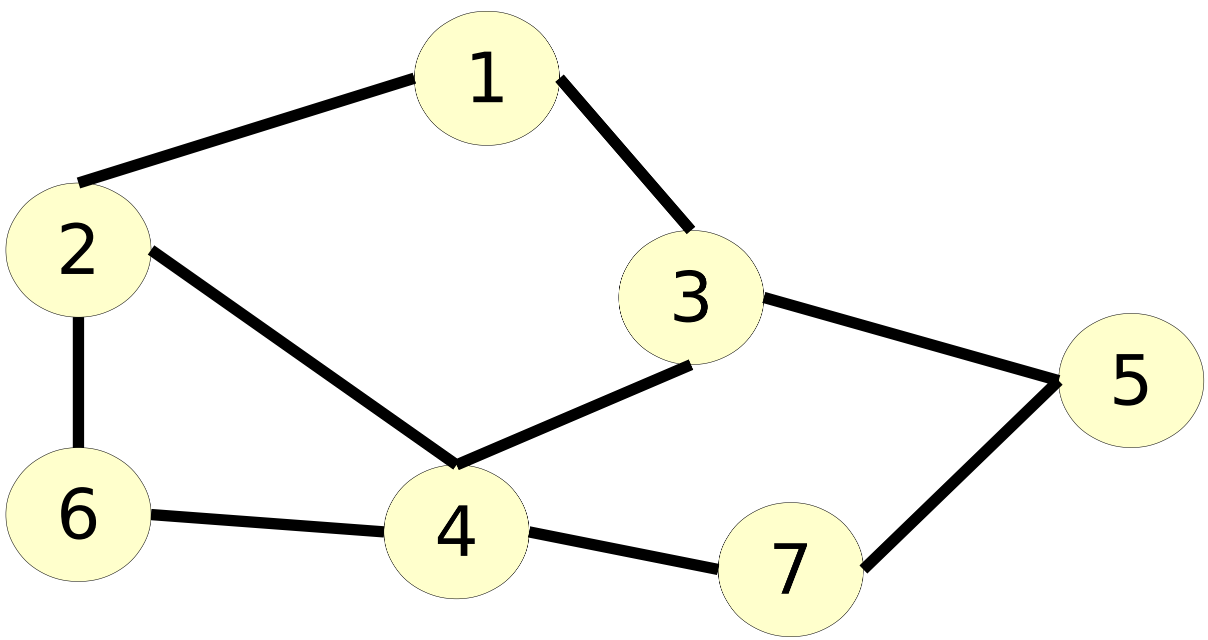
Arbres

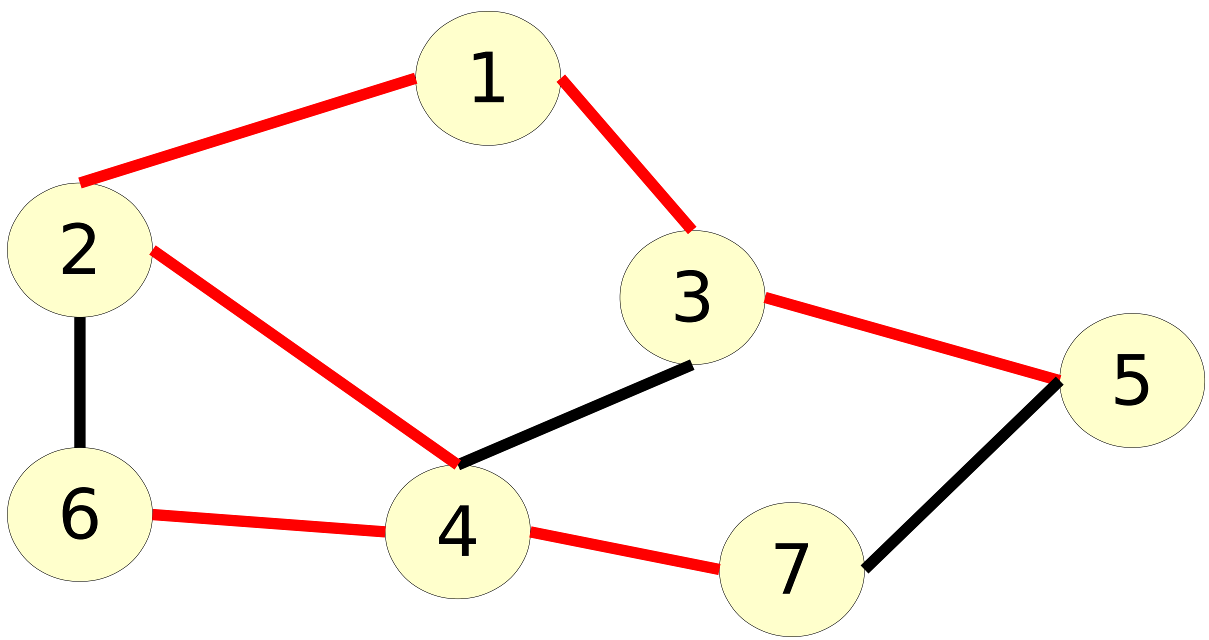
- G = (X, U).
  H = (X, T), T $\subseteq$ U est une forêt si H ne contient pas de cycle.
- Si G est connexe et que aucune arête ne peut être ajoutée sans créér de cycle, G est un arbre.

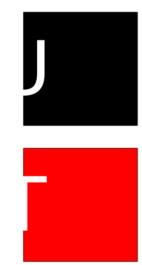


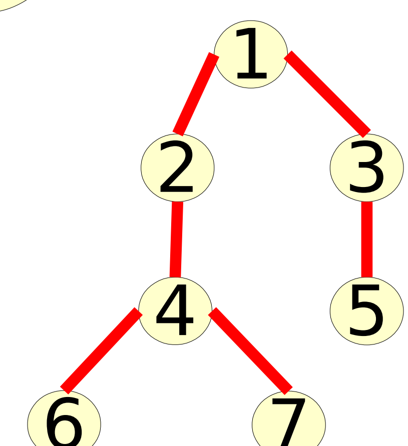
- Tout arbre contient |X|-1 arêtes

Arbres

- G = (X, U).
  H = (X, T), T $\subseteq$ U est une forêt si H ne contient pas de cycle.
- Si G est connexe et que aucune arête ne peut être ajoutée sans créér de cycle, G est un arbre.



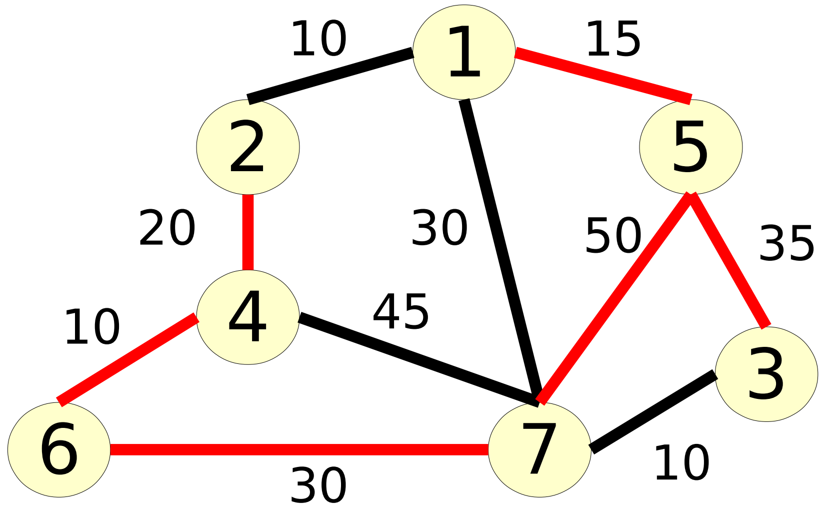




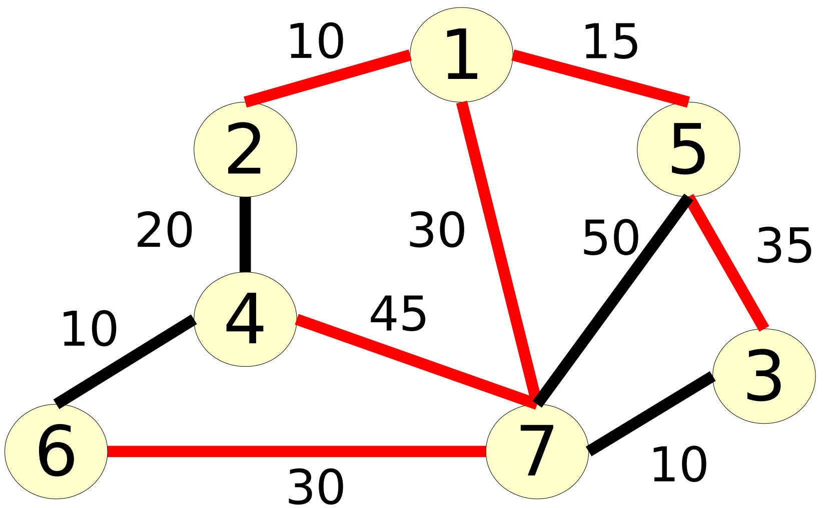
- Tout arbre contient |X|-1 arêtes

Arbre couvrant de poids minimum

- $G = (X, U)$, avec poids associés aux arêtes. Trouver, à partir d'un sommet S_0 , un arbre $H = (X', T)$, $X' \subseteq X$, $T \subseteq U$ t.q :
  – $|X'|$ soit maximum et
  – $\sum (w_{u \in T})$ soit minimum.
- H est un arbre couvrant de poids minimum



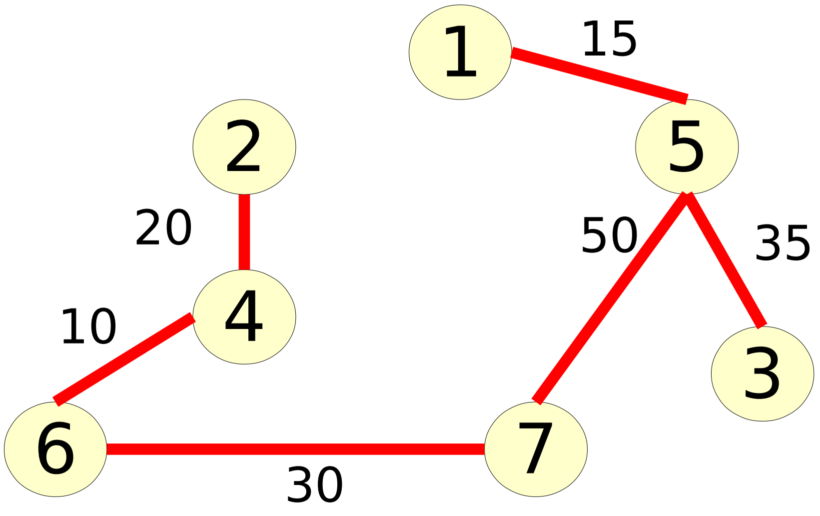
OU ?



Arbre couvrant de poids minimum

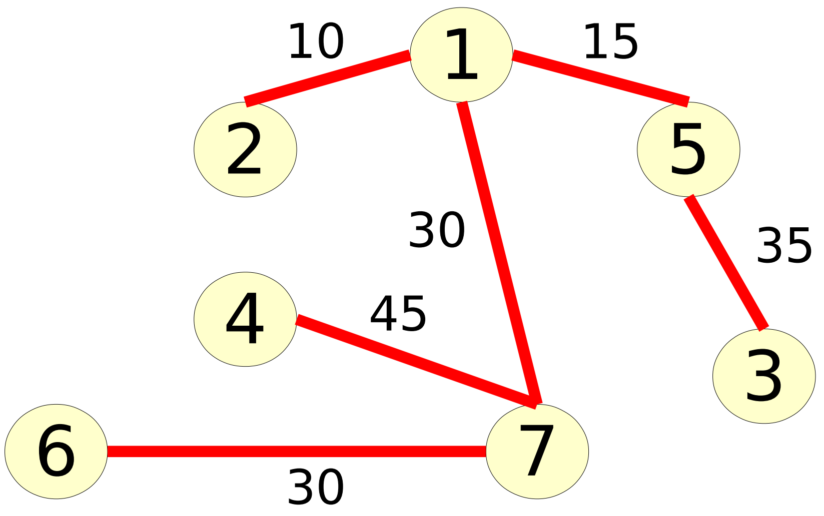
- Est-ce l'arbre couvrant de poids minimum ? Algorithme de Kurskal ou **Prim**

$\sum (w_{u \in T}) = 160$



OU ?

$\sum (w_{u \in T}) = 165$



Arbre couvrant de poids minimum
Algorithme de Prim

- A partir de S_0, ajout d'un sommet à l'arbre à chaque étape
  - une arête de connexion à l'arbre par sommet
  - N-1 étapes, N-1 arêtes

- Même principe de marquage que avec Dijkstra
  D[i], poids de l'arête de connexion u pour S_i
  a[i] = u, arête de connexion u pour S_i
- E, ensemble des sommets déjà connectés à l'arbre

Prim - énoncé

Prim(Graphe G = (X, U), sommet S_0)(Tableaux D, a)

1-Initialisation

E = {}

D[0] $\leftarrow$ -∞

$\forall S_i \in X$, $i \neq 0$, D[i] $\leftarrow$ +∞

$\forall$ S_i $\in$ X, a[i] $\leftarrow$ ∞

2-Itération courante

choisir S_i t.q

D[i] = $\min_{\{j \notin E\}}$D[j]

Si (D[i]=+∞) ou ($\forall S_i \in X$, a[i]>0) **fin**

Sinon E $\leftarrow$ E $\cup$ {i}

Pour toute arête u = ($S_i \leftrightarrow S_j$)

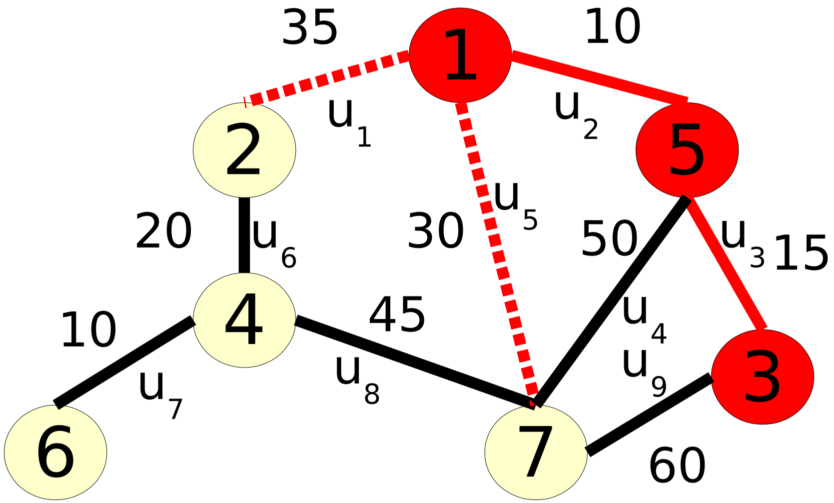
Si (w_u < D[j]) et (j $\notin$ E)

D[j] $\leftarrow$ w_u

a[j] $\leftarrow$ u

Arbre couvrant de poids minimum
une étape

E	S_1	S_2	S_3	S_4	S_5	S_6	S_7
D	$-\infty$	35	15	∞	10	∞	30
a	$+\infty$	u_1	u_3	∞	u_2	∞	u_5

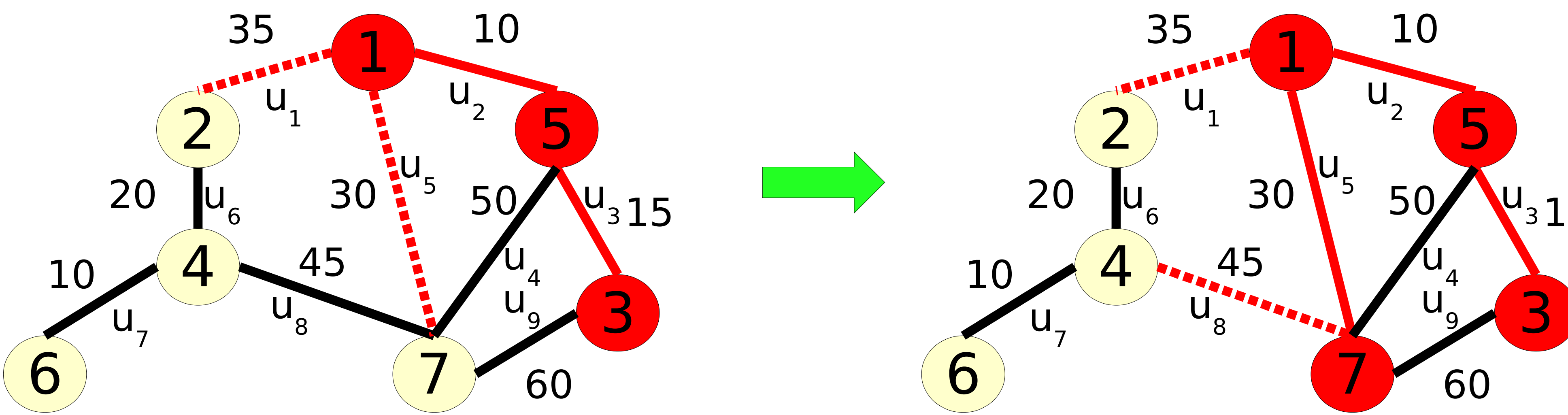


Arbre couvrant de poids minimum
une étape

E	S_1	S_2	S_3	S_4	S_5	S_6	S_7
D	$-\infty$	35	15	∞	10	∞	30
a	$+\infty$	u_1	u_3	∞	u_2	∞	u_5

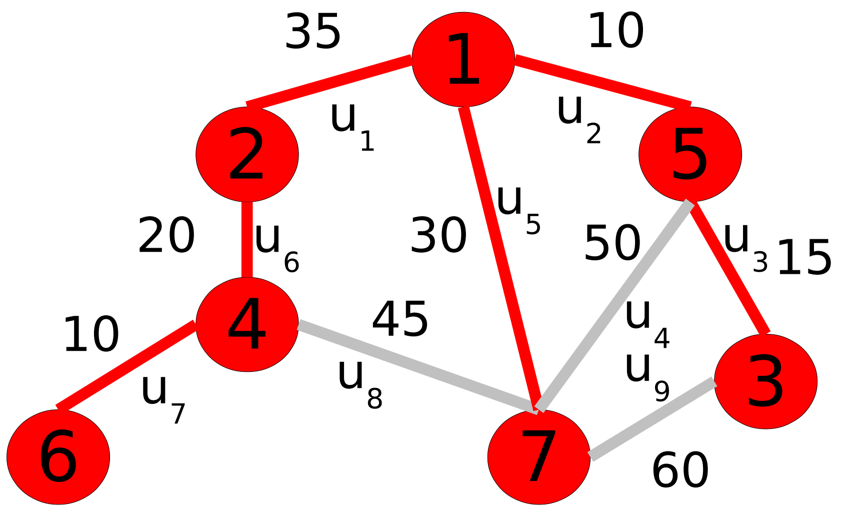
E	S_1	S_2	S_3	S_4	S_5	S_6	S_7
D	$-\infty$	35	15	45	10	∞	30
a	$+\infty$	u_1	u_3	u_8	u_2	∞	u_5

minimu
m



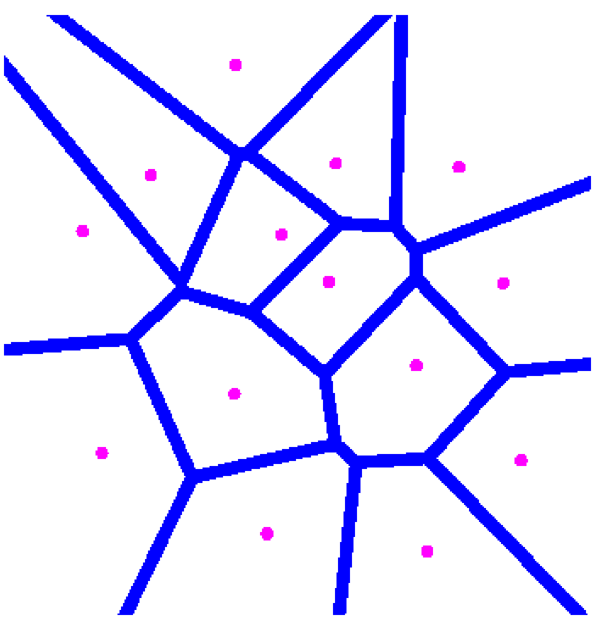
Arbre couvrant de poids minimum résultat

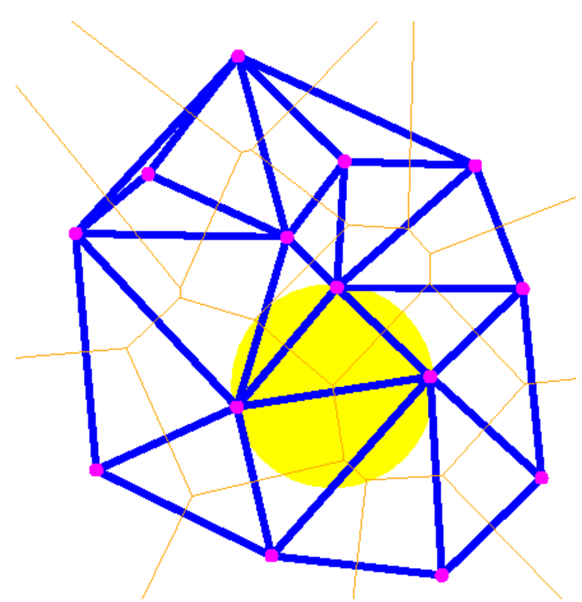
E	S_1	S_2	S_3	S_4	S_5	S_6	S_7
D	$-\infty$	35	15	20	10	10	30
a	$+\infty$	u_1	u_3	u_6	u_2	u_7	u_5

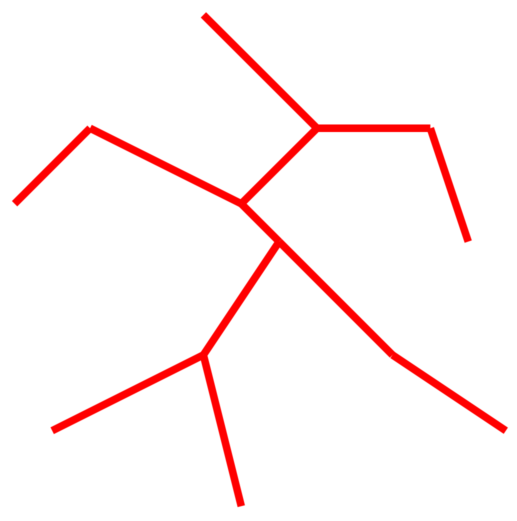


Arbre couvrant de poids minimum
Applications

- La construction/couverture de réseau (télephone, routier/transport, informatique)
- Souvent dans le plan
  - graphe complet ou
    diag. Voronoi -> triangles de Delaunay -> Arbre



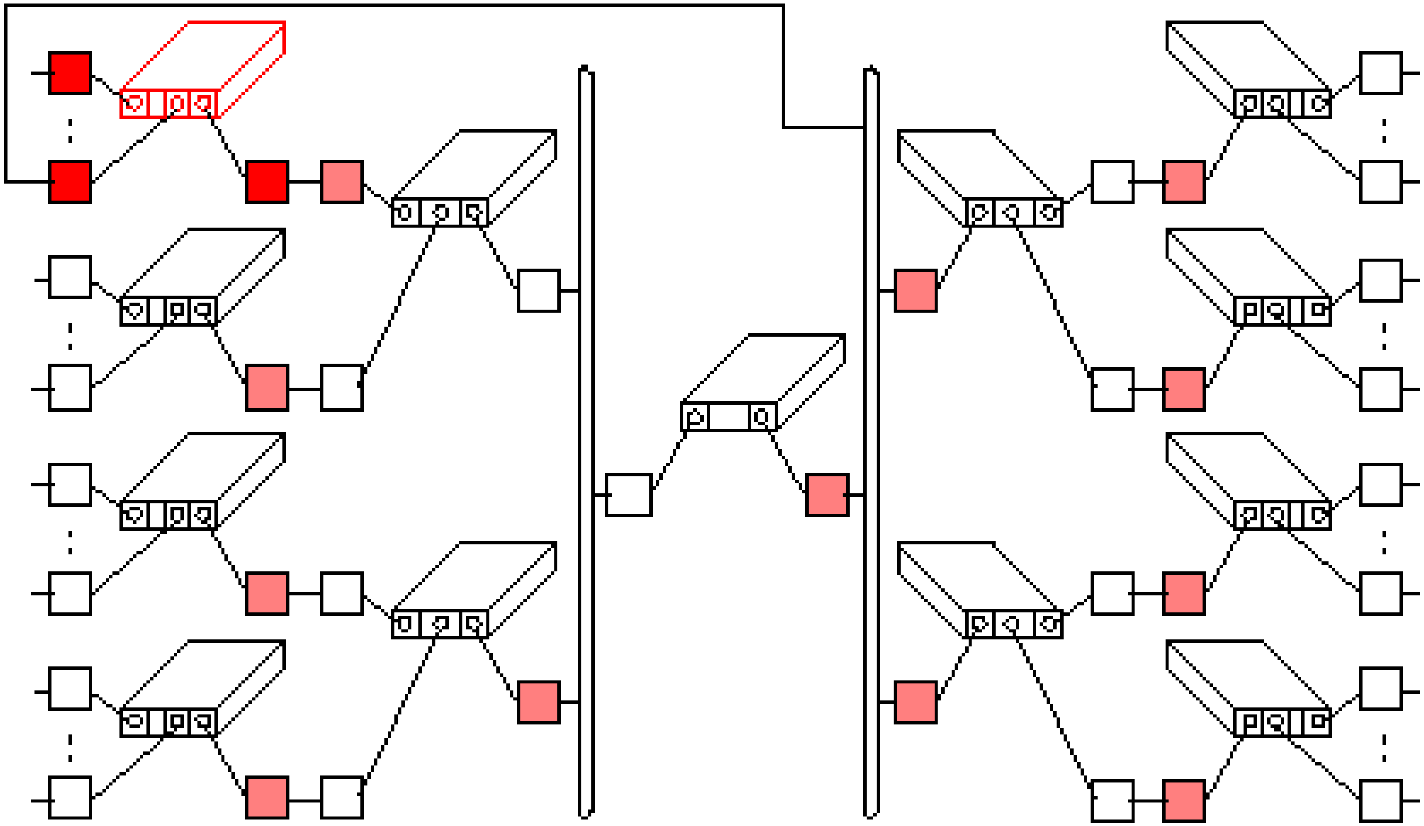




http://www-sop.inria.fr/prisme/fiches/Voronoi/index.html.fr

Arbre couvrant de poids minimum exemples

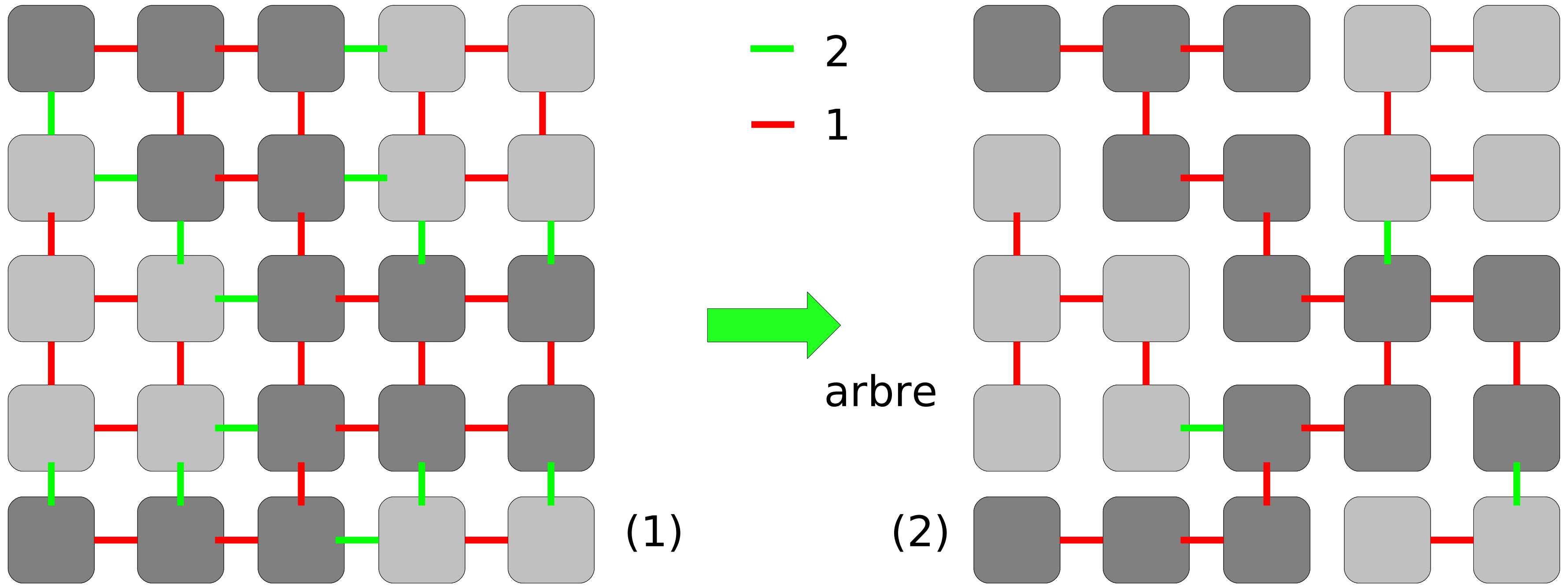
- STP (Spanning Tree Protocol) pour la diffusion Ethernet – éviter les rebouclages



http://www.cs.berkeley.edu/~bonachea

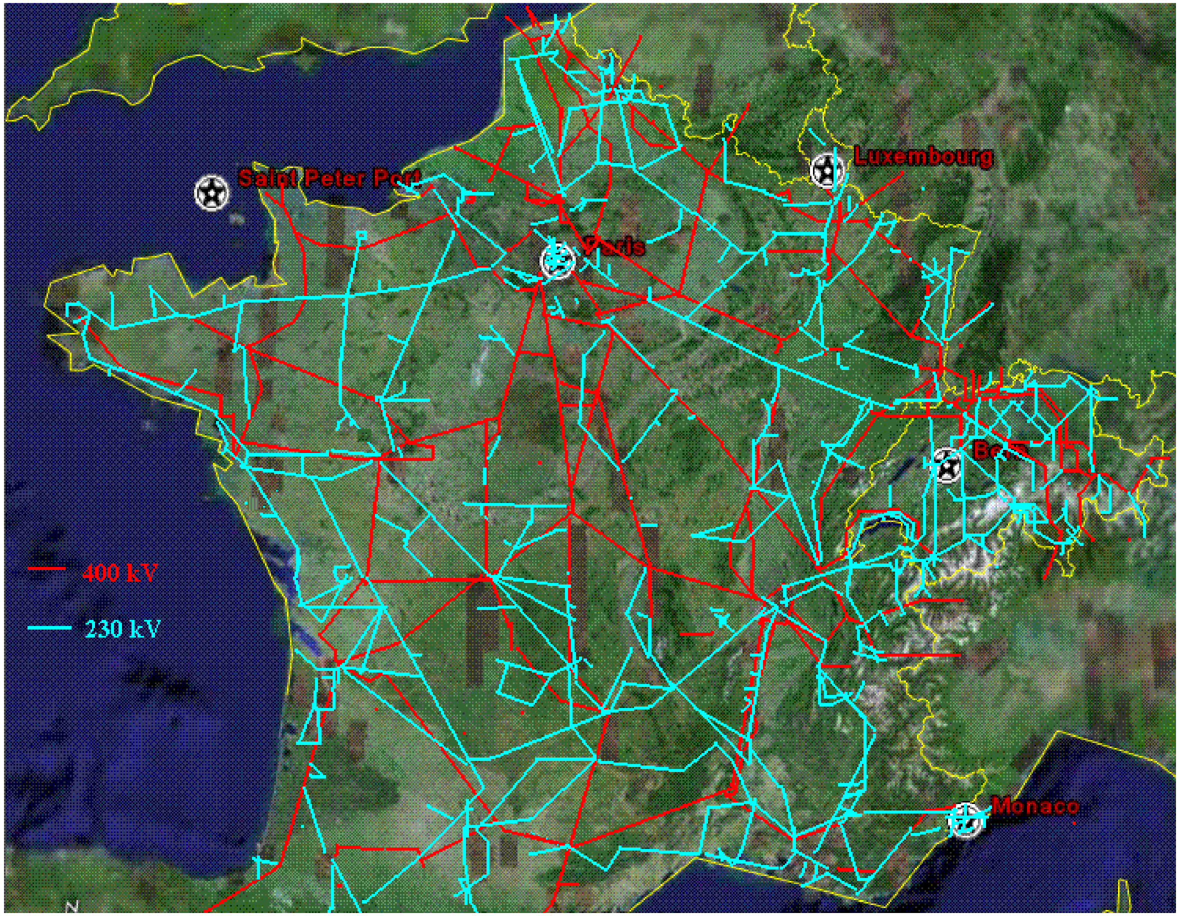
Arbre couvrant de poids minimum
Applications

- Définition de régions dans une image
  arêtes entre pixels voisins pondérées par la similarité entre les pixels (1)
  - Arbre de poids min/ simil max (2)
  - et coupe des arêtes de coût supérieur à un seuil



Problèmes de flot
réseau électrique

- L'ensemble de l'électricité produite est consommée :
- Il y a équilibre entre les producteurs et les consomateurs à tout moment



- Loi d'Ohm : les courants entrants et sortant s'équilibrent à chaque intersection

Problèmes de flot

- **Flux**: nombre f_u associé à chaque arc $u \in U$ de $G=(X, U, C)$
- **Flot**: L'ensemble des flux sur G constitue un flot ssi

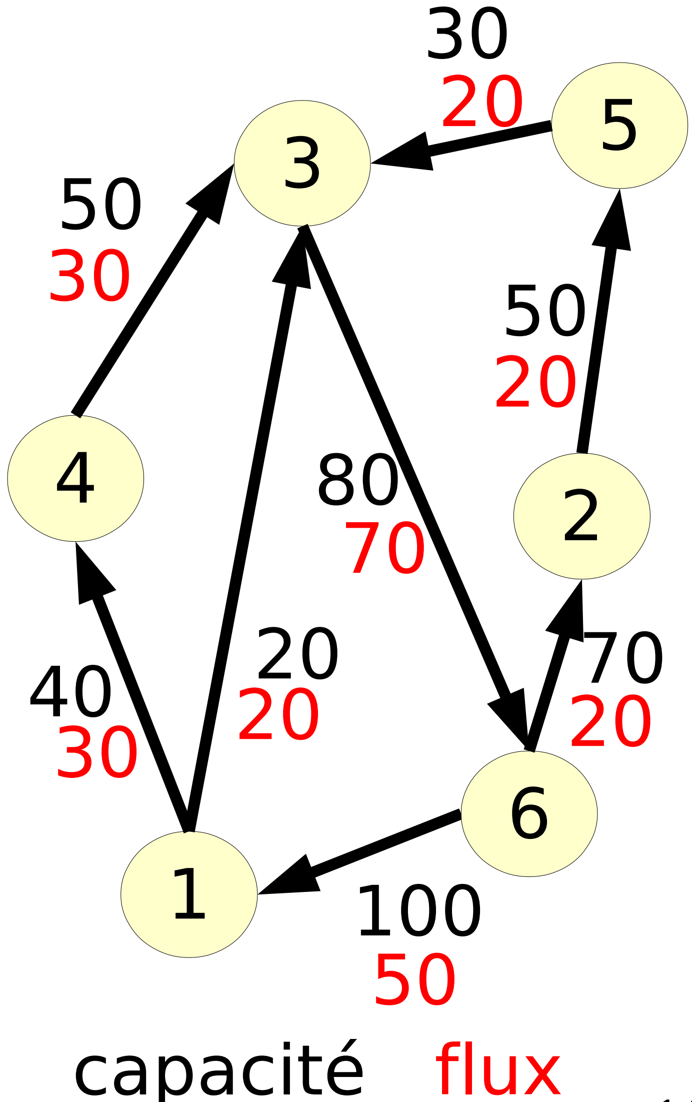
$$(1) \quad \sum_{u \in \omega+(i)} f_u = \sum_{u \in \omega-(i)} f_u$$

Loi de conservation

  - ω+(i) : arcs sortants de Si
  - ω-(i) : arcs entrants de Si

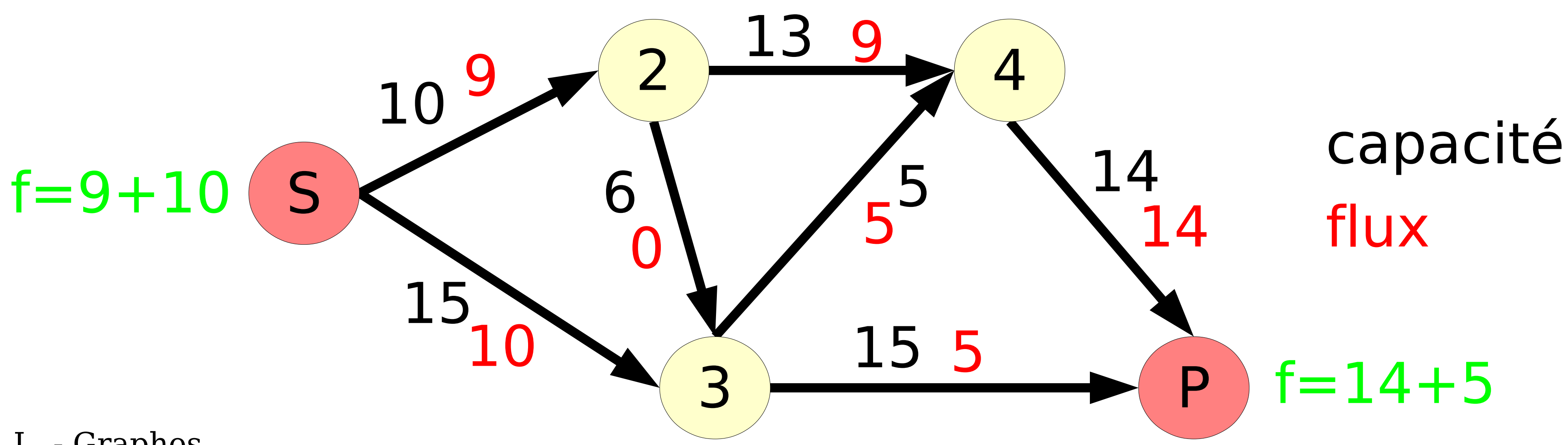
$$(2) \quad \forall u \in U, \quad f_u \leq c_u$$

Flux admissible



Graphe de flot

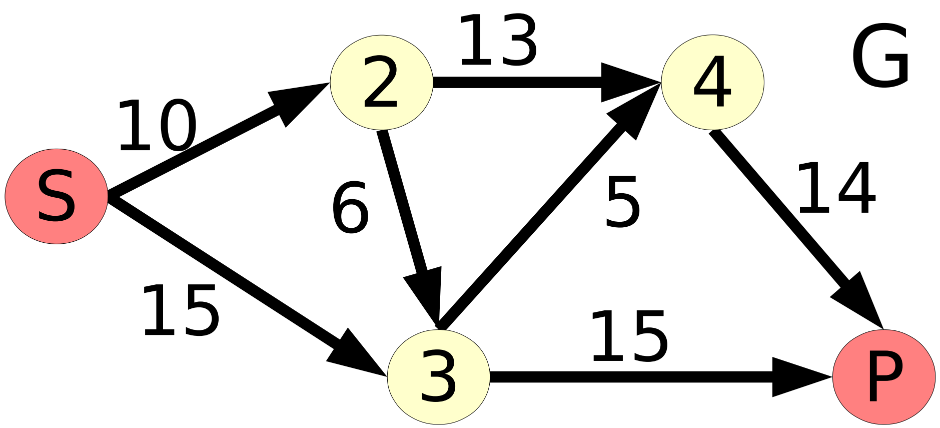
- G=(X, U, C)
  - Sommet **source** S, sans arc entrant
  - Sommet **puits** P sans arc sortant
- Flot de S à P, avec loi de conservation respectée sauf pour S et P
  - Valeur de flot : $f = \sum_{u \in \Gamma(S)} f_u = \sum_{u \in \Gamma^{-1}(P)} f_u$



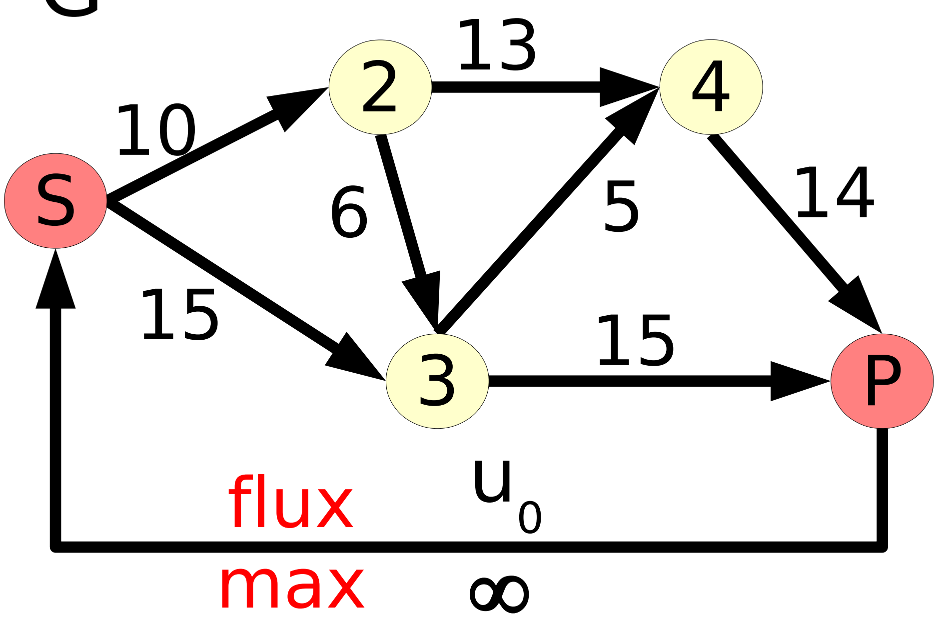
Flot maximum

- Maximiser la valeur flot de S à P dans G
  $\Leftrightarrow$ Flot maximum dans G^0
  - G^0: ajout de l'arc u_0, avec une capacité ∞, dans G
  - Flot maximum dans G^0 $\Leftrightarrow$ flux maximum pour u_0

G



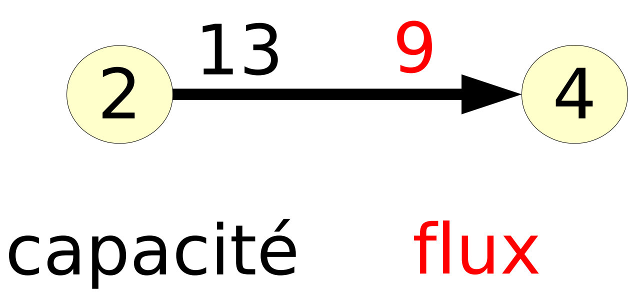
G^0



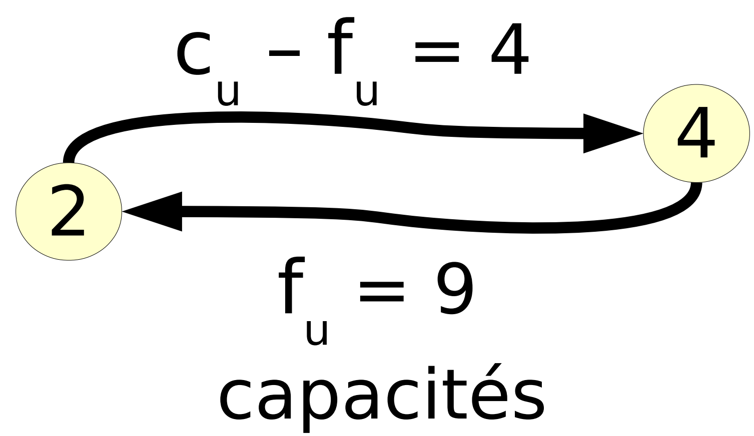
Graphe d'écart

- Soit un graphe de flot $G=(X, U, C)$ et un flot f sur G le graphe d'écart $G^{\Delta}(f) = (X, U^{\Delta}, C^{\Delta})$ est défini par :
  $\forall$ arc $u=(S_i \rightarrow S_j) \in U$
  - $u^+ = (S_i \rightarrow S_j) \in U^{\Delta}$, avec $c^{\Delta}(S_i \rightarrow S_j) = c_u - f_u$
  - $u^- = (S_j \rightarrow S_i) \in U^{\Delta}$, avec $c^{\Delta}(S_j \rightarrow S_i) = f_u$

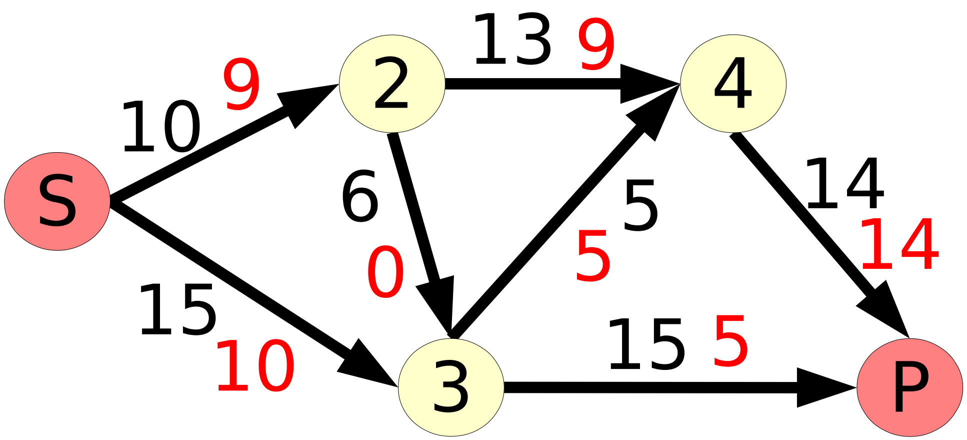
dans G :



dans G^{Δ} :



Graphe d'écart
exemple

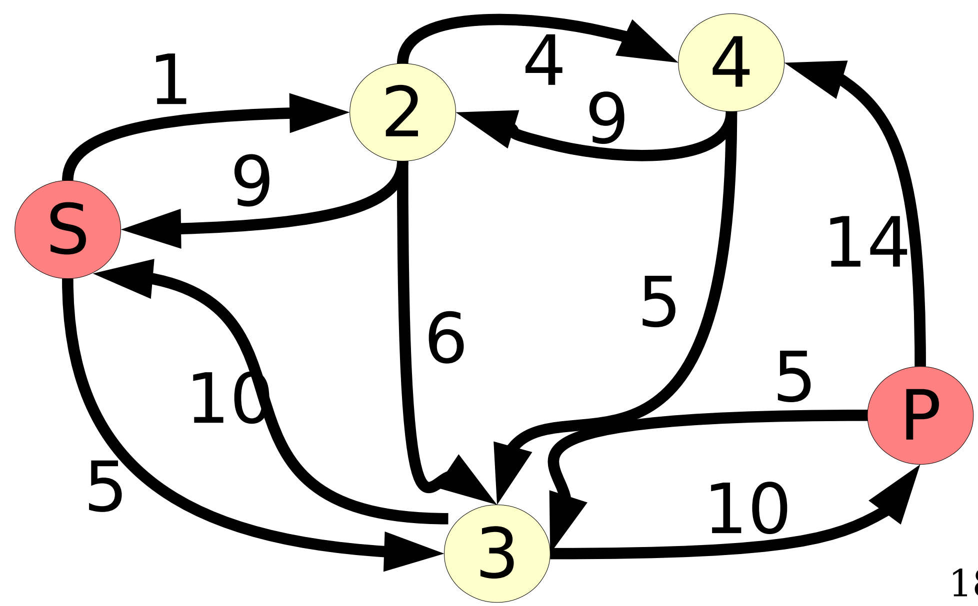


capacité flux

G

G^{Δ}

capacité



Flot maximum dans G
si il n'existe aucun
chemin $S \rightarrow P$ dans G^{Δ}

Flot max ici ?

Ford-Fulkerson - énoncé

FordFulkerson(Graphe G = (X, U, C))(Flot F)

1-Initialisation

Flot $f \leftarrow (0, 0, \ldots, 0)$

2-Itération courante

Trouver un chemin A
de $S \rightarrow P$ dans $G^{\Delta}(f)$

Si A non trouvé **fin**

Flot max

Sinon

$r \leftarrow \min_{u \in X} c^{\Delta}(u)$

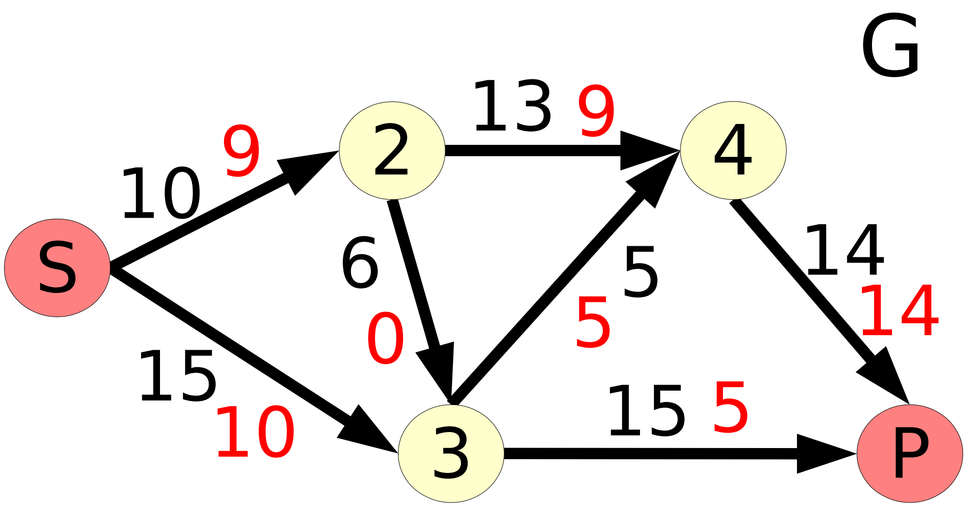
r: capacité résiduelle

$f \leftarrow f + r$

Si $u^{+} \in X$, $f_u \leftarrow f_u + r$

Si $u^{-} \in X$, $f_u \leftarrow f_u - r$

Ford-Fulkerson – une étape

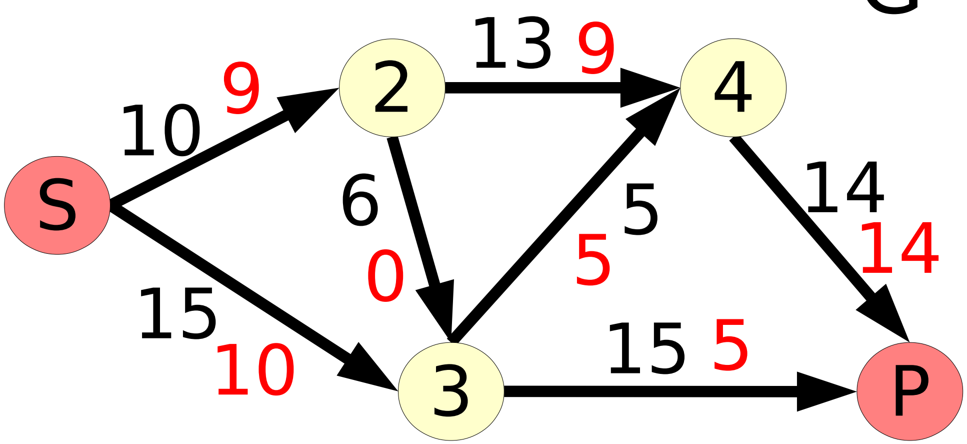


capacité flux

$f = 19$

construire $G^{\Delta}(f)$

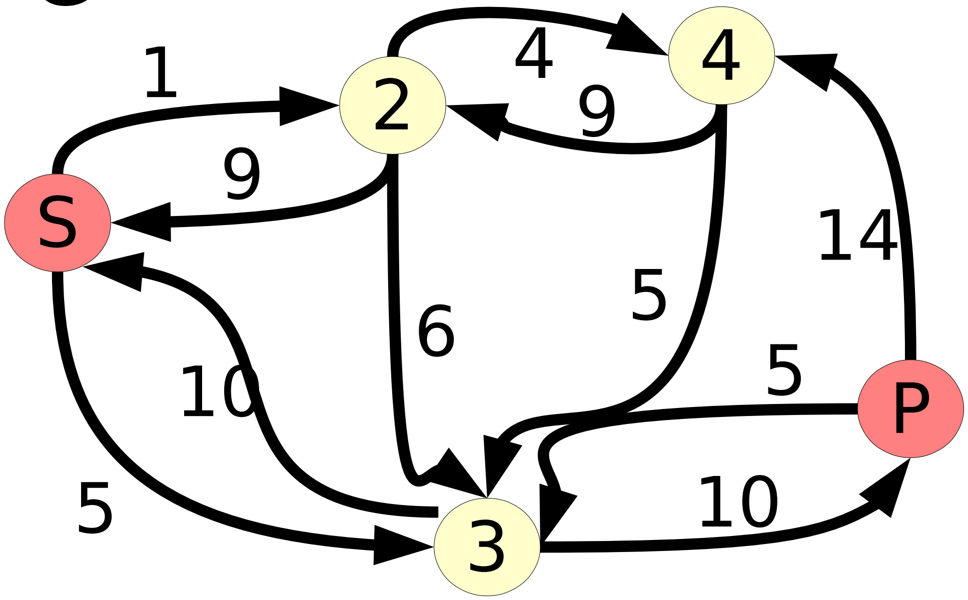
Ford-Fulkerson -- exemple



capacité flux

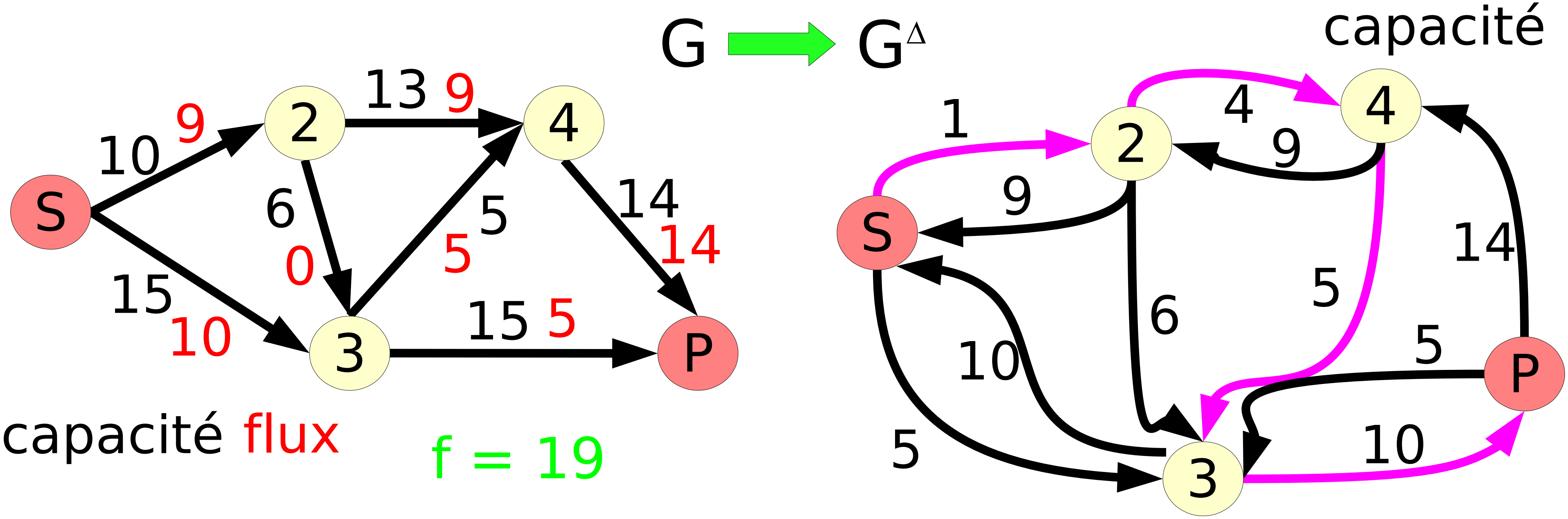
f = 19

G ⟹ G^Δ



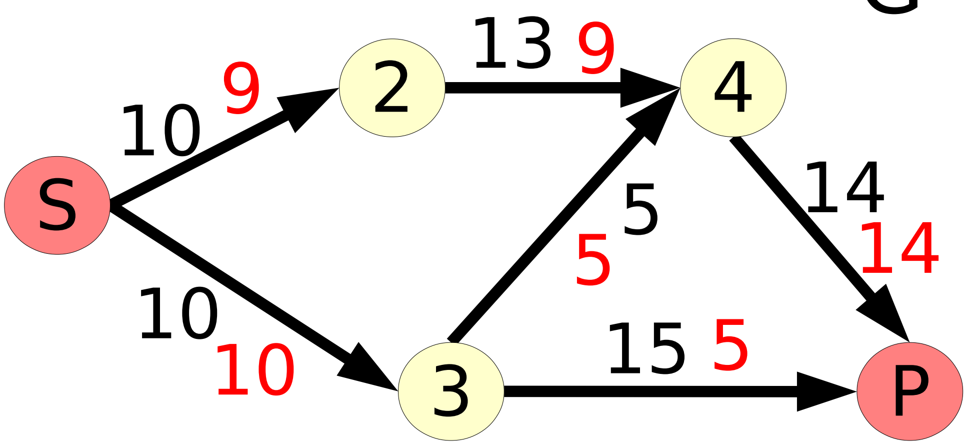
Trouver un chemin A
de S $\rightarrow$ P dans $G^\Delta(f)$

Ford-Fulkerson -- exemple



$r \leftarrow \min_{u \in A} c^{\Delta}(u) = 1$

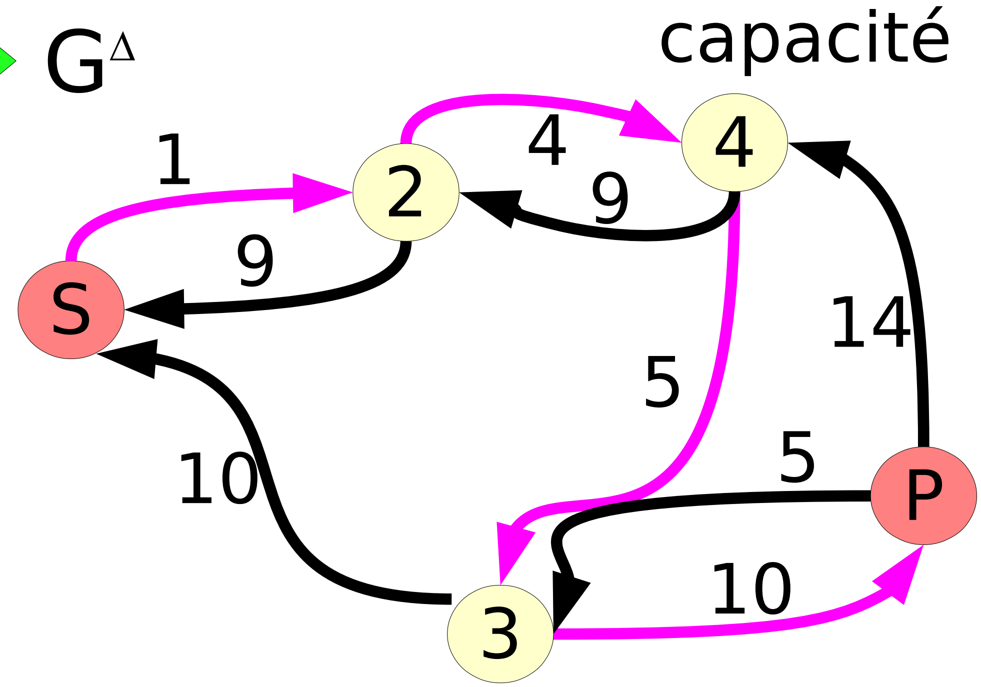
Ford-Fulkerson -- exemple



capacité flux

f = 19

G → G^Δ

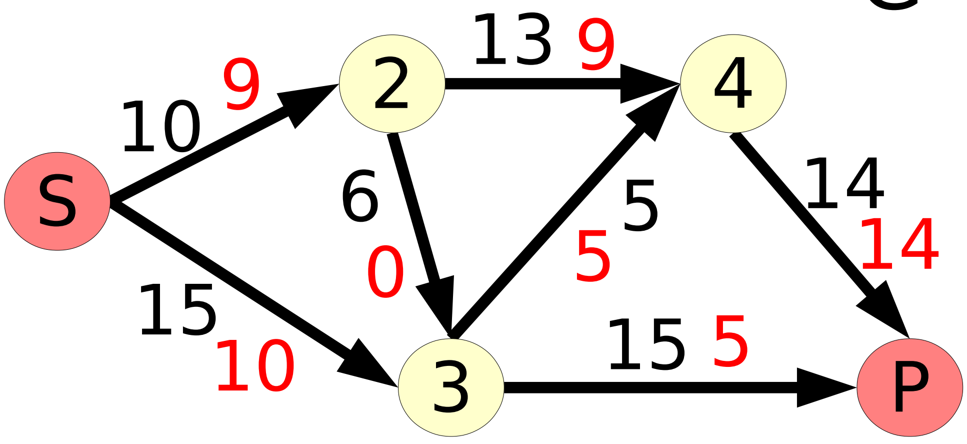


NOTEZ que c'est le seul chemin possible dans ce graphe modifié ...

$$r \leftarrow \min_{u \in A} c^\Delta(u) = 1$$

Ford-Fulkerson -- exemple

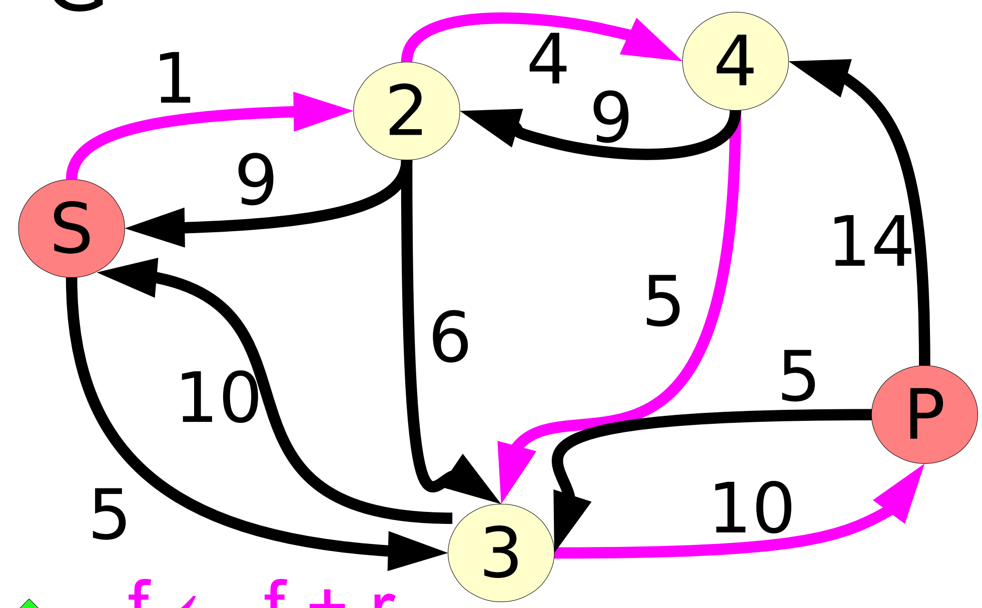
$G \Rightarrow G^{\Delta}$



capacité flux

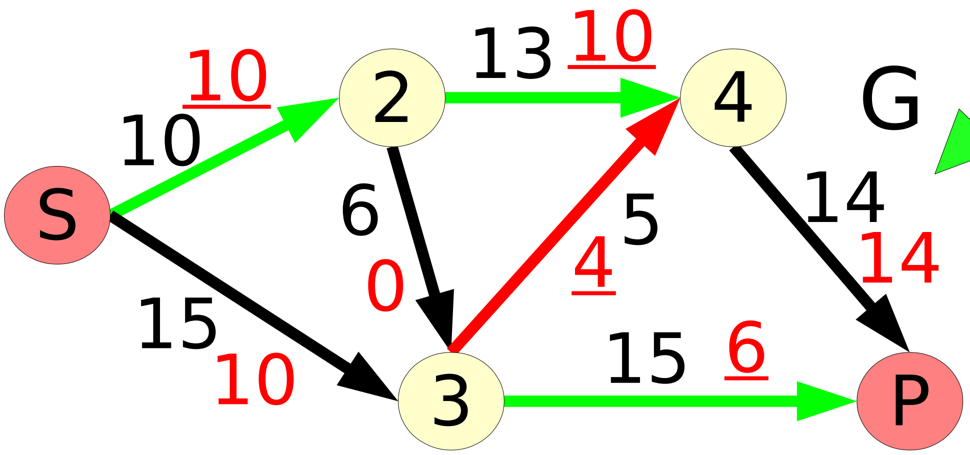
$f = 19$

capacité



$f \leftarrow f + r$

G



flux

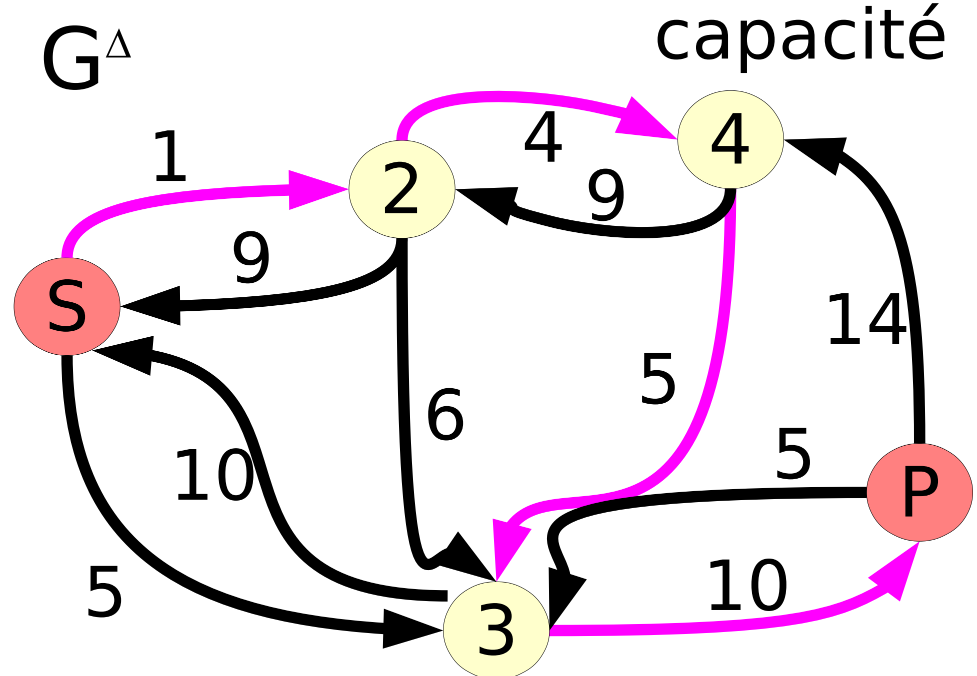
$f = 20$

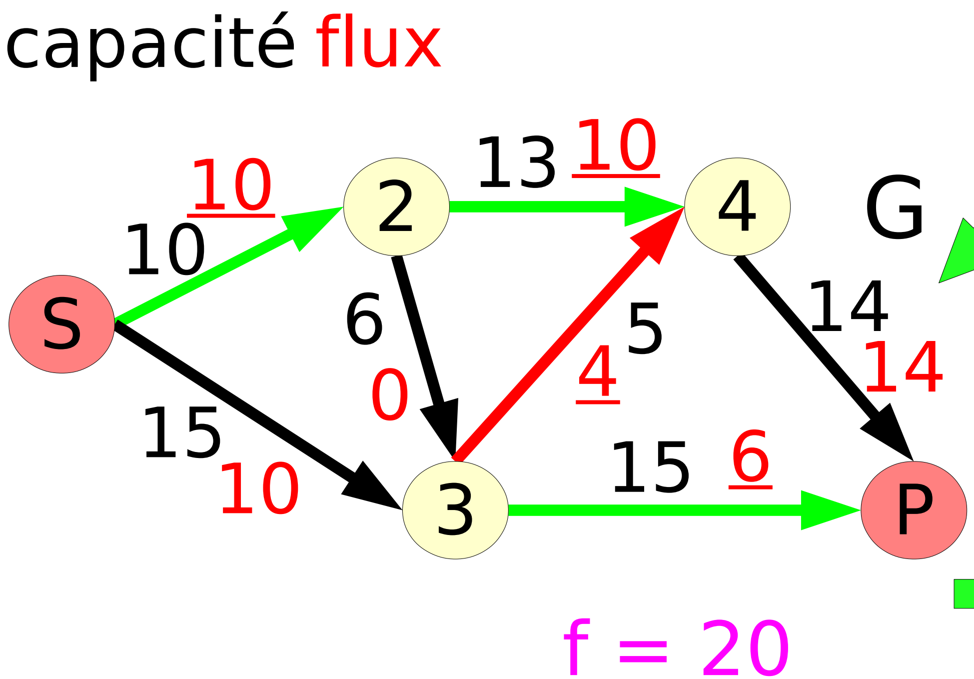
Si $u^{+} \in X$, $\quad f_u \leftarrow f_u + r$

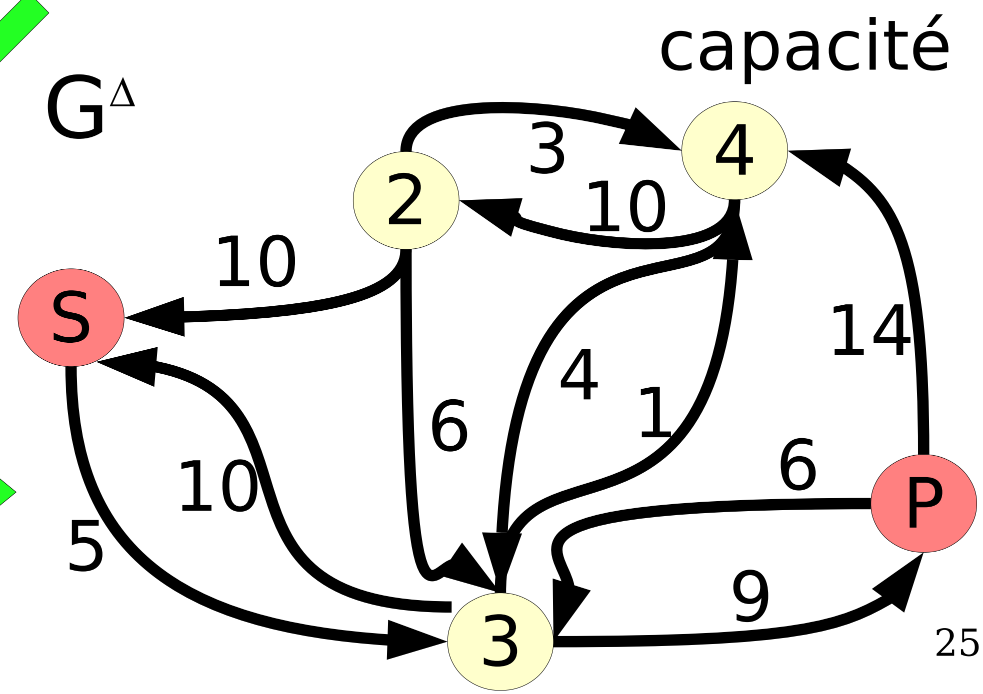
Si $u^{-} \in X$, $\quad f_u \leftarrow f_u - r$

Ford-Fulkerson -- exemple

- Etape suivante : mise à jour de G^Δ
  - à partir de G ou
  - à partir de G^Δ

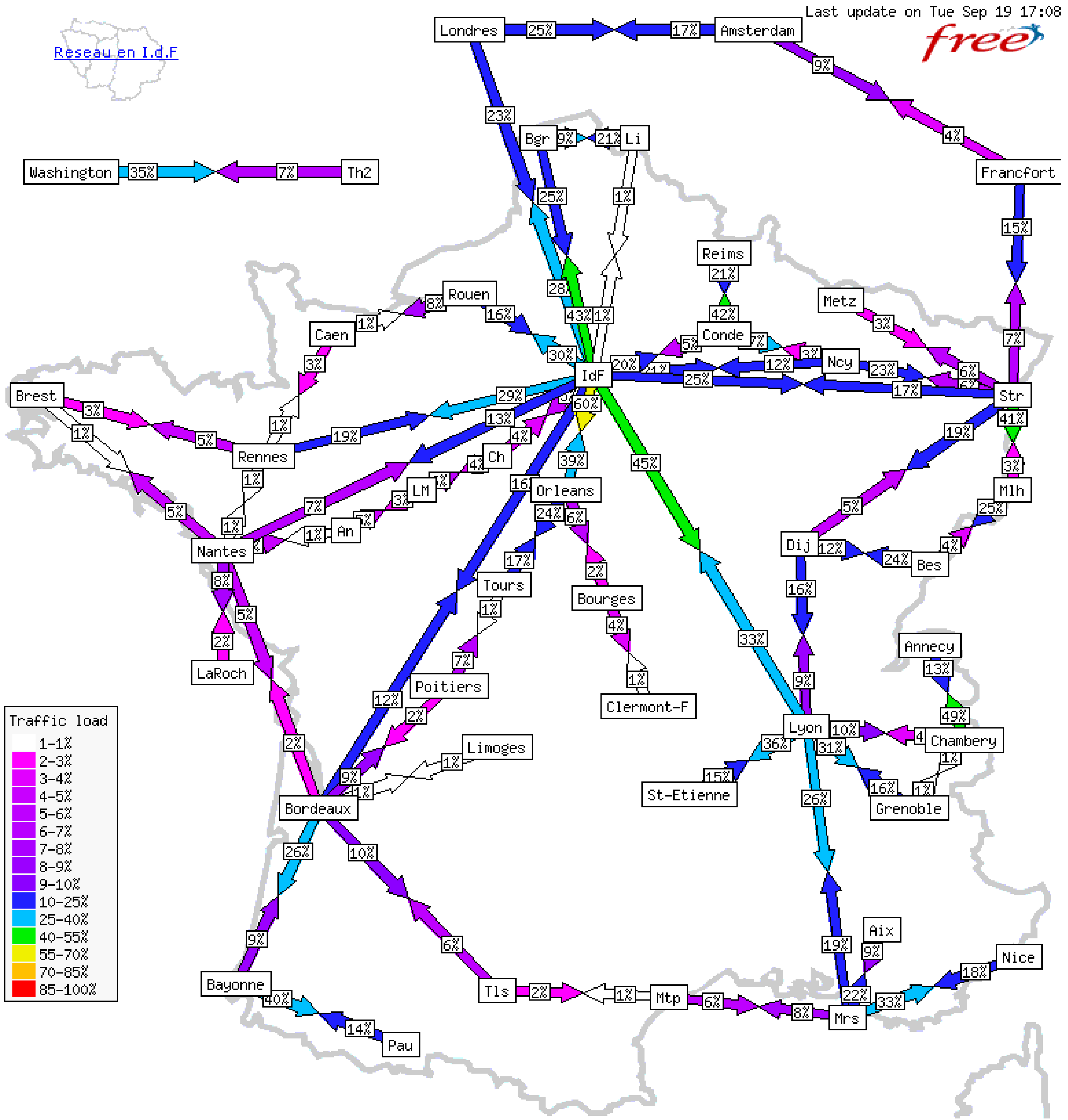






Problèmes de flot

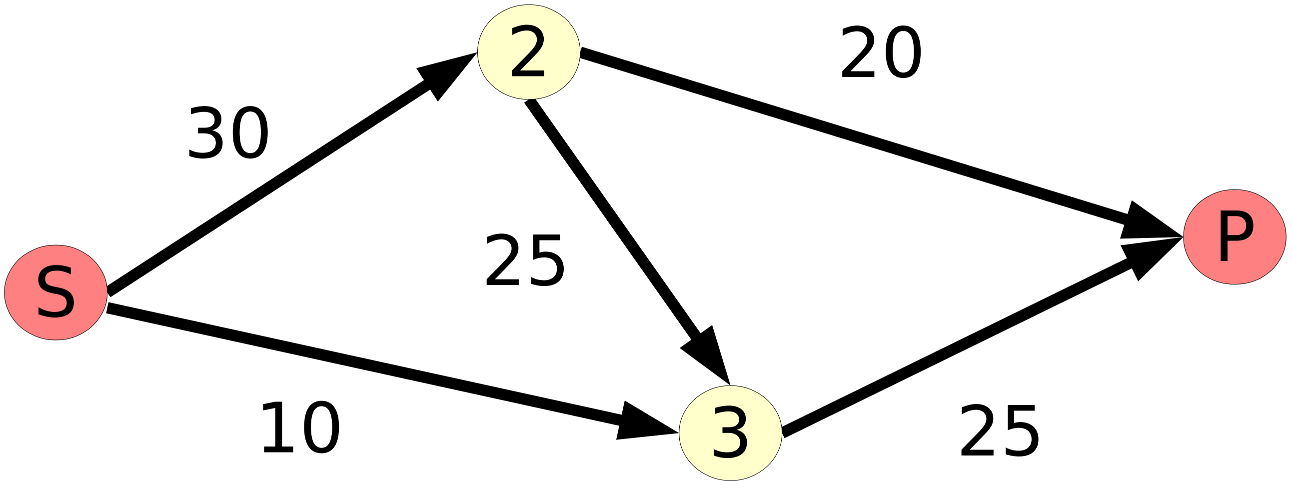
- Quelle est la capacité de transfert entre Brest et Nice ?



Flot maximum - variantes

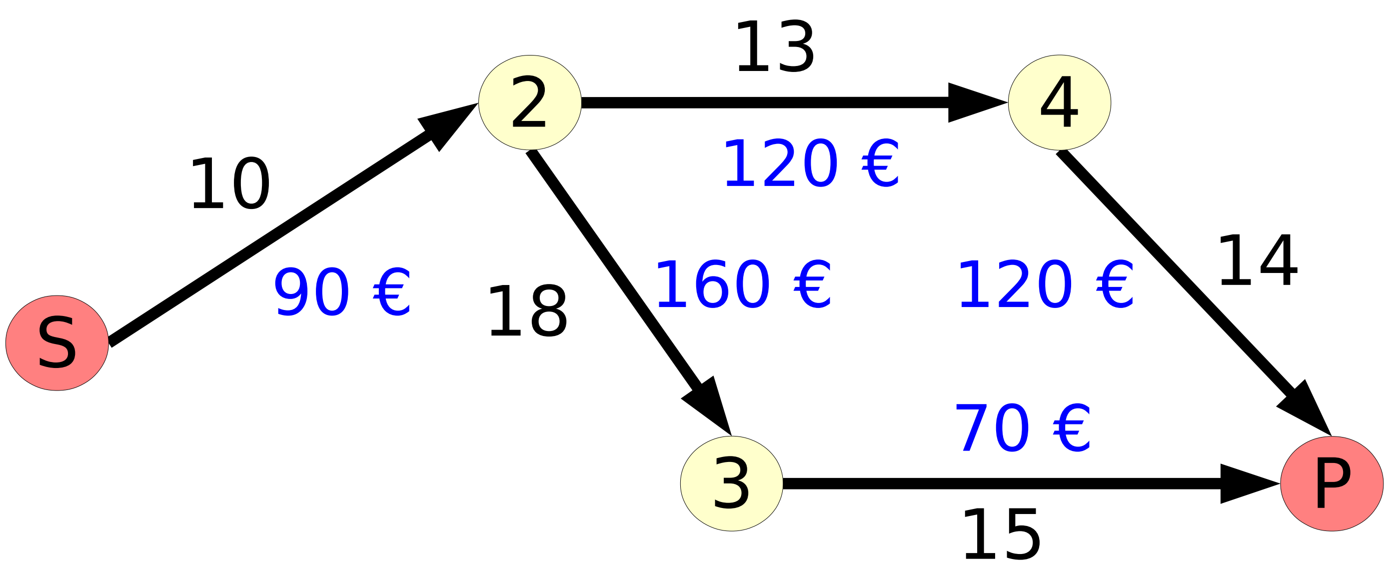
- Maximiser la capacité résiduelle r
  - Algorithme capa Max
    Edmonds-Karp

- Pas toujours optimal en nombre d'étapes



Flot maximum - variantes

- Flot maximum avec coût minimum
  - Coûts associés aux arcs, en plus des capacités
    Busaker et Gowen



Couplage maximal

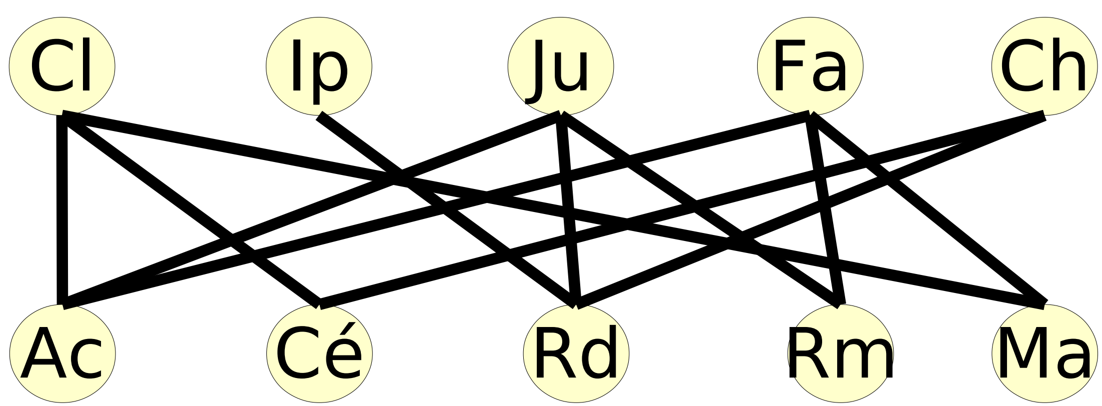
- Associer dans un graphe G=(X, U) les sommets 2 par 2 pour obtenir un couplage maximum :
  - Sommets dits compatibles reliés par des arêtes
  - Sélectionner un sous-ensemble $E \supseteq U$ d'arêtes non adjacentes (couples disjoints) t.q :

$$\sum_{u \in E} w_u \quad \text{est maximum}$$

- Si, $\forall u \in U$, $w_u = 1$, couplage de cardinalité maximale

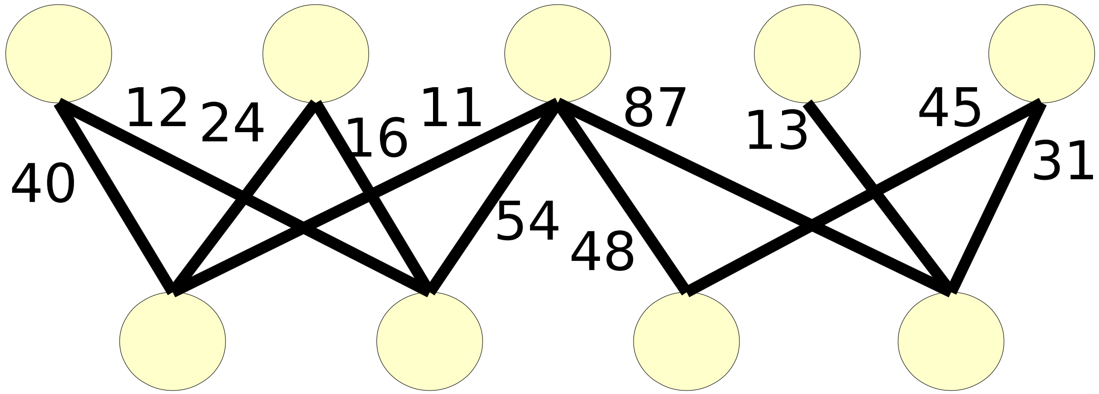
Couplage de cardinalité maximale exemple: maximiser le nombre de duos

	Cléopatre	Iphigénie	Juliette	Fanny	Chimène
Achille					
César					
Rodrigue					
Roméo					
Marius					



Couplage de poids maximal
Problème d'affectation

- Dans un graphe biparti
- On cherche à maximiser ou minimiser le coût du couplage, mesuré par le poids des arêtes sélectionnées



- Résolution par l'algorithme de Busaker et Gowen (il existe aussi la méthode hongroise)

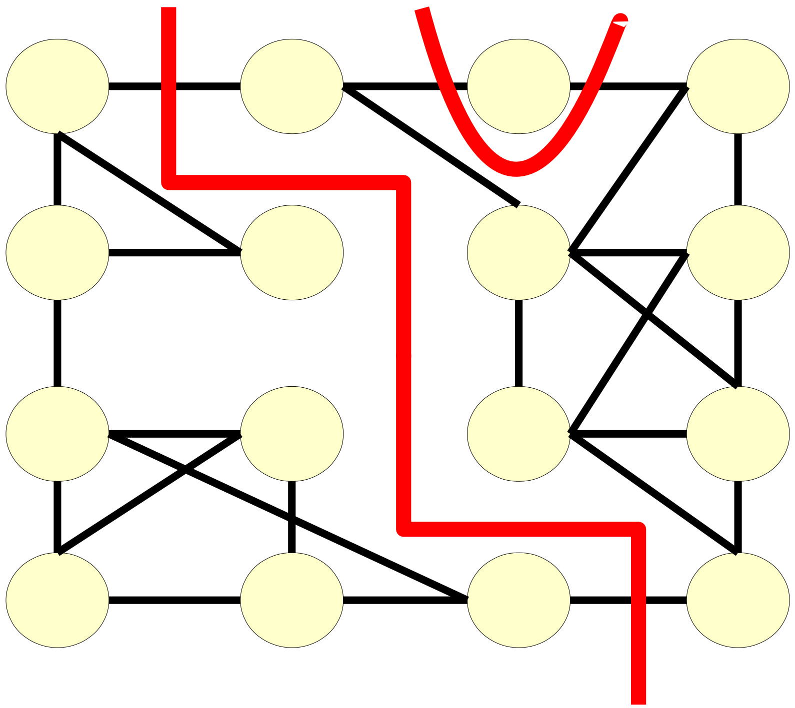
Couplage de poids maximal
Problème d'affectation

- Exemple : qui fait quoi ?

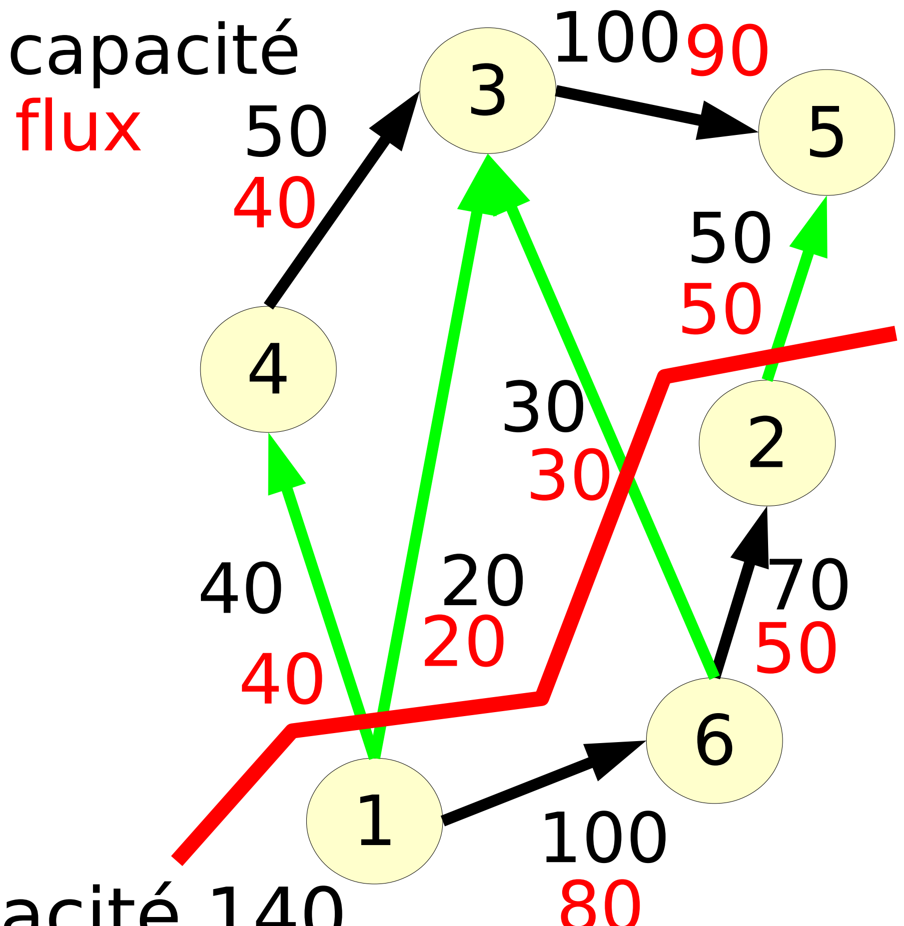
coût E/h	Jardin	Maison	Enfants
Toto	4	3	7
Titi	8	2	5
Tata	3	3	3

Flot maximum - applications
Flot Max / Coupe Min

- Le flot maximum d'un graphe de flot est égale à la coupe de taille/capacité minimale dans ce graphe



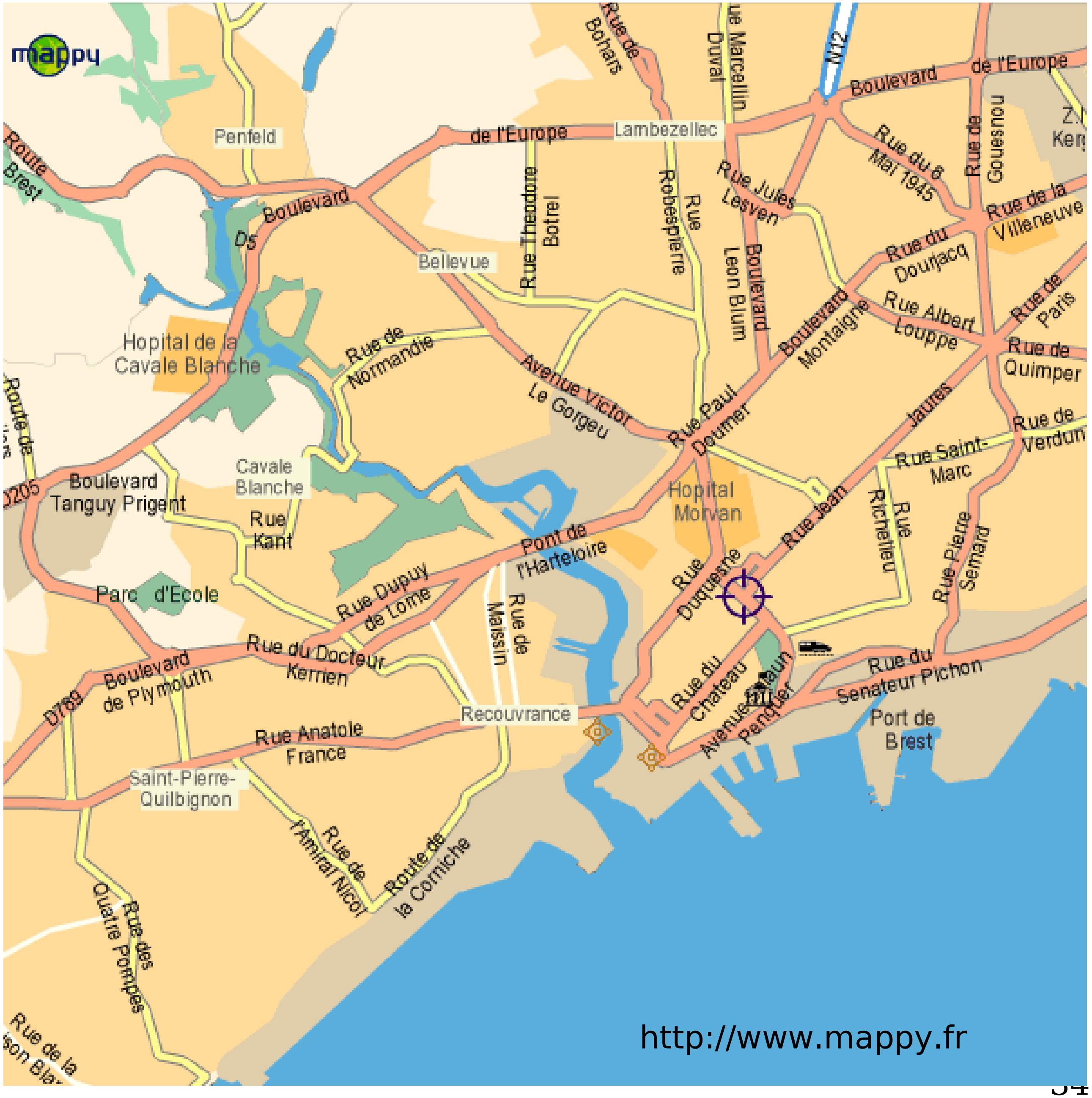
Coupes de taille 2



Coupe de capacité 140

Flot Max / Coupe Min

- Où sont le flot max et la coupe min dans Brest centre ?

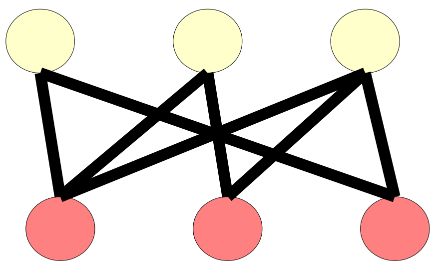


Coloration de graphes

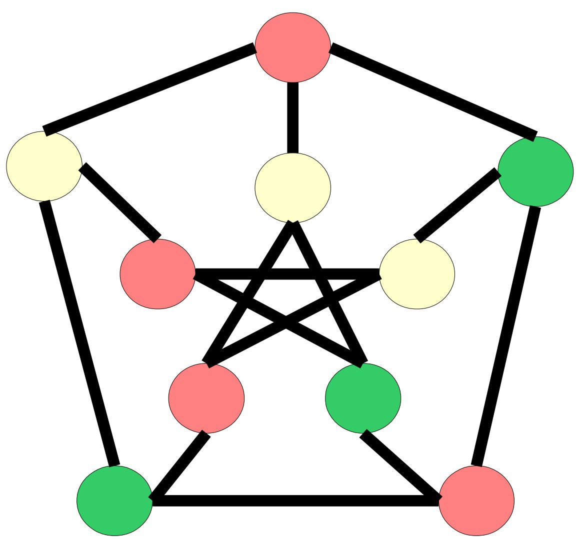
- On appelle **k-coloration** d'un graphe G = (X, U), la répartition des sommets de X en k ensembles disjoints $X_1, X_2, \ldots, X_k$ t.q. :
  - $\forall$ S $\in$ X, S $\in$ X_i $\Rightarrow C_S = i$, i $\in$ [1..k] est la couleur associée à S
  - $\forall$ u = (S1 $\rightarrow$ S2) $\in$ U, $C_{S1} \neq C_{S2}$. Deux sommets adjacents sont de d'une couleur différente

- **Nombre chromatique** : nombre de couleurs minimum pour colorer l'ensemble du graphe

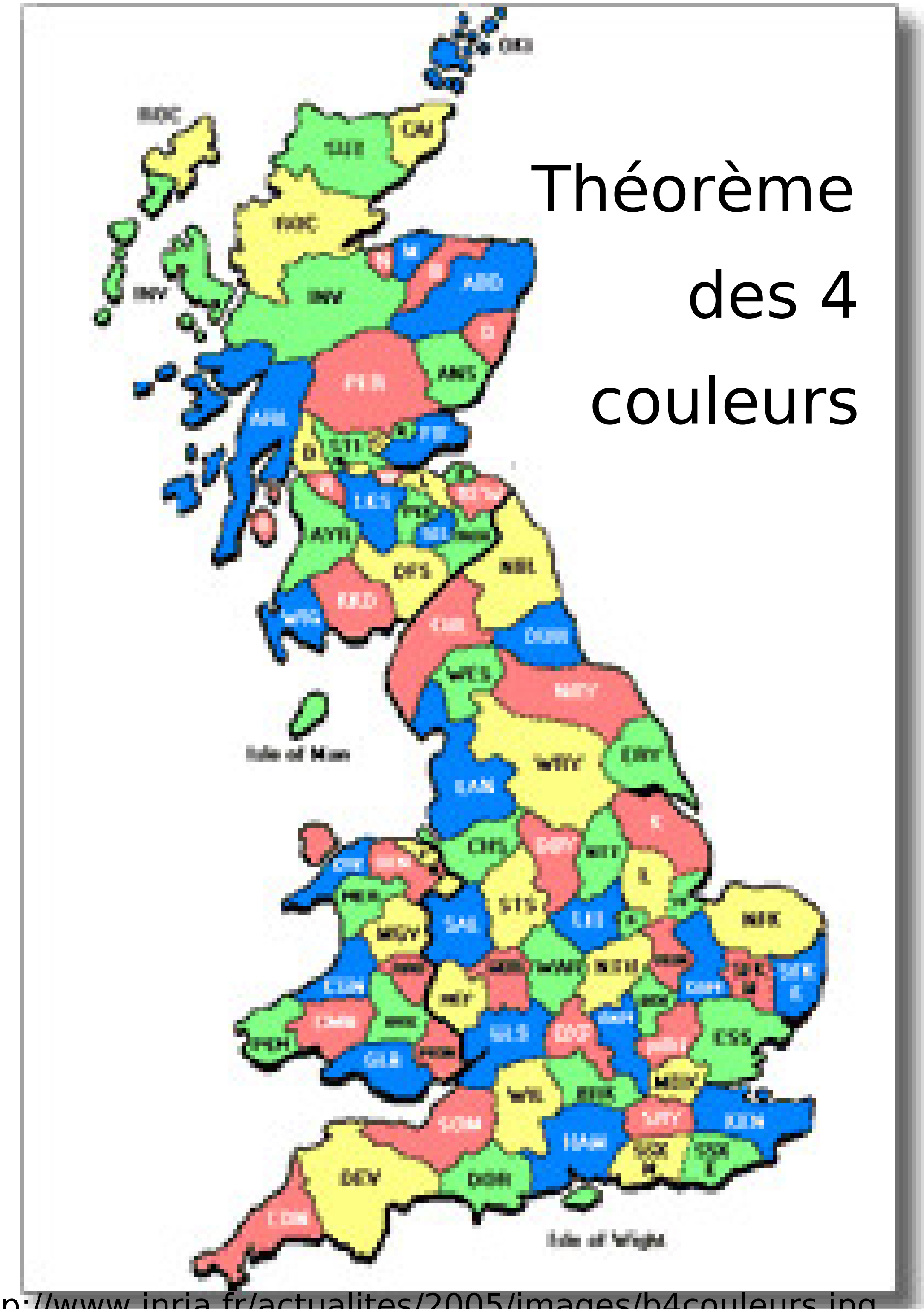
Exemples



Graphe biparti



Graphe de Pertersen



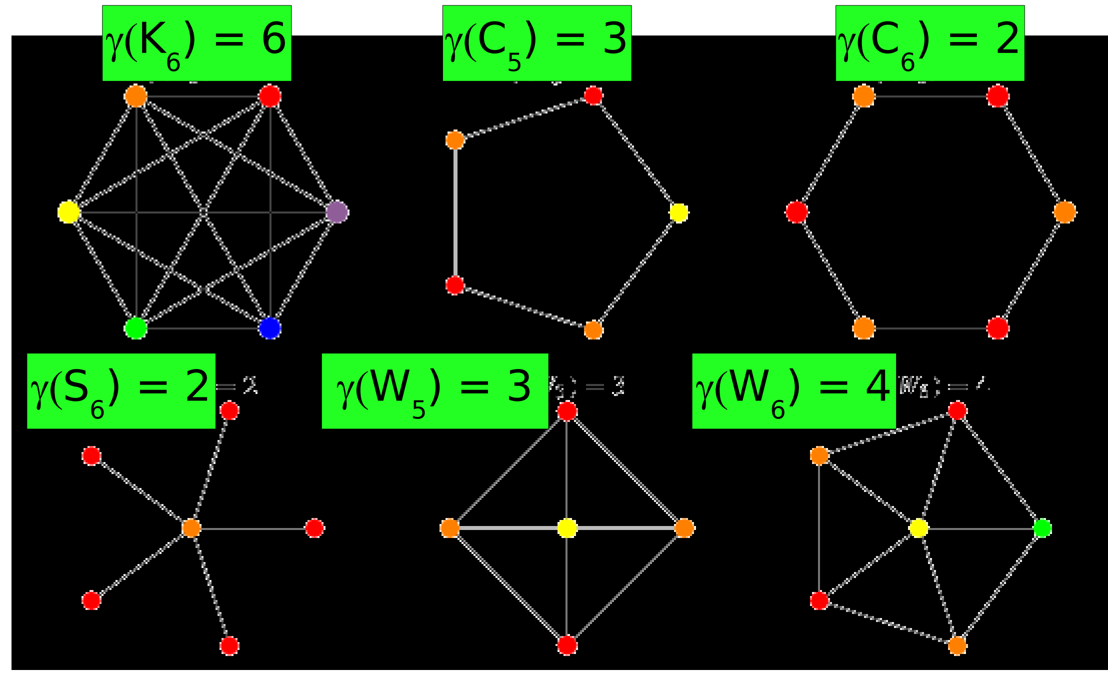
http://www.inria.fr/actualites/2005/images/b4couleurs.jpg

Nombre chromatique γ(G)

Théorème de Brooks :
Pour un graphe G, $\gamma(G) \leq \Delta$
(Δ : degré max des sommets)

Exceptions – $\gamma(G) = \Delta+1$:

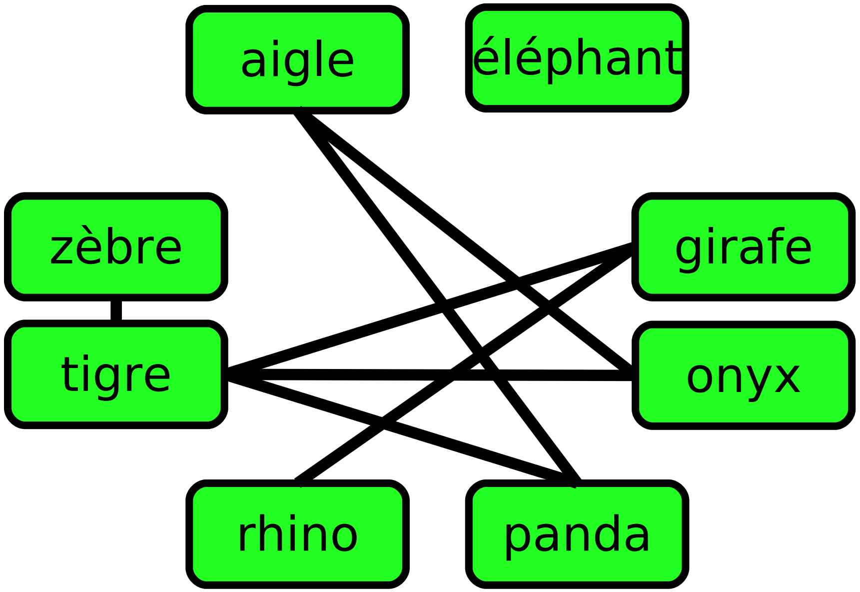
- Cycle impair
- Graphe complet



graphe G	γ(G)
graphe complet K_n	n
cycle C_n, n>1	3 si n impair 2 si n pair
étoile S_n, n>1	2
roue W_n, n>2	3 si n impair 4 si n pair

http://mathworld.wolfram.com/ChromaticNumber.html

Applications



Animaux compatibles au zoo

- Coloration générale
  - Couverture de graphes de compatibilité (classes disjointes d'éléments, par ex. EDT, allocation de ressources)
- Théorème des 4 couleurs
  - Coloration de cartes
    Téléphonie mobile , ...

Réduire les fréquences utilisées pour couvrir une région par un ensemble de cellules

http://research.yale.edu/.../articles-logic-graph.gif

Algorithme glouton - énoncé

coloriage(Graphe G = (X, U))(tableau C de couleurs)

1-Initialisation

couls $\leftarrow$ 0

$\forall S_i \in X$, C[i] $\leftarrow$ 0

2-Itération courante

Tant que

$\exists\, S_i \in X$ non colorié

Si $\exists$ couleur c $1 \leq c \leq$ couls t.q.

$\forall S_j \in$ adjacents(S_i), C[j] $\neq$ c

Alors

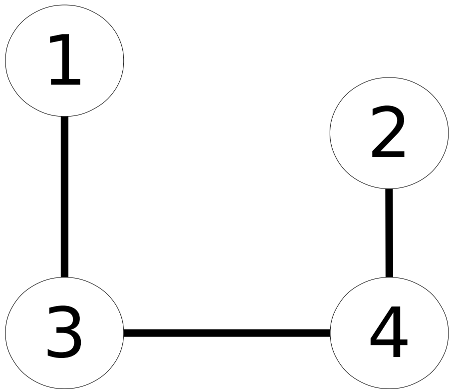
C[i] $\leftarrow$ c

Sinon

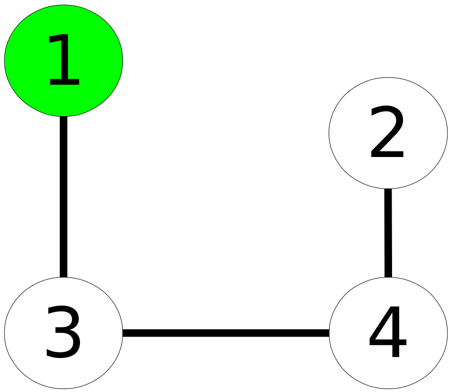
couls $\leftarrow$ couls + 1

C[i] $\leftarrow$ couls

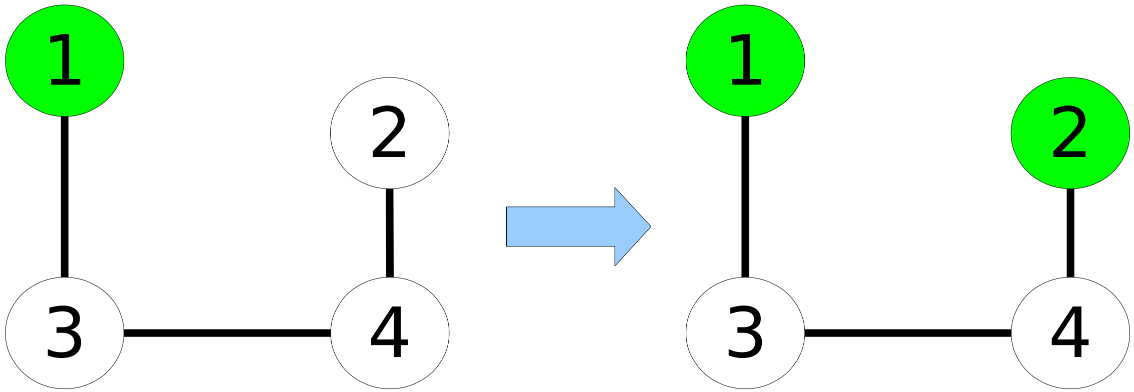
Exemple



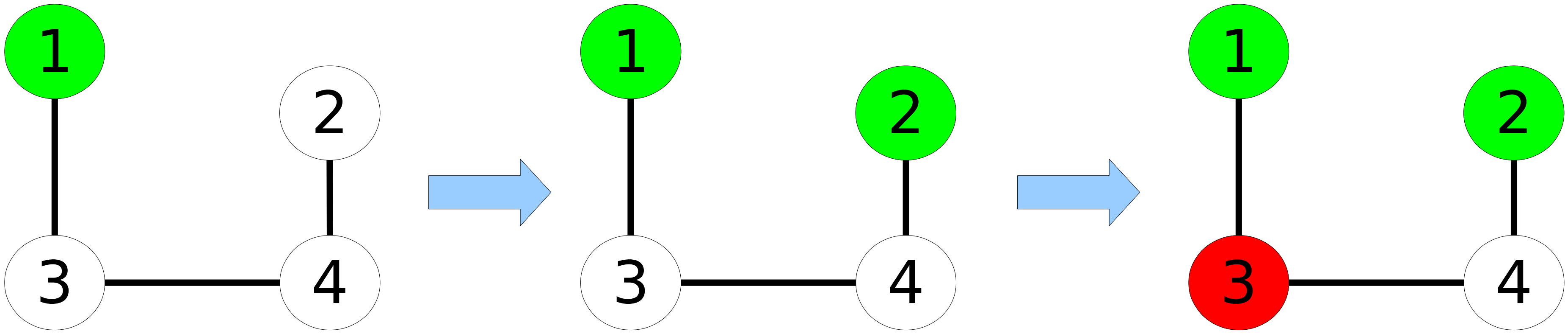
Exemple



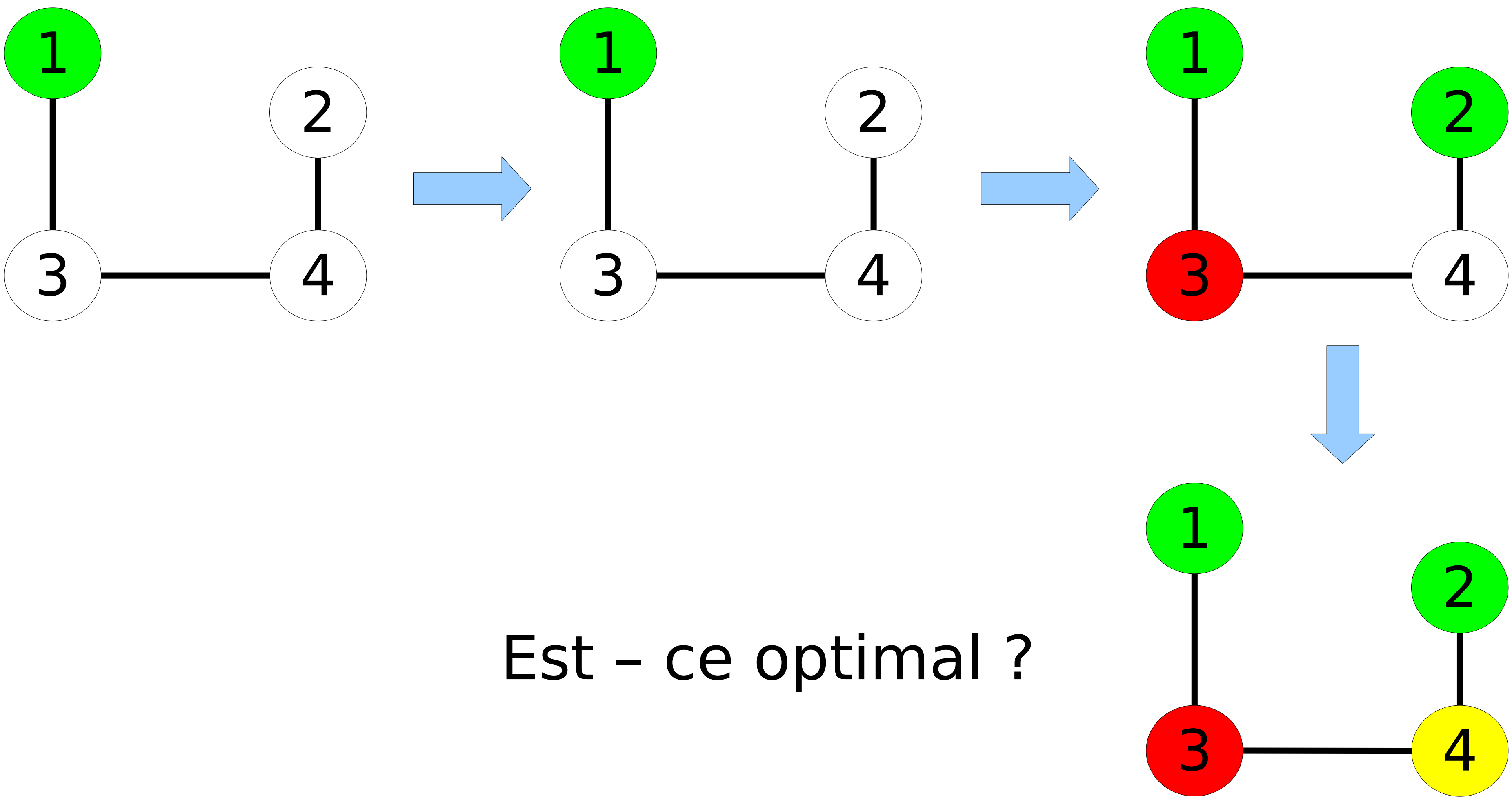
Exemple



Exemple



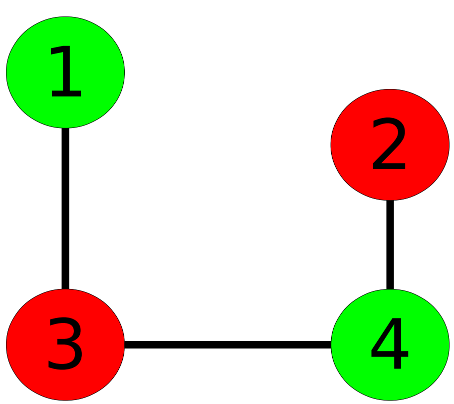
Exemple



Est – ce optimal ?

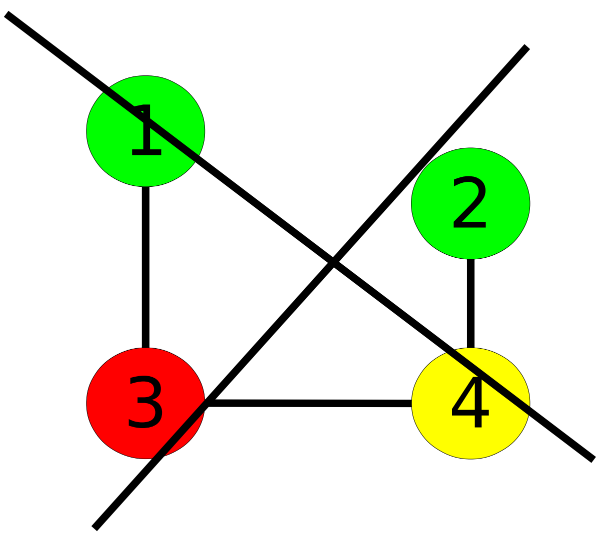
Conclusion sur l'exemple

- Heuristique (donne une solution approchée)
  - Rapide (O(n))
  - Meilleure solution non garantie (depend de l'ordre d'examen des sommets)
- Algorithme optimal en $O(e^n)$



Ok

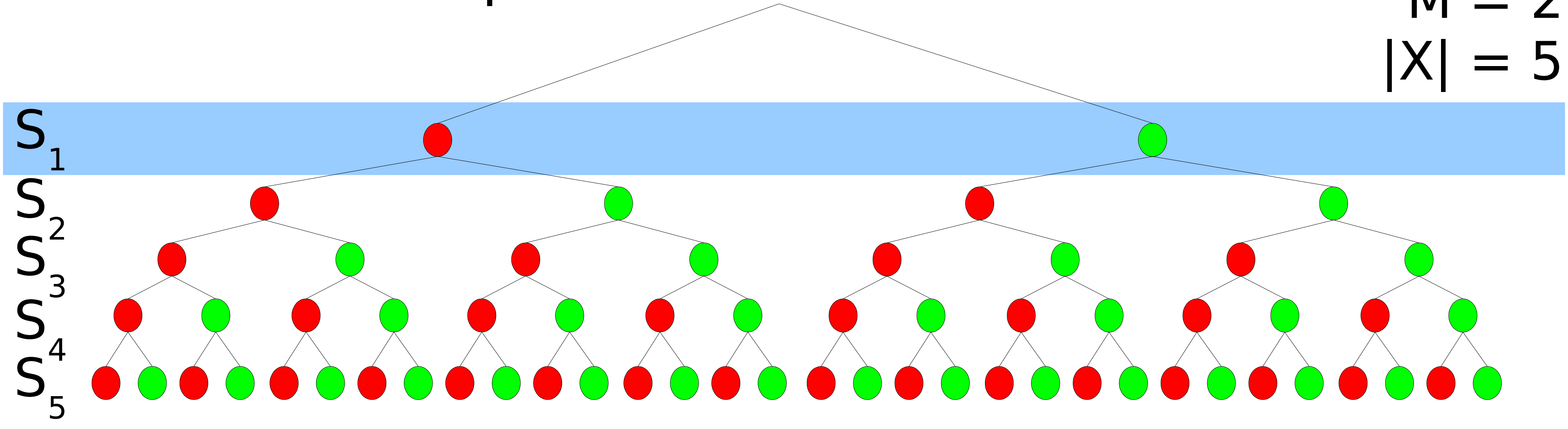
Non !



Coloration de graphe
Algorithme par énumération

- Toutes les colorations en au plus M couleurs pour G= (X, U)
  - Enumerer toutes les solutions ($|X|^M$)
  - Sommets déjà colorés
    Colorer le sommet suivant avec toutes les couleurs disponibles

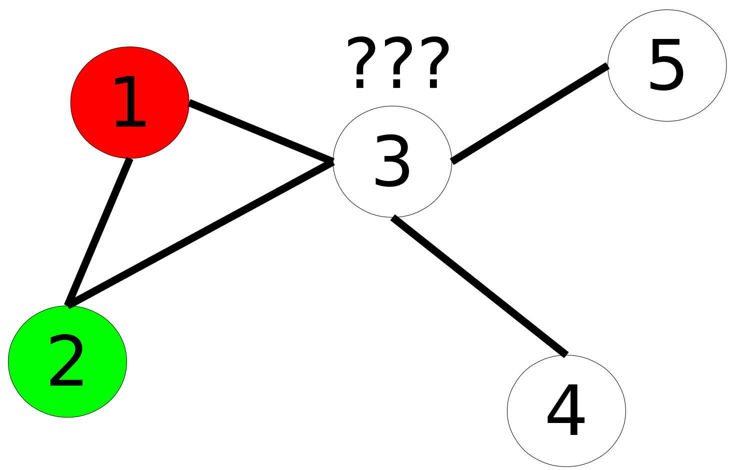
M = 2
|X| = 5

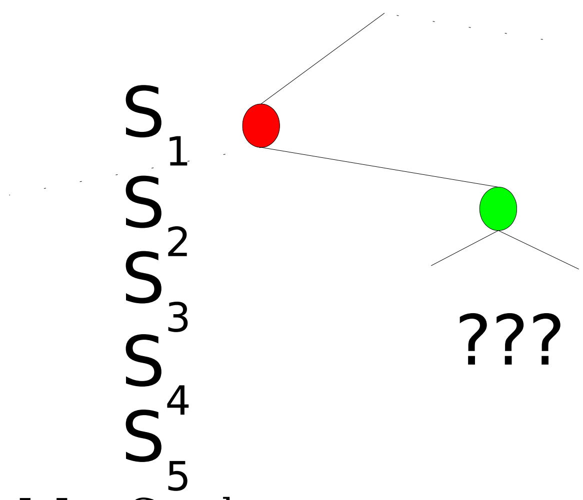


Arbre de recherche

Coloration de graphe
Enumération implicite

- Dans certains cas, inutile de colorier tous les sommets

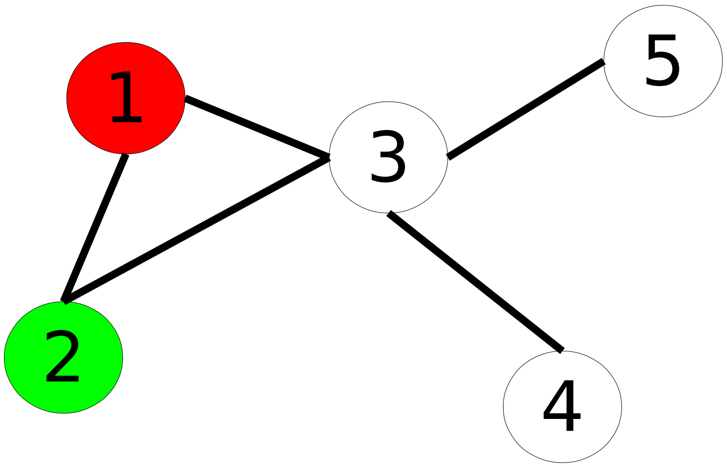


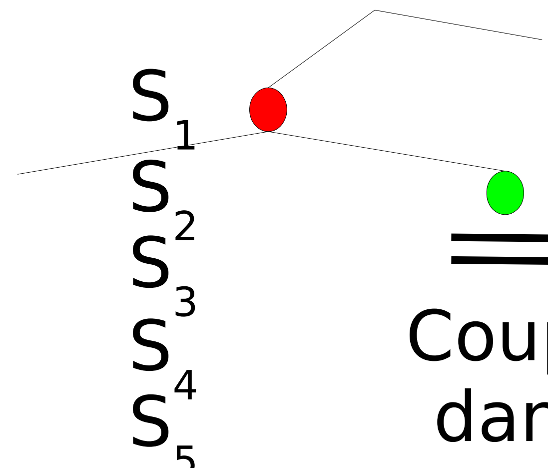


M = 2
|X| = 5

Coloration de graphe
Enumération implicite

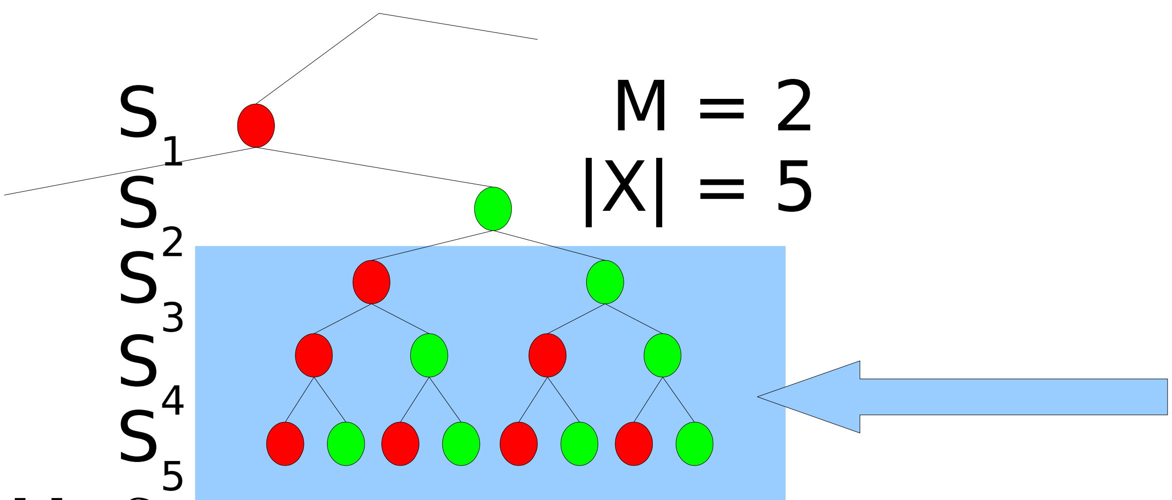
- Dans certains cas, inutile de colorier tous les sommets





M = 2
|X| = 5

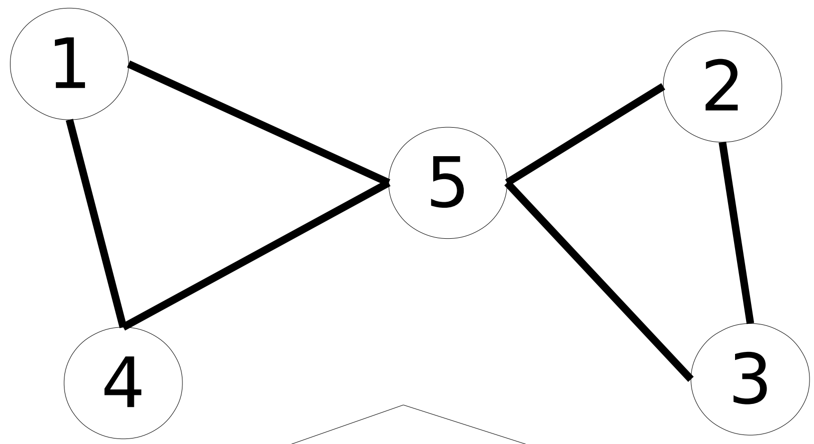
Coupe
dans
l'arbre de
recherche



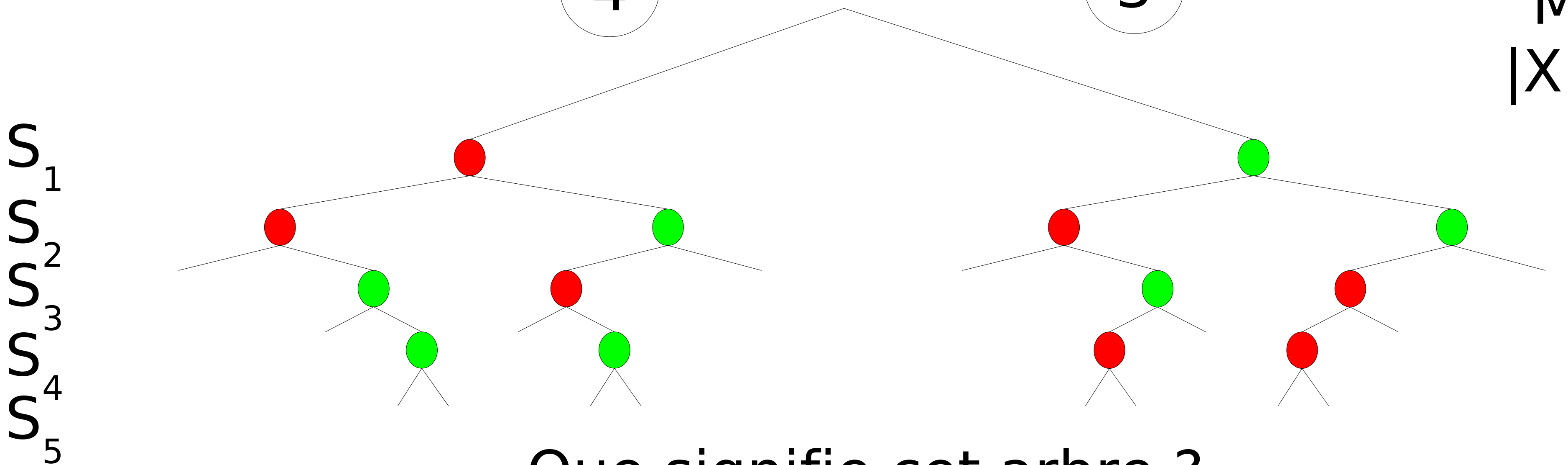
Solutions énumérées
implicitement

Coloration de graphe
Arbre de recherche

- Toutes les colorations en au plus M couleurs pour G= (X, U)



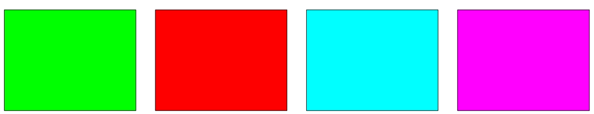
M = 2
|X| = 5

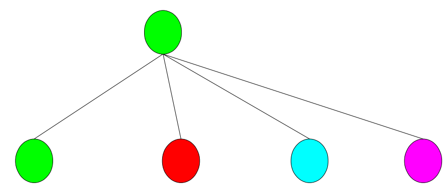


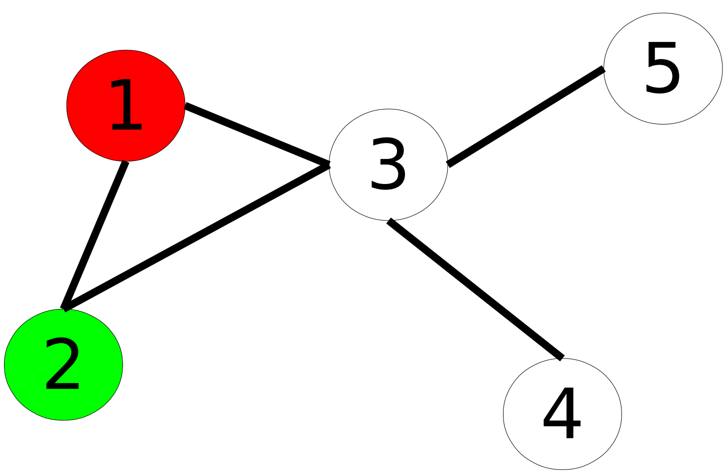
Que signifie cet arbre ?

Algorithme par énumération

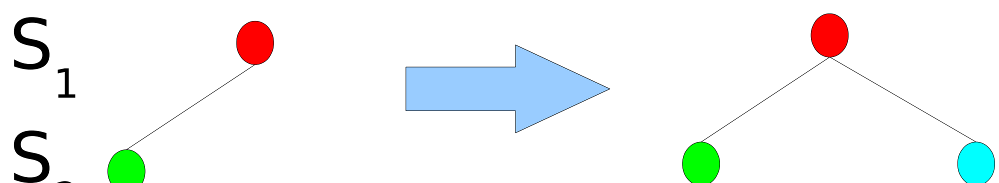
- C[i] = $0 \leq c \leq M+1$
  - 0 pour non colorié
  - M+1 si impossible à colorier
- suivant(C, s) détermine la couleur suivant pour S_s







suivant(C, 2) :



2	1	0	0	0

2	3	0	0	0

Algorithme par énumération

- $C[i] = 0 \leq c \leq M+1$
  - 0 pour non colorié
  - M+1 si impossible à colorier
- suivant(C, s) détermine la couleur suivant pour S_s

coloriage(Graphe G = (X, U), M)(tableau C de couleurs)

1-Initialisation

$\forall S_i \in X, C[i] \leftarrow 0$ — Aucune couleur attribuée

$s \leftarrow 1$ — Sommet courant

$fin \leftarrow faux$

Algorithme par énumération

2-Itération courante

suivant(C, s)

<u>cas 1</u> : (C[s] ≤ M) et (s < |X|)

    s ← s+1

<u>cas 2</u> : (C[s] ≤ M) et (s = |X|)

    afficher(C)

<u>cas 3</u> : (C[s] > M) et (s > 1)

    C[s] ← 0

    s ← s-1

<u>cas 4</u> : (C[s] > M) et (s = 1)

    fin ← vrai

Avancer dans l'arbre

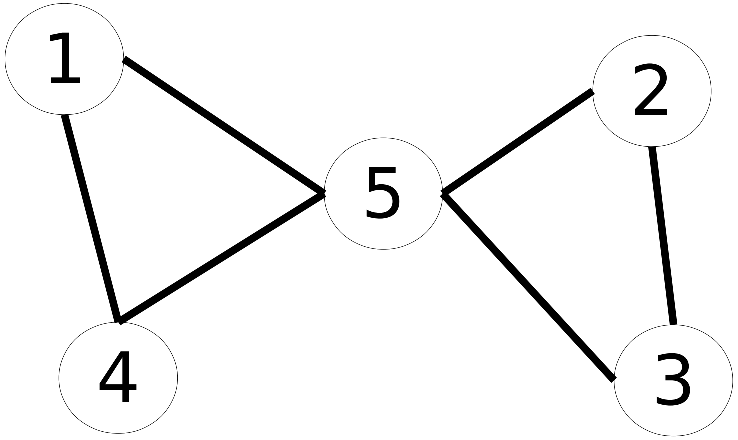
Une solution trouvée

Non coloriable

Retour à la racine

Algorithme par énumération déroulement

- Dérouler l'algorithme sur le graphe suivant ...



	C						
#	C[1]	C[2]	C[3]	C[4]	C[5]	Cas	S
Init	0	0	0	0	0	1	0
1	1	0	0	0	0	1	1
2	1	1	0	0	0	1	2
3	1	1	2	0	0	1	3
...	...	...	...	...	...	...	...

Coloration de graphe application

- Emploi du temps
  contrainte de partage de ressource entre une tâche T_i et une tâche T_j

- Graphe G = (X, U)
  - X, ensemble des tâches T_i
  - $u = (T_i \to T_j) \in U$ si T_i et T_j ne peuvent s'exécuter simultanément

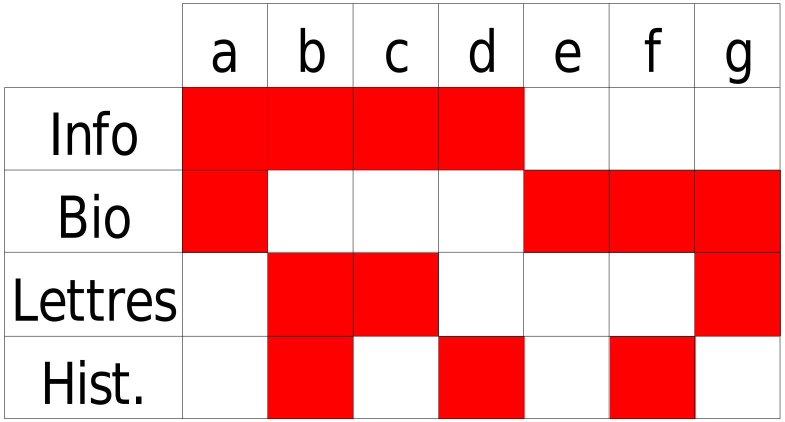
- Trouver une affectation des tâches respectant les contraintes d'exclusion, sachant que toutes les tâches ont le même temps d'exécution

Coloration de graphe exemple

- Examens pour les étudiants :
  - a, b, c et d : info
  - a, e, f, g : bio
  - e, f, g : lettres
  - b, d, f : histoire
- Minimiser le nombre de creneaux d'examens
  - formulation du problème d'optimisation
  - graphe associé
  - application

Coloration de graphe exemple

- Examens pour les étudiants :
  - a, b, c et d : info
  - a, e, f, g : bio
  - e, f, g : lettres
  - b, d, f : histoire



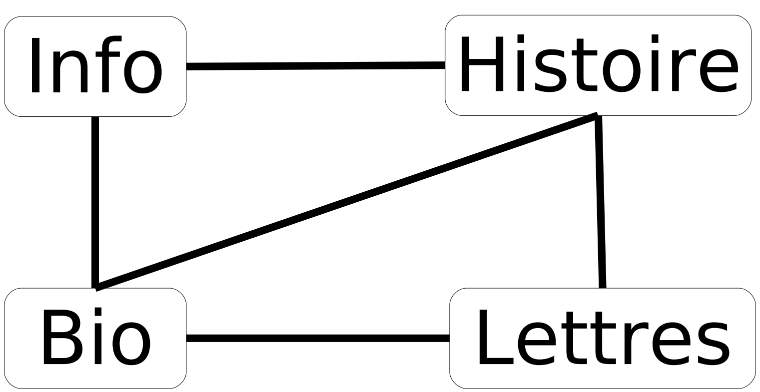
- Formulation du problème d'optimisation
  - Correspond à trouver le nombre chromatique du graphe d'incompatibilité

Coloration de graphe exemple

- Examens pour les étudiants :
  - a, b, c et d : info
  - a, e, f, g : bio
  - e, f, g : lettres
  - b, d, f : histoire

	a	b	c	d	e	f	g
Info							
Bio							
Lettres							
Hist.							

- graphe associé

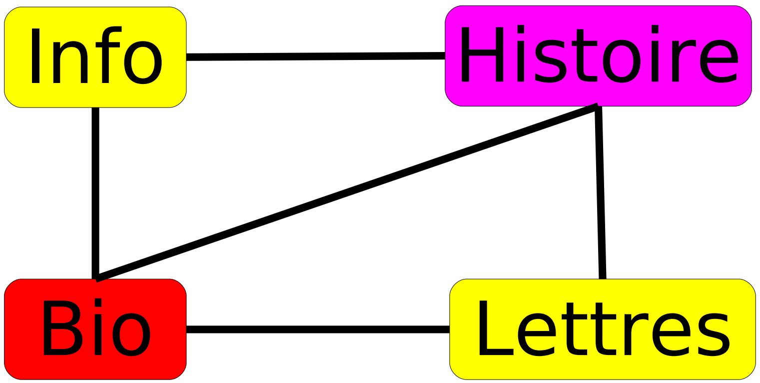


incompatibilités

	Bio	Lettres	Hist.
Info			
Bio			
Lettres			

Coloration de graphe exemple

- Examens pour les étudiants :
  - a, b, c et d : info
  - a, e, f, g : bio
  - e, f, g : lettres
  - b, d, f : histoire
- 3 creneaux d'examens, info même temps que lettres



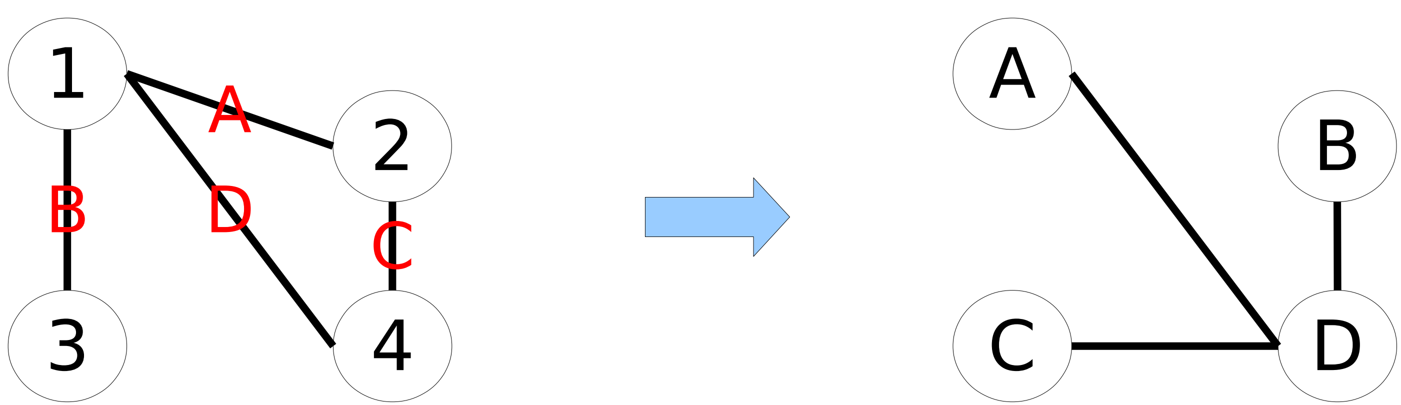
Coloration de graphe
un autre exemple

6	8		3	5				
				1			2	
		7		6		9		
1								
3		5		9		1		2
								9
		3		2		8		
	6			7				
				4	5		3	1

- Carré Latin (Sudoku)
  - Un carré A de taille N x N contient sur chaque ligne et colonne les nombres de 1 à N une et une seule fois
- 9 couleurs (1 par nombre de la grille)
- 81 sommets de type (x,y)
  - 2 sommets de la même ligne, 2 sommets de la même colonne sont adjacents
  - 2 sommets d'un même groupe sont adjacents

Coloration de graphe
Coloriage des arêtes

- Colorier les arêtes : 2 arêtes adjacentes ne peuvent avoir la même couleur

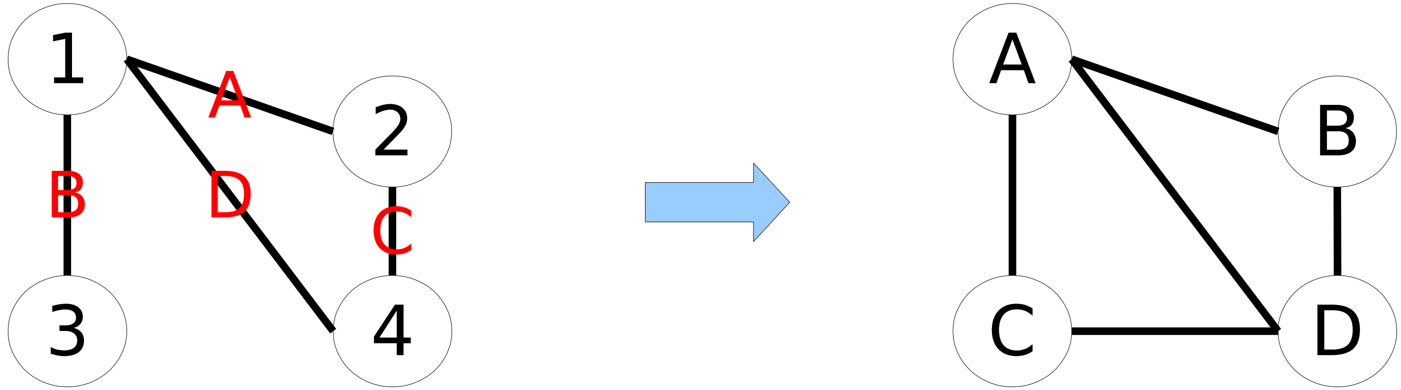


Graphe G

Arête D adjacente donc incompatible avec A, B et C

Coloration de graphe
Coloriage des arêtes

- Colorier les arêtes : 2 arêtes adjacentes ne peuvent avoir la même couleur



Graphe G' à colorier

Coloration de graphe
définition

- [rappel] Nombre chromatique
  - Nombre minimum de couleurs pour colorier *les sommets* d'un graphe.

- Indice chromatique
  - Nombre minimum de couleurs pour colorier *les arêtes* d'un graphe.

Coloration de graphe
preuve de l'existence d'une grille

- Le coloriage permet de prouver que le carré latin de la matrice A existe
- Graphe biparti G = (X, U) avec $X = L \cup C$,
  - L correspond aux indices de lignes, C aux indices de colonnes de A.
  - G est complet ($\forall x_1 \in L, \forall x_2 \in C, (x_1 \leftrightarrow x_2) \in U$)

Coloration de graphe
preuve de l'existence d'une grille

- [propriété] Dans un graphe biparti, de degré maximal des sommets Δ, l'indice chromatique est Δ.
  - $x_1 \in$ L adjacent à tous les sommets de C
    $\rightarrow$ clique, donc nombre chromatique dans G' = Δ

- Dans le graphe de la matrice, Δ = N. La coloration se fait en N couleurs, correspondant chacune à un des nombres du carré.

  arête ($x_a \leftrightarrow x_b$) est coloriée en couleurs[i],

  $\Rightarrow A_{ab} = i$

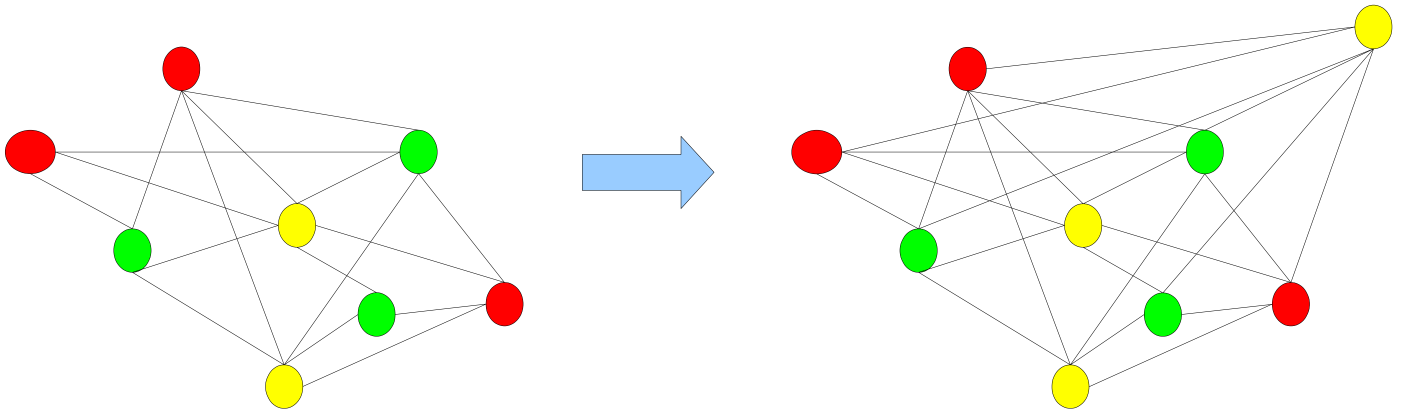
Coloration de graphe
un 3ème exemple

- Mot de passe à cryptage public
  - Graphe 3-colorable public, qui sert de login
  - Coloriage sert de mot de passe
    - La difficulté (pb NP-complet) sert de preuve

- Générer un graphe 3-colorable
  - Méthode inductive
- Prouver que l'on sait le colorier sans révéler le coloriage
  - Réutilisable
  - Pas de stockage sur le serveur

Coloration de graphe
générer un graphe 3-couleurs

- A partir d'un graphe 3 couleurs
  - Ajouter un sommet, avec une couleur aléatoire (couleurs 1 à 3)
  - Le connecter à un nombre aléatoire d'autres sommets de couleurs différentes de la sienne



- Graphe assez grand, nombre d'arêtes suffisant pour rendre la solution introuvable en temps limité

Coloration de graphe
vérification de la capacité à colorier

- Choix par l'observateur de 2 sommets au hasard
  - Dévoiler les 2 couleurs (elles doivent être différentes si sommets adjacents)
  - Permuter aléatoirement toutes les couleurs (sans changer de coloriage)
- Au bout de r essais, la probabilité d'une erreur approche 100% si le coloriage n'est pas connu.
  - Permuter les couleurs entre chaque test permet de préserver le secret du coloriage global.
  - Tiers de confiance nécessaire pour faire les permutations