Algorithmes efficaces et Complexité

Du Page Rank au problème **P** = **NP**

Richard Lassaigne

IMJ/Logique mathématique
CNRS/Université Paris Diderot

04/05/2015

Quelques questions

- Qu'est-ce qu'un algorithme efficace ?
- Existe-t-il des problèmes "intrinsèquement" difficiles ?
- Comment mesurer la complexité d'un problème ?
- Comment comparer les problèmes entre eux ?
- L'existence de problèmes "intrinsèquement" difficiles peut-elle être un atout ?

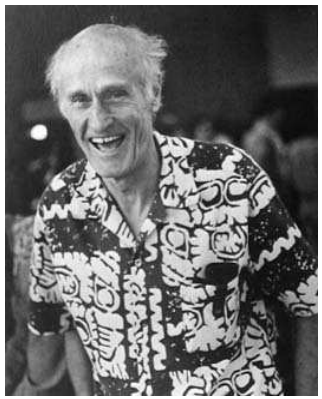
Quelques sujets non traités ici

Début des années 1930 :

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

Alan Turing

A. Church



S. C. Kleene

Kurt Gödel

- Mise en évidence des limites : il existe des fonctions non calculables et des problèmes indécidables !

Quelques réponses

- Un exemple d'algorithme efficace
- Les mesures de complexité (le temps, l'espace)
- Un exemple de problème "intrinsèquement" difficile
- Le problème **P** $=$ **NP** et la notion de réduction
- La sécurité prouvable

Quelques exemples d'algorithmes "efficaces"

- L'interrogation de votre base de données préférée (train, avion, voyages,...)
- La localisation à l'aide de votre GPS
- La recherche du plus court chemin jusqu'à votre lieu de vacances
- La recherche d'informations dans votre moteur préféré

Exemple choisi :
Le calcul du Page Rank dans un moteur de recherches

Fonctionnement d'un moteur de recherches

Données : une question (chaîne de caractères)
Résultat : une liste ordonnée d'URL associées à la question

1. Extraction du contenu de la question (i.e. quelques mots clés)
2. Recherche des pages WEB qui contiennent ces mots clés
3. Tri et affichage d'une liste ordonnée d'URL

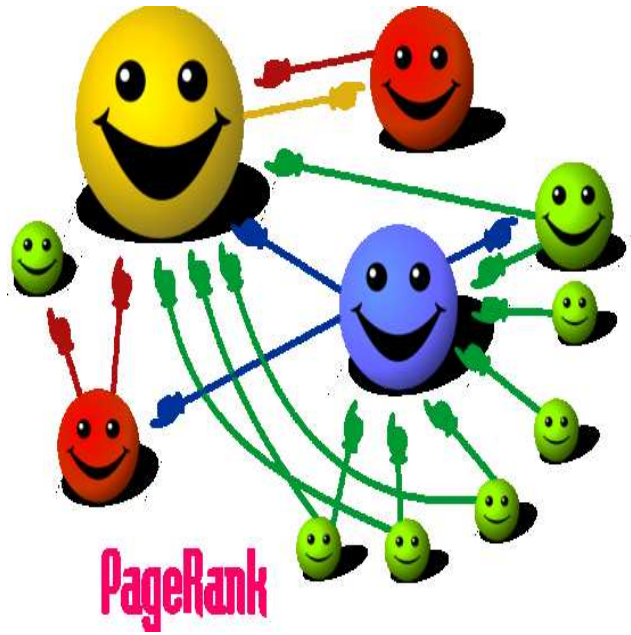
- Un graphe gigantesque
- Une base de données gigantesque
- Des algorithmes efficaces

Principe de l'algorithme de Page Rank

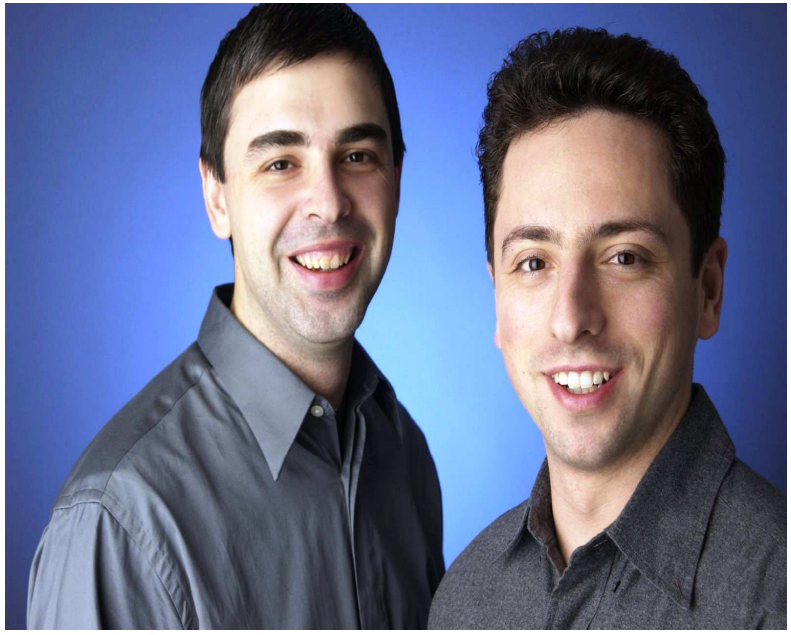
Idée :

- Eviter à tout prix la sémantique
- L'importance d'une page ne dépend que de la structure des liens entre les pages html
- Formulation récursive de l'importance d'une page

...And that, in simple terms, is how you increase your ranking on search engines."



Principe de l'algorithme de Page Rank



S. Brin, L. Page, R. Motwani, T. Winograd

Une page a un score d'autant plus élevé qu'elle est référencée par des pages ayant un score élevé

Le graphe du WEB

- Graphe orienté
- Les sommets sont les pages html
- Les arcs correspondent aux hyperliens entre ces pages

L'exploration du graphe du WEB est un problème difficille d'informatique distribuée (des programmes appelés robots suivent les liens)

- Une page standard contient au plus une centaine de liens vers d'autres pages (degré sortant borné)
- Graphe peu dense représentable efficacement
- Algorithmes parallélisables

Un modèle linéaire

Une sorte de flot

Soit $R_n(i)$ le coefficient Page Rank de la page i
à l'étape n du calcul et soit $R_{n+1}(j,i)$ la quantité qui traverse l'arc $j \to i$ entre les étapes n et $n+1$

Equation

$R_{n+1}(i) = \sum_{j \to i} R_{n+1}(j,i)$

Equirépartition sur les liens sortant d'une page j

$R_{n+1}(j,i) = \frac{1}{deg(j)} R_n(j)$ pour tout arc sortant de la page j

Conséquence : $R_{n+1}(i) = \sum_{j \to i} \frac{1}{deg(j)} R_n(j)$

Modèle linéaire (suite)

- $R_{n+1} = A^t R_n$ où A est une sorte de matrice d'incidence
- $A(i,j) = \frac{1}{deg(i)}$ s'il existe un arc $i \to j$ et 0 sinon
- $\sum_j A(i,j) = 1$

Interprétation :

- A est une matrice stochastique
- la limite de $R_n(i)$ peut se voir comme la probabilité qu'un surfeur aléatoire visite la page i

Le vecteur R final est la distribution stationnaire d'une marche aléatoire sur le graphe du WEB

Modèle du surfeur aléatoire : le facteur ZAP

- On initialise à $1/N$ le coefficient de PageRank de toutes les pages, où N est le nb total de pages du graphe
- $R_{n+1}(i) = \frac{\alpha}{N} + (1-\alpha)\dot{\sum}_{j \to i} \frac{1}{degre(j)} R_{n+1}(j)$
- Le surfeur peut choisir :
  soit de suivre un lien sortant avec probabilité $(1-\alpha)$
  soit de "zapper" sur une page aléatoire avec probabilité α
- on propose de prendre le facteur ZAP α entre 0.1 et 0.2

Point fixe

Le graphe devient fortement connexe

Transformation $T : \mathbb{R}^N \longrightarrow \mathbb{R}^N, T(x) = \alpha x_0 + (1-\alpha)Ax$
où x_0 est le vecteur dont toutes les composantes valent $1/N$

Point fixe

Si A est un matrice stochastique, alors l'application T est contractante de rapport $(1-\alpha)$ et quel que soit le vecteur initial x_0, la suite $x_{n+1} = T(x_n)$ converge vers un unique point fixe μ

Vitesse de convergence

$| x_{n+1} - \mu | \leq \frac{(1-\alpha)}{\alpha} | x_{n+1} - x_n |$ avec $| y | = \sum_i | y_i |$
Test d'arrêt :
Il suffit de choisir une borne d'approximation ε et de tester
$| x_{n+1} - x_n |$

Pourquoi PageRank est-il tant utilisé ?

- Convergence très rapide
  En pratique une centaine d'itérations avec $\alpha = 0.15$
- Le calcul se parallélise simplement
- Plusieurs interprétations mathématiques intéressantes
- Extrême robustesse expérimentale
  Peu dépendant des conditions initiales (point fixe unique)

Le(s) secret(s) de Google

- Comment calculer en temps quasi-réel le PageRank sur plus de 1000 milliards de pages ?
- Puissance et nombre des datacenters de Google
  Le trafic des DC Google représente 7%
  du trafic sur le web
- Google ne résout pas le problème brutalement
  mais utilise des méthodes approchées
  similaires à des marches aléatoires qui
  simulent le comportement du surfeur

Le PageRank : Pour et Contre

Pour :

- Le calcul se fait offline
- C'est une notion robuste

Contre :

- Le PR favorise les pages recevant beaucoup de liens
- Les thématiques marginales ne ressortent jamais

Le retour de la sémantique :
surf intelligent ou pagerank thématique ?

Le Surf Intelligent

- Idée : modifier le comportement du surfeur aléatoire pour le rendre sensible à la sémantique
- Sauter de page en page de manière probabiliste mais avec des probailités conditionnées par le contenu des pages et des requêtes

Problème : nécessite une puissance de calcul hors de portée

Le PageRank thématique

- Idée : 1 vecteur de PR par page
  (chaque composante correspond à un sujet)
- Granularité plus importante (niveau de la thématique)

Scénario d'utilisation :

- Requête de l'utilisateur
- Recherche des sujets les plus en phase
- Classement des pages par popularité à l'aide
  des vecteurs PR des pages pour ces sujets

Retour sur le fonctionnement d'un moteur de recherches

1. Précalcul d'un fichier inversé des pages WEB, dans une gigantesque Base de Données distribuée
2. Extraction du contenu de la question (mots clés)
3. Recherche de toutes les pages WEB contenant ces mots clés et calcul d'un score pondéré pour chaque page
4. Filtrage des pages résultats à l'aide d'un profil utilisateur (langue, adresse IP, académique versus commercial)
5. Tri des pages obtenues à l'aide de PageRank (plus des astuces secrètes) et affichage de cette liste ordonnée

Algorithme et complexité

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

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

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.

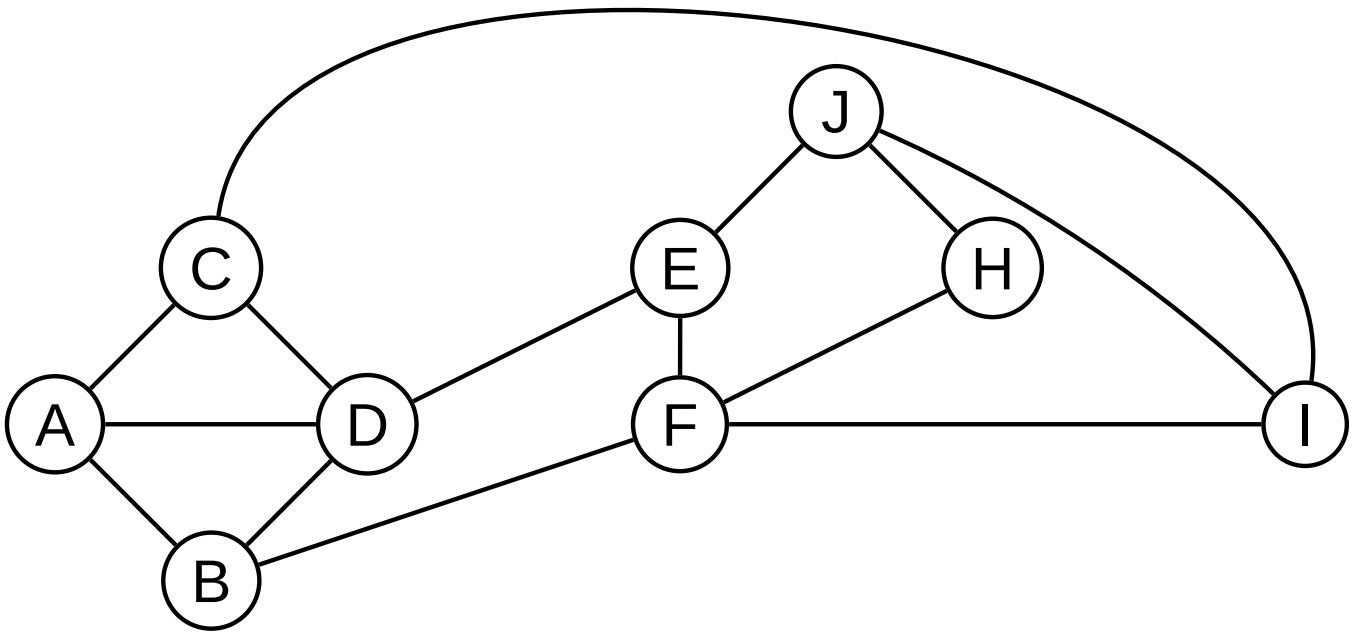
Tout cela reste à définir...

Un exemple : le circuit hamiltonien

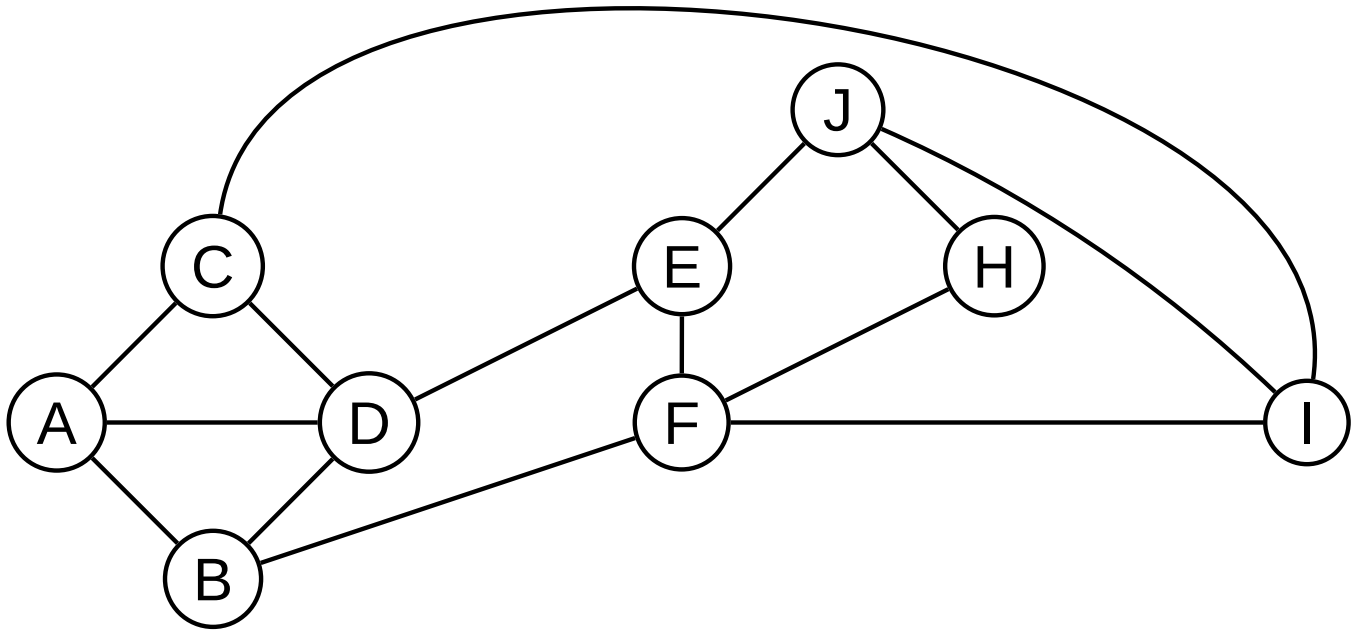
Circuit hamiltonien

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 ?

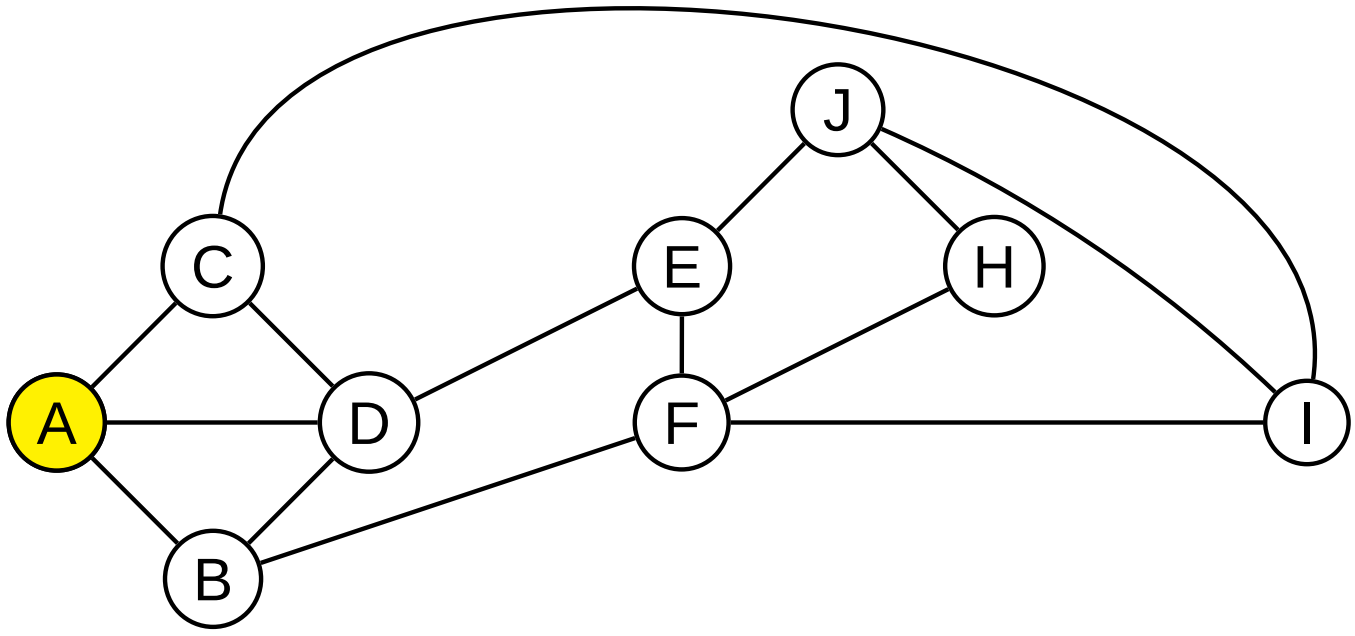
Aussi : voyageur de commerce (graphe valué).



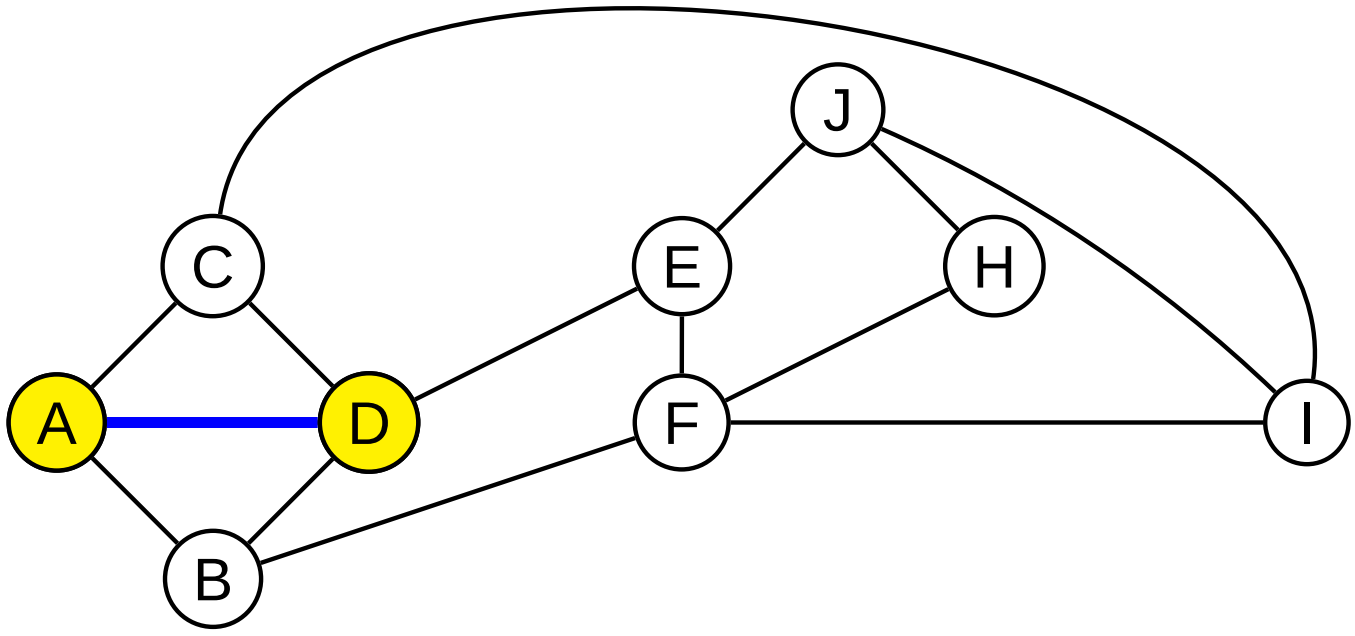
Un exemple : le circuit hamiltonien



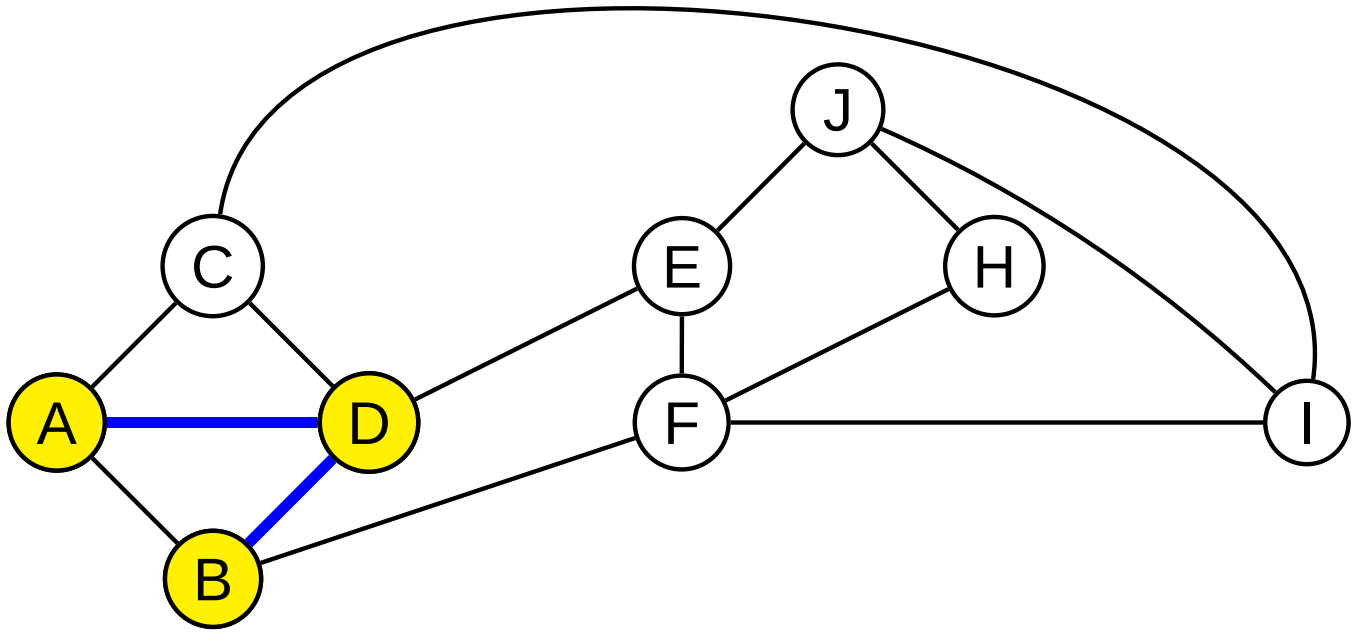
Un exemple : le circuit hamiltonien



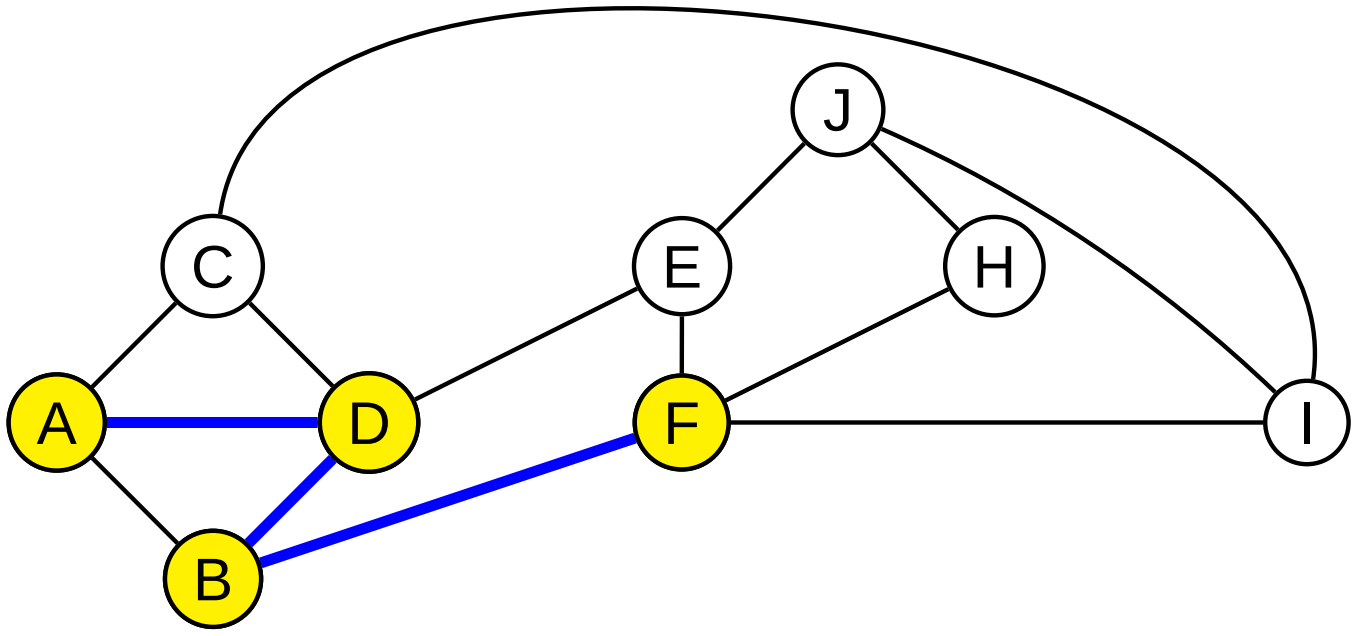
Un exemple : le circuit hamiltonien



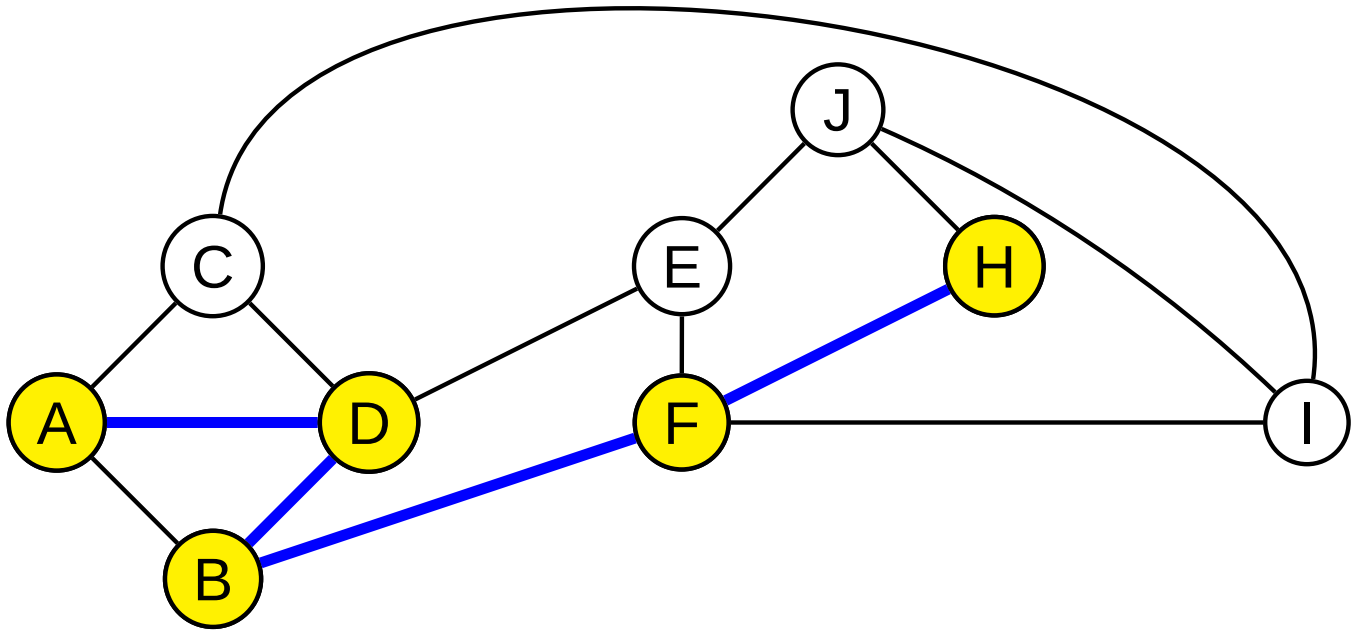
Un exemple : le circuit hamiltonien



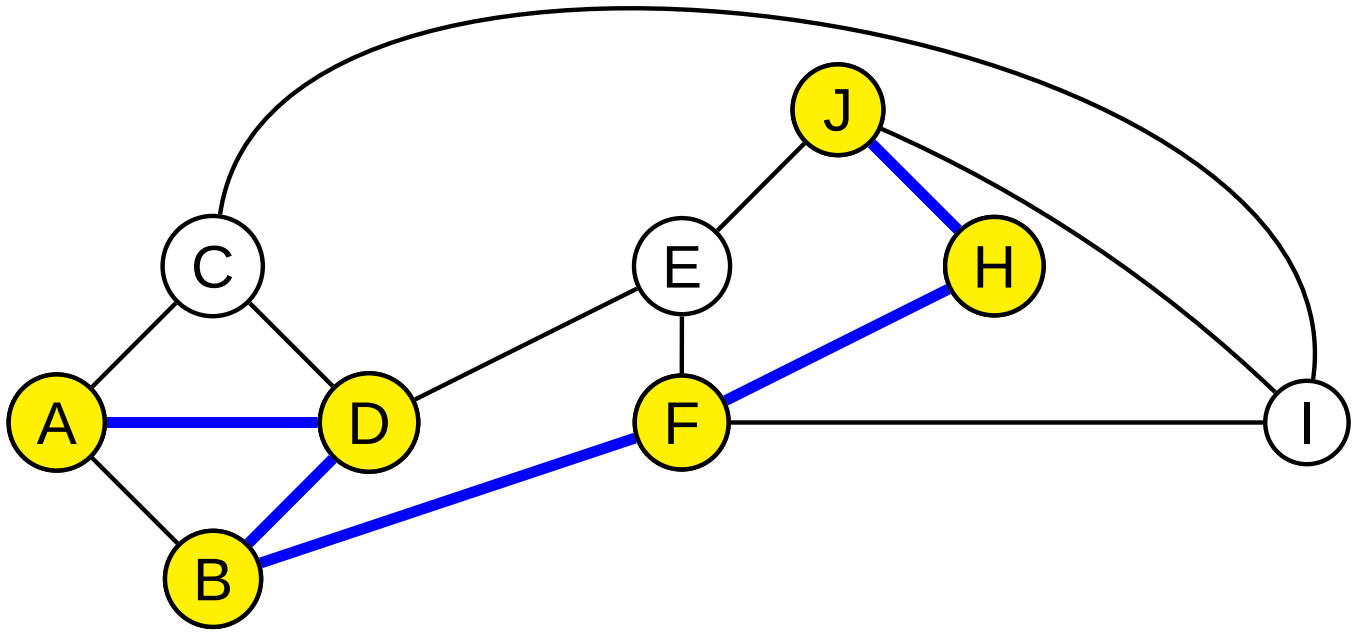
Un exemple : le circuit hamiltonien



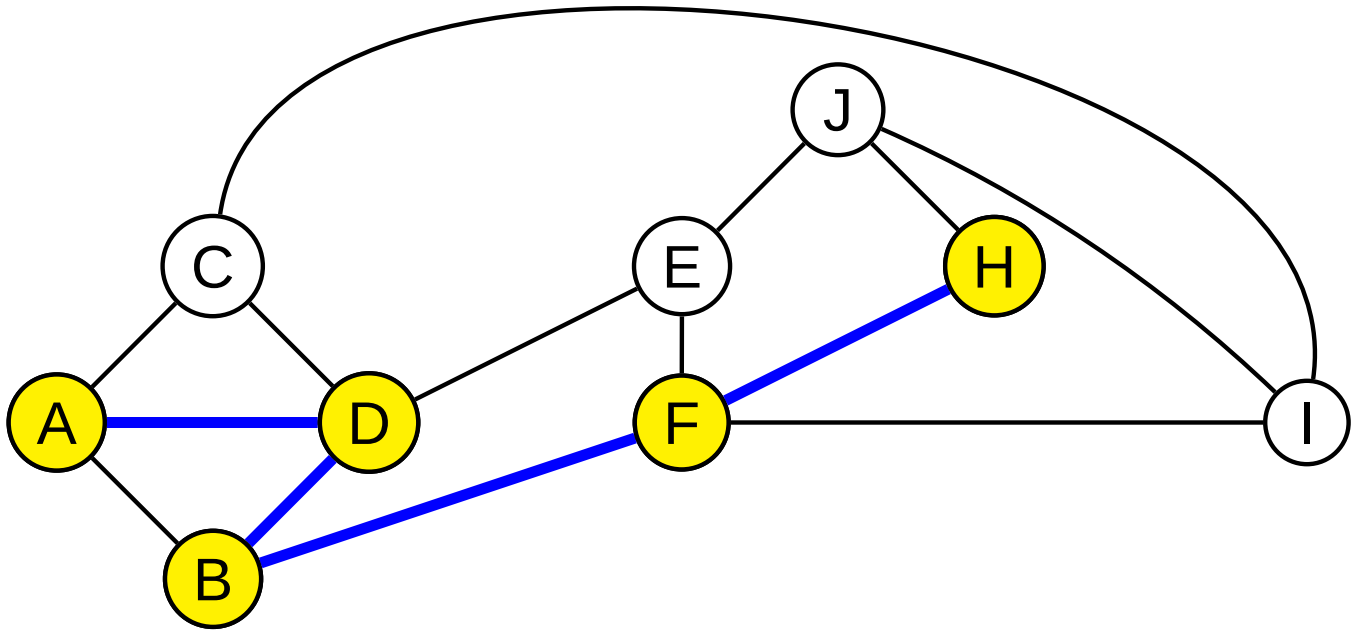
Un exemple : le circuit hamiltonien



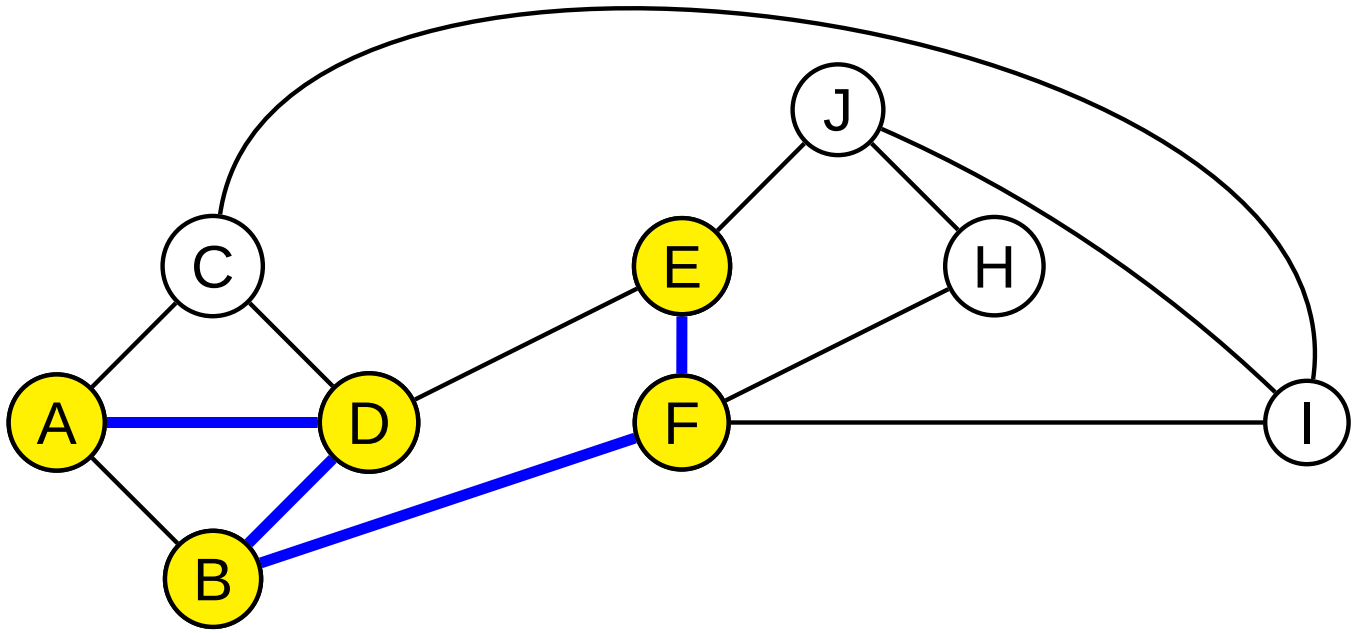
Un exemple : le circuit hamiltonien



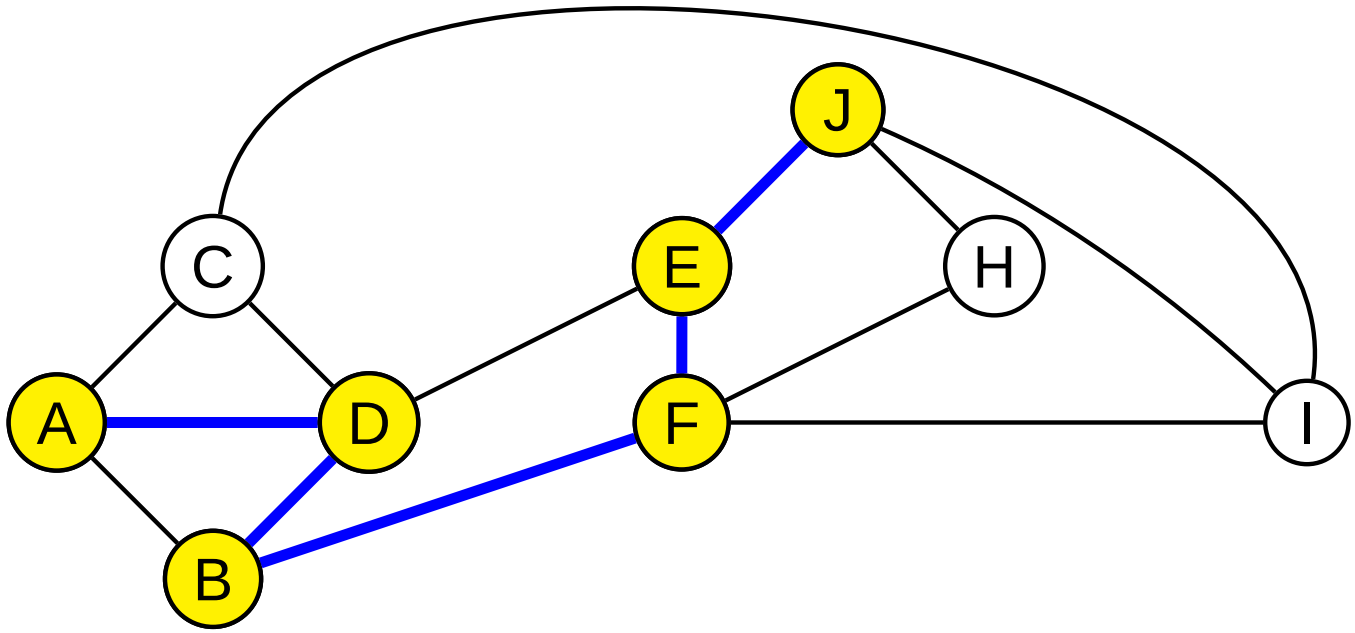
Un exemple : le circuit hamiltonien



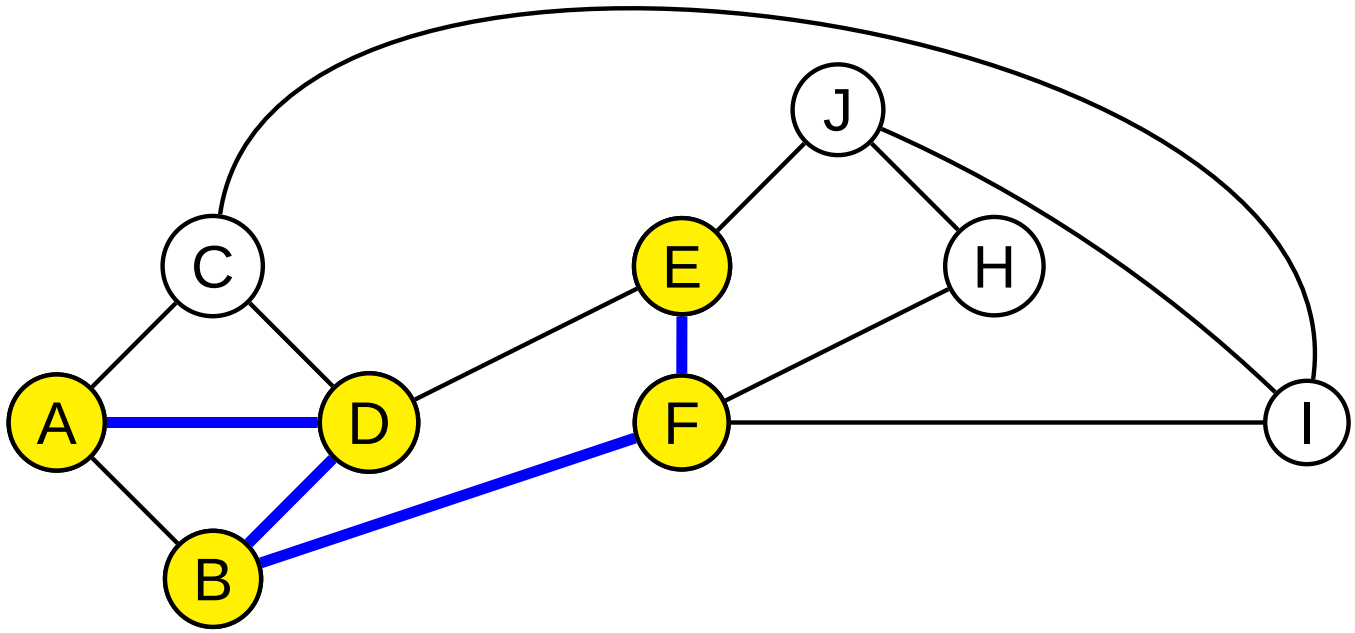
Un exemple : le circuit hamiltonien



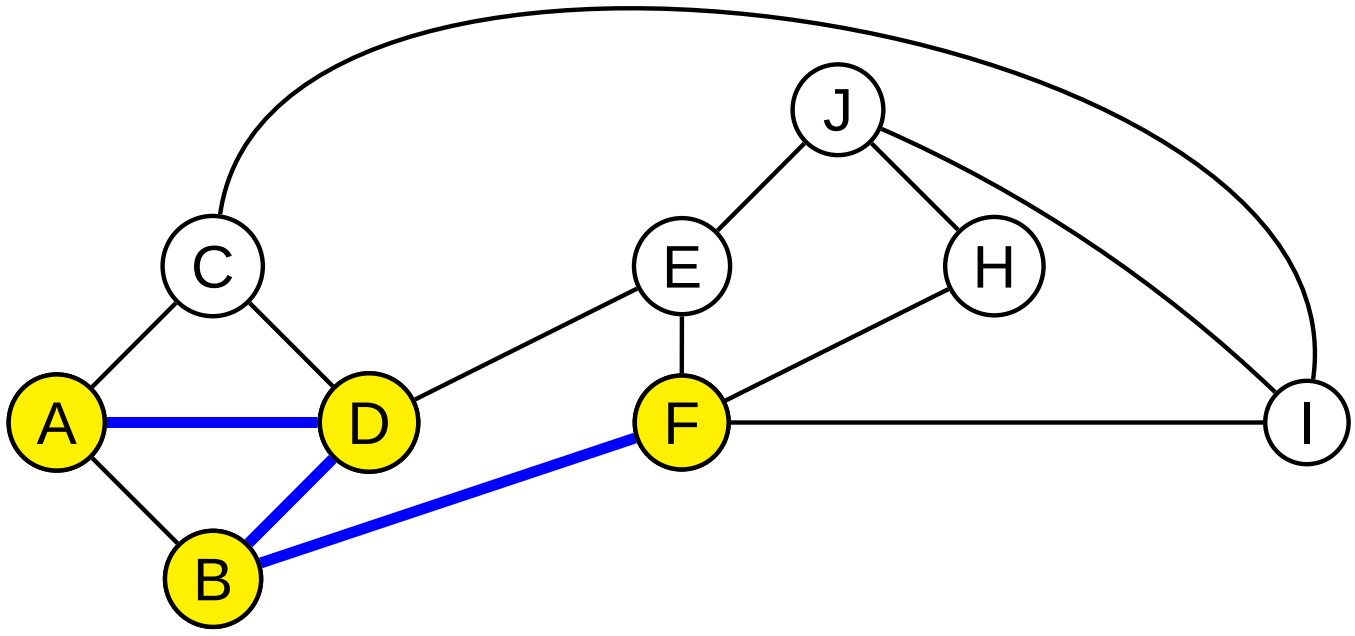
Un exemple : le circuit hamiltonien



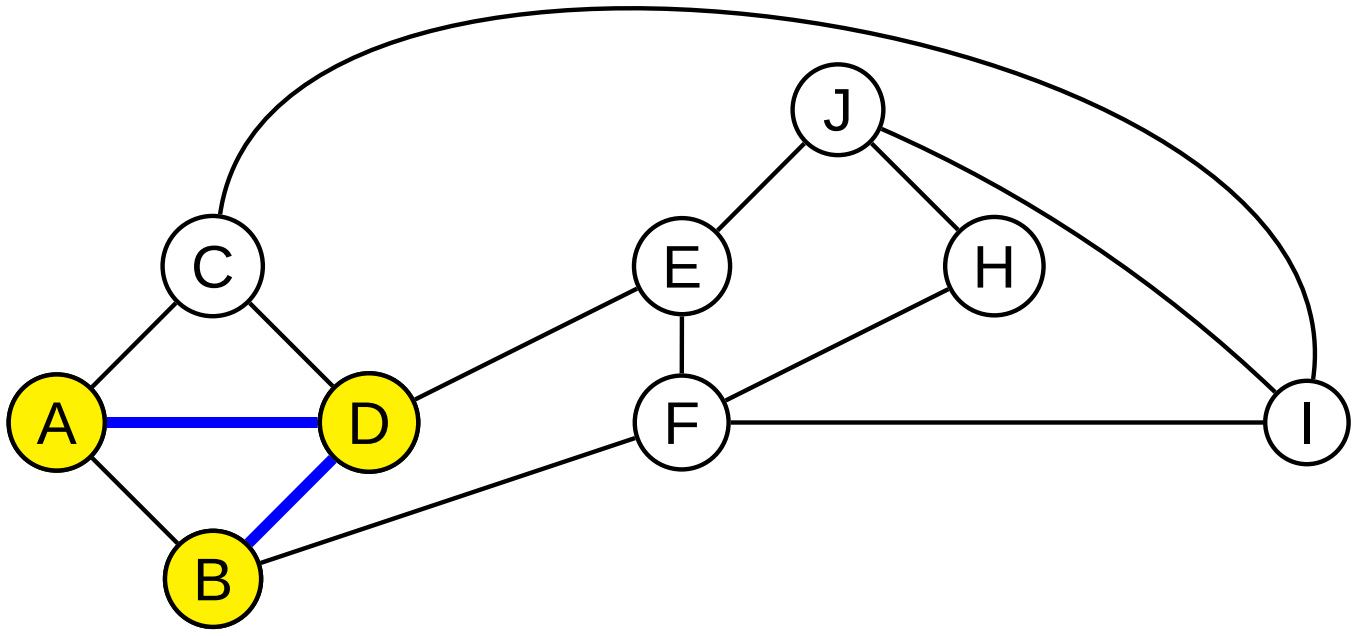
Un exemple : le circuit hamiltonien



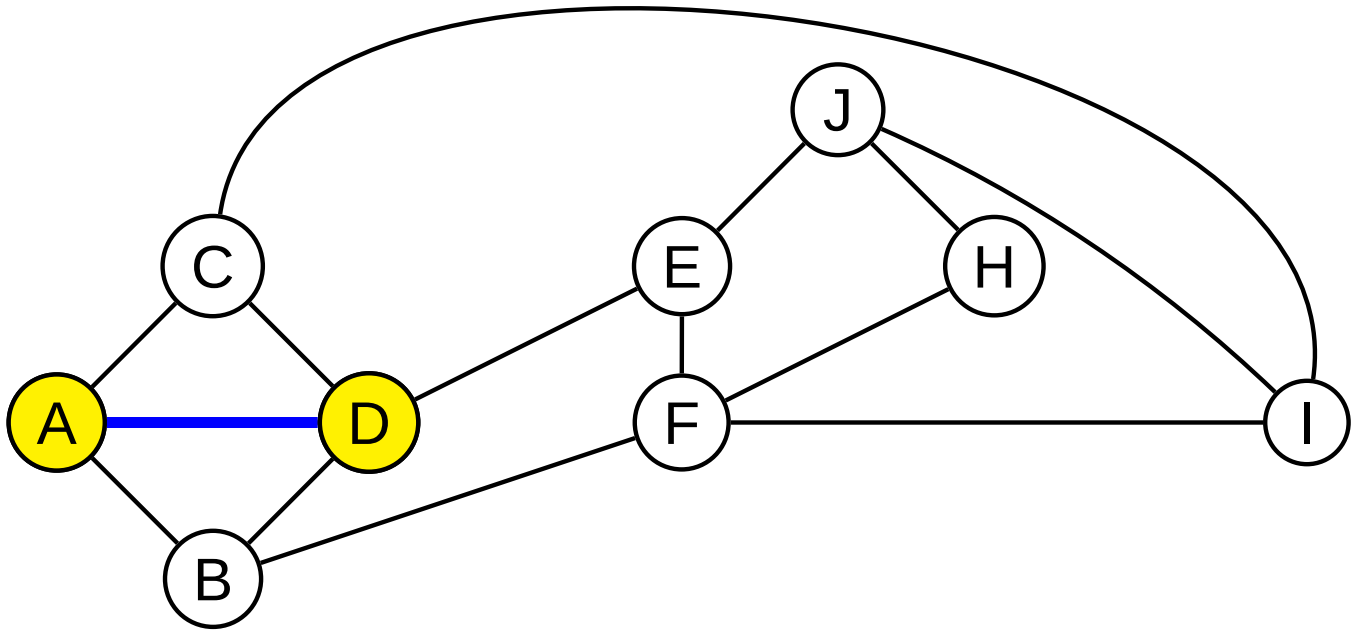
Un exemple : le circuit hamiltonien



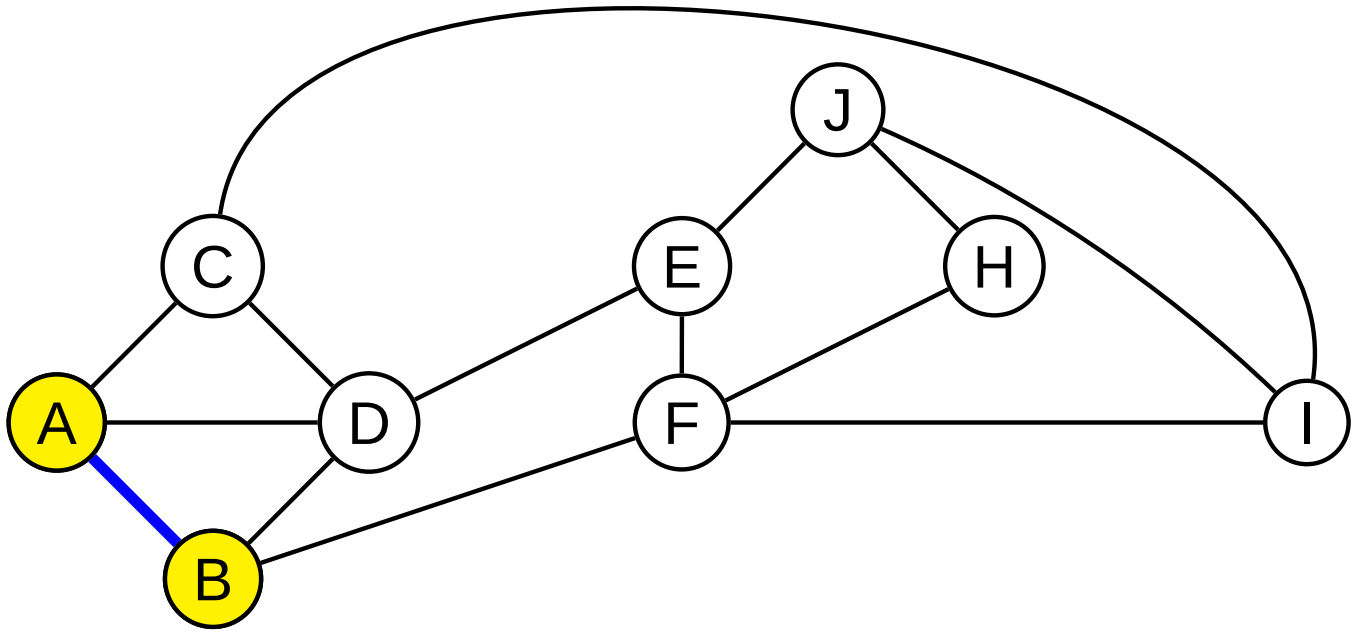
Un exemple : le circuit hamiltonien



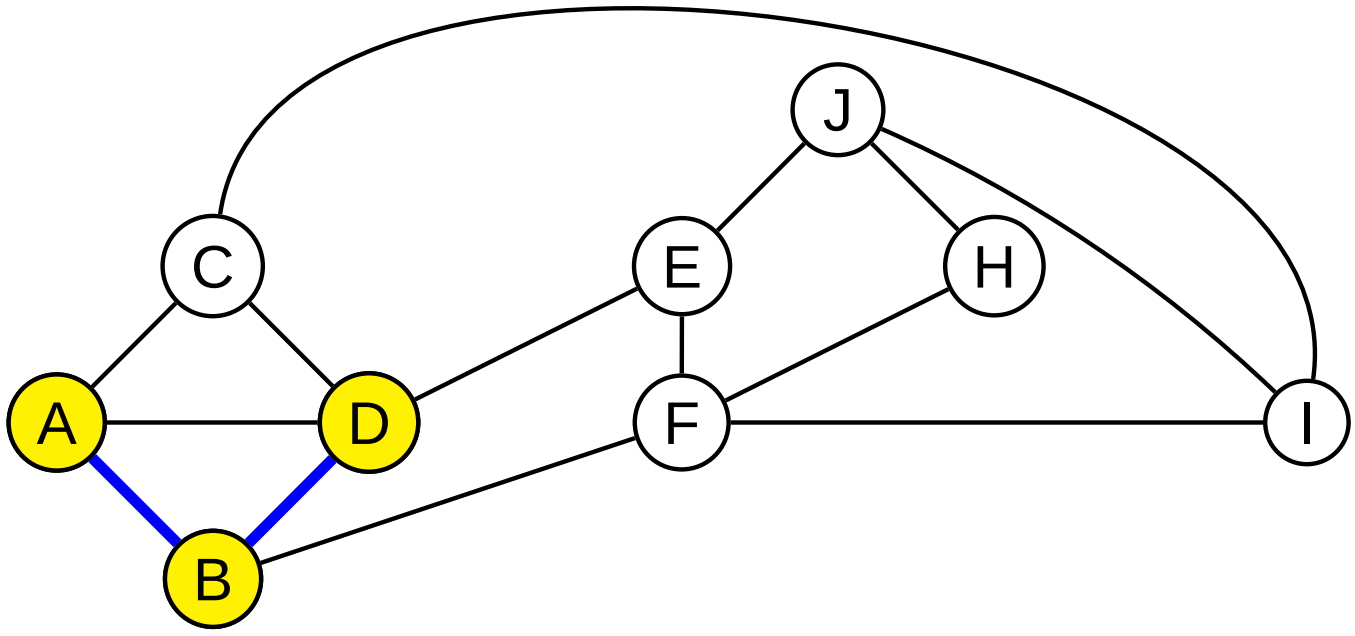
Un exemple : le circuit hamiltonien



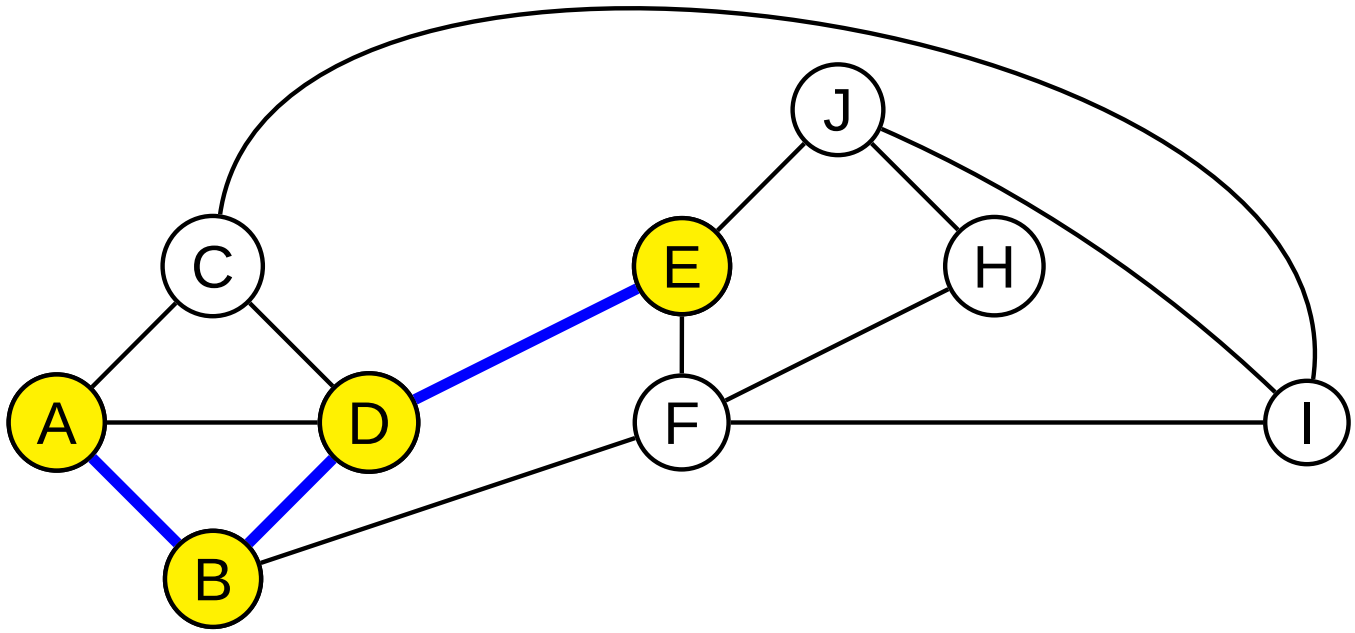
Un exemple : le circuit hamiltonien



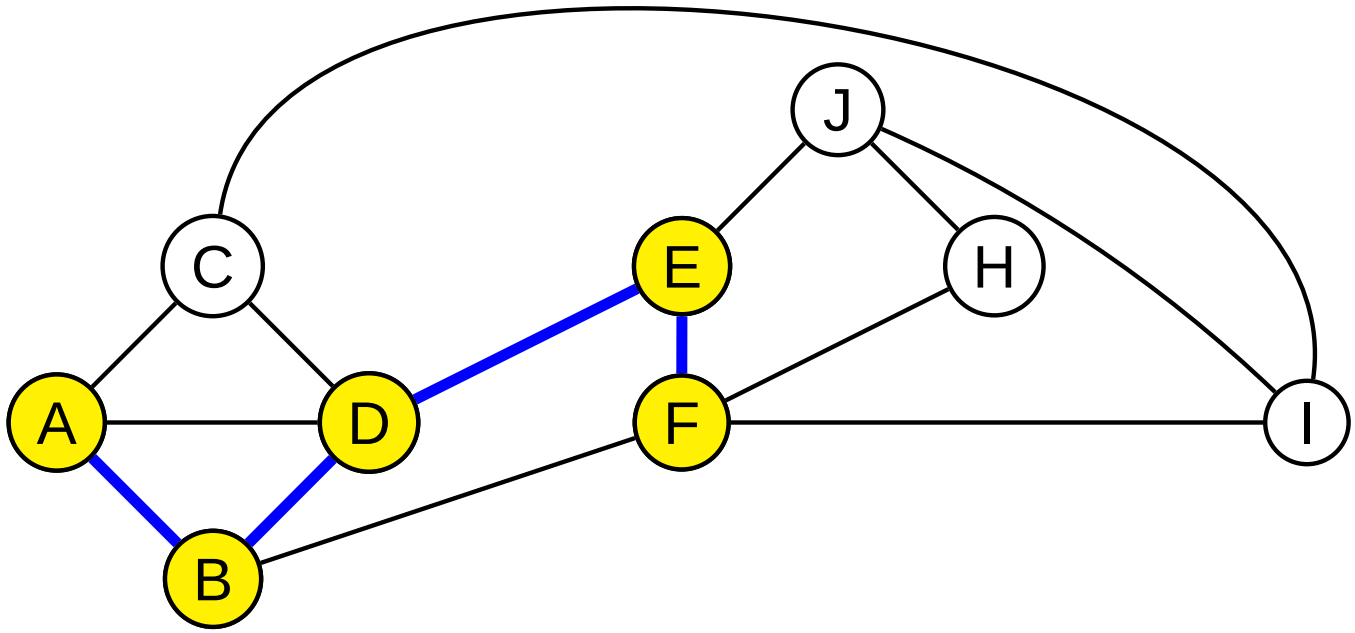
Un exemple : le circuit hamiltonien



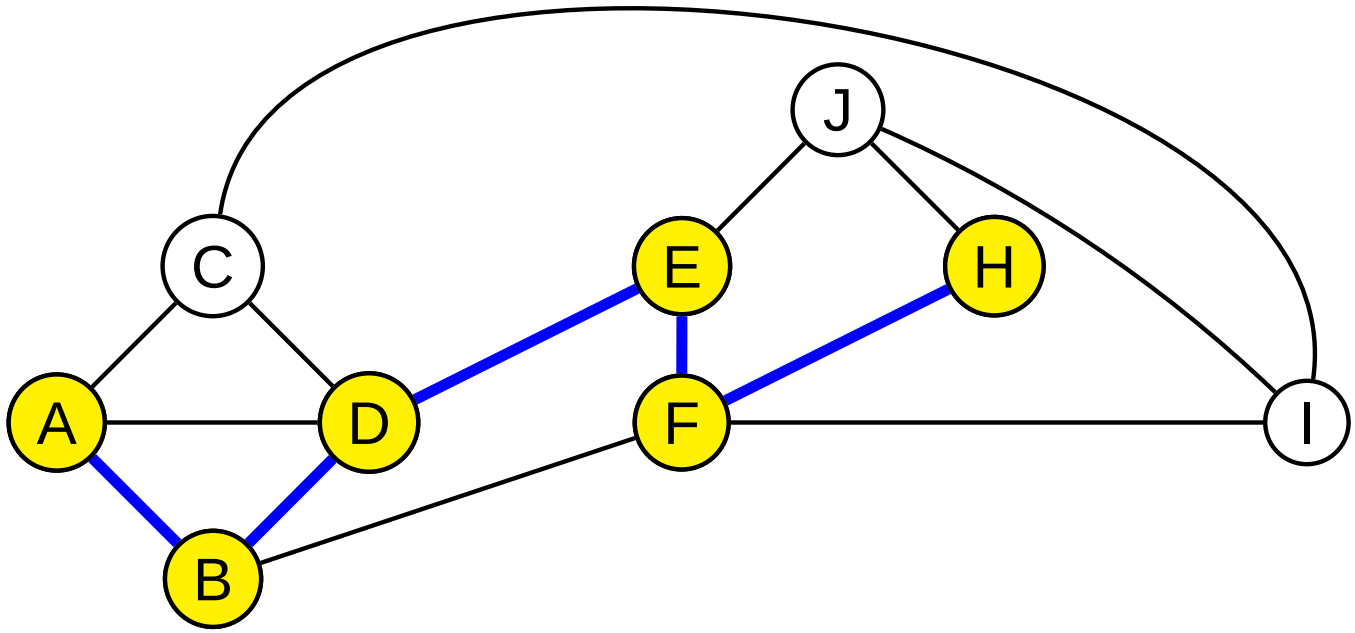
Un exemple : le circuit hamiltonien



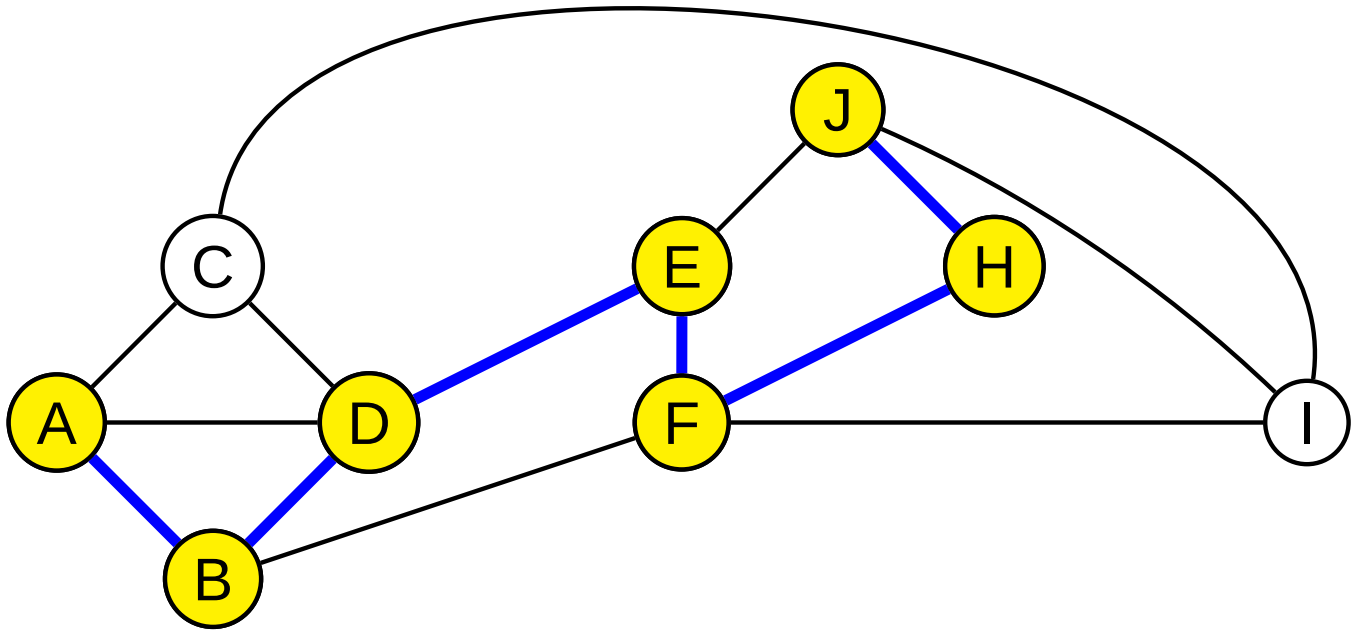
Un exemple : le circuit hamiltonien



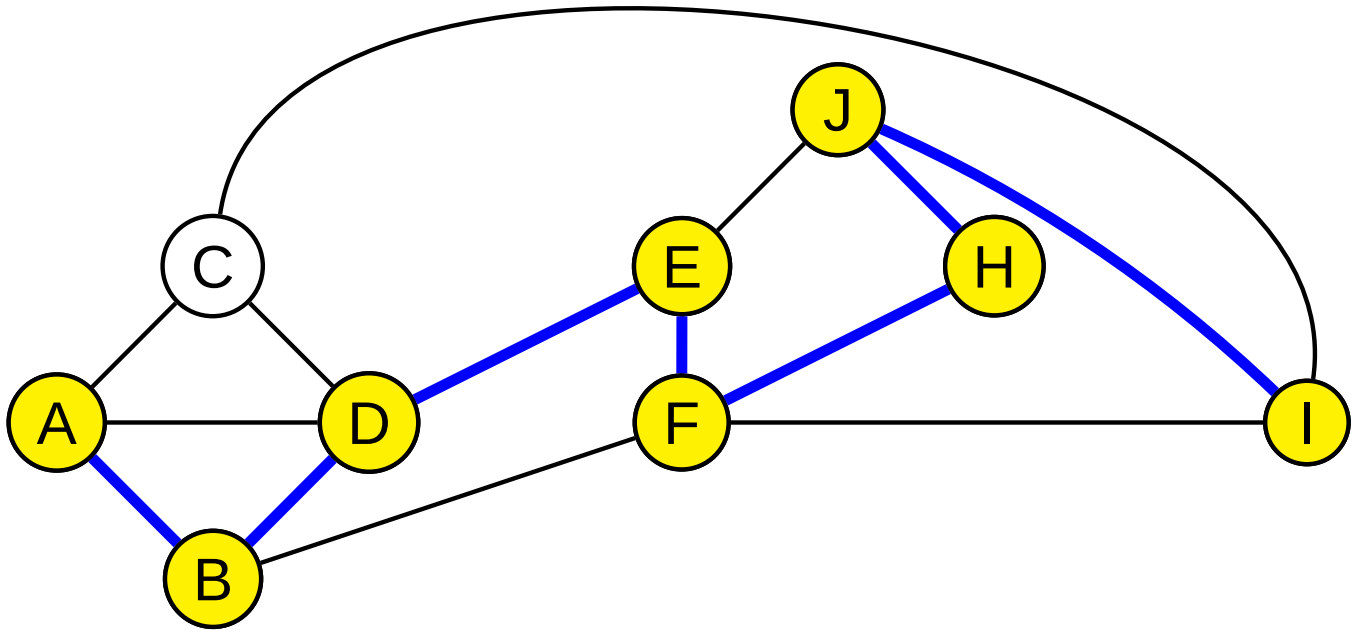
Un exemple : le circuit hamiltonien



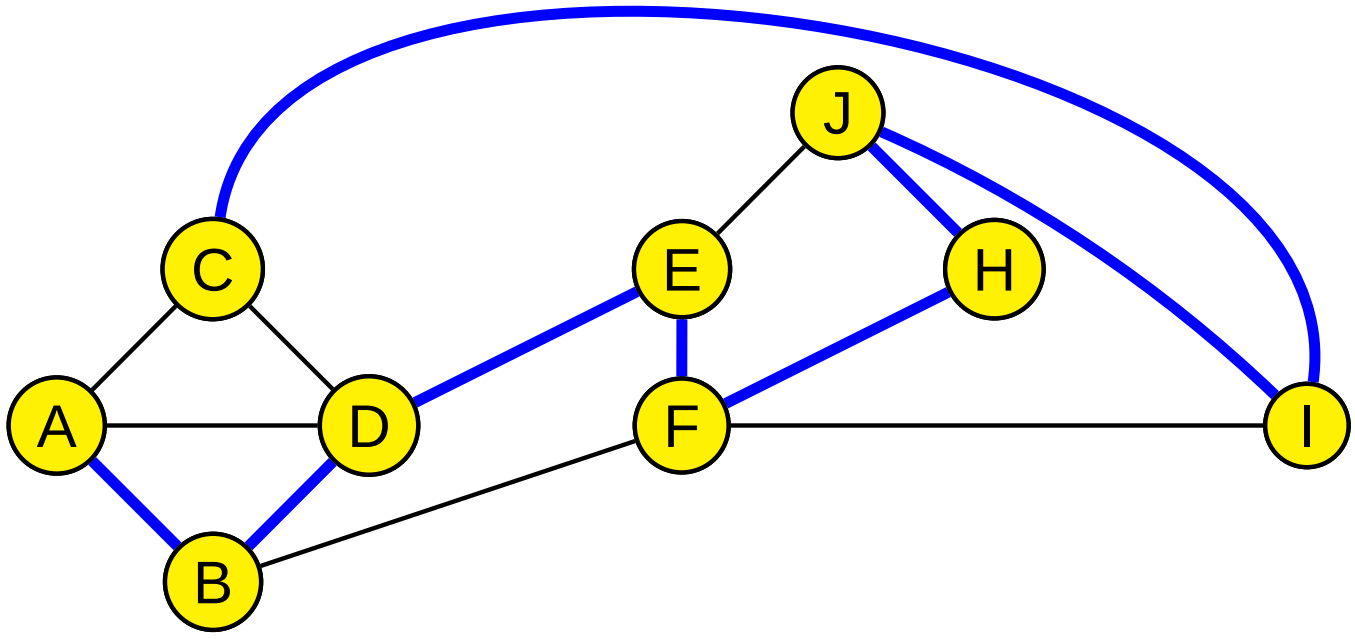
Un exemple : le circuit hamiltonien



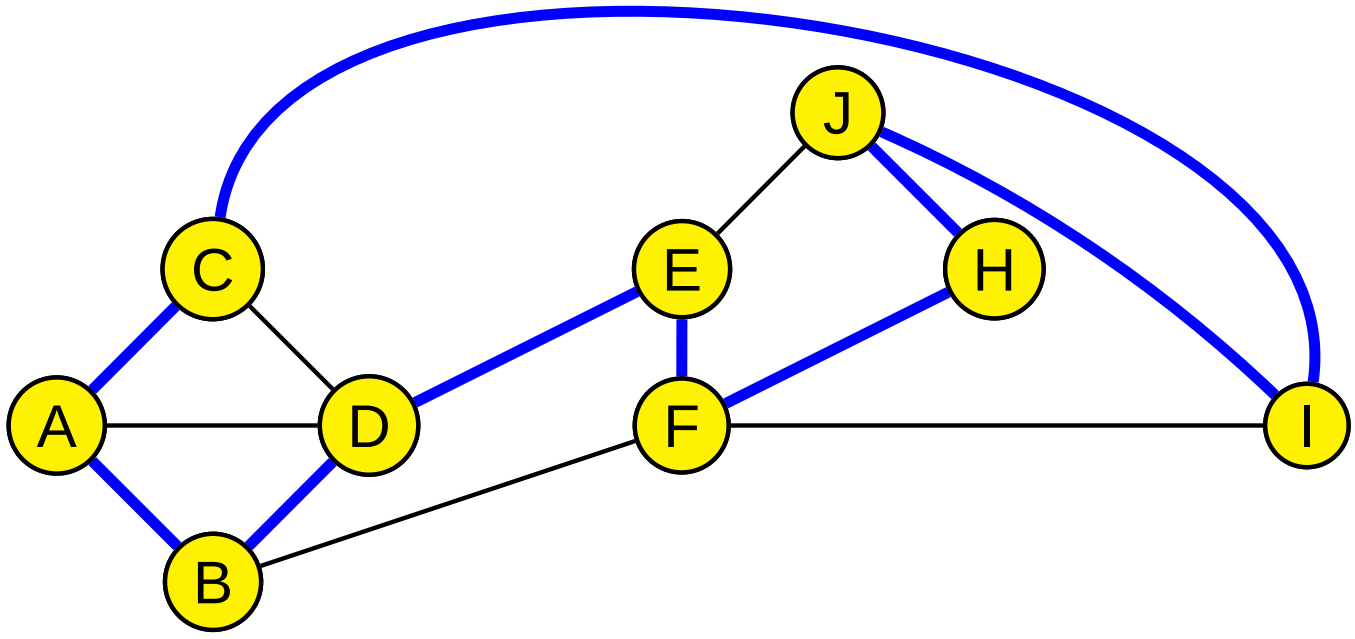
Un exemple : le circuit hamiltonien



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 :

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

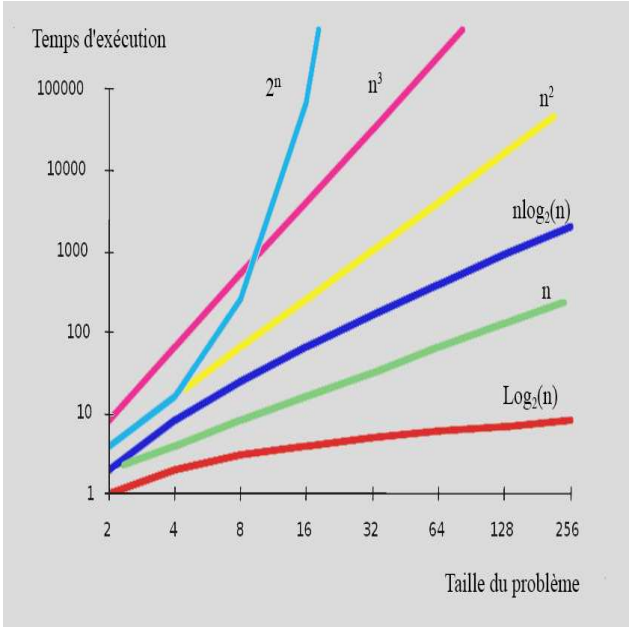
Pas de stratégie franchement meilleure (en dessous de 2^n) connue...

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

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)

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).

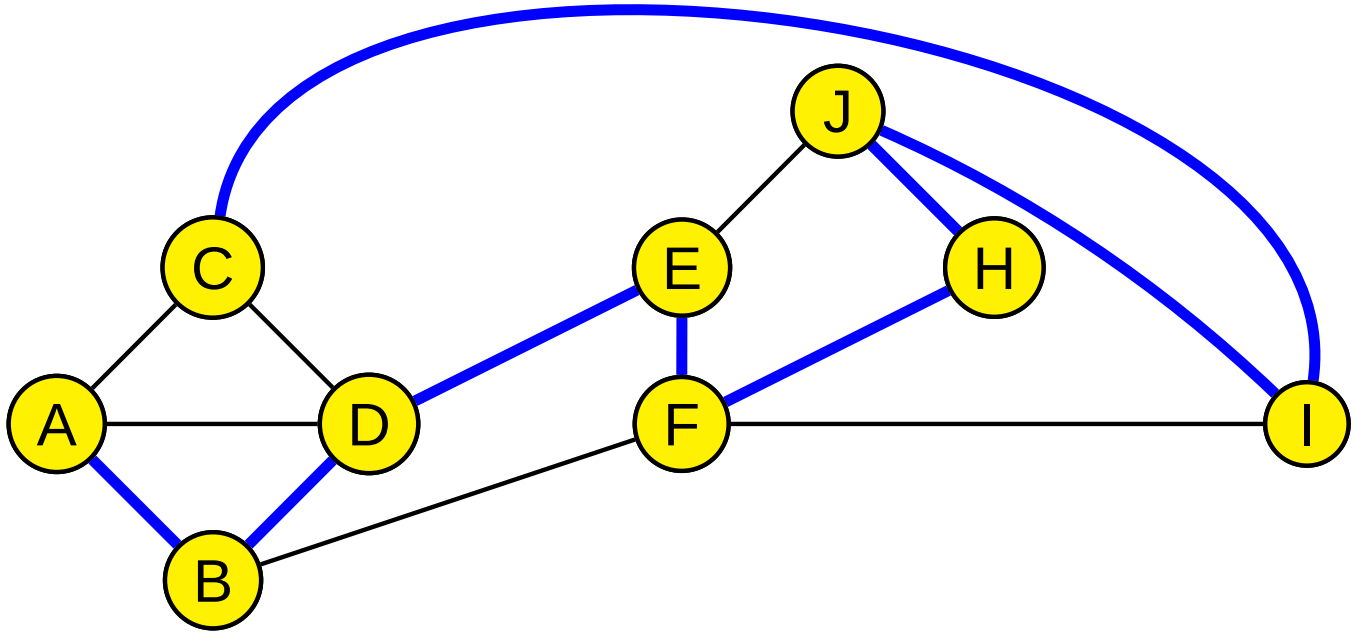
Trouver un témoin... et vite

Le problème “difficile” que l’on a vu possède la propriété suivante :

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

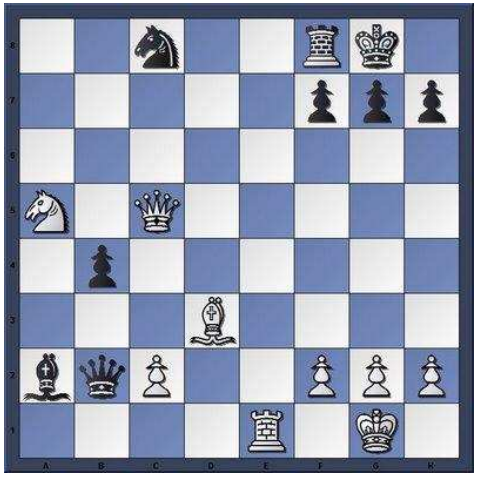
Soient trois entiers n, p, q, déterminer si $p \neq 1, n$ et $p \times q = n$ se fait en temps polynomial...

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).

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

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

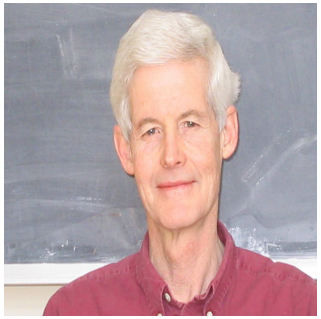
Les problèmes vérifiables en temps polynomial sont-ils aussi décidables en temps polynomial ? Autrement dit, existe-t-il des problèmes dont les solutions sont faciles à vérifier mais sont dures à trouver ?

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

S. Cook (1971)
Le problème SAT est NP-complet
Outil principal : algorithme polynomial pour "réduire" un problème à un autre, une réduction.



R. Karp (1972) :
Réduction entre problèmes combinatoires
"Computational intractability is the rule rather than the exception"

Un exemple de réduction

Dans les graphes

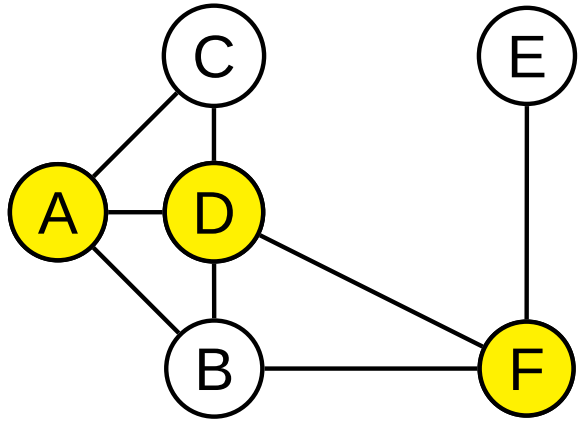
Recouvrement de sommets minimal
sous-ensemble S de taille minimale tel que toute arête a un de ses sommets dans S.

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

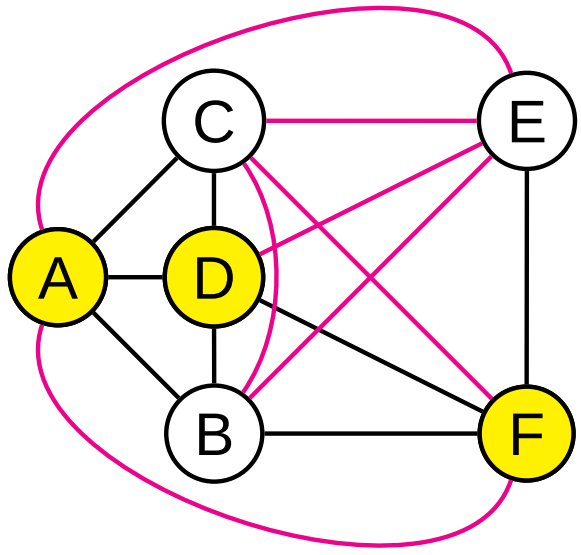
A partir d'un graphe G, construire en temps polynomial un graphe G' tel que

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

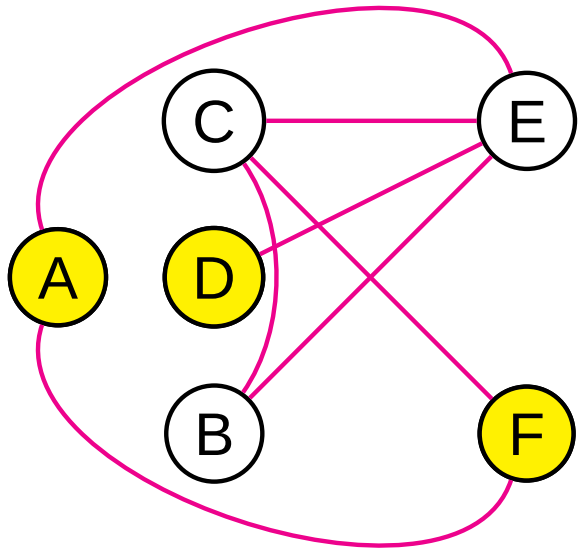
Un exemple de réduction



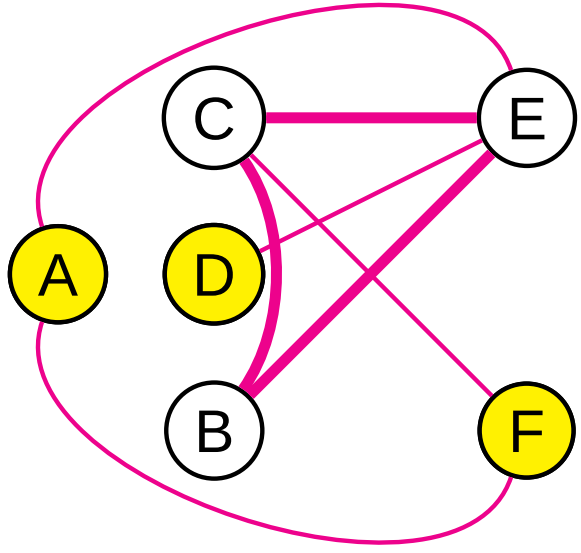
Un exemple de réduction



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...

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

Remarque : Exemple du calcul du PageRank
C'est une approche probabiliste couplée avec une méthode d'approximation qui fournit un algorithme efficace

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

Objectifs de la cryptographie

- Confidentialité : le contenu d'un message est caché
- Authenticité : l'auteur d'un message est bien identifié
- Intégrité : les messages n'ont pas été altérés

Secret parfait vs secret en pratique

- Aucune information ne peut être extraite du message crypté, même par un adversaire super-puissant (temps et puissance de calcul)
  $\rightarrow$ théorie de l'information
- En pratique : les adversaires sont limités en temps/puissance
  $\rightarrow$ théorie de la complexité

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 difficile

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 (problèmes difficiles)
- d'un modèle d'adversaire
  (limitation de la puissance de calcul)

Méthode générale

Preuves de sécurité

- un modèle de propriété (exemple : confidentialité)
- une réduction : si un adversaire peut casser la propriété, alors un algorithme composé (simulateur + adversaire) peut résoudre facilement un problème réputé difficile
- existence de problèmes difficiles (factorisation d'entiers, log discret, fonctions à sens unique,...)

On modélise les participants au protocole
(légitimes et adversaire)
par des algorithmes probabilistes en temps polynomial

Factorisation d'entiers

Etant donné un entier $n = pq$ (p, q premiers),
déterminer les facteurs premiers p et q

Records

Digits	Date	Longueur (bits)
130	avril 1996	431 bits
140	février 1999	465 bits
155	août 1999	512 bits
160	avril 2003	531 bits
200	mai 2005	664 bits
232	décembre 2009	768 bits

Complexité

768 bits $\rightarrow 2^{64}$ op.	3072 bits $\rightarrow 2^{128}$ op.
1024 bits $\rightarrow 2^{80}$ op.	7680 bits $\rightarrow 2^{192}$ op.
2048 bits $\rightarrow 2^{112}$ op.	15360 bits $\rightarrow 2^{256}$ op.

Réduction

Adversaire calculant en temps $t \rightarrow$
Algorithme calculant en temps $T = f(t)$

- Réduction paresseuse : $T = k^3 \times t$

Longueur (bits) Module	Complexité Adversaire	Complexité Algorithme	Meilleure Complexité
$k = 1024$	$t < 2^{80}$	$T < 2^{110}$	2^{80}
$k = 2048$	$t < 2^{80}$	$T < 2^{113}$	2^{112}
$k = 3072$	$t < 2^{80}$	$T < 2^{115}$	2^{128}

- Réduction fine : $T \approx t$
  Avec $k = 1024$ et $t < 2^{80}$, on obtient $T < 2^{80}$

Conclusion : utilité de la théorie de la complexité

- Comme dans l'exemple du PageRank, les algorithmes réellement efficaces font souvent appel aux méthodes d'approximation et aux algorithmes probabilistes
- Les mesures de complexité (le temps, l'espace) permettent la comparaison des algorithmes
- De nombreux domaines d'application fournissent des exemples de problèmes "intrinsèquement" difficiles
- La notion de réduction est au coeur de la problématique de la complexité
- La complexité (problèmes difficiles, réduction, modélisation de l'adversaire) permet de formaliser la sécurité prouvable

Quelques références

M.R. Garey et D.S. Johnson

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

M. Habib

Modélisation et moteurs de recherche sur le web
2012

R. Lassaigne et M. de Rougemont

Logic and Complexity. Springer-Verlag, 2004

C.H. Papadimitriou

Computational Complexity. Addison Wesley, 1994

D. Pointcheval

Quelles garanties avec la cryptographie ?
Collège de France, 27/04/2011

Merci pour votre attention