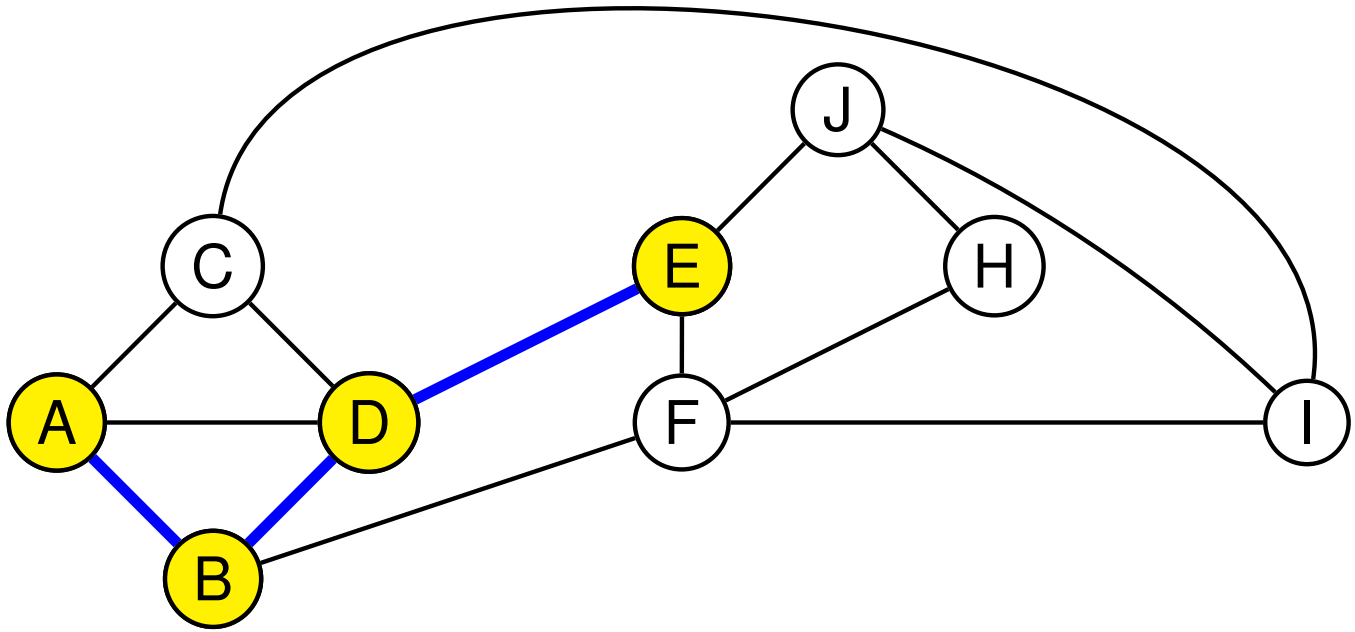
Un exemple : le circuit hamiltonien



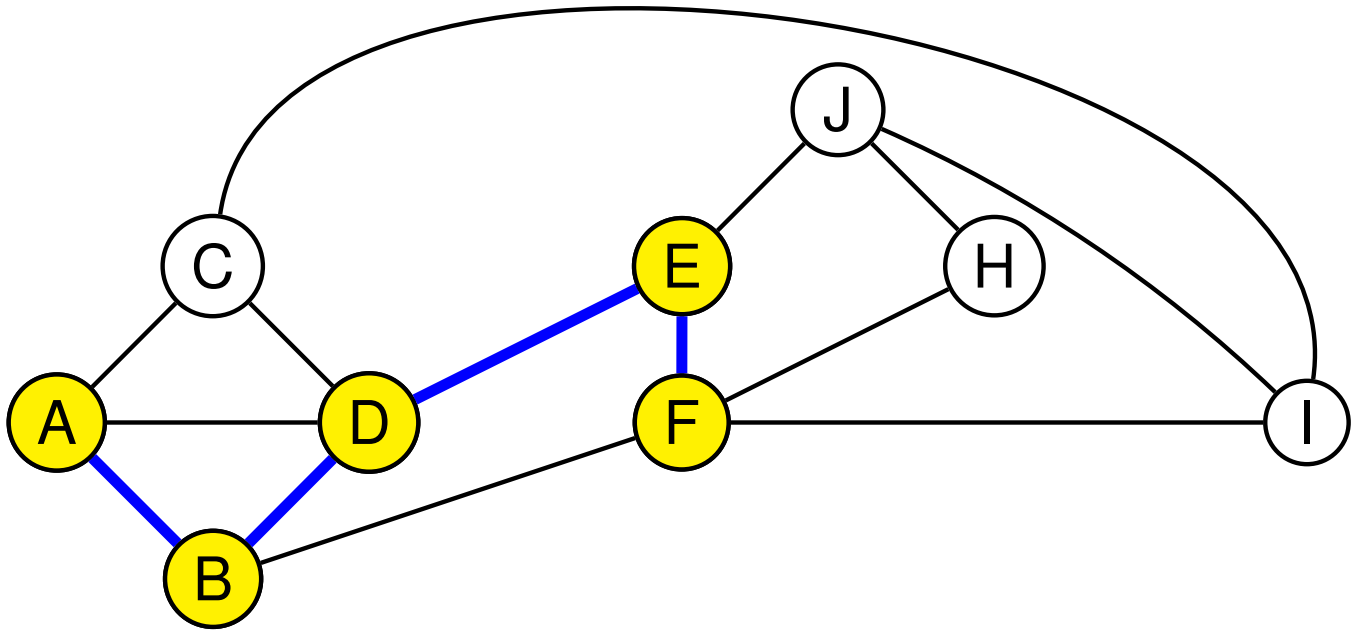
Un exemple : le circuit hamiltonien



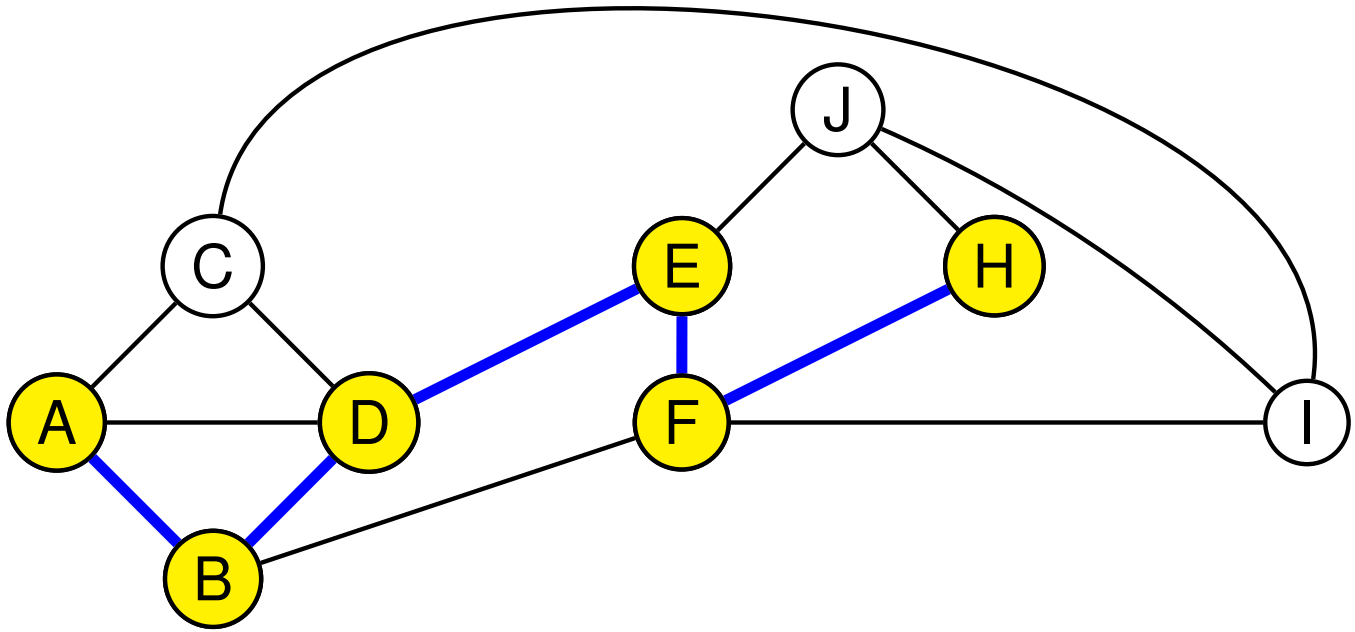
Un exemple : le circuit hamiltonien



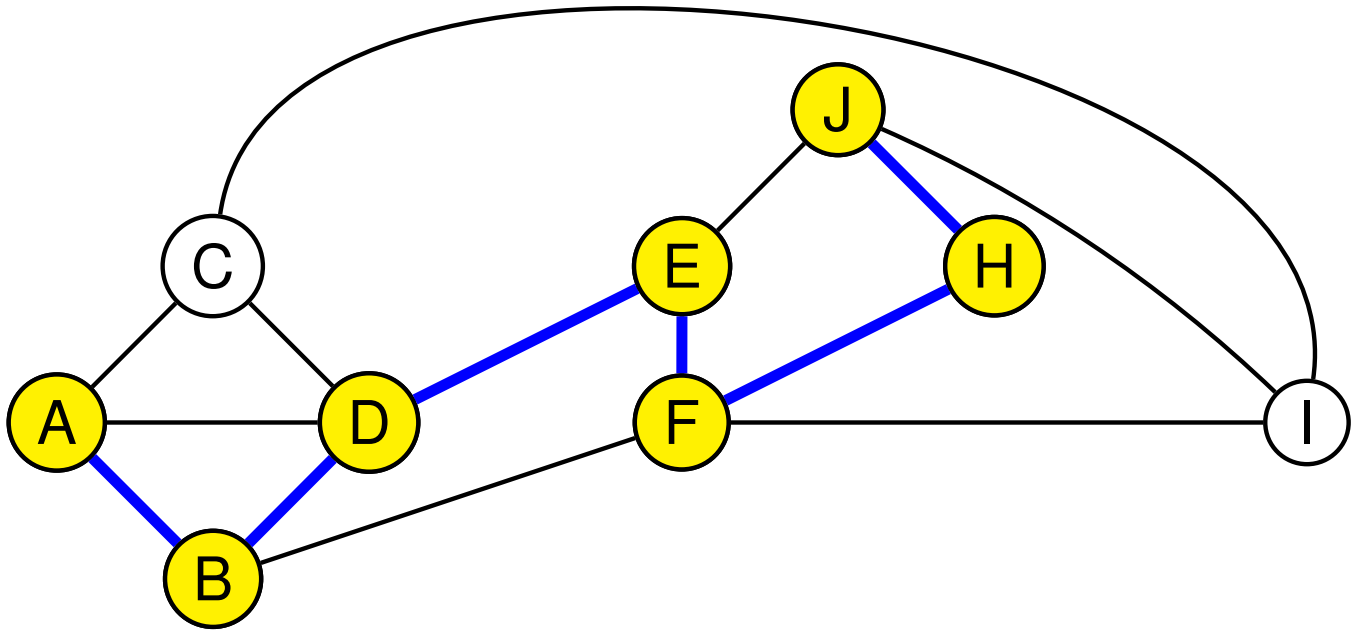
Un exemple : le circuit hamiltonien



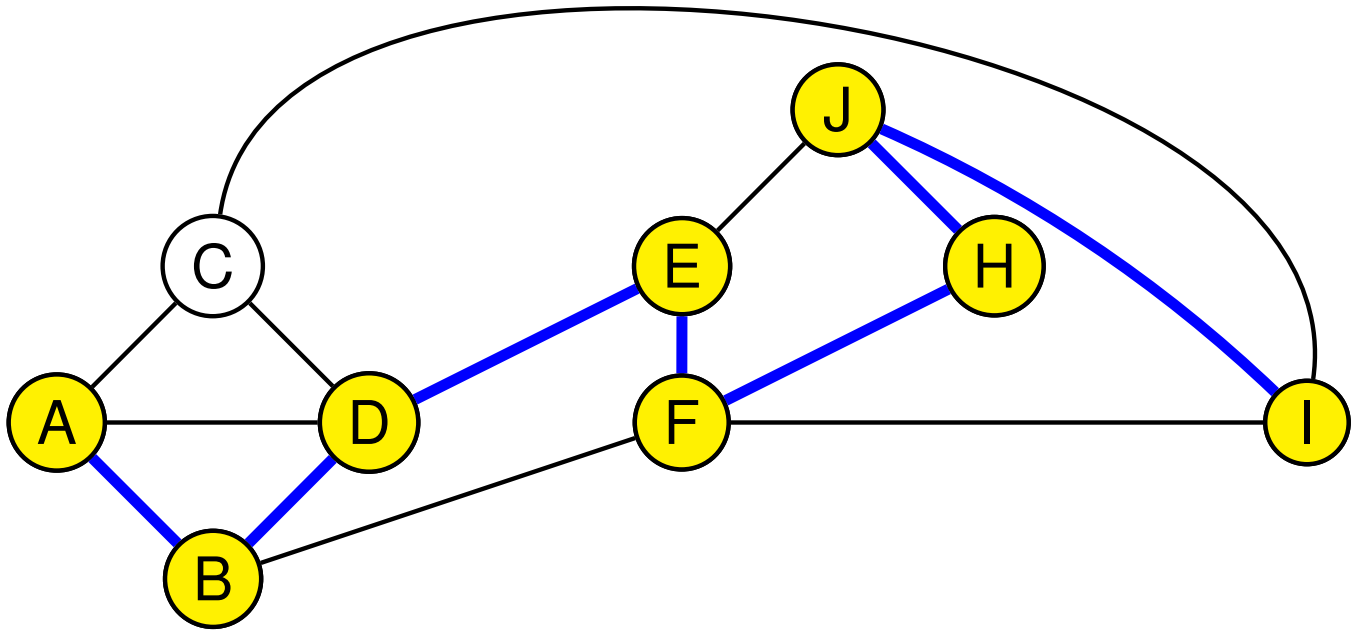
Un exemple : le circuit hamiltonien



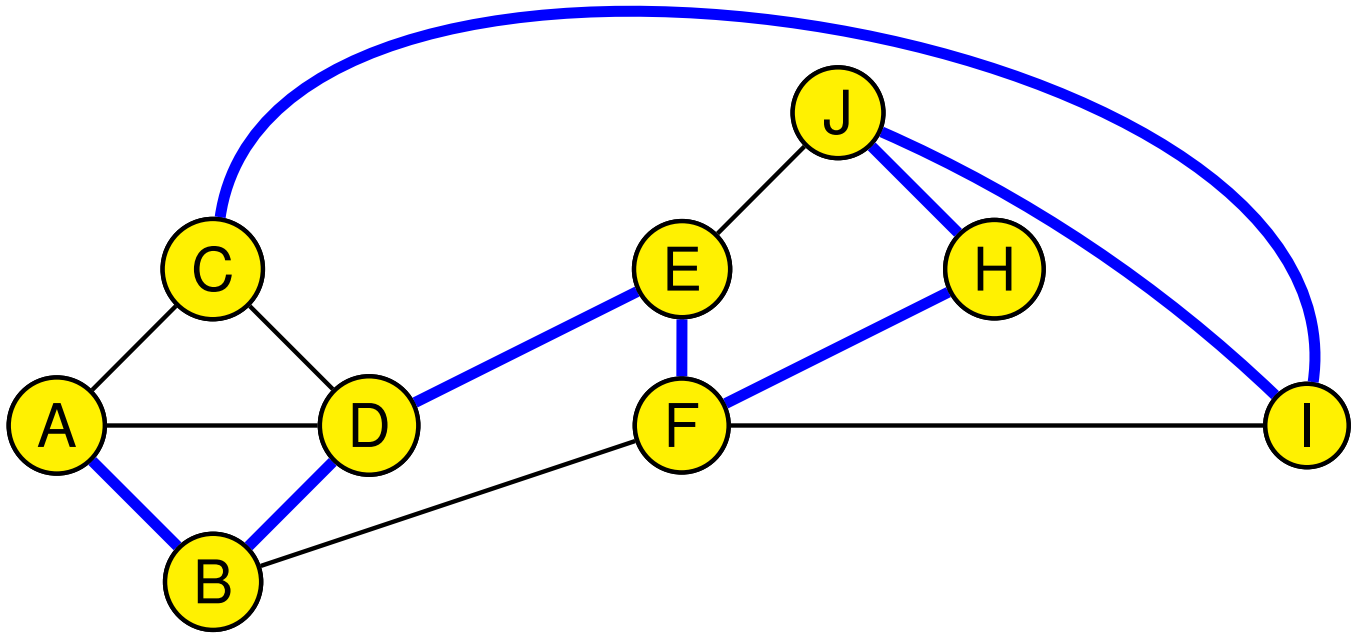
Un exemple : le circuit hamiltonien



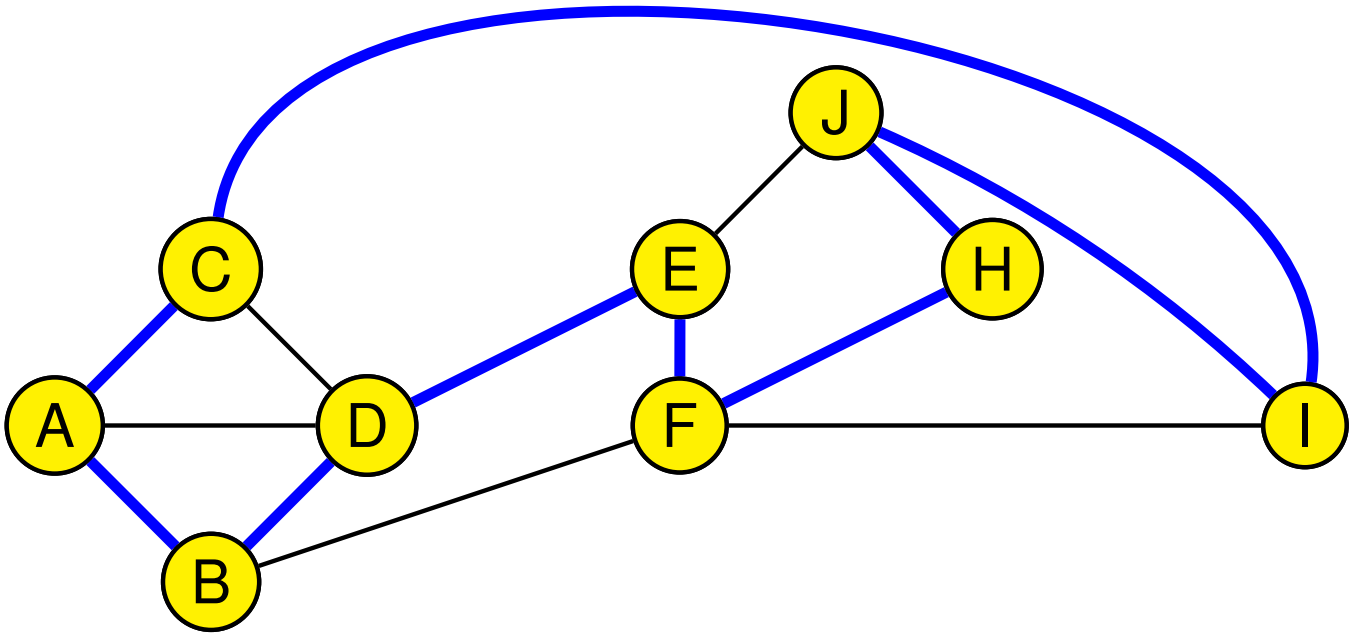
Un exemple : le circuit hamiltonien



Un exemple : le circuit hamiltonien



Un exemple : le circuit hamiltonien



Circuit hamiltonien : bilan

Problème difficile !

Taille du graphe = nombre d'arêtes + nombre de sommets $= |E| + |V|$

Nombre d'étapes

On peut être amené à essayer tous les chemins possibles... Il y en a :

$$(n-1)(n-2)\cdots 2 = (n-1)! \text{ avec } n = |V|$$

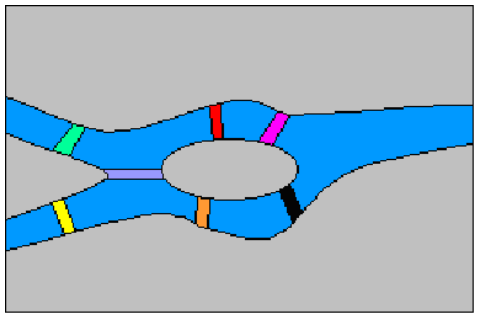
Pas de stratégie franchement meilleure (en dessous de 2^n) connue...

Le nombre d'étapes est exponentiel en la taille du graphe.

Plus facile : le circuit eulérien

Circuit eulérien

Etant donné un graphe $G = (V, E)$, peut-on trouver un circuit passant une seule fois par chaque arête du graphe ?

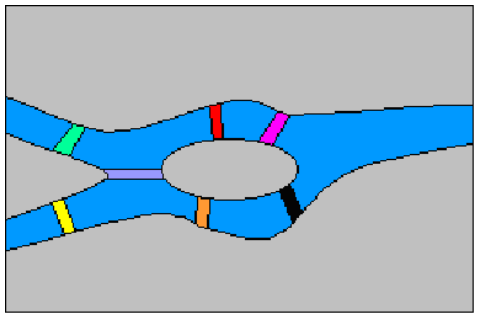


Les 7 ponts de Koenigsberg

Plus facile : le circuit eulérien

Circuit eulérien

Etant donné un graphe $G = (V, E)$, peut-on trouver un circuit passant une seule fois par chaque arête du graphe ?



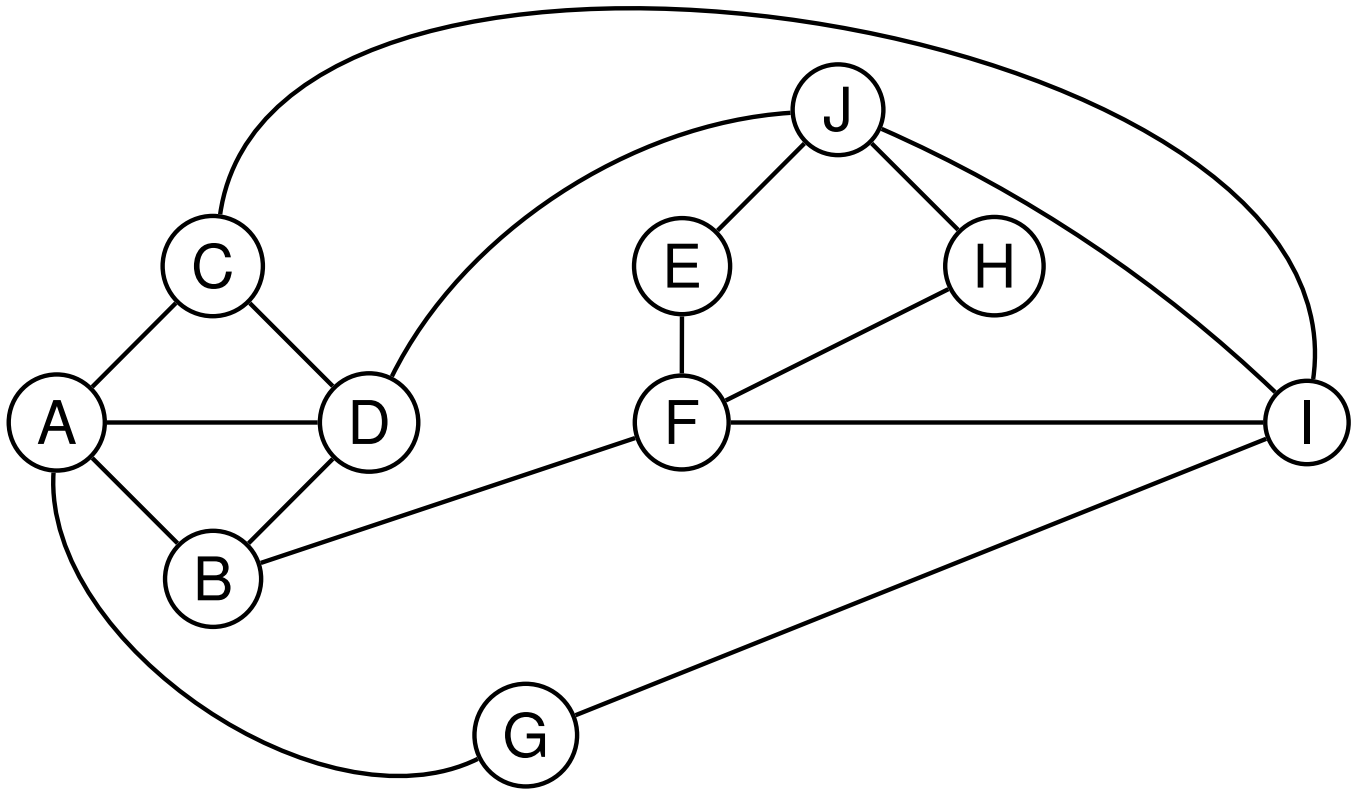
Les 7 ponts de Koenigsberg

Caractérisation

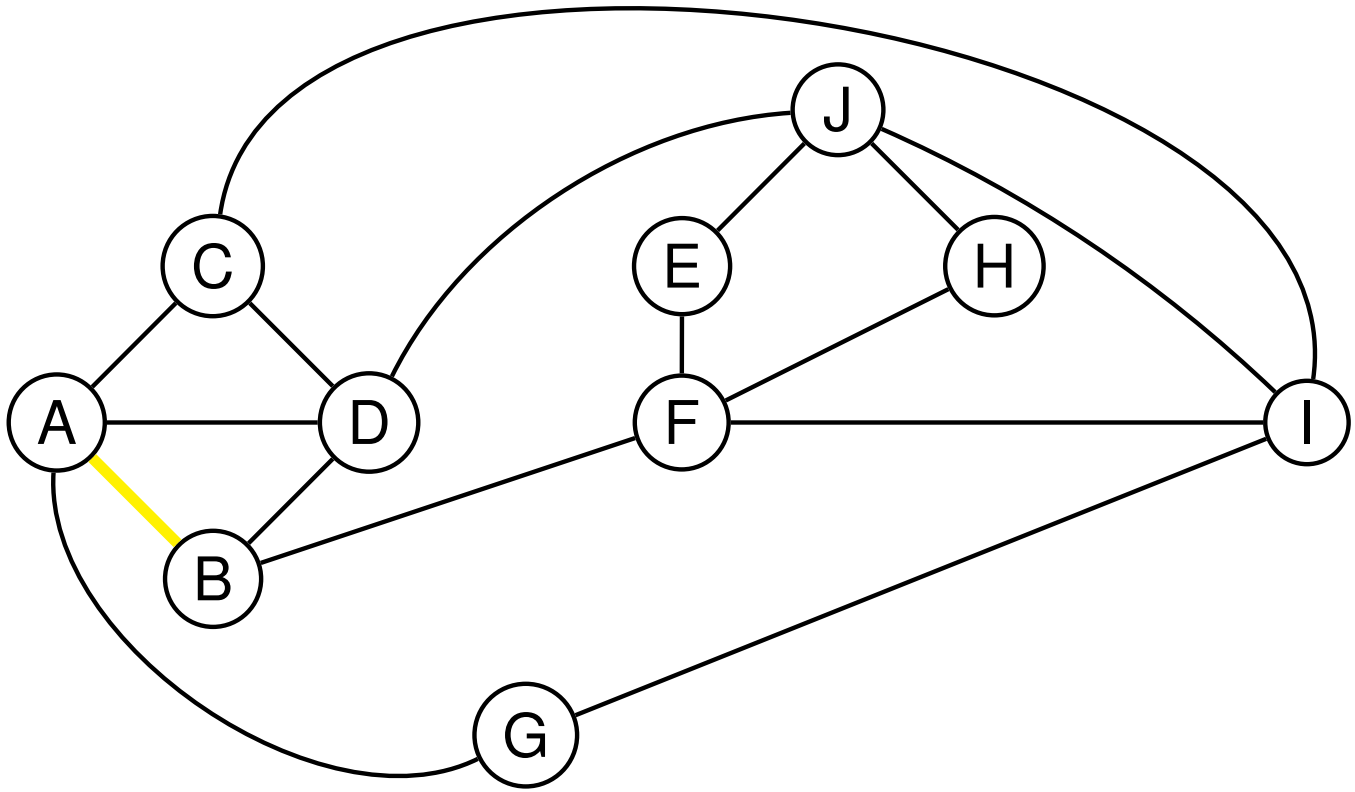
Un graphe $G = (V, E)$ admet un circuit eulérien si et seulement si tout sommet $v \in V$ est de degré pair.

Degré d'un sommet v : nombre de sommets reliés à v (ses "voisins")

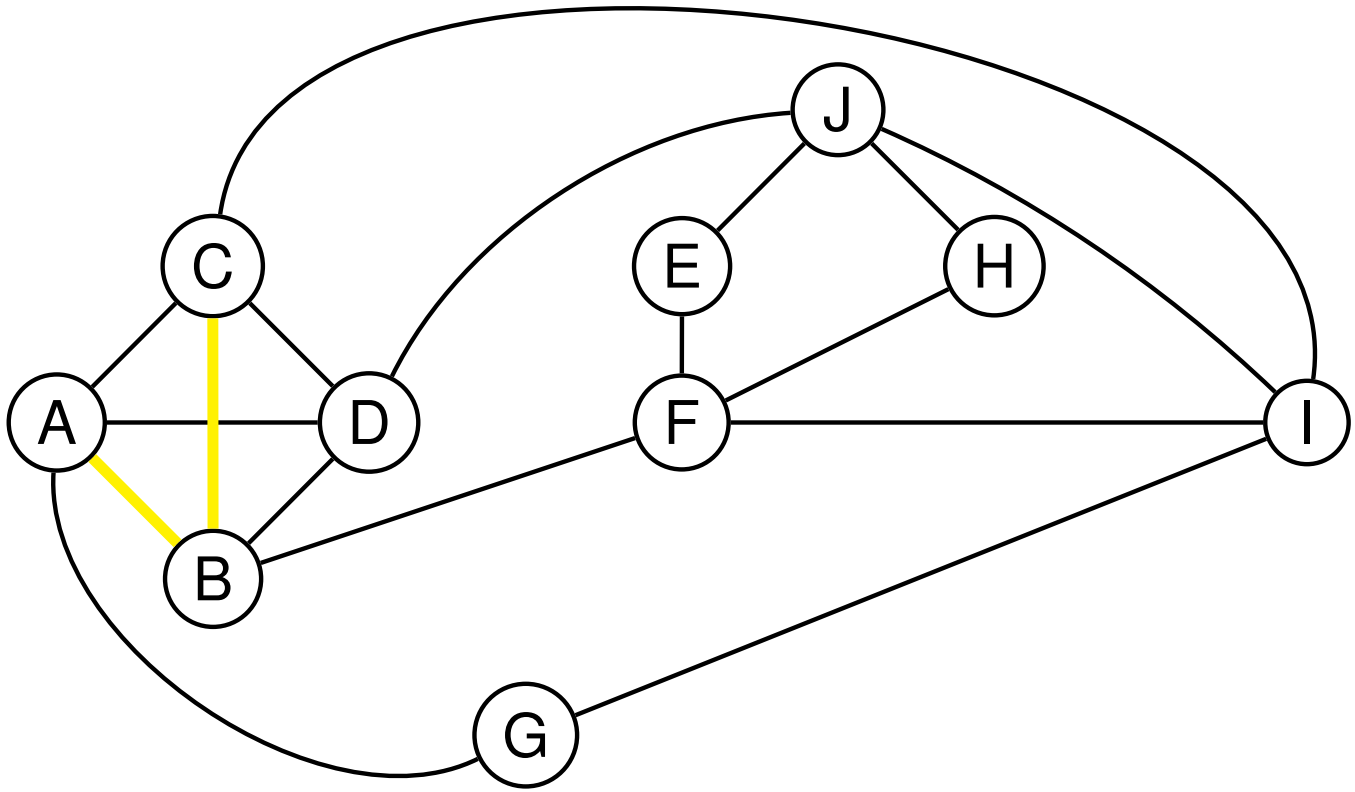
Circuit eulérien



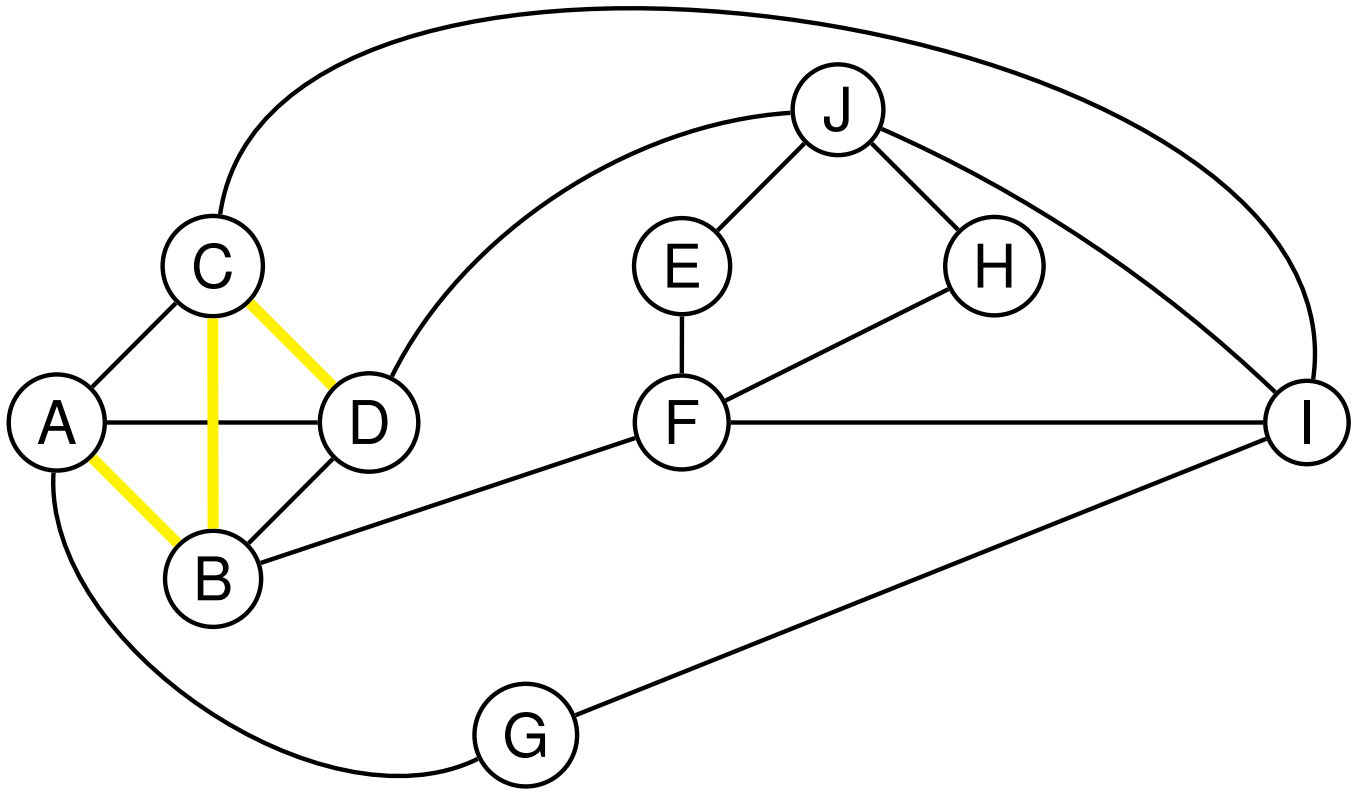
Circuit eulérien



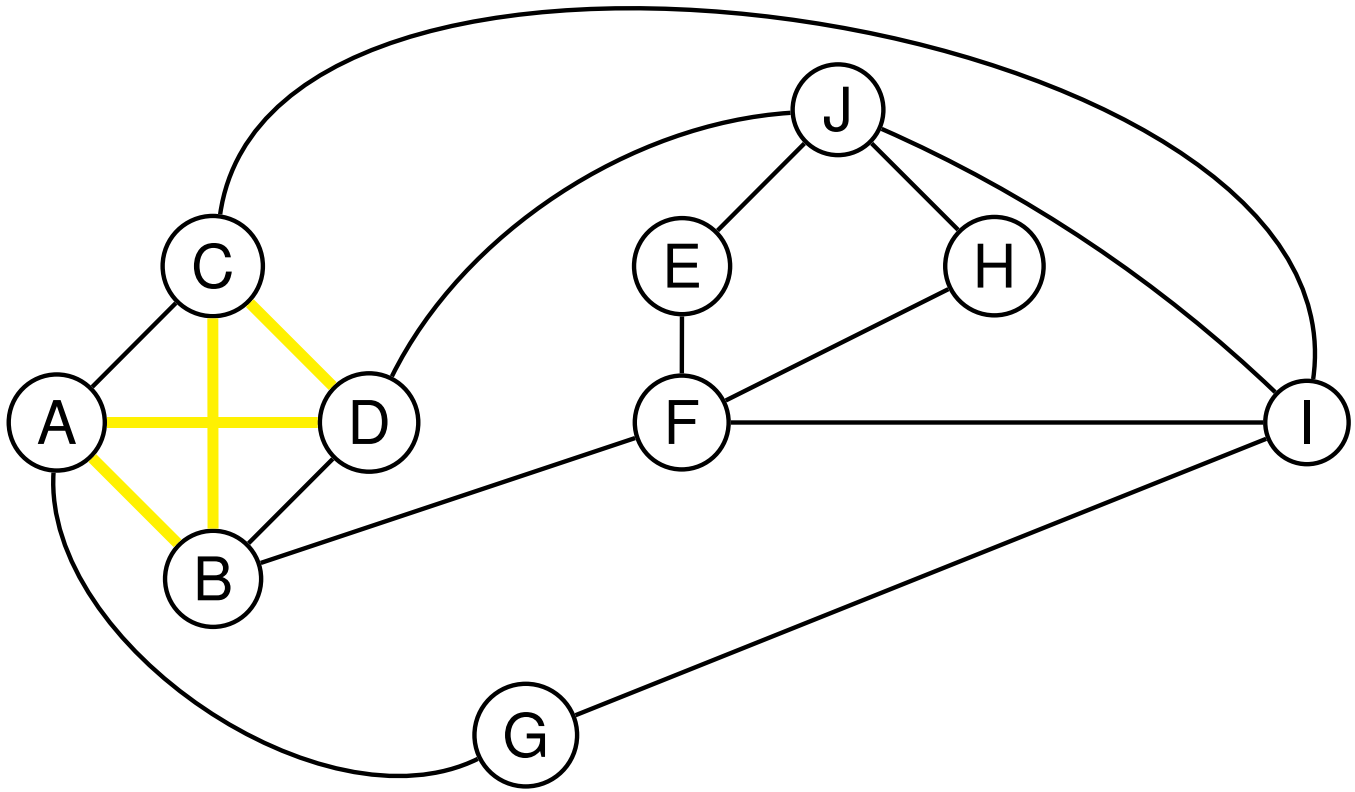
Circuit eulérien



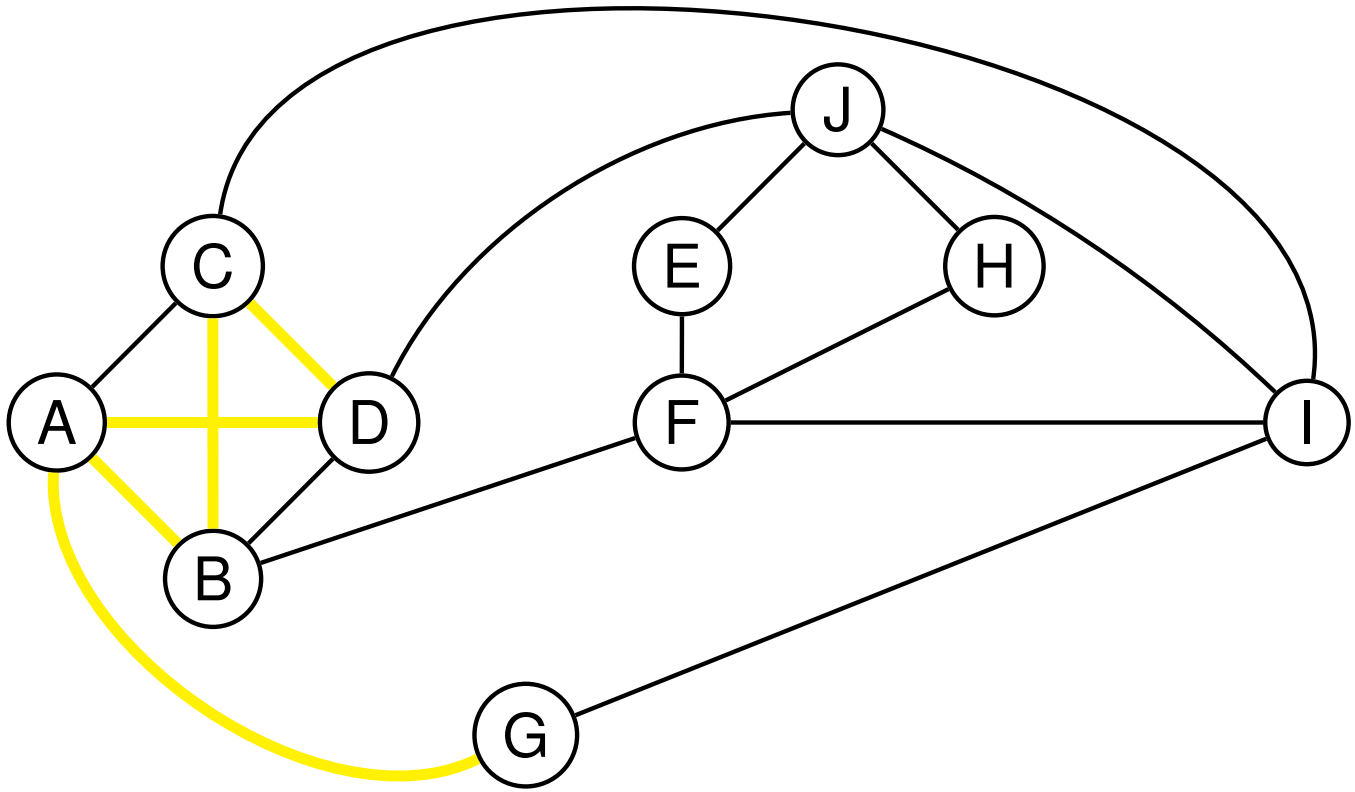
Circuit eulérien



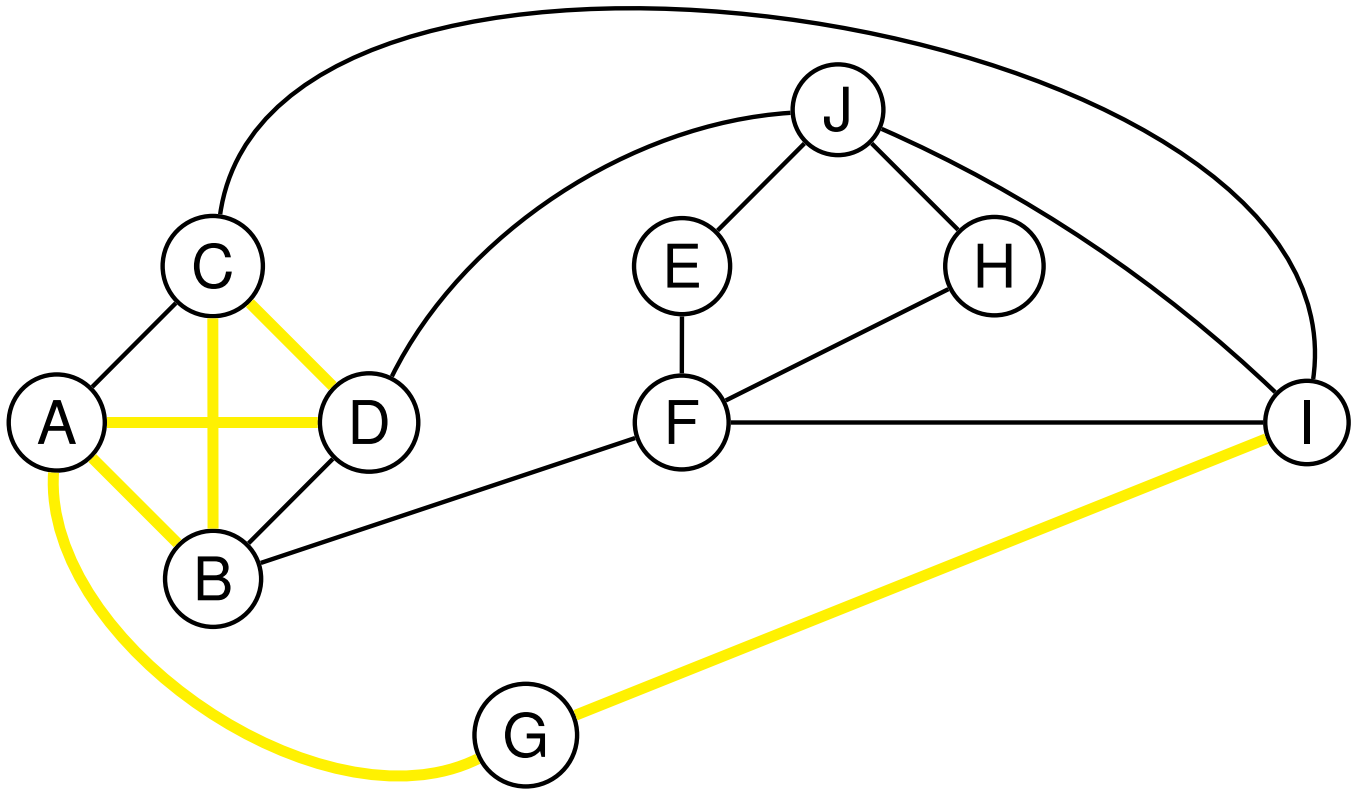
Circuit eulérien



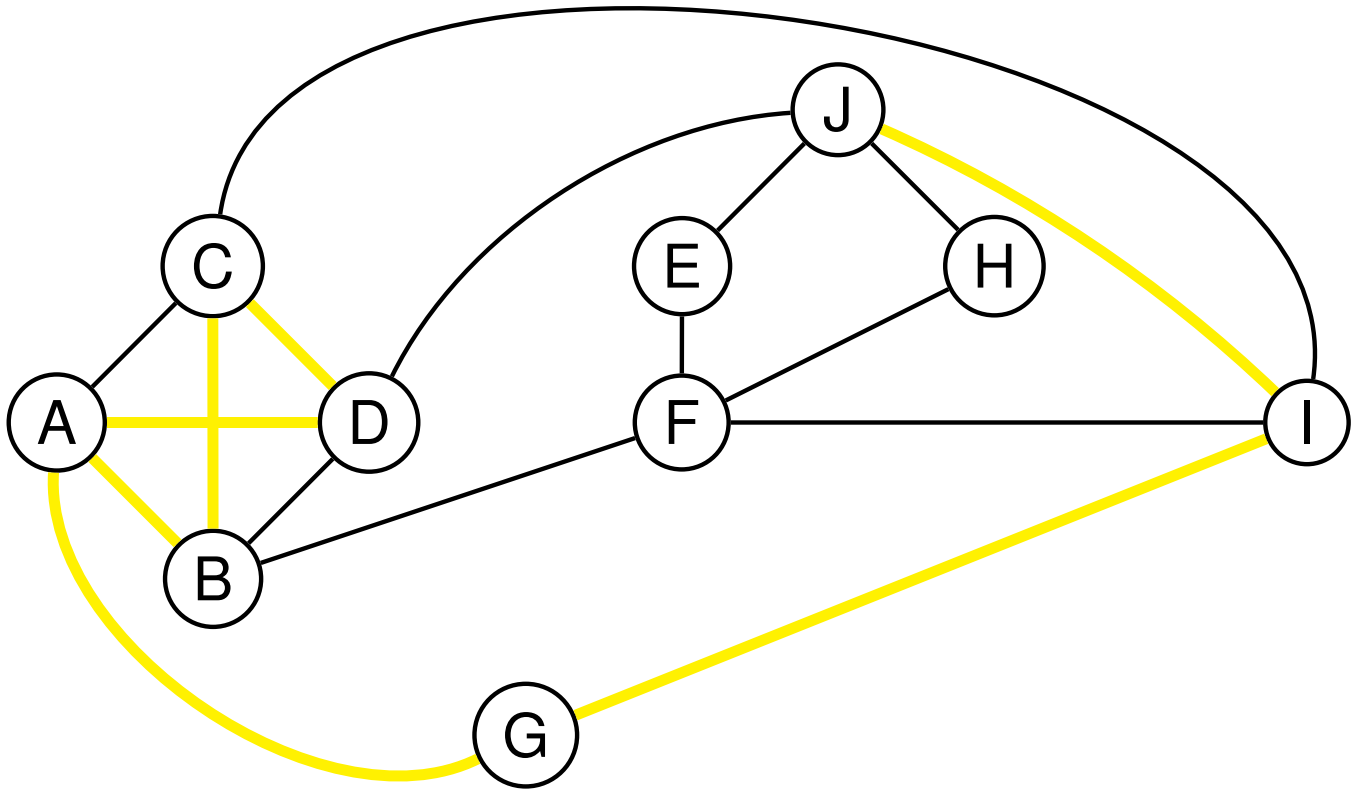
Circuit eulérien



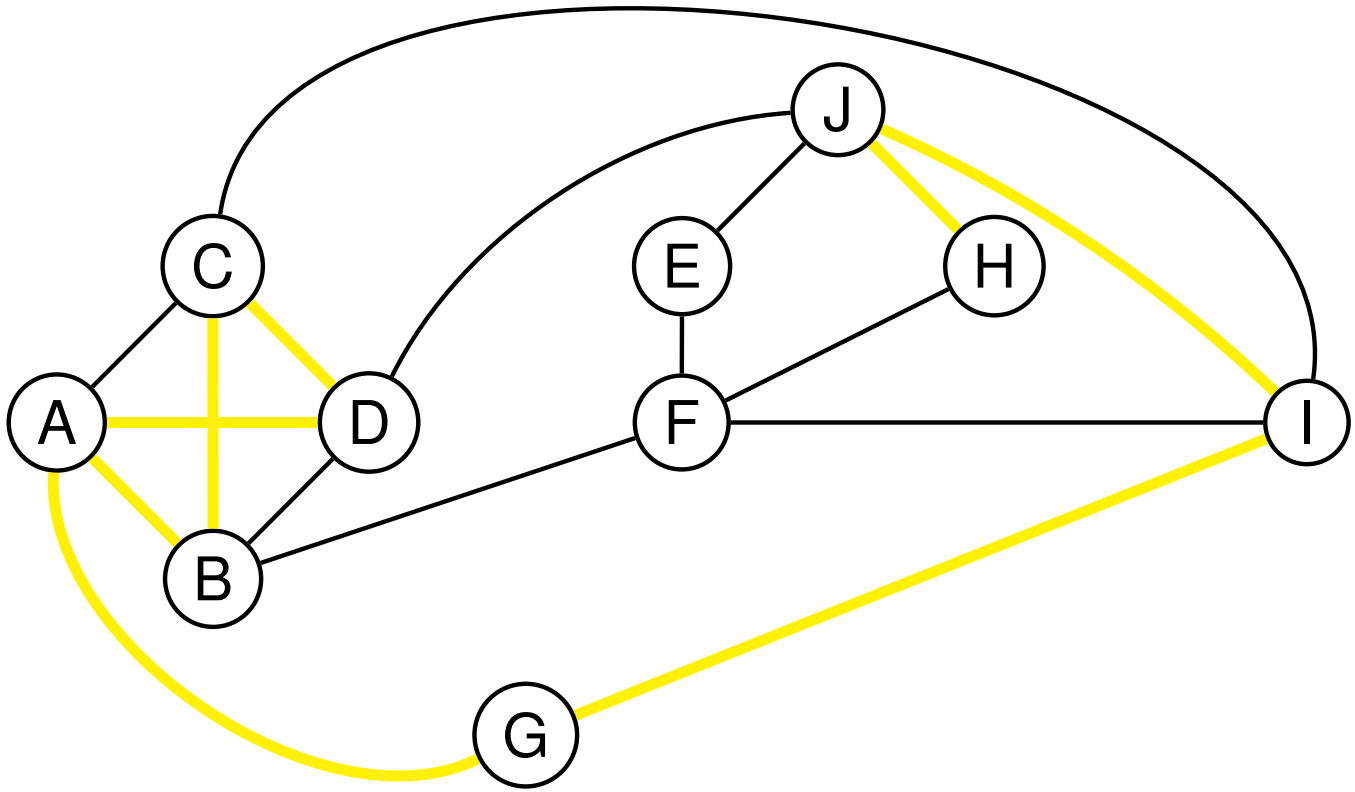
Circuit eulérien



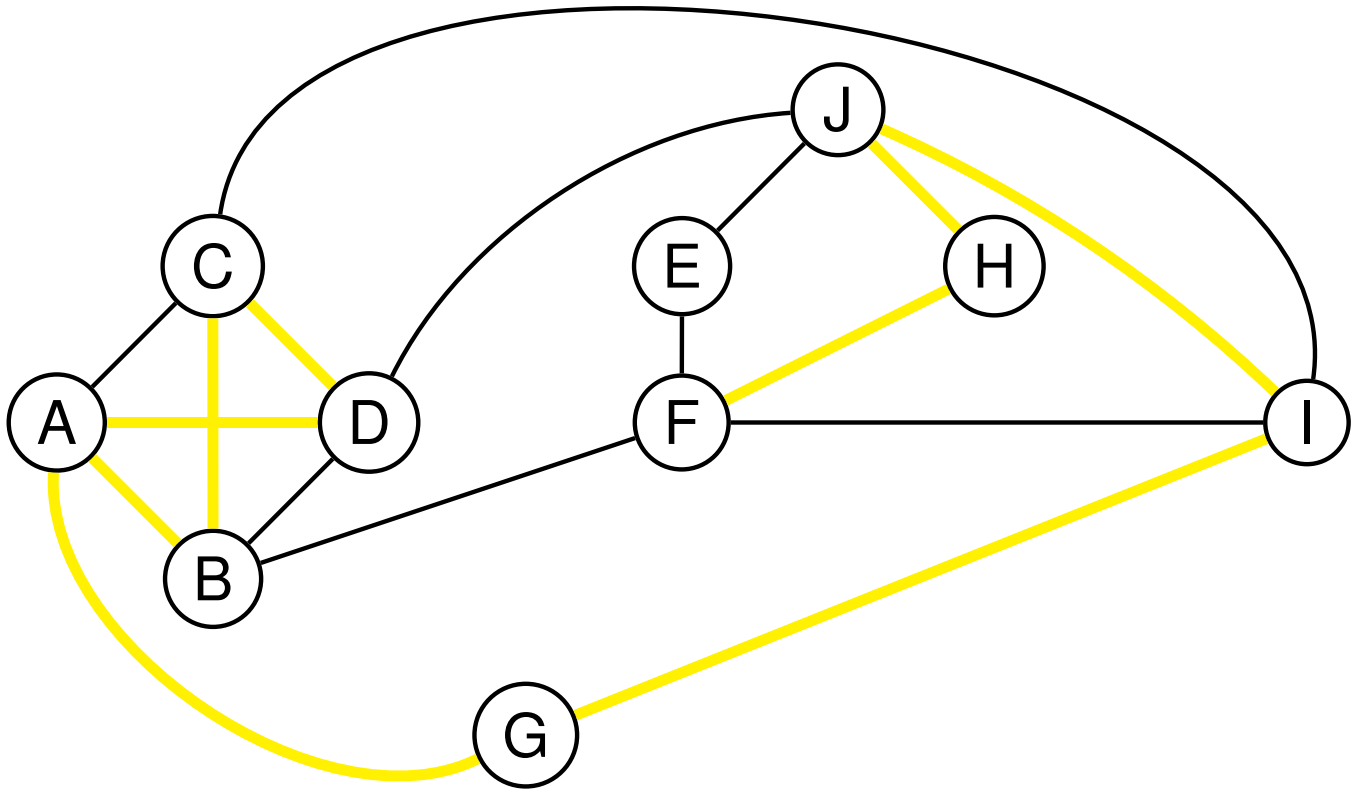
Circuit eulérien



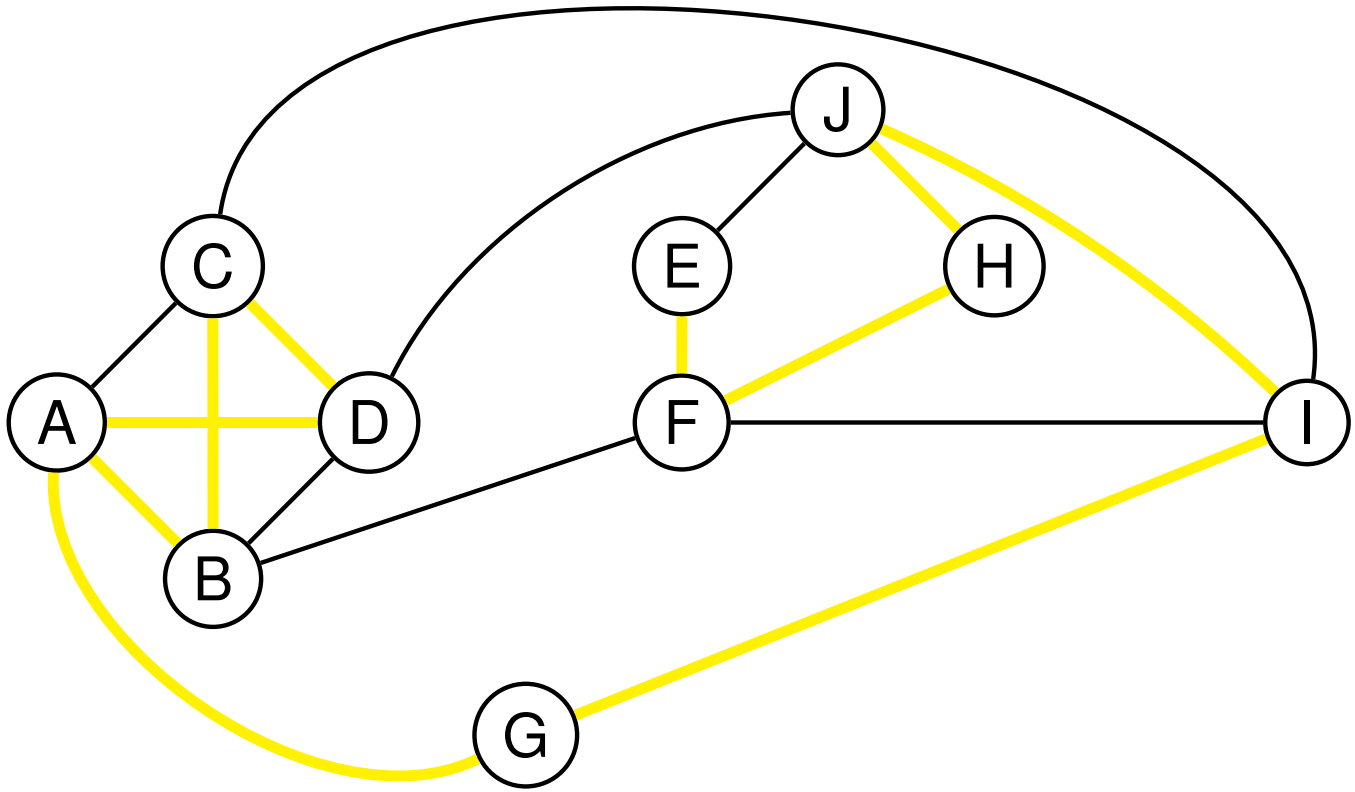
Circuit eulérien



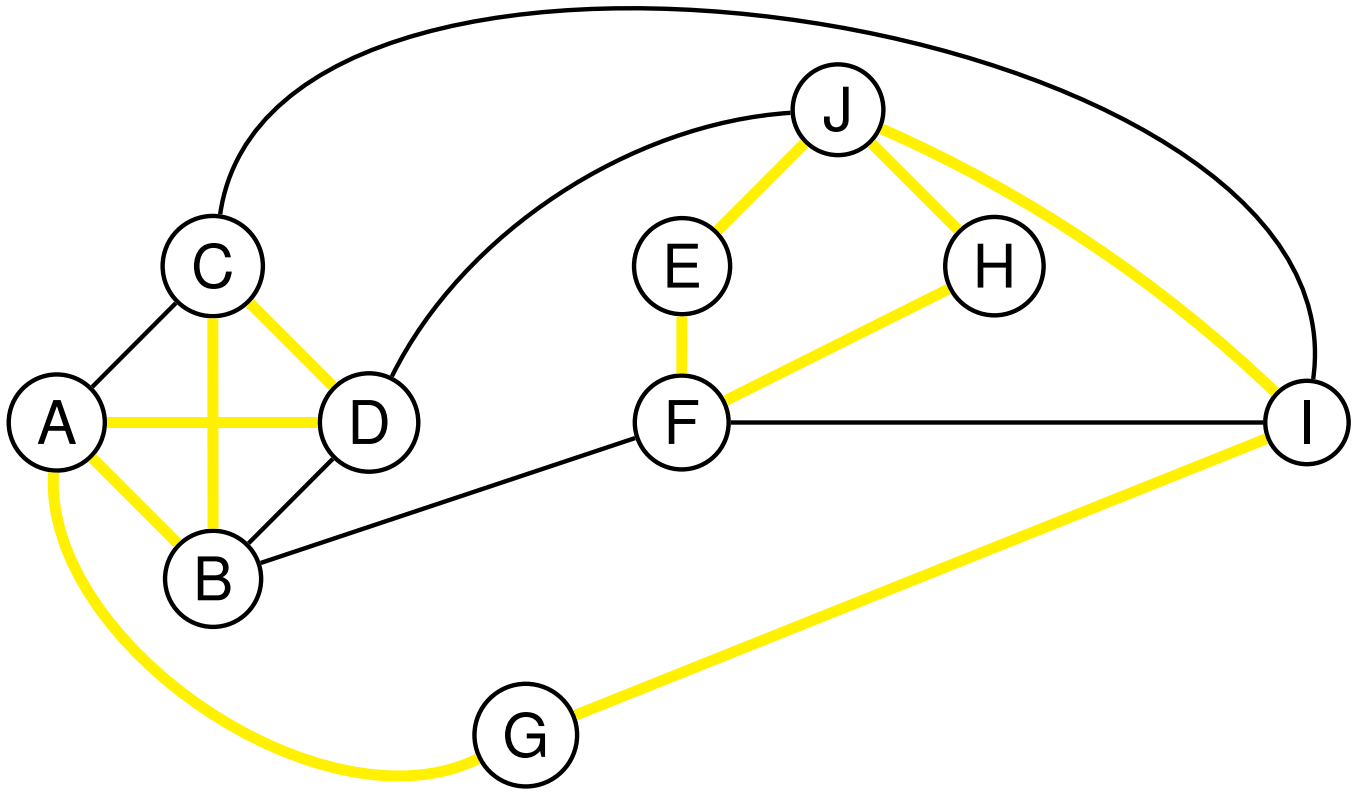
Circuit eulérien



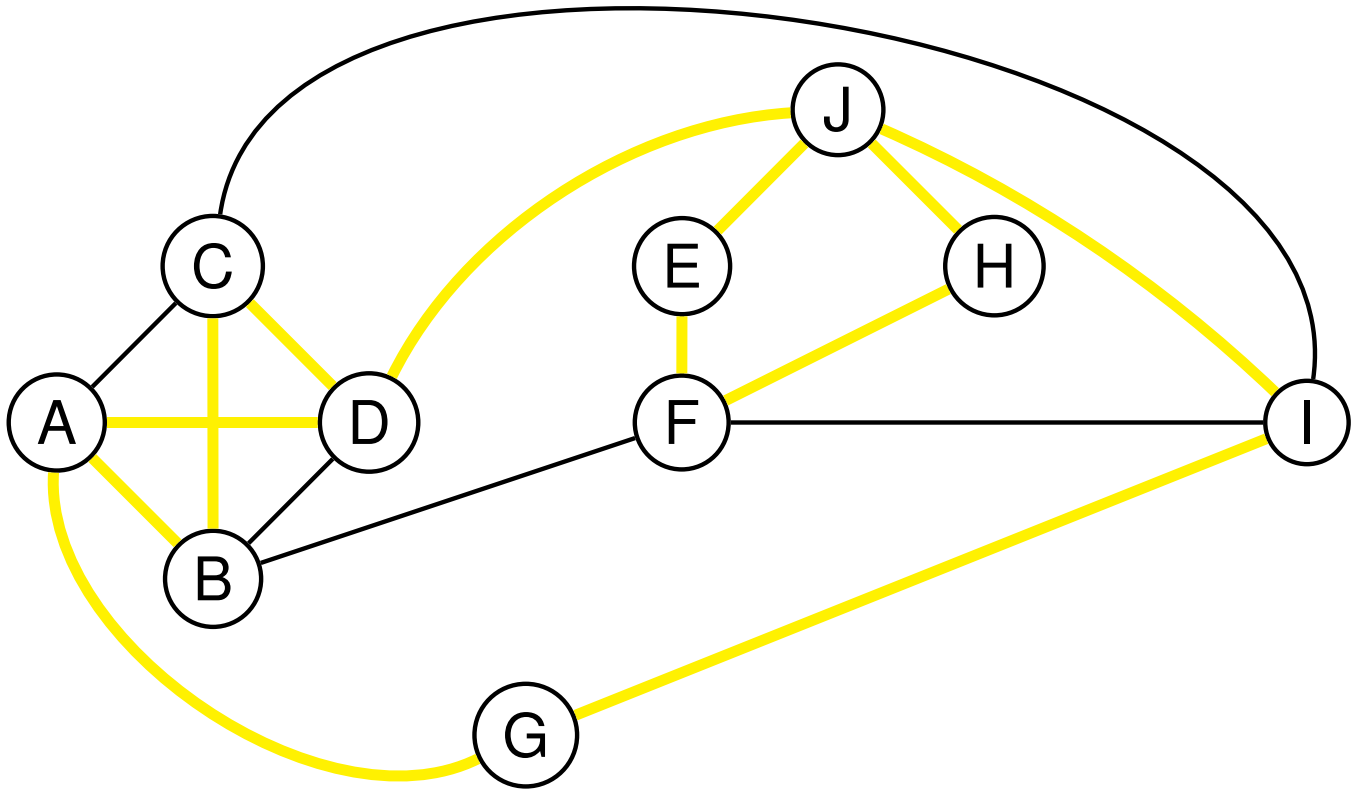
Circuit eulérien



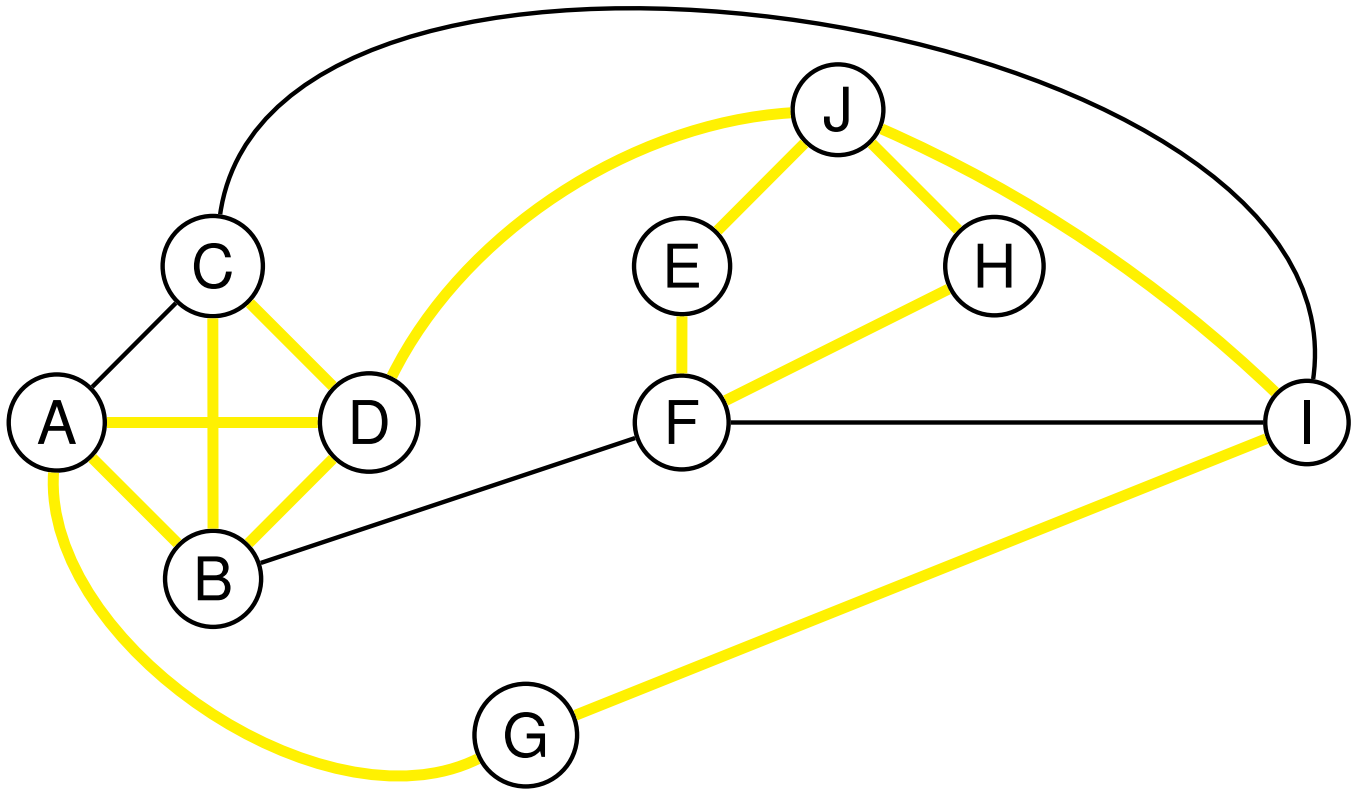
Circuit eulérien



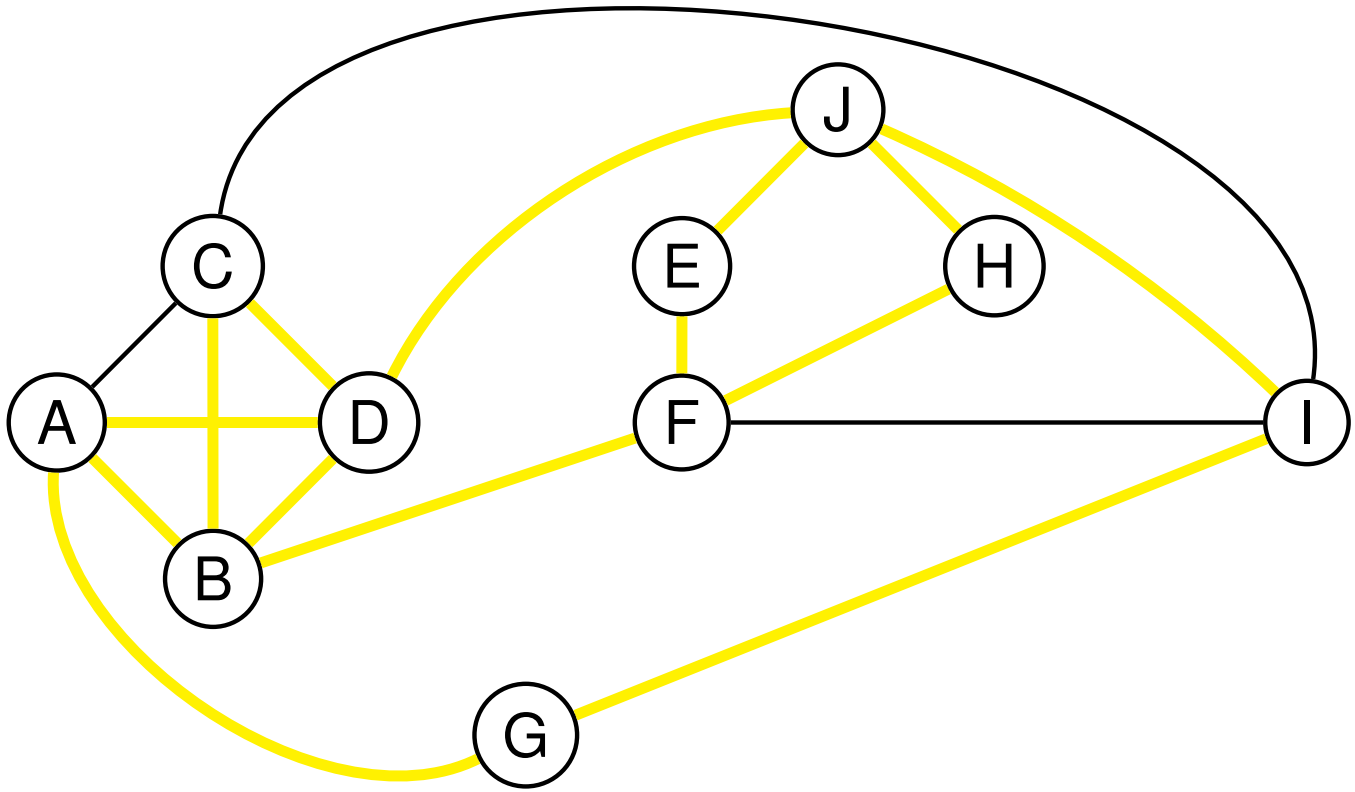
Circuit eulérien



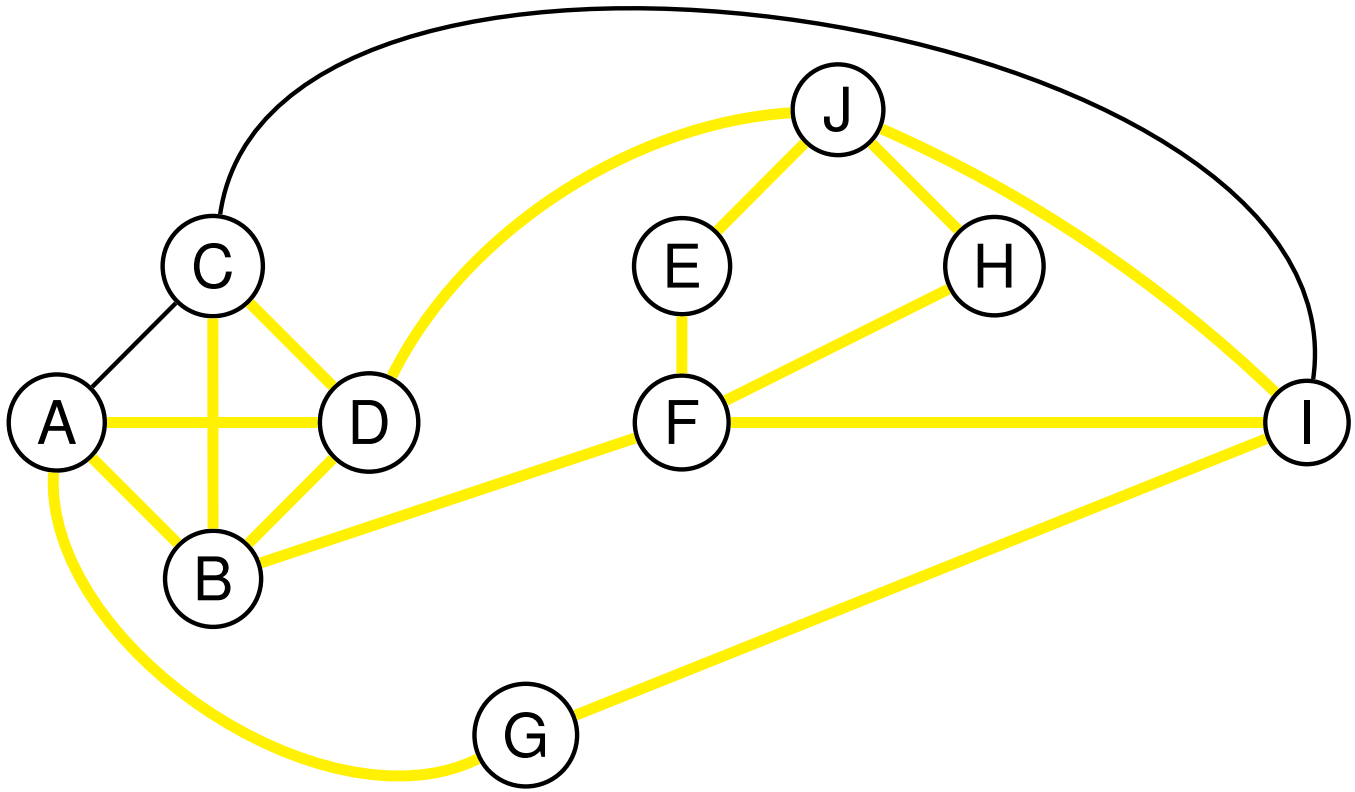
Circuit eulérien



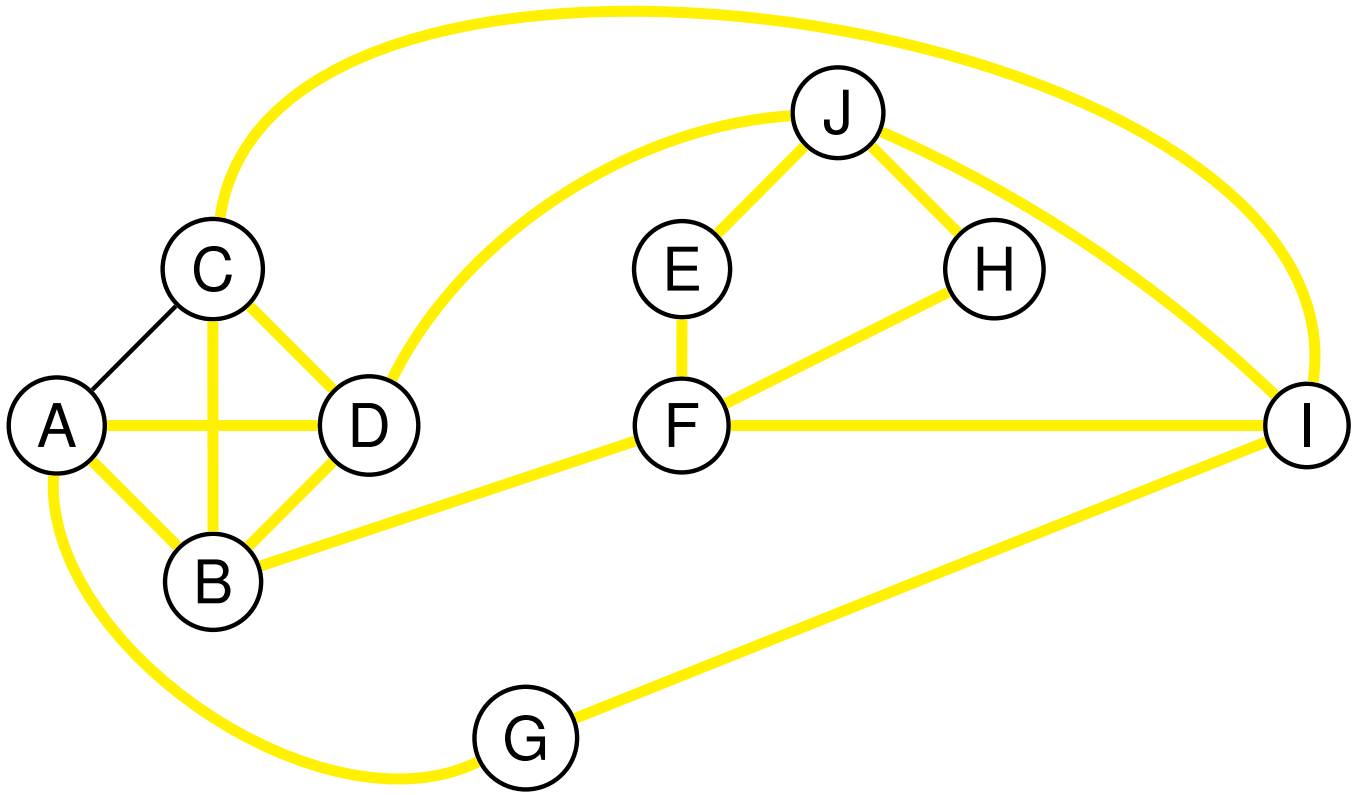
Circuit eulérien



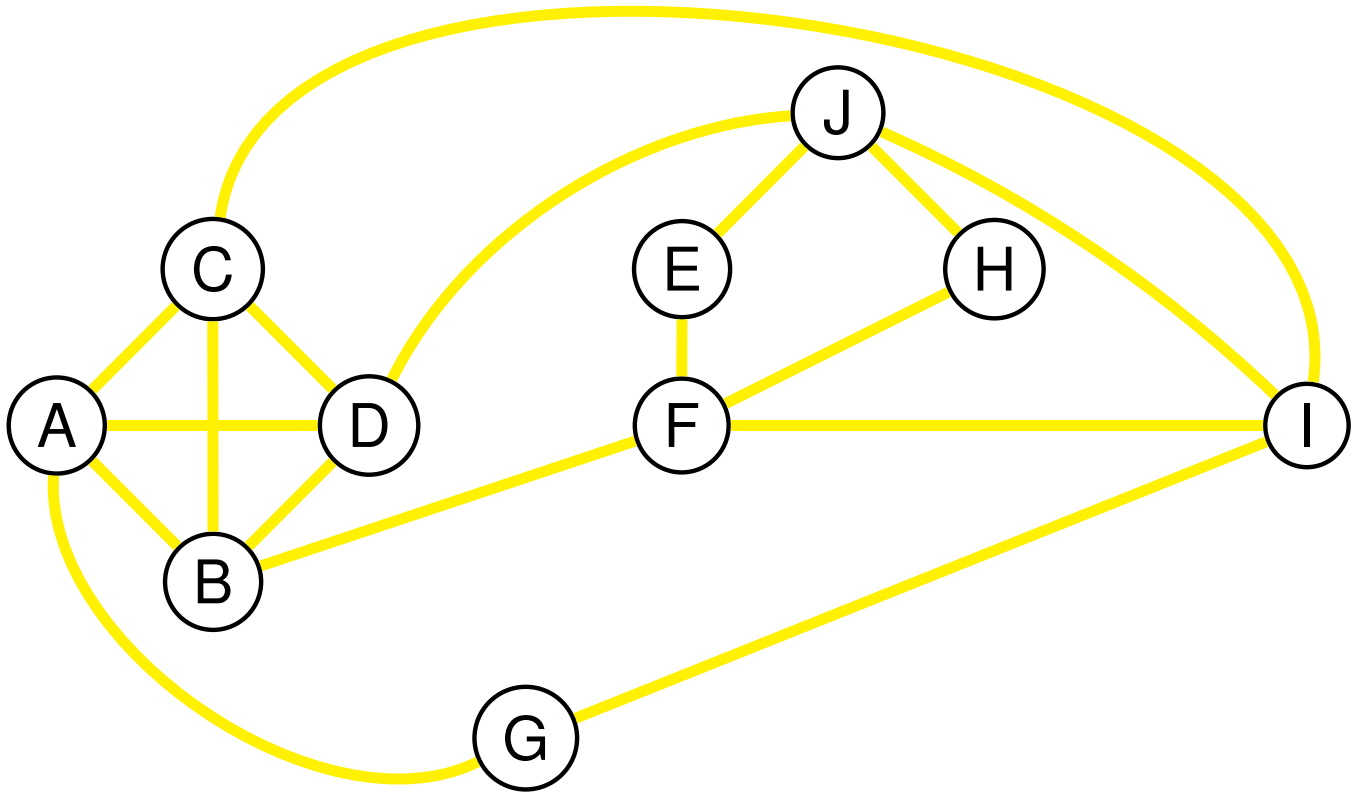
Circuit eulérien



Circuit eulérien



Circuit eulérien



Bilan

Nombre d'étapes pour le circuit eulérien

Besoin de tester le degré de chaque sommet. En tout, $|V|^2$ étapes (au grand maximum... $O(|E|)$ en fait).

Le nombre d'étapes est polynomial en la taille du graphe.

Bilan

Circuit hamiltonien : a priori difficile (exponentiel)
Circuit eulerien : facile (polynomial)

k-colorabilité

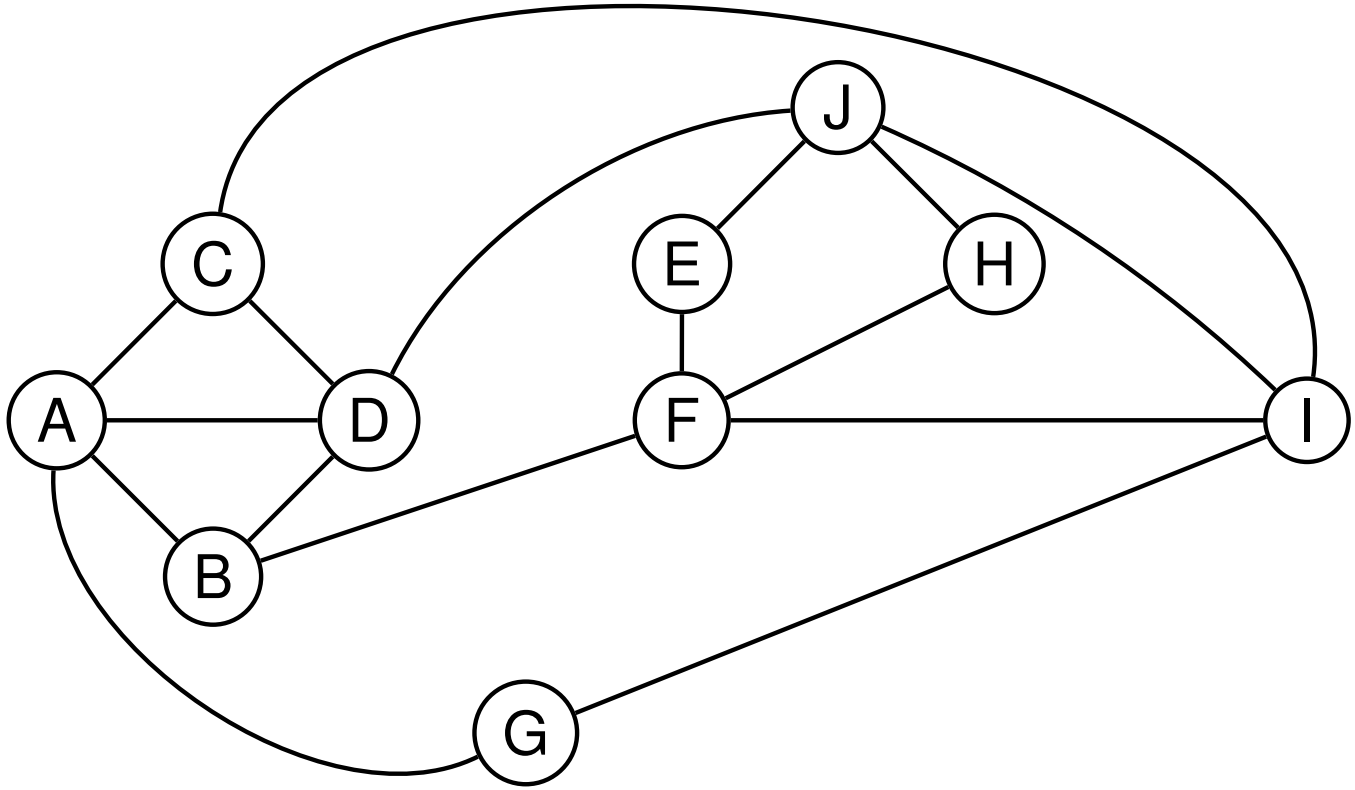
k-colorabilité

Etant donnés un graphe G et un entier k, peut-on colorier les sommets de G avec au plus k couleurs de manière à ce que deux sommets adjacents ne portent jamais la même couleur ?

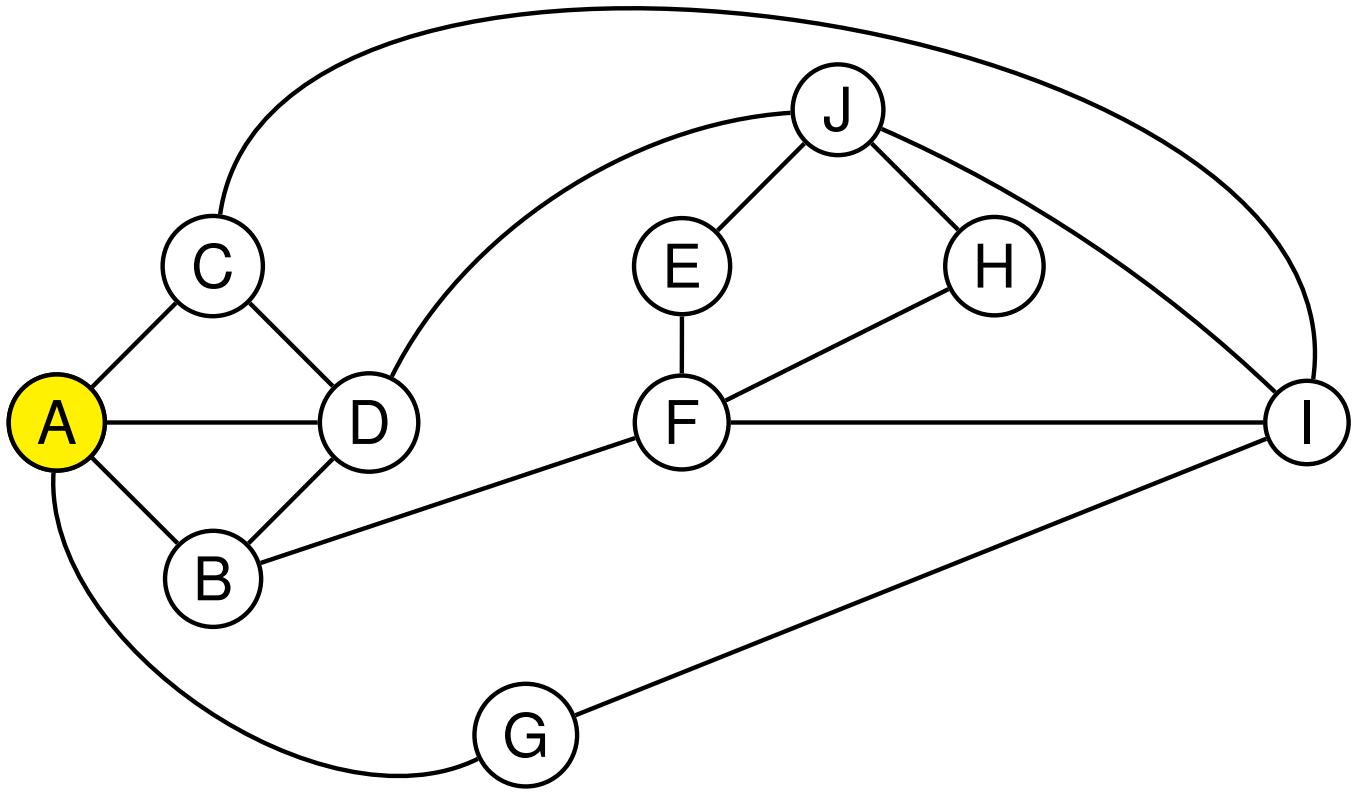
Si le graphe G est une carte et $k \geq 4$, la réponse est toujours oui.

3-colorabilité

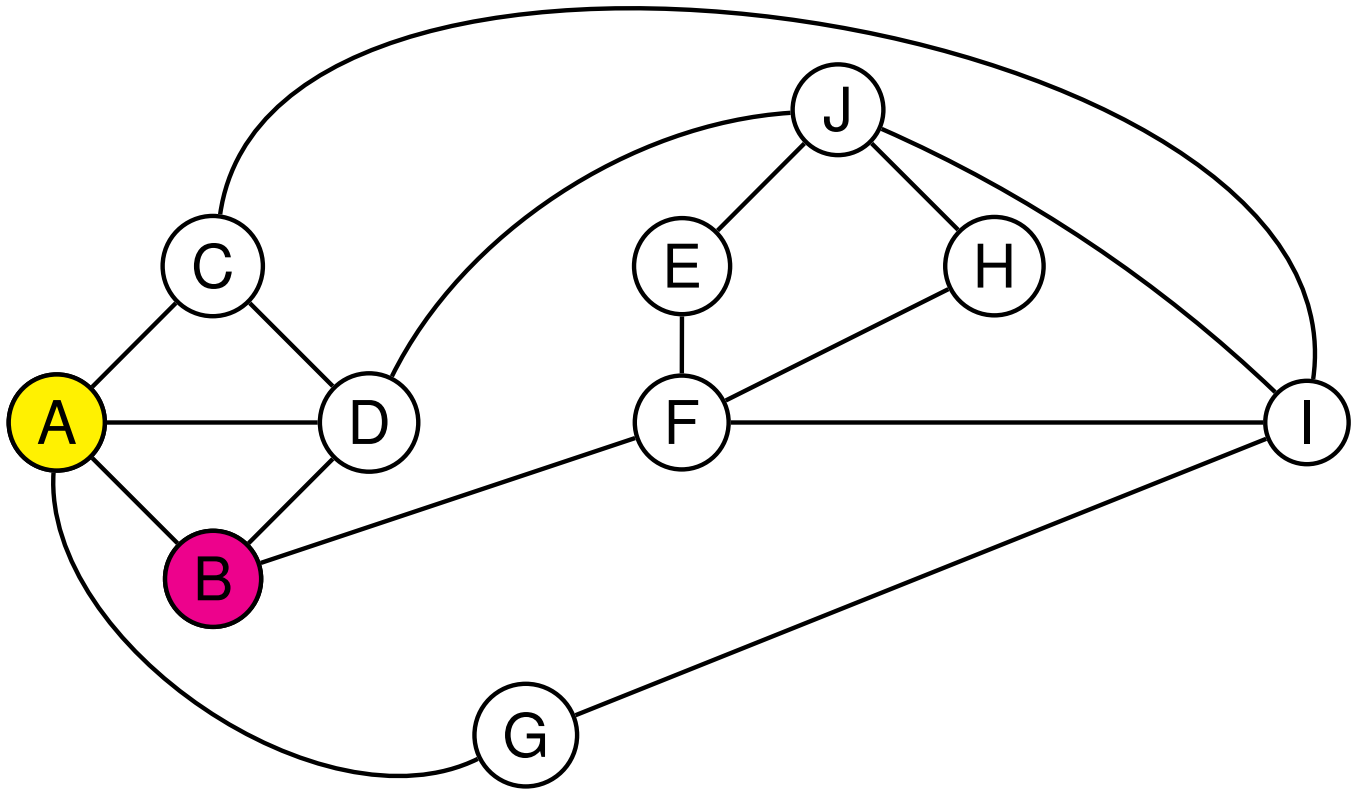
3 couleurs : jaune,bleu,rouge



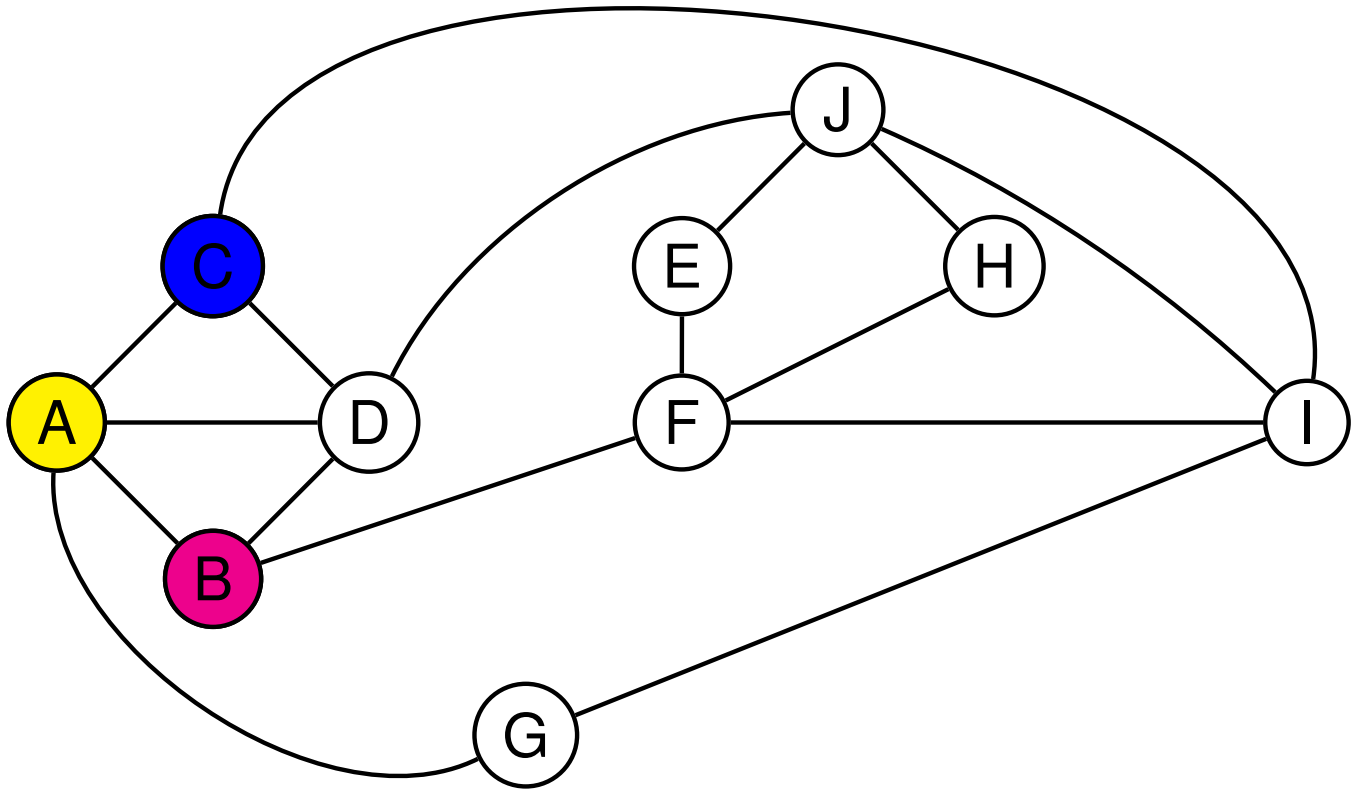
3-colorabilité



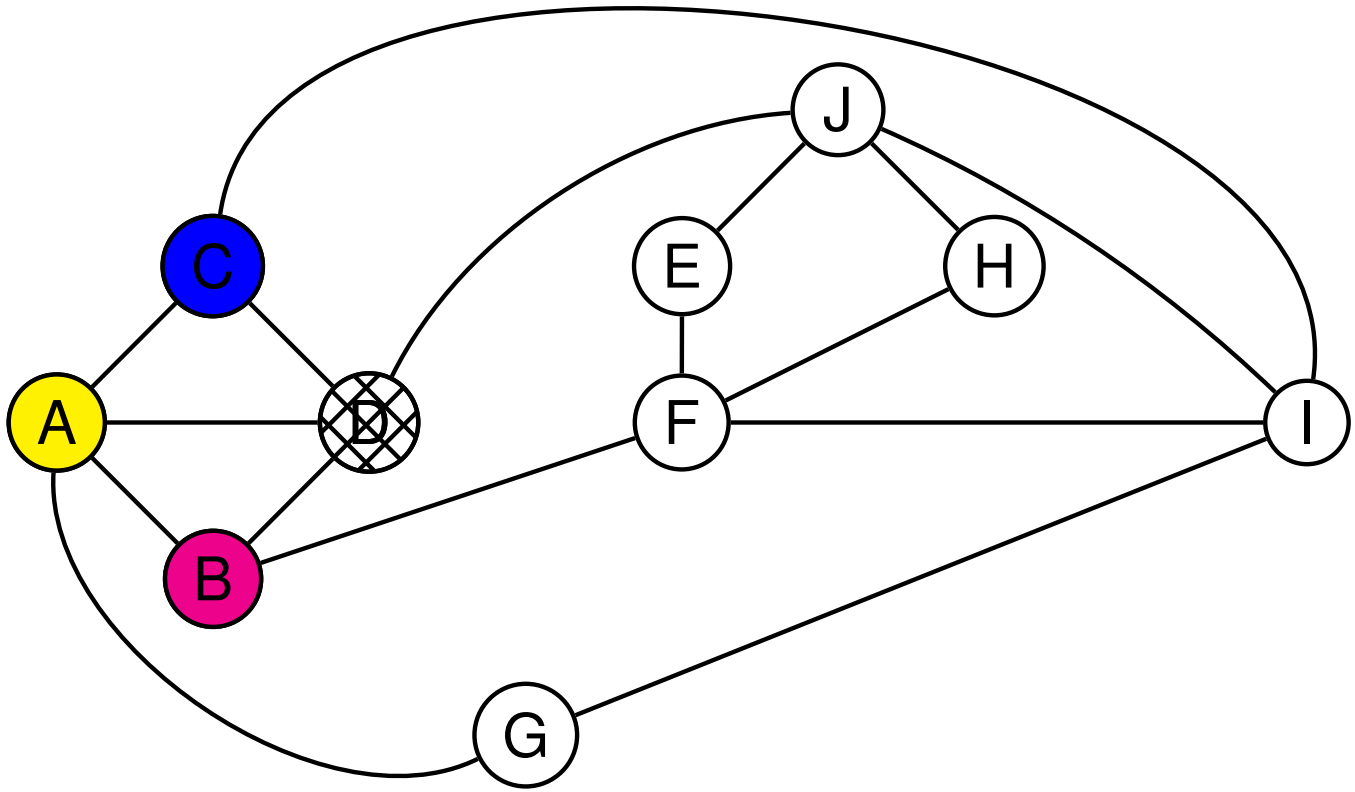
3-colorabilité



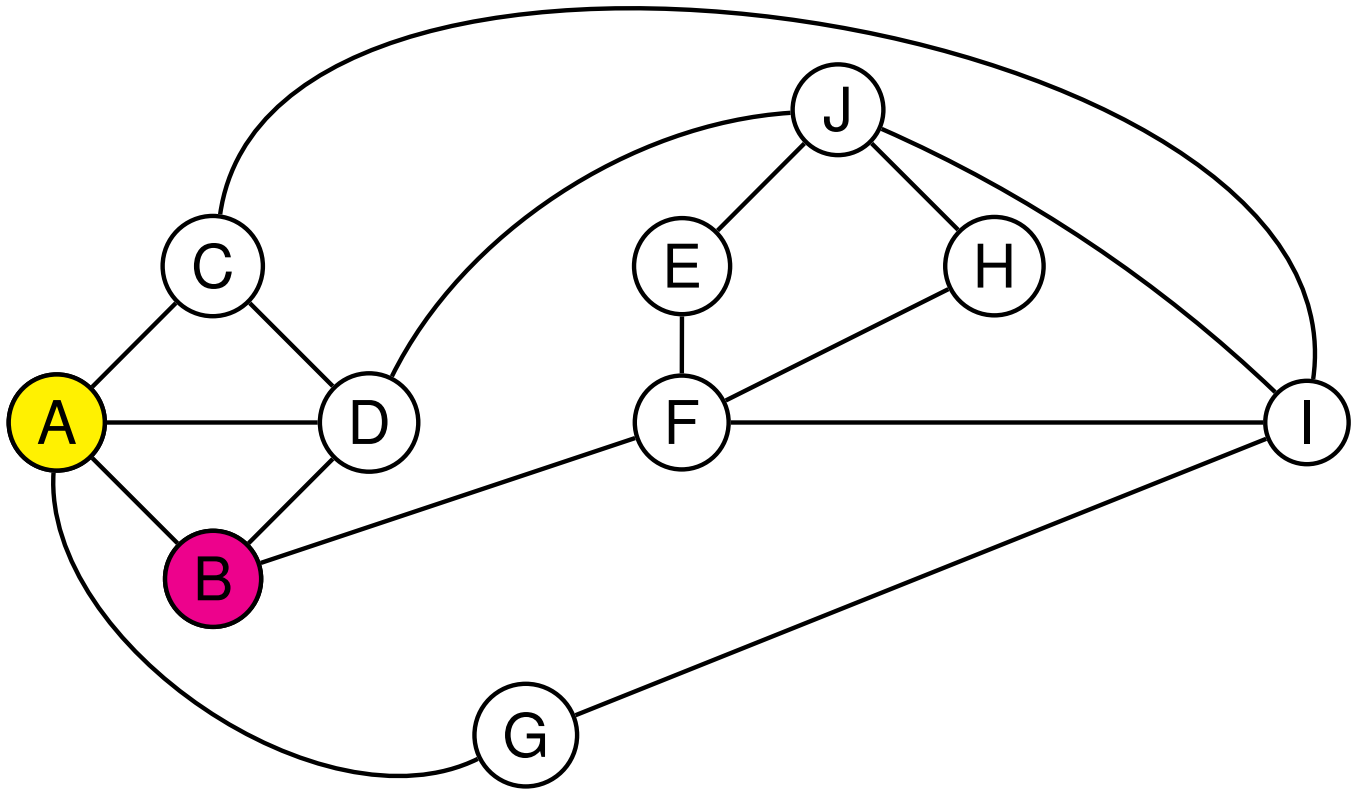
3-colorabilité



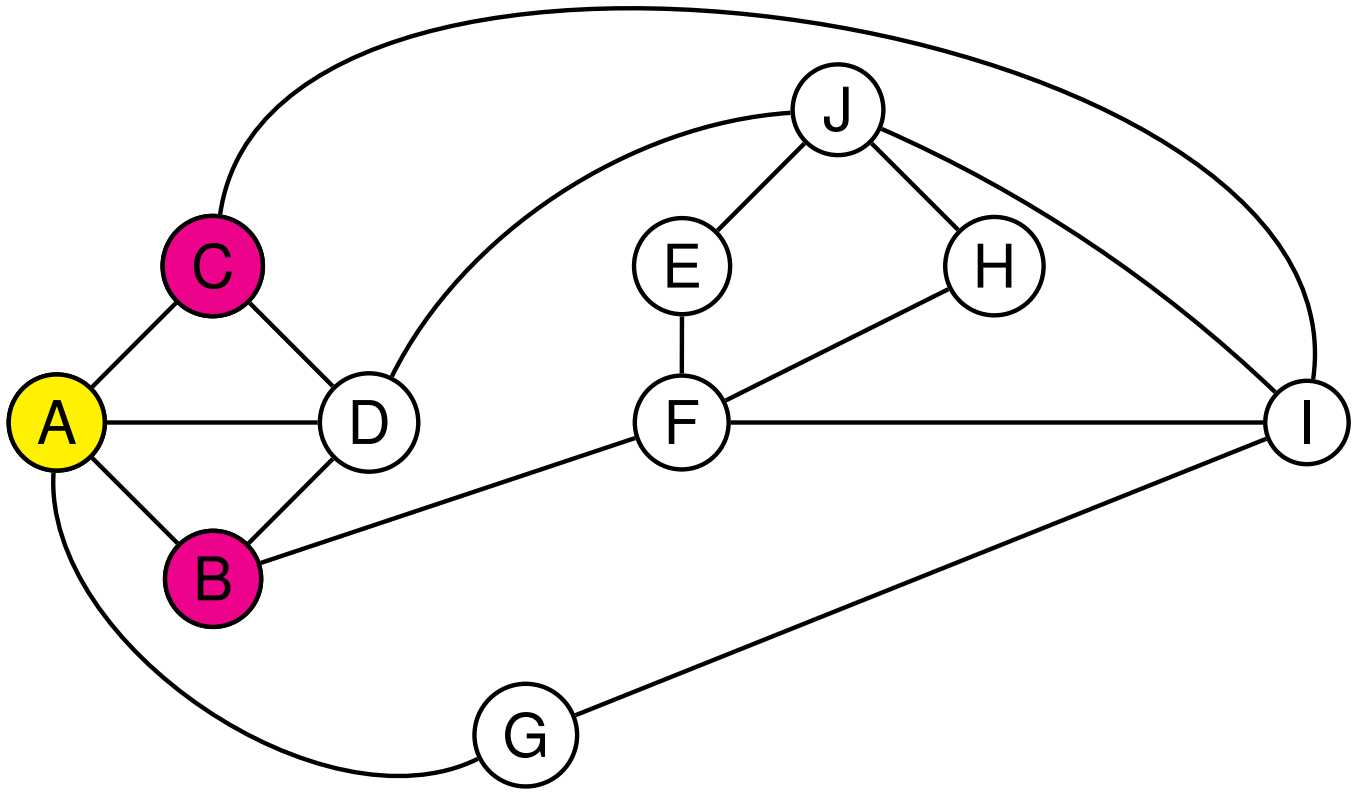
3-colorabilité



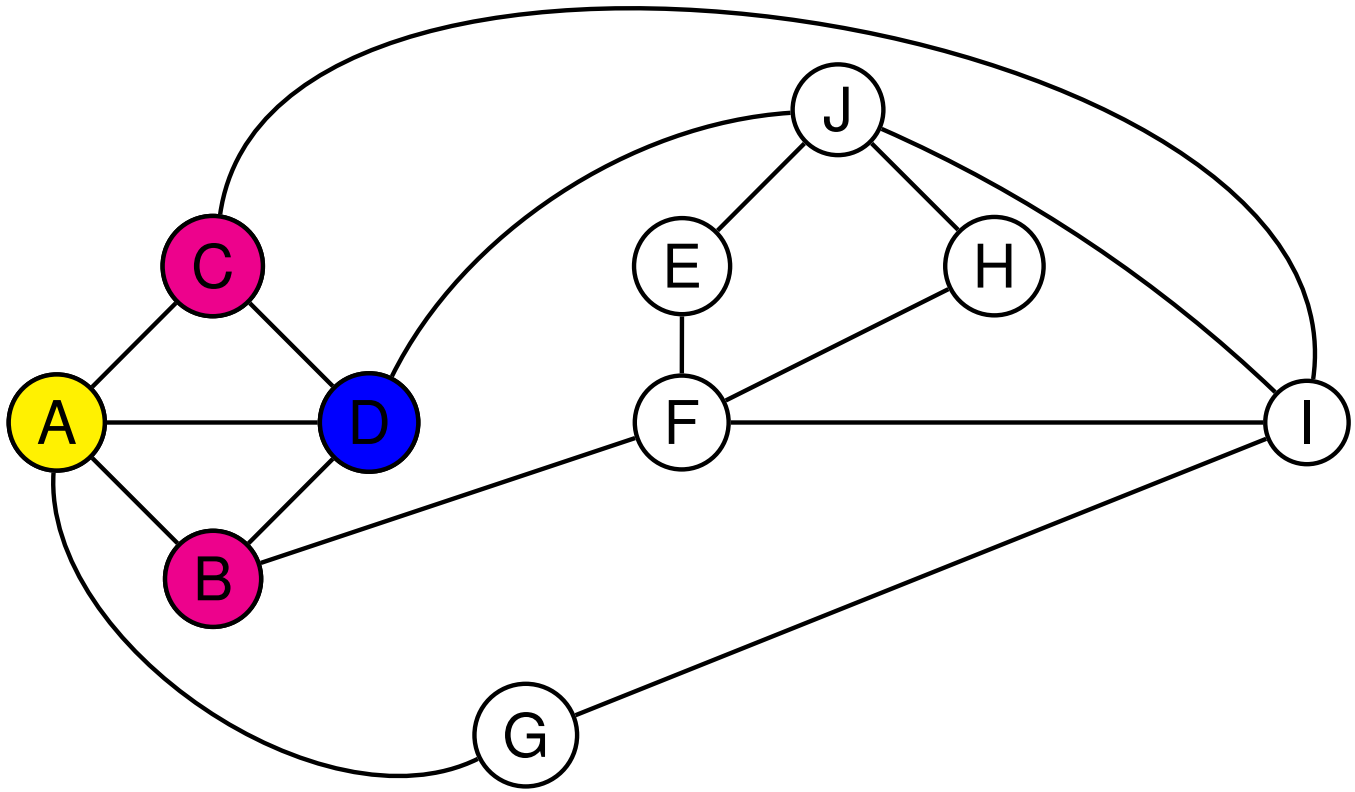
3-colorabilité



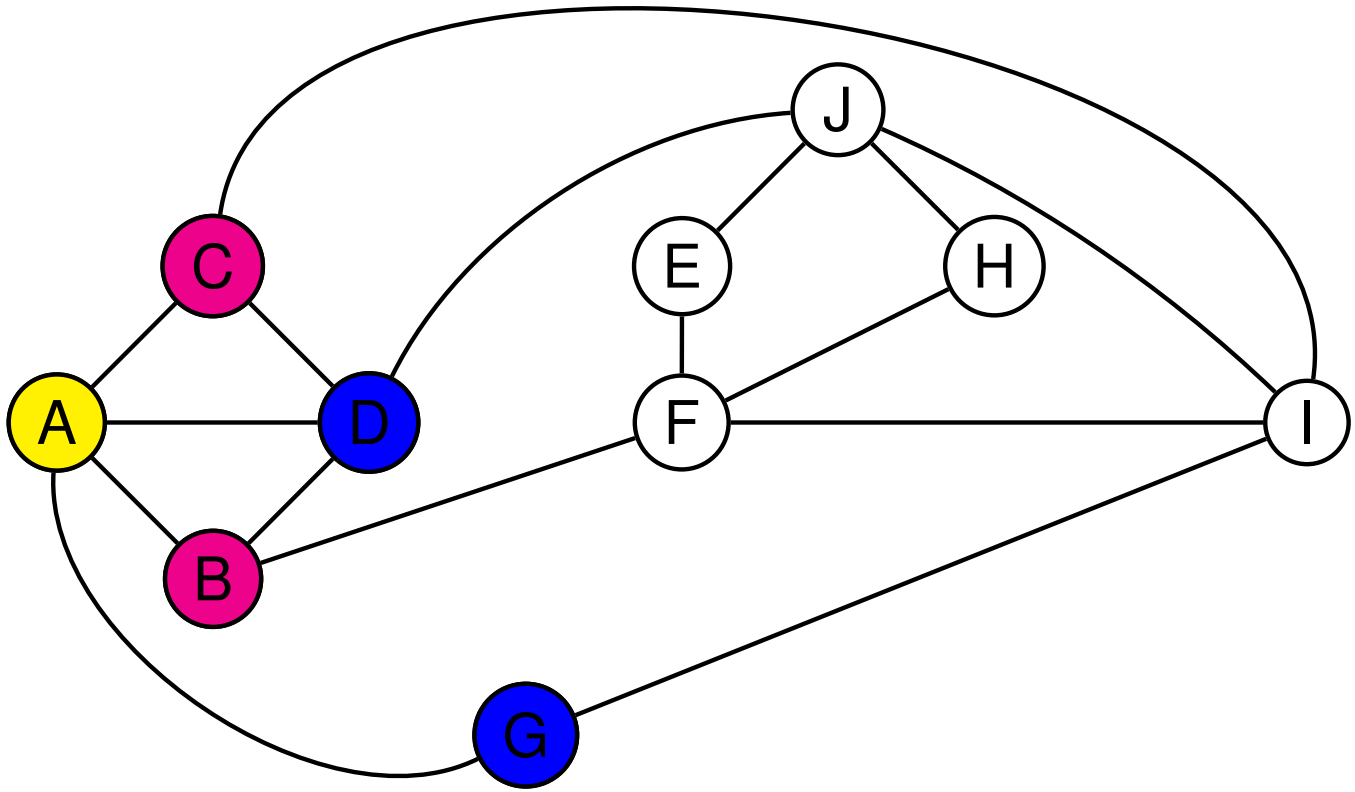
3-colorabilité



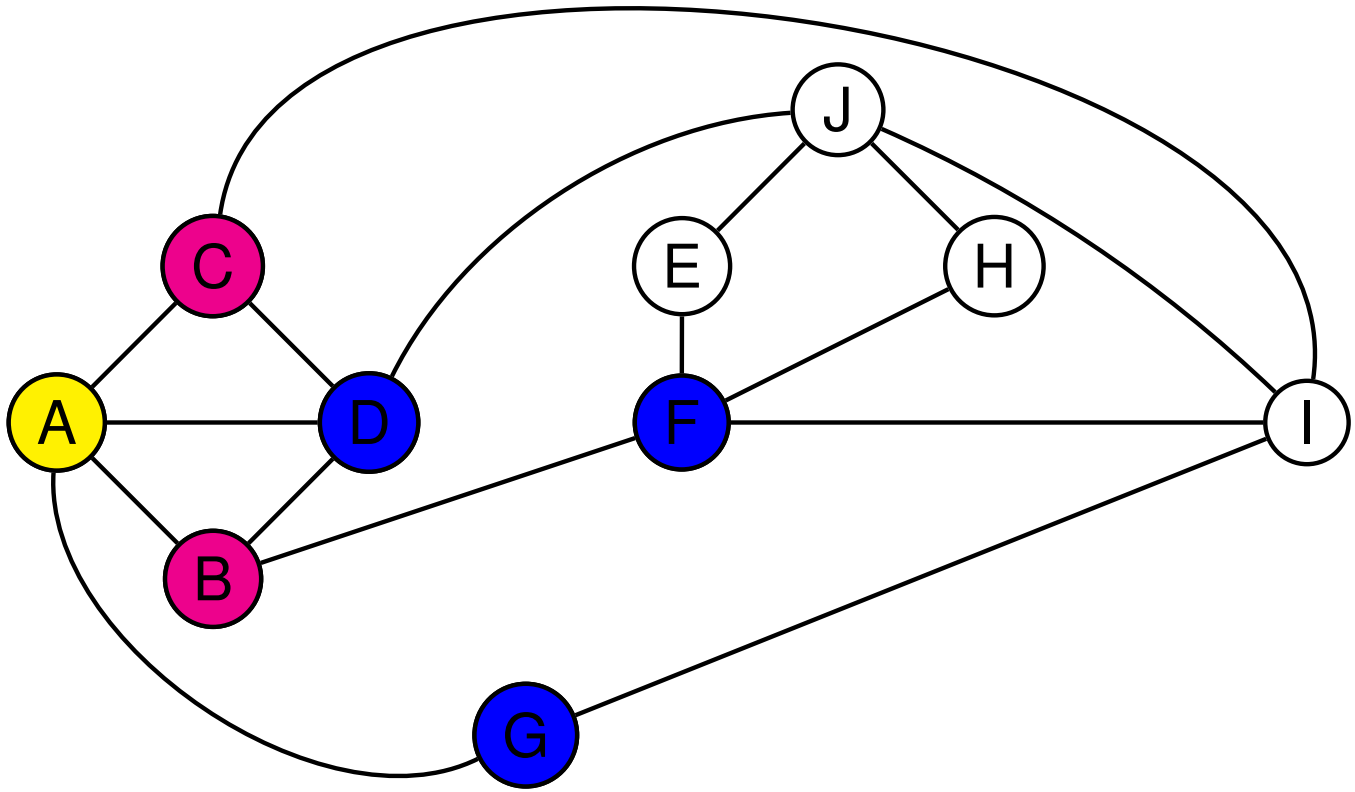
3-colorabilité



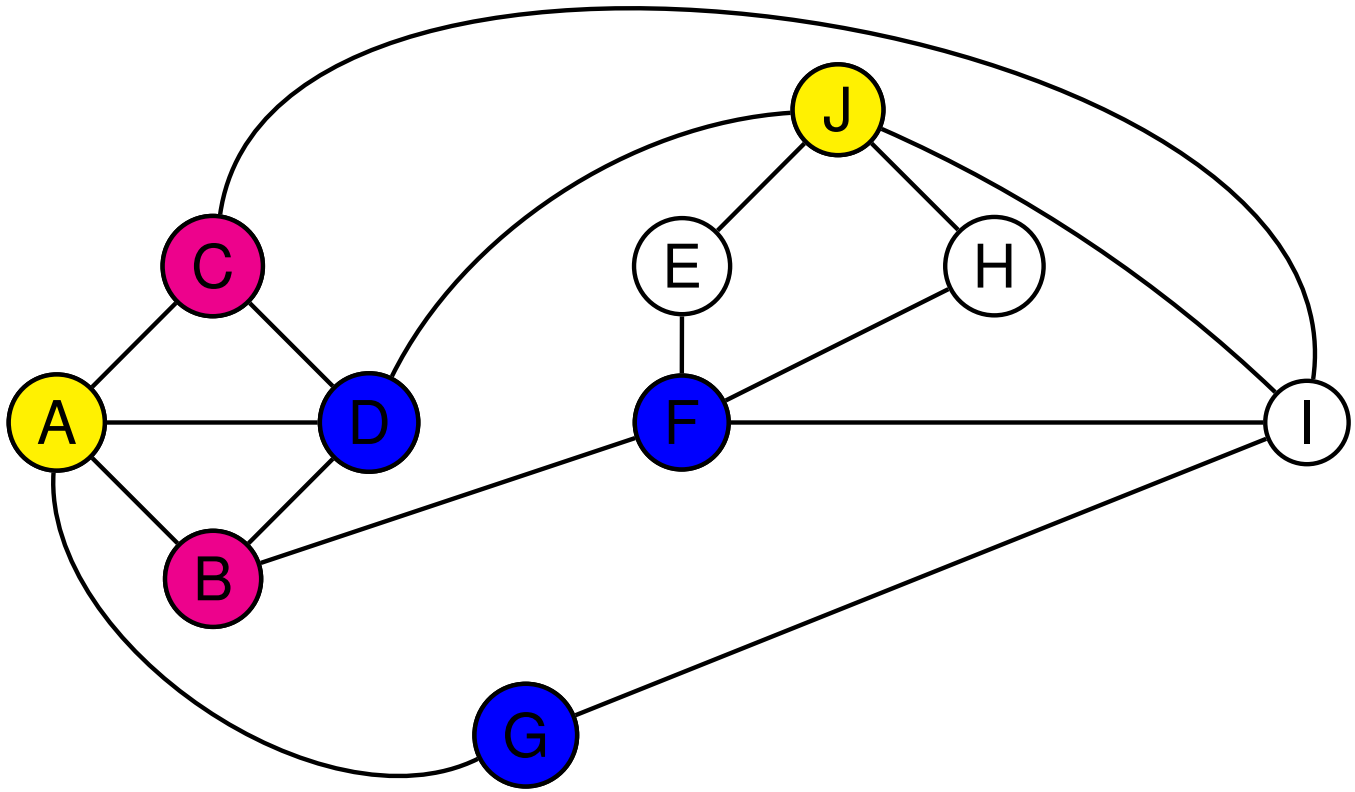
3-colorabilité



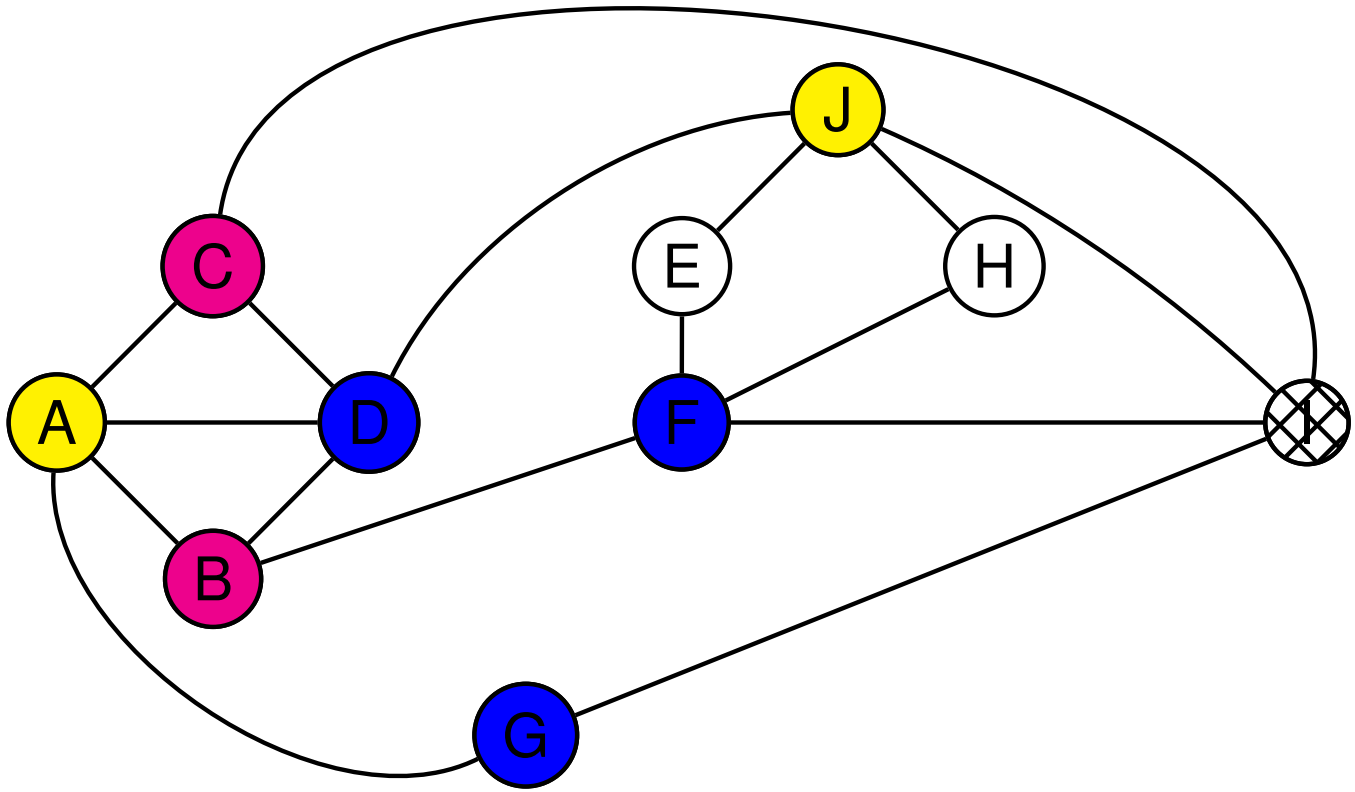
3-colorabilité



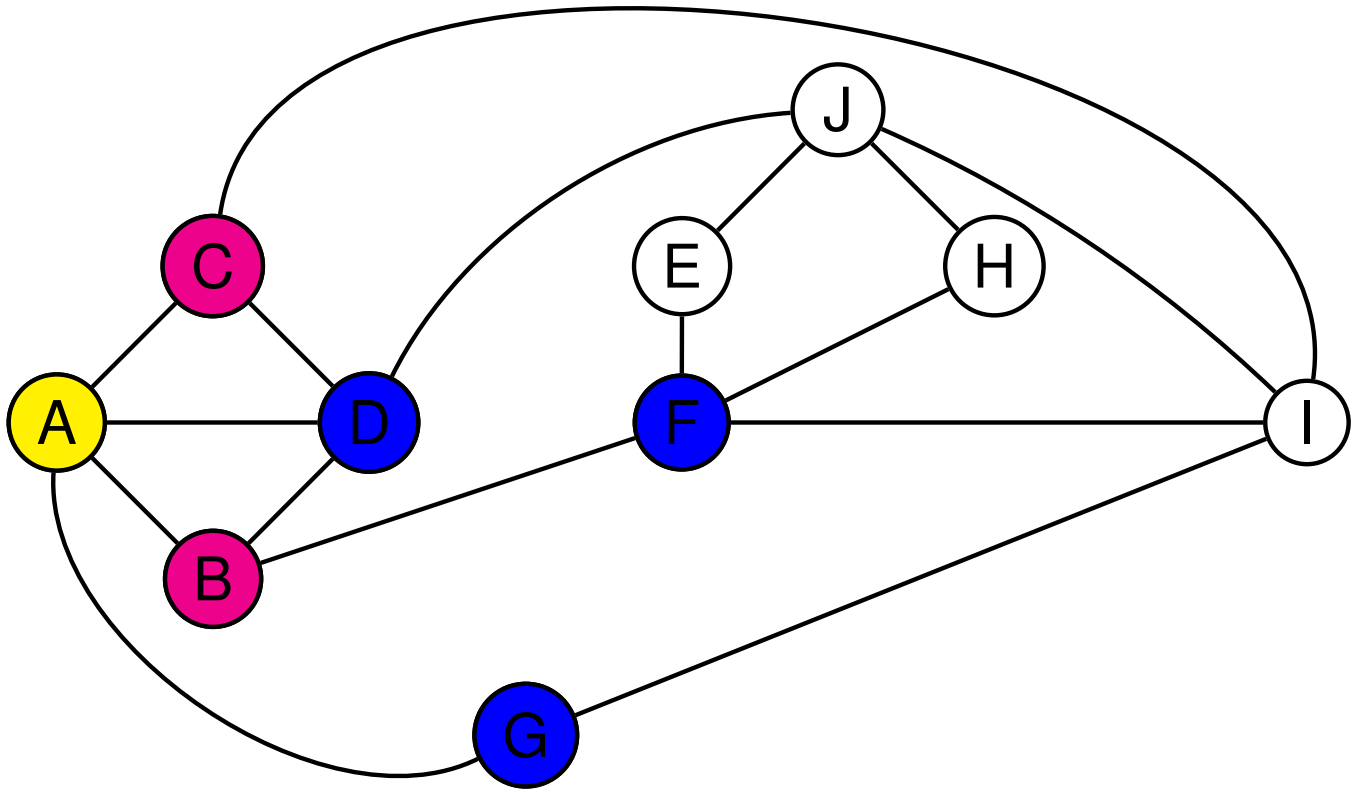
3-colorabilité



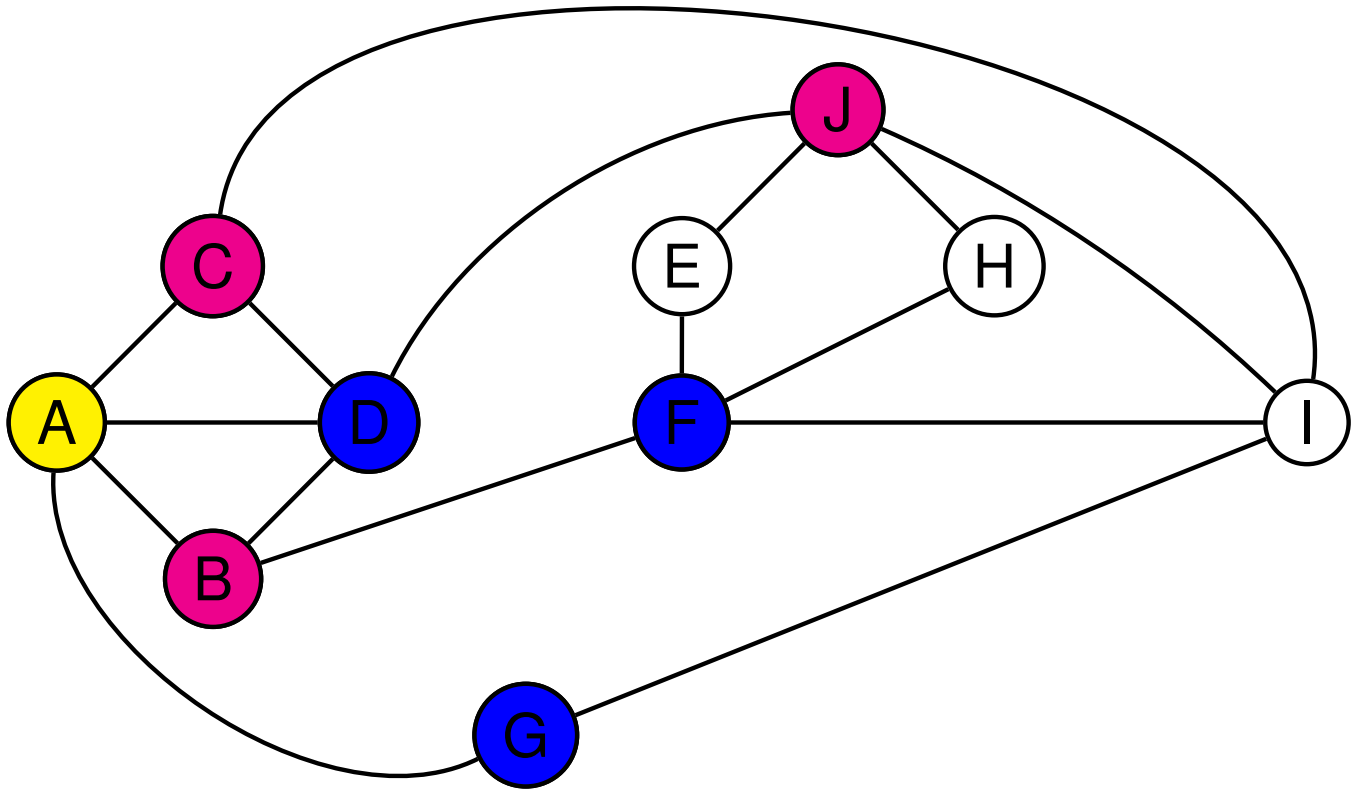
3-colorabilité



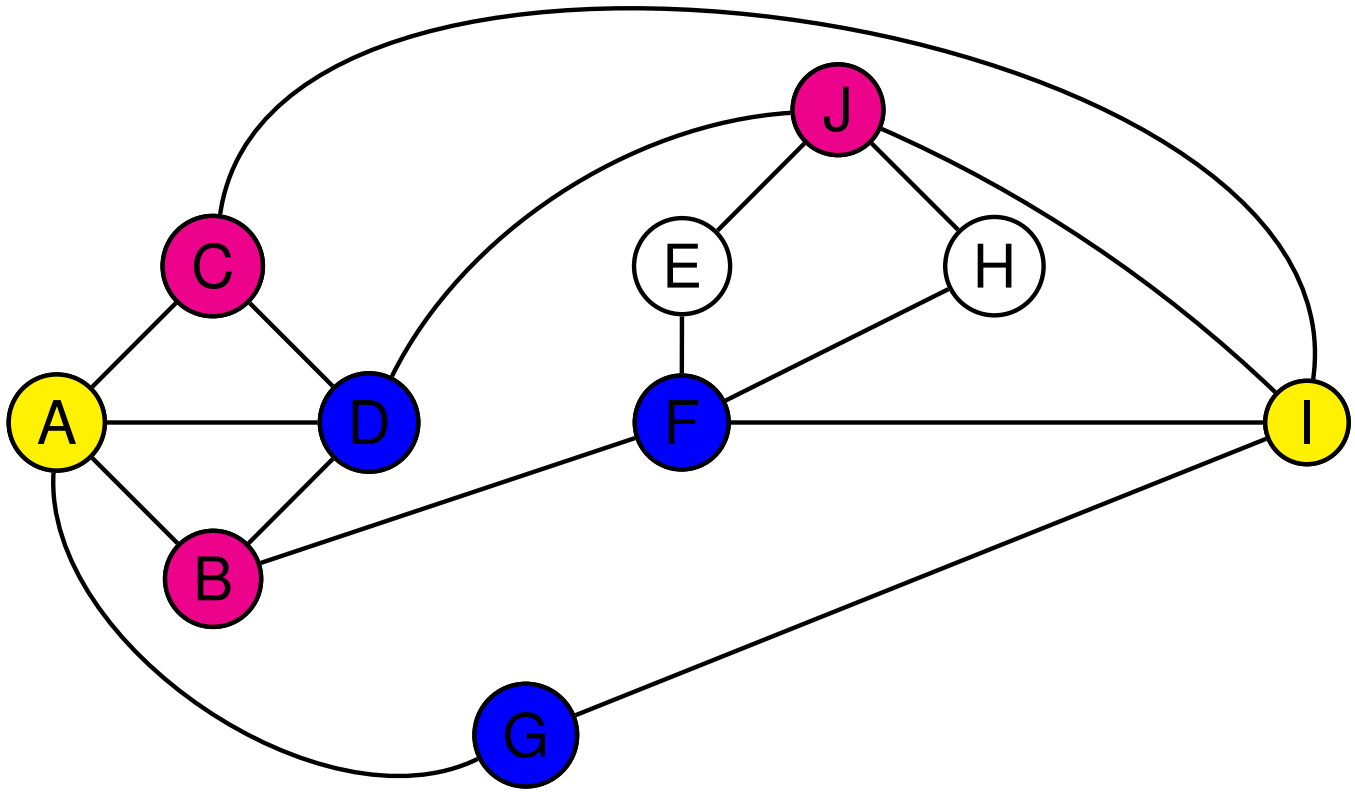
3-colorabilité



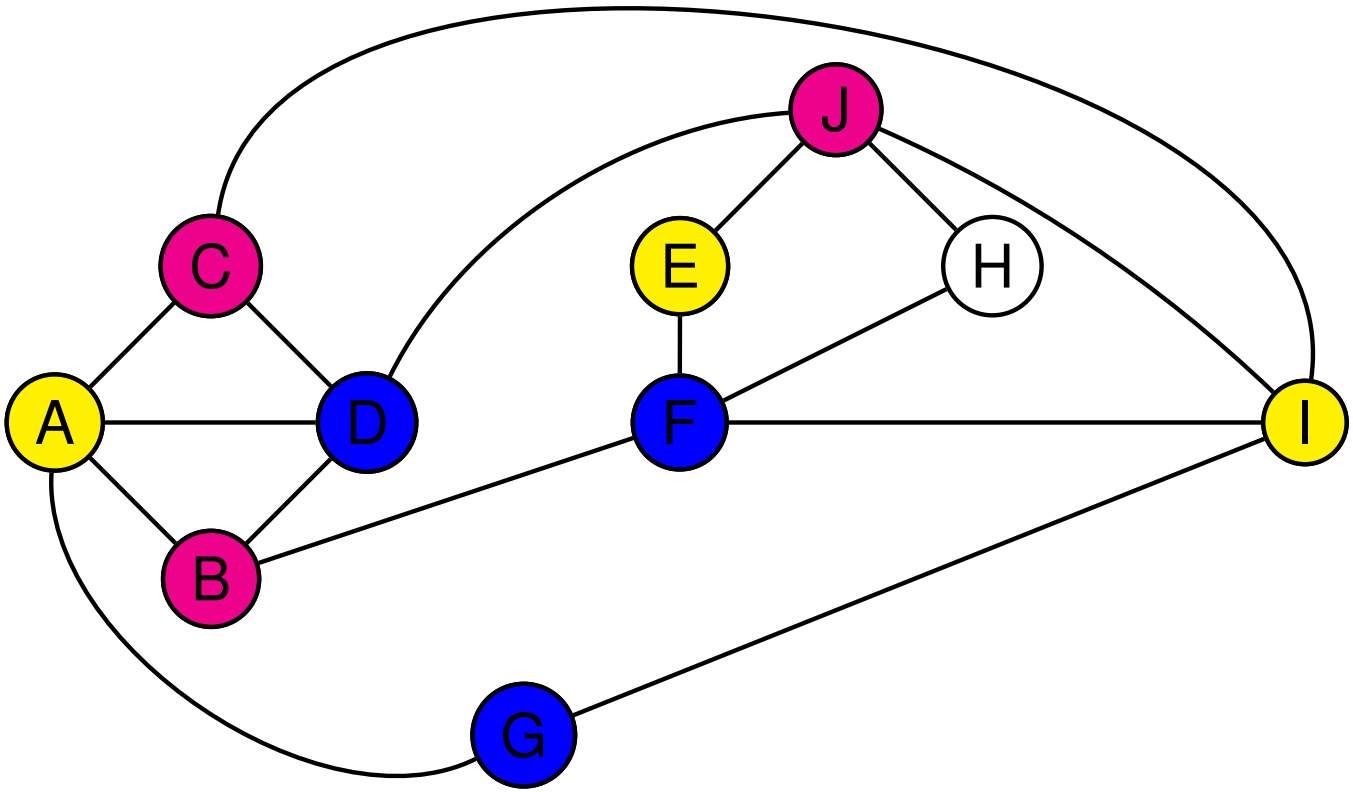
3-colorabilité



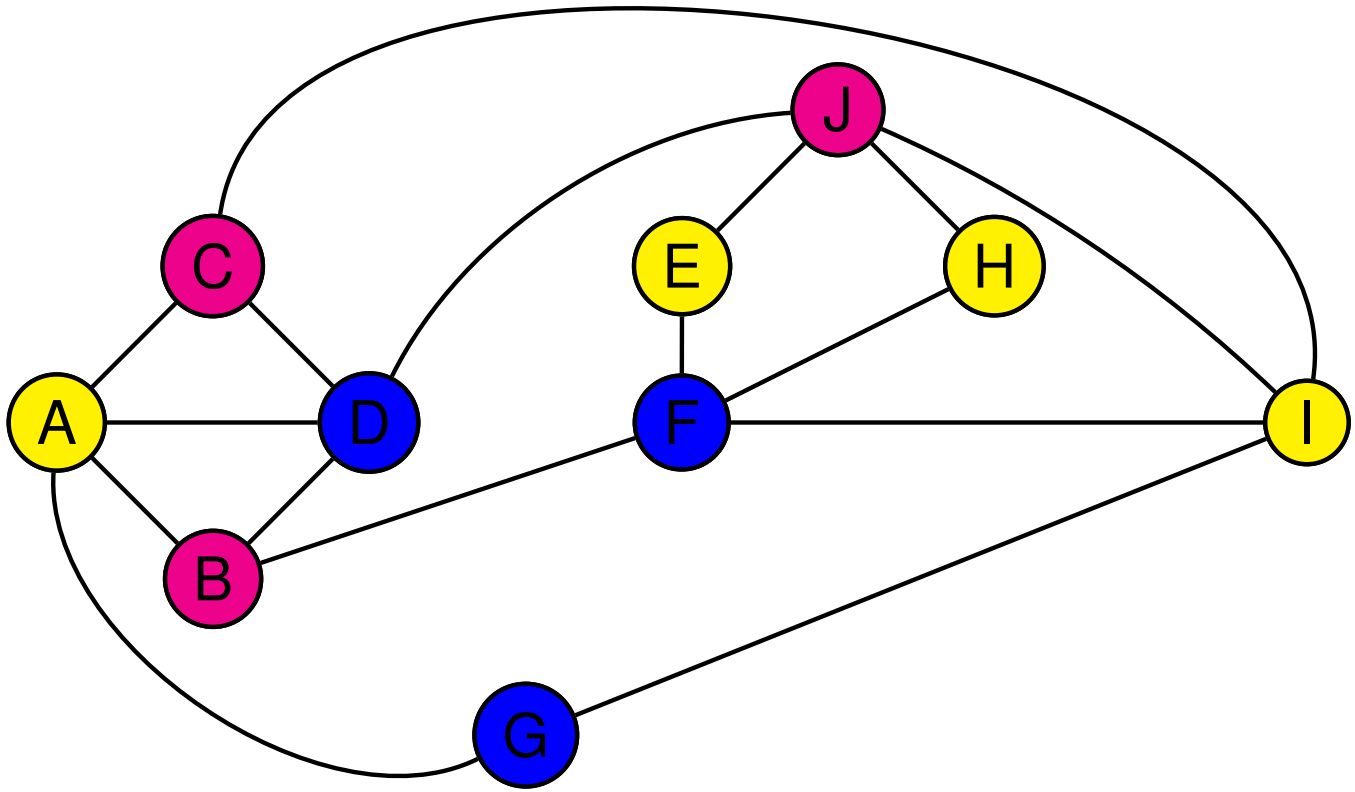
3-colorabilité



3-colorabilité

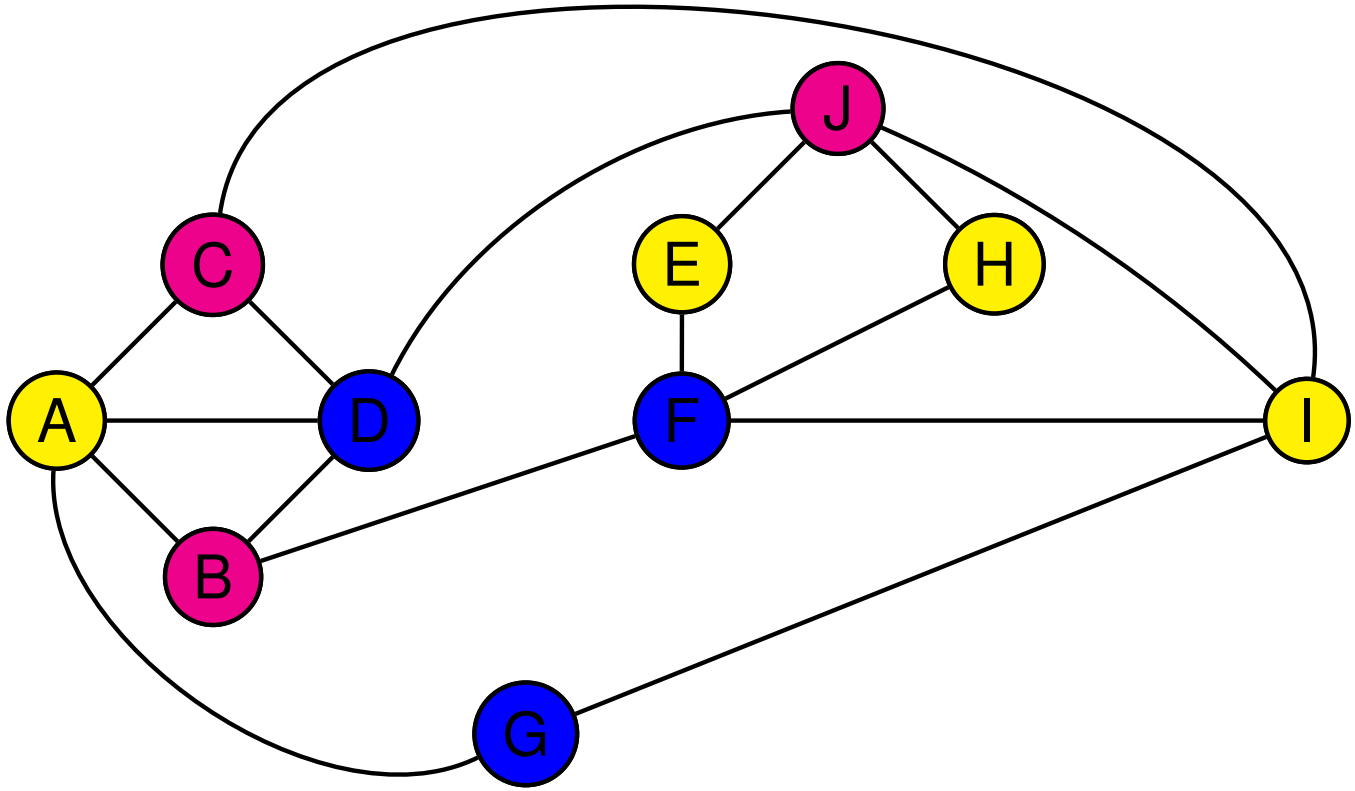


3-colorabilité



3-colorabilité : bilan

A nouveau quelques retours en arrière. Tester toutes les possibilités : 3^n où n est le nombre de sommets. Pas d'algorithme connu avec un nombre polynomial d'étapes.



Polynomial vs exponentiel

Quelle différence ?

n	10	100	1000
n^2	100	10^4	10^6
n^5	10^5	10^{10}	10^{15}
2^n	2^{10}	$> 10^{25}$	$> 10^{250}$

PC actuels : entre 1 et 4 milliards d'opérations par seconde

Machines les plus rapides (-> Petaflops) : 10^{15} (1 million de milliards d'opérations par seconde).

Nombre d'atomes dans l'univers observable : 10^{80}

Problèmes artificiels ou pratiques ?

Problématique naturelle en mathématiques, mais aussi..

- Bases de données
- Sécurité (cryptographie...)
- Algorithmique des réseaux
- Ordonnancement (planification de tâches)
- Optimisation
- Économie (systèmes électoraux, équilibres)
- Jeux
- Intelligence artificielle (raisonnement, apprentissage)
- Bio-informatique

Le temps polynomial

Le temps polynomial **P**

Un problème de décision *A* est dans **P** s'il est résoluble (i.e. décidable) par un algorithme polynomial.

Proposition comme classe de problèmes faciles ou raisonnables : Cobham (1964), Edmonds (1965), Rabin (1966)

Le temps polynomial

Le temps polynomial **P**

Un problème de décision A est dans **P** s'il est résoluble (i.e. décidable) par un algorithme polynomial.

Proposition comme classe de problèmes faciles ou raisonnables : Cobham (1964), Edmonds (1965), Rabin (1966)

Attention, **P** $\neq$ "facile" de plusieurs points de vue

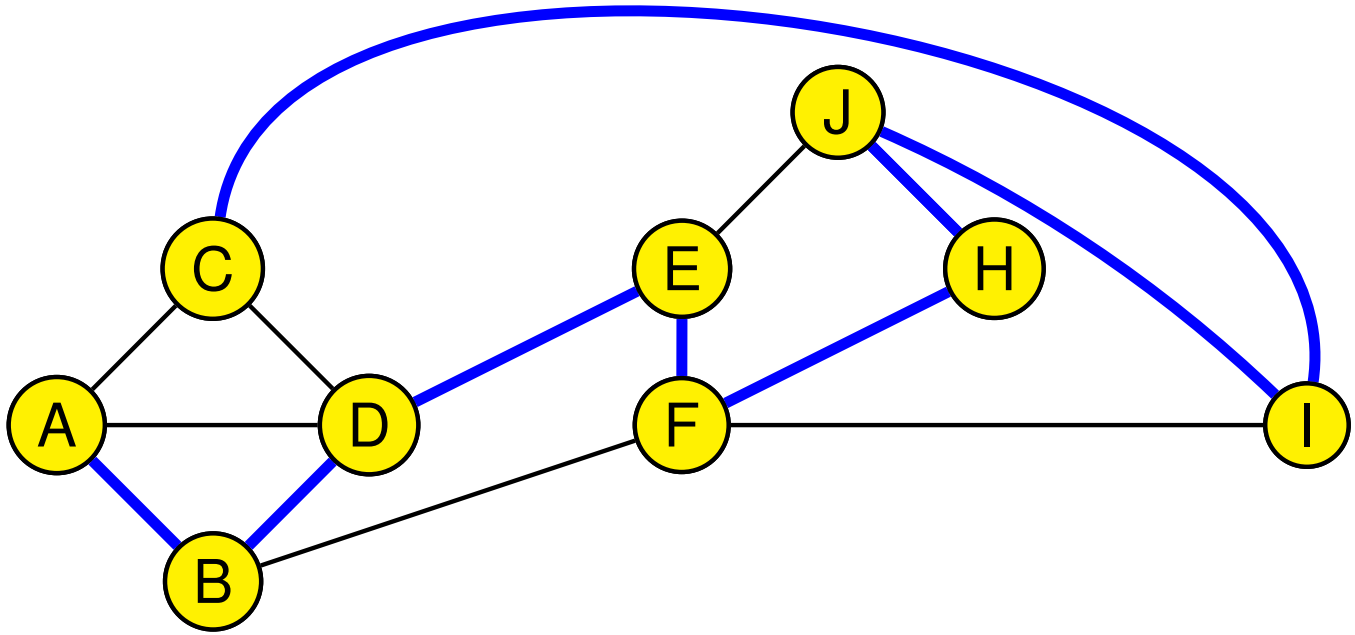
- n^{100} n'est pas une complexité "acceptable".
- Certains algorithmes polynomiaux ont été très durs... à trouver (et à comprendre).

Trouver un témoin... et vite

Les problèmes “difficiles” que l’on a vu sont très particuliers :
on peut vérifier facilement qu’un candidat est une solution !

Vérification d’une solution

Si on a $G = (V, E)$ et un circuit $f : \{1, .., n\} \longrightarrow V$, déterminer si f décrit un circuit hamiltonien est facile...



Trouver un témoin... et vite

Situation fréquente et naturelle...

k-colorabilité

Soient $G = (V, E)$ et un coloriage des sommets $c : V \longrightarrow \{1, ..., k\}$, déterminer si c décrit un “bon” k-coloriage se fait en temps polynomial.

Pareil pour voyageur de commerce, circuit eulérien, etc.

Non primalité

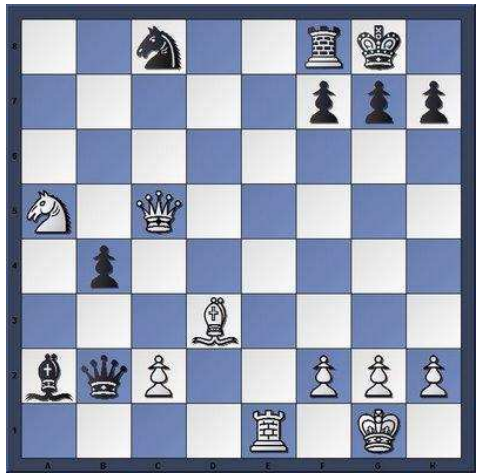
Soient trois entiers n, p, q, déterminer si $p \neq 1, n$ et $p \times q = n$ se fait en temps polynomial...

Trouver un témoin... et vite

Remarque

Tous les problèmes "difficiles" (ici, exponentiels) n'ont pas forcément cette propriété

Par exemple, déterminer l'existence d'une stratégie gagnante dans certains jeux classiques ou leur généralisation (les échecs, le Go, ...).



Le temps polynomial non déterministe **NP**

Vérification en temps polynomial

Un problème B est vérifiable en temps polynomial si pour toute instance x de ce problème :

- si $x \in B$, il existe un témoin y (de taille polynomiale en la taille de x) permettant de décider en temps polynomial que $x \in B$.
- si $x \notin B$, il n'existe pas de tel témoin

Le temps polynomial non déterministe **NP**

Un problème de décision B est dans **NP** s'il est vérifiable en temps polynomial

Formalisé par Cook (1971) et Levin (1973).
Notions similaires : Edmonds (1966) mais aussi Gödel (1956 - Lettre à Von Neumann).

$\mathbf{P} = \mathbf{NP}$: un problème de témoin

$\mathbf{P} = \mathbf{NP}$?

Les problèmes vérifiables en temps polynomial sont-ils aussi décidables en temps polynomial ? Autrement dit, existe-t'il des problèmes dont les solutions sont faciles à vérifier mais sont dures à trouver ?

Montrer $\mathbf{P} = \mathbf{NP}$: montrer que tous les problèmes de **NP** sont dans **P**.

Réductibilité : un seul pour les représenter tous

Certains problèmes dans **NP** sont représentatifs de la difficulté de la classe. Ils sont dits **NP**-complets

Outil principal : algorithme polynomial pour "réduire" un problème à un autre, une réduction.

Cook (1971), Levin (1973)

Le problème *SAT* (satisfaisabilité d'une formule propositionnelle) est **NP**-complet

Depuis lors, plusieurs centaines de problèmes prouvés *NP*-complets (depuis le travail de Karp en 1972)...

Un exemple de réduction

Dans les graphes

Recouvrement de sommets minimal

sous-ensemble S de taille minimale tel que toute arête a un de ses sommets dans S.

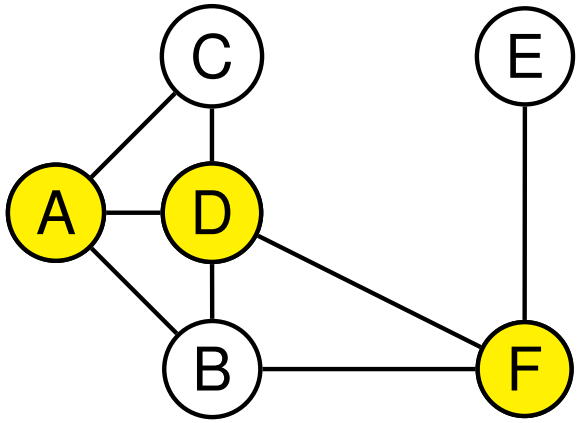
Clique maximale

sous-ensemble T de taille maximale tel que toute paire de sommets de T est liée par une arête.

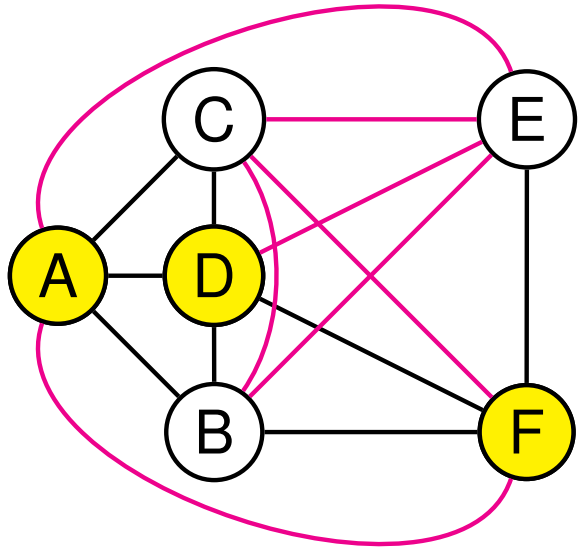
A partir d'un graphe G, construire en temps polynomial un graphe G' tel que

G a un recouvrement de taille minimale ssi
G' a une clique de taille maximale.

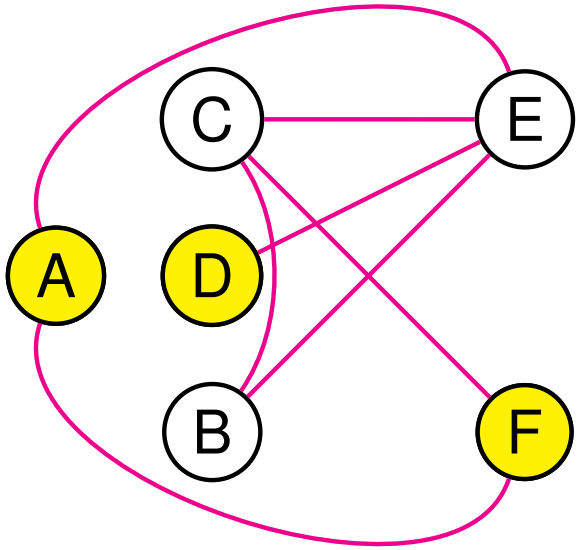
Un exemple de réduction



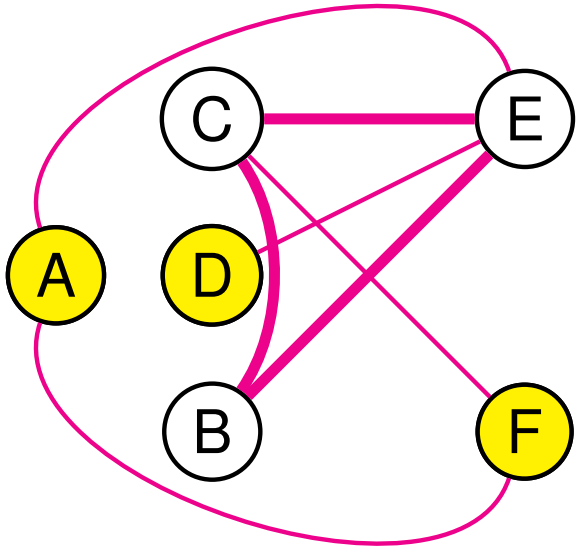
Un exemple de réduction



Un exemple de réduction



Un exemple de réduction



Réduction

Pour trouver un recouvrement minimal dans G, on cherche une clique maximale dans un autre graphe G', "facilement" constructible.

Réductibilité

Dans l'exemple précédent, trouver un recouvrement minimal se réduit à trouver une clique maximale.
Donc trouver une clique maximale est au moins aussi dur que trouver un recouvrement minimal.

Un problème de la classe **NP** est **NP**-complet

si tout problème de **NP** se réduit à lui.

Conséquence

Trouver un algorithme polynomial pour un problème **NP**-complet donnerait automatiquement un algorithme polynomial pour tous les problèmes de **NP**
A priori, plein de manières d'attaquer la conjecture (chacun peut choisir son problème préféré...).

Quelques remarques sur la conjecture

Arguments pour $P \neq NP$:

- Pragmatique : beaucoup de gens d'horizons très différents ont cherché un algorithme polynomial pour un problème **NP**-complet...

Utilisation de **P** $\neq$ **NP** pour la sécurité

Qu'est-ce qu'un schéma cryptographique sûr ?

Sécurité prouvable

- Si un adversaire est capable de casser le schéma cryptographique,
- alors on peut résoudre facilement un problème réputé difficille

Pour prouver la sécurité d'un schéma cryptographique, on a besoin

- d'un modèle formel de sécurité
- de la notion de réduction
- d'hypothèses de complexité raisonnables (problèmes difficilles)
- d'un modèle d'adversaire (limitation de la puissance de calcul)

Quelques approches pour la résolution pratique

1. Heuristique : toutes les instances d'un problème sont loin d'être "uniformément" difficiles
2. Approche probabiliste : trouver une solution avec une très grande probabilité
3. Approximation : trouver une solution proche de l'optimum
4. Approximation probabiliste : trouver une solution proche de l'optimum avec une très grande probabilité
5. Famille d'instances facile : Déterminer des familles entières d'instances pour lesquelles "**P** $=$ **NP**"

Quelques références

M.R. Garey et D.S. Johnson

Computers and Intractability : a guide to NP-completeness.
Freeman and Co, 1979

C.H. Papadimitriou

Computational Complexity. Addison Wesley, 1994

R. Lassaigne et M. de Rougemont

Logic and Complexity. Springer-Verlag, 2004