Graphes et algorithmique des graphes

cours: Vincent Bouchitté
rédaction: Brice Goglin
relecture: Jean-Baptiste Rouquier.

École normale supérieure de Lyon
3^{e} année de licence d'informatique

Version provisoire non relue par l'enseignant.

OpenContent License Version 1.0, July 14, 1998.
The original version of this document may be found at http ://opencontent.org/opl.shtml

LICENSE

Terms and Conditions for Copying, Distributing, and Modifying

Items other than copying, distributing, and modifying the Content with which this license was distributed (such as using, etc.) are outside the scope of this license.

1. You may copy and distribute exact replicas of the OpenContent (OC) as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty ; keep intact all the notices that refer to this License and to the absence of any warranty ; and give any other recipients of the OC a copy of this License along with the OC. You may at your option charge a fee for the media and/or handling involved in creating a unique copy of the OC for use offline, you may at your option offer instructional support for the OC in exchange for a fee, or you may at your option offer warranty in exchange for a fee. You may not charge a fee for the OC itself. You may not charge a fee for the sole service of providing access to and/or use of the OC via a network (e.g. the Internet), whether it be via the world wide web, FTP, or any other method.

2. You may modify your copy or copies of the OpenContent or any portion of it, thus forming works based on the Content, and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions :

a- You must cause the modified content to carry prominent notices stating that you changed it, the exact nature and content of the changes, and the date of any change.

b- You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the OC or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License, unless otherwise permitted under applicable Fair Use law.

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the OC, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the OC, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Exceptions are made to this requirement to release modified works free of charge under this license only in compliance with Fair Use law where applicable.

3. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to copy, distribute or modify the OC. These actions are prohibited by law if you do not accept this License. Therefore, by distributing or translating the OC, or by deriving works herefrom, you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or translating the OC.

NO WARRANTY

4. Because the opencontent (oc) is licensed free of charge, there is no warranty for the oc, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the oc "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk of use of the oc is with you. Should the oc prove faulty, inaccurate, or otherwise unacceptable you assume the cost of all necessary repair or correction.

5. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may mirror and/or redistribute the oc as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the oc, even if such holder or other party has been advised of the possibility of such damages.

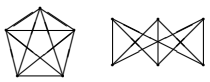
Table des matières

Chapitre 1

Généralités

En 1736, Euler s'intéresse au problème des ponts de Koenigsberg. Il s'agit de savoir s'il existe un chemin passant une fois et une seule par chaque pont de la ville. Ce problème deviendra ensuite le problème du postier qui vise à minimiser la distance parcourue par un postier voulant desservir toute la ville.

En 1856, Hamilton étudie un problème apparemment aussi simple, celui de trouver un chemin passant une fois et une seule par chaque sommet d'un graphe. En fait, ce problème est beaucoup plus difficile à résoudre.



K_5 $K_{3,3}$

Au début des années 1930, Kuratowski s'intéresse à la décomposition des graphes planaires. Il montre qu'on peut «décomposer» en temps linéaire tout graphe non-planaire en graphes K_5 ou $K_{3,3}$.

Le dernier grand résultat concernant les graphes a été obtenu en 1976 par Appel et Haken qui ont montré qu'on peut colorier les faces de tout graphe planaire avec seulement 4 couleurs (les faces tangentes doivent être coloriées avec des couleurs différentes). Cependant, on ne connaît toujours pas de démonstration simple de ce résultat.

Définition (Graphe non-orienté). Un graphe non-orienté est un couple (X, E) où X est un ensemble de ***sommets*** et E est inclus dans $\mathcal{P}_2(X)$ (ensemble des parties à deux éléments de X). Les éléments de E sont appelés ***arêtes***.

Parfois, on autorise les boucles (arêtes de mêmes extrémités), ou les arêtes multiples. L'arête $\{x, y\}$ est souvent notée xy.

Définition (Graphe orienté). Un graphe orienté est un couple (X, U) où X est un ensemble de ***sommets*** et U est inclus dans $X \times X \setminus \{(x, x) \mid x \in X\}$. U est un ensemble d'***arcs***.

On autorise également parfois les boucles ou arcs multiples.

Définitions (Adjacence, incidence, voisinage, degré). Deux sommets x et y d'un graphe $G = (X, E)$ sont ***adjacents*** si $xy \in E$.
Deux arêtes e et f d'un graphe $G = (X, E)$ sont ***adjacentes*** si $e \cap f \neq \varnothing$.
L'arête xy est ***incidente*** aux sommets x et y.
Le ***voisinage*** d'un sommet x est défini par $\Gamma(x) = \{y \in X | xy \in E\}$. Le ***degré*** du sommet x est alors $d(x) = |\Gamma(x)| = Card\ \Gamma(x)$.

Pour un graphe orienté, les définitions peuvent être précisées :

Définitions (Voisinages extérieurs et intérieurs). Soit $G = (X, U)$ un graphe orienté et x un sommet.
Le ***voisinage extérieur*** de x est défini par $\Gamma^+(x) = \{y \in X \mid (x, y) \in U\}$.
Son ***voisinage intérieur*** est $\Gamma^-(x) = \{y \in X \mid (y, x) \in U\}$.
Le voisinage $\Gamma(x)$ est la réunion de $\Gamma^+(x)$ et $\Gamma^-(x)$.

On appelle parfois voisinages ***sortant*** et ***entrant*** les voisinages extérieurs et intérieurs. On peut alors définir les degrés sortant d^+ et entrant d^- par le cardinal des ensembles $\Gamma^+(x)$ et $\Gamma^-(x)$.

Définitions (Point isolé, pendant, dominant). Si $d(x) = 0$, le sommet x du graphe est dit ***isolé***.
Si $d(x) = 1$, le sommet x est dit ***pendant***.
Si $d(x) = |X| - 1$, le sommet x est dit ***dominant***.

Dans toute la suite du cours, on notera $n = |X|$ et $m = |E|$ ou $|U|$.

Proposition 1.1. *Soit $G = (X, E)$ un graphe non-orienté.*

1. $m \leqslant \frac{1}{2}n(n-1)$;
2. $\sum_{x \in X} d(x)$ *est pair ;*
3. *Il y a un nombre pair de sommets de degré impair.*

Démonstration.

1. Le graphe étant supposé sans boucle ni arête multiple, ce résultat est immédiat.
2. La somme des degrés des sommets est le nombre d'extrémités d'arêtes. Chaque arête possède deux extrémités et compte donc pour deux dans la somme des degrés. On a donc $\sum_{x \in X} d(x) = 2m$.
3. Ce résultat se déduit immédiatement du 2. ■

Proposition 1.2. *Soit $G = (X, E)$ un graphe. Il existe deux sommets différents ayant le même degré.*

Démonstration. Supposons que tous les sommets aient des degrés différents. Ces degrés sont des entiers compris entre 0 et $n-1$. Ils prennent donc une fois et une seule chaque valeur entière entre 0 et $n-1$. Il y a donc un sommet isolé et un sommet dominant, ce qui est impossible simultanément. Il y a contradiction. ■

Remarque. Pour un entier n donné, il existe deux uniques graphes différents possédant deux sommets de mêmes degrés et les autres sommets de degrés tous différents.

Définitions (Sous-structures). $G' = (X, E')$ est un ***graphe partiel*** de $G = (X, E)$ si $E' \subset E$.
$G' = (X', E')$ est un ***sous-graphe*** de $G = (X, E)$ si $X' \subset X$ et $E' = E \cap \mathcal{P}_2(X')$.
$G' = (X', E')$ est un ***sous-graphe partiel*** de $G = (X, E)$ si $X' \subset X$ et $E' \subset E \cap \mathcal{P}_2(X')$.

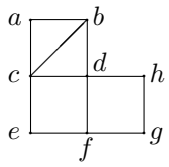
Définitions (Graphes particuliers). On note K_n le graphe complet à n sommets et S_n le graphe sans arête à n sommets.

Un sous-graphe de G isomorphe à K_p est appelé ***clique*** tandis qu'un sous-graphe isomorphe à S_p est appelé ***stable***.

Définition (Parcours, chaîne, cycle, longueur). Un ***parcours*** de G est une suite d'arêtes $\mu = (x_1, \ldots, x_k)$ telle que $x_i x_{i+1} \in E$ pour tout $i \in \{1, \ldots, k-1\}$.
Un parcours $(x_1, \ldots, x_k)$ est ***fermé*** si $x_i = x_k$.
Une ***chaîne*** de $G = (X, E)$ est un parcours sans répétition d'arête.
Un ***cycle*** est une chaîne fermée.
Une ***chaîne élémentaire*** est un chemin sans répétition de sommet.
La ***longueur*** d'une chaîne est le nombre d'arêtes que contient cette chaîne.

Remarque. On peut noter que la longueur d'une chaîne est toujours définie tandis que pour un parcours, ce n'est pas toujours le cas (un parcours peut être infini s'il contient des cycles).

Exemple.



- (a, b, c, d, c, a) est un parcours fermé.
- $(e, f, d, c, b, d, h, g, f)$ est une chaîne.
- (a, b, d, c, a) est un cycle élémentaire.
- (a, b, d, c) est une chaîne élémentaire.

Proposition 1.3. *Soit $G = (X, E)$ un graphe non orienté.*
De toute chaîne, on peut extraire une chaîne élémentaire ayant les mêmes extrémités.

Démonstration. Soit $\mu = (x_1, \ldots, x_k)$ une chaîne. Si μ n'est pas élémentaire, il existe $i < j$ tel que $x_i = x_j$. Alors $\mu' = (x_1, \ldots, x_i, x_{j+1}, \ldots, x_k)$ est une sous-chaîne de μ, de mêmes extrémités. La longueur de μ' est strictement inférieure à la longueur de μ. On construit donc une suite de chaînes de mêmes extrémités, et de longueur strictement décroissante. Comme ces longueurs sont des entiers, cela ne peut pas se poursuivre indéfiniment. L'hypothèse de base devient donc fausse à partir d'un certain nombre d'itérations. La chaîne obtenue est alors élémentaire. ■

Remarque. Dans le cas des graphes orientés, on définit non pas des chaînes et des cycles mais des ***chemins*** et des ***circuits*** (sans répétition d'arcs). Les notions de chaîne et cycle existent et font abstraction des orientations des arcs.

Définitions (connexité). Deux sommets sont ***connectés*** s'il existe une chaîne ayant ces deux sommets pour extrémités. On construit ainsi une relation d'équivalence. Les classes d'équivalence de cette relation sont appelées ***composantes connexes***. Un graphe ne possédant qu'une seule composante connexe est dit ***connexe***.

Chapitre 2

Arbres

Définition (Arbre). Un graphe $G = (X, E)$ est un ***arbre*** s'il est connexe sans cycle. Un graphe sans cycle est une ***forêt*** (réunion d'arbres).

Lemme 2.1. *Soit $G = (X, E)$ un graphe non-orienté.*

1. *Si G est sans cycle et $|E| \geqslant 1$ alors G possède au moins 1 sommet de degré égal à 1.*
2. *Si G est connexe et $m = n - 1$ alors G possède au moins 1 sommet de degré égal à 1.*

Démonstration.

1. Soit $\mu = (x_1, \ldots, x_k)$ une chaîne élémentaire de longueur maximale (que l'on ne peut pas prolonger).

   $|E| \geqslant 1$ donc il existe au moins une arête donc $k \geqslant 2$. Si $d(x_1) \neq 1$, on trouve y voisin de x_1 et différent de x_2.

   Si $y \in \{x_3, \ldots, x_k\}$, on a un cycle, ce qui est impossible.

   Donc y est différent de tous les x_i donc la chaîne peut être prolongée, ce qui est impossible par hypothèse car la chaîne est supposée maximale.
2. G est supposé connexe donc le degré de chaque sommet est supérieur ou égal à 1. Supposons que ce degré soit toujours différent de 1. La somme des degrés des sommets est alors supérieure ou égale à $2n$. Or, cette somme est égale au double du nombre d'arêtes, qui par hypothèse vaut $n - 1$. ■

Théorème 2.2. *Les propriétés suivantes sont équivalentes :*

1. *$G = (X, E)$ est un arbre ;*
2. *G est sans cycle et $m = n - 1$;*
3. *G est connexe et $m = n - 1$;*
4. *G est sans cycle maximal ;*
5. *G est connexe minimal ;*
6. *Deux sommets quelconques sont reliés par une unique chaîne élémentaire.*

Démonstration.

1 ⇒ 2 On raisonne par récurrence sur n.

Si G est un arbre à un seul sommet ($n = 1$), le nombre d'arêtes m est forcément nul. G est donc bien sans cycle et tel que $m = n - 1$.

Supposons que tout arbre à $n-1$ ($n \geqslant 2$) sommets est sans cycle et vérifie $m = n - 1$. Soit G un arbre à n sommets. G est connexe avec au moins 2 sommets donc possède au moins une arête. D'après le lemme 1, il possède au moins un sommet x de degré 1. Considérons alors le graphe G' construit en retirant le sommet x de G. On a $n(G') = n(G) - 1$. Comme le degré de x est 1, supprimer x dans G revient à supprimer une seule arête. On a donc $m(G') = m(G) - 1$.

G étant sans cycle, le sous-graphe G' est également sans cycle. G' est forcément connexe car le sommet x est de degré 1 dans G. G' est donc un arbre à $n - 1$ sommets. Par hypothèse de récurrence, G' est sans cycle et $m(G') = n(G') - 1$. Donc $m(G) = n(G) - 1$. G étant un arbre, il est également sans cycle. L'hypothèse est donc vérifiée au rang n.

2 ⇒ 3 Soit G un graphe sans cycle vérifiant $m = n - 1$. Montrons que G est connexe. On raisonne également par récurrence. A l'ordre $n = 1$, le résultat est évident.

Soit $n \geqslant 2$. $m \geqslant 1$ donc, d'après le lemme 1, il existe un sommet x de degré 1. Considérons le graphe G' égal à G privé du sommet x. G' est également sans cycle. Par hypothèse $m(G) = n(G) - 1$ donc $m(G') = n(G') - 1$. L'hypothèse de récurrence impose donc la connexité de G'. G est alors connexe car chaque sommet de G' peut être relié au sommet où est relié x.

3 ⇒ 4 On raisonne à nouveau par récurrence, l'initialisation étant évidente.

Soit G un graphe connexe vérifiant $m = n - 1$. Montrons que G est sans cycle maximal.

D'après le lemme 1, il existe un sommet x de degré 1. Soit G' obtenu en retirant x de G. G' est connexe et vérifie $m = n - 1$. Par hypothèse, il est sans cycle maximal.

Si G avait un cycle, ce cycle passerait par x (car ce cycle n'est pas dans G'). Or c'est impossible car $d(x) = 1$.

Montrons maintenant qu'en rajoutant une arête, on crée un cycle et un seul. Soit une arête $xy \notin E$ (x quelconque). Il existe un chemin μ reliant x à y (car G est connexe) mais alors $\mu + xy$ est un cycle.

Supposons que $G+xy$ possède deux cycles distincts $\nu_1 = (x, \ldots, z, t_1, \ldots, y)$ et $\nu_2 = (x, \ldots, z, t_2, \ldots, y)$ avec $(x, \ldots, z)$ le plus long préfixe commun à ν_1 et ν_2. Notons $\nu'_1 = (z, t_1, \ldots, y)$ et $\nu'_2 = (z, t_2, \ldots, y)$. On définit z' le premier sommet de ν'_1 différent de z et apparaissant dans ν'_2. Les sous-chaînes allant de z à z' dans ν'_1 et ν'_2 sont différentes. On peut donc en déduire un cycle dans G, ce qui est absurde.

4 ⇒ 5 Montrons qu'un graphe sans cycle maximal est connexe minimal. Soit G sans cycle maximal. Soit xy une arête de G. Si $xy \in E$, x et y sont connectés. Sinon, $G + xy$ contient un cycle $\nu = (\underbrace{x, \ldots, y}_{\mu}, x)$. μ est une chaîne connectant x et y. Donc G est connexe.

Montrons sa minimalité. Soit $xy \in E$. Si $G \setminus \{xy\}$ est connexe, il existe une chaîne μ reliant x et y dans $G \setminus \{xy\}$. Dans ce cas, $\mu + xy$ est un cycle

de G, ce qui est absurde.

Par ailleurs, en enlevant une seule arête d'un graphe connexe, on ne peut obtenir qu'au plus 2 composantes connexes.

5 ⇒ 6 Soit G connexe minimal. Deux points x et y sont connectés par une chaîne élémentaire par connexité (par la propriété 3, on peut toujours se ramener à une chaîne élémentaire).

Supposons qu'il existe deux chaînes élémentaires ν_1 et ν_2 reliant x à y. Soit zz' la première arête de ν_1 qui n'est pas dans ν_2. Le graphe $G \setminus \{zz'\}$ reste connexe car z (respectivement z') est relié à x (resp. y) par ν_1 tandis que x et y sont reliés par ν_2. Ceci contredit l'hypothèse de minimalité.

6 ⇒ 1 Soit G un graphe tel que deux sommets $x \neq y$ sont toujours reliés par une unique chaîne élémentaire. G est connexe à l'évidence. Montrons qu'il est sans cycle. Si G admet un cycle $(x, \ldots, y, x)$, les chaînes $(x, \ldots, y)$ et (y, x) relient x à y de deux façons différentes. On peut extraire une chaîne élémentaire de $(x, \ldots, y)$ et cette chaîne sera différente de (x, y) car il n'y a pas répétition d'arête dans le cycle. On contredit donc l'hypothèse de départ. ■

Chapitre 3

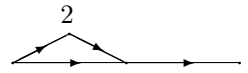
Parcours dans les Graphes

3.1 Représentation des graphes

On s'intéresse au cas des graphes orientés. On code la relation d'adjacence sous une des deux formes habituelles :

- ***La matrice d'adjacence :*** $M = (a_{ij})_{1 \leqslant i,j \leqslant n}$ avec $a_{ij} = 1$ s'il y a un arc allant de i à j, et 0 sinon.
- ***Le tableau des listes d'adjacences :*** A chaque sommet, on associe la liste de ses voisins.

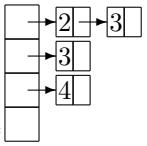
Exemple.



Matrice

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Tableau des listes



Place mémoire occupée La matrice occupe un espace mémoire de l'ordre de n^2 tandis que le tableau des listes n'occupe qu'un espace en $O(n + m)$.

Existence d'un arc Pour savoir si un arc ij existe dans le graphe, il faut $O(1)$ opérations dans la matrice d'adjacence tandis qu'il faudra parcourir toute la liste dans le tableau, c'est à dire $O(d^+(i))$ opérations.

Degré d'un sommet Dans la matrice, la recherche du degré d'un sommet nécessite $O(n)$ opérations alors que dans le tableau des listes, cela prend $O(d^+(i))$. Si on cherche les degrés de tous les sommets, on obtient $O(n^2)$ dans la matrice et $O(n + m)$ dans les listes.

Remarque. Pour les graphes non-orientés, on code symétriquement : l'arête ij sera codée par les arcs ij et ji.

3.2 Structure générale des parcours

À quoi peuvent servir les parcours ? À déterminer les plus courts chemins, des composantes connexes ou fortement connexes (connexes dans les deux «sens»,

sachant que nos graphes sont ici orientés), à savoir si un graphe est planaire (possibilité d'aplanir le graphe sans croisement d'arêtes), ...

3.2.1 Algorithme général

▷ initialisation
▷ tant que non-fini faire
  ▷ sélection d'un sommet
  ▷ traitement du sommet

3.2.2 Description

Un sommet a trois statuts possibles :
- non-atteint : l'algorithme ne le connaît pas encore.
- atteint : l'algorithme a rencontré le sommet mais ne l'a pas encore traité.
- examiné : l'algorithme a traité le sommet, il le connaît ainsi que tous ses voisins. On ne teste jamais si un sommet est dans cet état, il sert uniquement à une bonne compréhension de l'algorithme.

3.2.3 algorithme détaillé

```
à_traiter := ∅
foreach x ∈ X do
    état[x] := non_atteint
foreach x ∈ X do
    if état[x] = non_atteint then
        état[x] := atteint
        père[x] := ⊥ (*x est une racine*)
        à_traiter ∪= {x}
        while à_traiter ≠ ∅ do
            y := choix(à_traiter)
            à_traiter ⧵= {y}
            foreach z ∈ Γ⁺(y) do
                if état[z] = non_atteint then
                    père[z] := y
                    état[z] := atteint
                    à_traiter ∪= {z}
            état[y] := examiné
```

Algorithme 1: schéma général du parcours d'un graphe

Chaque sommet entre une fois et une seule dans à_traiter : quand il y entre, il change d'état, ce qui l'empêche d'y entrer une seconde fois.

La boucle **while** garantit que tout sommet sortira de l'algorithme en étant passé à l'état examiné.

Lorsqu'un sommet y est sorti de à_traiter, on examine son voisinage $\Gamma^+(y)$. Chaque arc sera donc vu une fois et une seule.

Dans père, on aura une forêt orientée. En effet, on part d'un sommet x pour réaliser un parcours, ce sommet x sera racine d'un des arbres de la forêt père, les autres sommets visités après s ne seront pas racine.

3.2.4 Complexité

On suppose qu'un ajout ou une suppression dans à_traiter coûte $O(1)$ (c'est vrai pour tous les algorithmes). On suppose que le choix dans à_traiter coûte $O(1)$, ce qui n'est pas toujours vrai (cela dépend des critères de choix et d'efficacité).

L'initialisation (la première boucle **foreach**) coûte $O(n)$.

Seule la dernière boucle **foreach** et son test sont effectués $d^+(y)$ fois pour un sommet y, soit au total $O(m)$. Les autres opérations sont effectuées une cois par sommet, avec donc une complexité totale $O(n)$.

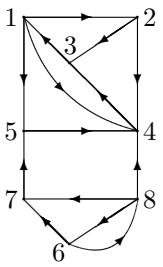
L'algorithme a donc un coût total $O(n+m)$. Attention, si l'on avait eu une fonction de choix en $O(n)$, comme on l'appelle n fois, on aurait un coût en $O(n^2)$.

3.3 Parcours en largeur

En fait, c'est une généralisation des parcours hiérarchiques des arbres (on regarde la racine puis les voisins, puis les voisins des voisins, ...). On va donc gérer la variable à_traiter comme une file FIFO (*First In First Out*).

Pour parcourir le graphe en largeur, on va ajouter deux règles : on fait défiler les sommets dans l'ordre des numéros ; on examine les successeurs d'un sommet dans le même ordre.

Exemple.



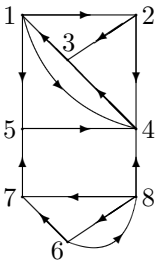
- On commence par 1 et ses voisins 2, 4 et 5.
- On regarde le plus petit voisin, c'est 2.
- On regarde les voisins de 2, ce sont 3 et 4. On a déjà vu 4 (il est déjà dans la pile). On ne tient donc pas compte de l'arc$(2,4)$.
- Le sommet suivant est 4. Son seul voisin est 3, que l'on a déjà vu.
- Le sommet suivant est 5. Son seul voisin est 4, que l'on a déjà empilé.
- Le sommet suivant est 3. Le seul voisin, qui est 1, a déjà été traité.
- La file est alors vide. On remonte donc à un sommet non-atteint, dans l'ordre des numéros.
- On traite donc le sommet 6. Ses voisins sont 7 et 8.
- On traite le sommet 7 dont le seul voisin est 5, qui a déjà été traité.
- On passe ensuite au sommet 8 dont les voisins 4, 6 et 7 ont tous déjà été traités.
- La file est alors vide et on a traité tous les sommets.

3.4 Parcours en profondeur

On part d'un sommet, on va voir un de ses voisins puis un voisin du voisin... jusqu'à être bloqué, alors on va en arrière. Le critère de choix sera ici que le dernier sommet atteint sera examiné. On utilisera une pile LIFO (*Last In First Out*).

La différence fondamentale entre les deux parcours est que le parcours en profondeur donne une structure au graphe, ce qui est utile pour étudier les composantes fortement connexes.

Exemple. On reprend l'exemple utilisé dans le cas du parcours en largeur.



- On commence par 1.
- On visite ensuite un voisin de 1, par exemple 2.
- On visite ensuite 3, qui est un voisin de 2.
- On est alors bloqué. On revient à 2 et on visite le seul voisin restant, 4.
- Il faut alors revenir à 1, et visiter le seul voisin restant, 5.
- On est alors bloqué. Il faut prendre un des sommets restant car il n'y a plus de voisin non-visité. On va donc à 6.
- On visite alors 7, qui est voisin de 6.
- 7 n'a pas de voisin non-visité. On retourne donc à 6 pour visiter le seul voisin restant, 8. Le parcours est alors terminé.

3.4.1 Description du parcours en profondeur

Le résultat du parcours sera deux étiquettes par sommet, OE est l'ordre d'empilement (par exemple 5 si 4 sommets on été empilés avant lui), OD l'ordre de dépilement.

On va utiliser une fonction récursive `parcours` qui fait l'essentiel du travail.

Fonction : `parcours` (x)
état$[x]$ = *atteint*
$OE[x] := i\,;\ i{+}{+}$
foreach $y \in \Gamma^+(x)$ **do**
  if état$[y]$ = *non_atteint* **then**
    père$[y] := x$
    `parcours`(y)
état$[x]$ = examiné
$OD[x] := j\,;\ j{+}{+}$

Dans l'exemple cité plus haut, on utilise en fait `parcours`(1) et `parcours`(6).

3.4.2 Classification des arcs

Une fois le parcours en profondeur terminé, on peut classer les arcs du graphe initial de la façon suivante : :

- Les ***arcs de la forêt*** (comme $(1,2)$ ou $(2,4)$). Si on les note (x,y), ils vérifient $OE[x] < OE[y]$ et $OD[x] > OD[y]$.

```
i := 1 ; j := 1
foreach x ∈ X do
  état[x] := non_atteint
foreach x ∈ X do
  if état[x] = non_atteint then
    père[x] := ⊥ (*x est une racine*)
    parcours(x)
```

Algorithme 2: parcours en profondeur

- Les ***arcs de transitivité*** au sens de la forêt (comme $(1, 4)$). Cela signifie pour (x, y) que l'on peut aller de x à y en n'utilisant que des arcs de la forêt. Ils vérifient également $OE[x] < OE[y]$ et $OD[x] > OD[y]$.
- Les ***arcs de retour*** (comme $(3, 1)$ et $(8, 6)$). Il y a également un chemin de x à y qui n'utilise que des arcs de la forêt. Ici, ils vérifient $OE[x] > OE[y]$ et $OD[x] < OD[y]$.
- Les ***arcs traversiers*** (comme $(4, 3)$ et $(7, 5)$). Ce sont les autres arcs. Ils vérifient $OE[x] > OE[y]$ et $OD[x] > OD[y]$.

3.5 Plus court chemin

Définition (Distance). Soit $G = (X, E)$ un graphe non-orienté. Soient x et y 2 sommets. La ***distance*** $\delta(x, y)$ entre les sommets x et y est la longueur d'une plus courte chaîne entre x et y si une telle chaîne existe. Dans le contraire, la distance sera par définition $+\infty$.

Définitions.

- Le ***diamètre*** d'un graphe est la maximum des distances entre deux sommets.
- L'***excentricité*** d'un sommet d'un graphe est le maximum de ses distances aux autres sommets.
- Le ***rayon*** d'un graphe est le minimum des excentricités des différents sommets.

Problème

Étant donné un sommet $s \in X$, on veut écrire un algorithme qui calcule $\delta(s, x)$ pour tout sommet x du graphe. On veut également calculer l'excentricité de s et le maximum des $\delta(s, x)$ (cela permet de repérer le sommet le plus loin de x).

En fait, on va faire un parcours en largeur à partir de s. On ajoute un compteur excentricité qui conserve la distance à partir de s.

```
foreach x ∈ X do d[x] := +∞
à_traiter := {s}
δ[s] := 0
excentricité := 0
while à_traiter ≠ ∅ do
    x := pop(à_traiter)
    foreach y ∈ Γ⁺(x) do
        if δ[y] = +∞ then
            à_traiter ∪= y
            δ[y] := δ[x] + 1
            excentricité := δ[y]
```

Algorithme 3: plus courts chemins dans un graphe non valué

Chapitre 4

Graphes orientés sans circuit

4.1 Reconnaissance d'un graphe orienté sans circuit

Lemme 4.1. *Si $G = (X, U)$ est un graphe orienté sans circuit, il existe $x \in X$ tel que $d^-(x) = 0$.*

Démonstration. Considérons un chemin maximal $\mu = (x_1, \ldots, x_k)$. Supposons $d^-(x_1) > 0$, appelons y un sommet de $\Gamma^-(x)$. Alors $(y, x_1, \ldots, x_k)$ est un chemin si y n'est pas dans μ, ce qui contredit la maximalité. Si y est dans μ, alors $(y, x_1, \ldots, x_i, y)$ est un circuit, ce qui est impossible. ■

Théorème 4.2. *Un graphe orienté $G = (X, U)$ est sans circuit si et seulement si $\exists x \in X$ tel que $d^-(x) = 0$ et $\forall x$ tel que $d^-(x) = 0$, $G \setminus \{x\}$ est sans circuit.*

Démonstration.

- $\Rightarrow$ Supposons G sans circuit. D'après le lemme précédent, $\exists x \in X$ tel que $d^-(x) = 0$ et tout sous-graphe d'un graphe sans circuit est sans circuit.
- $\Leftarrow$ Considérons $x \in X$ tel que $d^-(x) = 0$ (c'est possible par hypothèse). $G \setminus \{x\}$ est sans circuit par hypothèse. Donc, si G possède un circuit, il passe nécessairement par x. Donc $d^-(x) > 0$, ce qui est absurde. ■

Algorithme de reconnaissance

Algorithme : schéma de reconnaissance des graphes sans circuit
while $\exists x \in X \quad d^-(x) = 0$ **do** $G := G \setminus \{x\}$
if $G \neq \varnothing$ **then** G sans circuit **else** G a un circuit

On peut supposer que l'on représente le graphe sous la forme d'un tableau de listes. Il faut alors balayer tout le graphe pour trouver un x de degré entrant nul, puis recommencer... Chaque étape est de complexité $O(n+m)$, on voudrait que ce soit la complexité globale de l'algorithme.

Théorème 4.3. *Un graphe orienté $G = (X, U)$ est sans circuit si et seulement s'il existe une permutation $\sigma = (x_1, \ldots, x_n)$ des sommets de G tel que $d^-_{G_i}(x_i) = 0$, où $G_i = G[\{x_i, \ldots, x_n\}]$ (on supprime les $i-1$ premiers).*

Démonstration.

- $\Rightarrow$ Supposons G sans circuit. D'après le lemme, $\exists x \in X$ tel que $d^-(x) = 0$. D'après le théorème précédent, $G \setminus \{x\}$ est sans circuit. Par hypothèse de récurrence, il existe une permutation de $G \setminus \{x\}$, $\sigma = (y_1, \ldots, y_{n-1})$, satisfaisant aux conditions du théorème. La permutation $(x, y_1, \ldots, y_{n-1})$ convient.
- $\Leftarrow$ On suppose qu'il existe une permutation. Le graphe G_n à un seul sommet est sans circuit. Par le théorème précédent, si G_{i+1} est sans circuit, G_i est sans circuit. On en déduit que $G_1 = G$ est sans circuit. ■

Utilisons cette caractéristique pour obtenir un algorithme efficace.

Algorithme

```
foreach x ∈ X do d(x) := 0
foreach x ∈ X do foreach y ∈ Γ+(x) do d(y)++
à_traiter := ∅
nb_sommets := 0
foreach x ∈ X do
    if d−(x) = 0 then
        à_traiter ∪= {x}
        nb_sommets++
while à_traiter ≠ ∅ do
    x := premier(à_traiter)
    à_traiter \= {x}
    foreach y ∈ Γ+(x) do
        d−(y)--
        if d−(y) = 0 then
            à_traiter ∪= {y}
            nb_sommets++
if nb_sommets = n then G sans circuit else G a un circuit
```

Algorithme 4: reconnaissance des graphes sans circuits

La première étape de l'algorithme coûte $O(n)$, la seconde $O(n+m)$. Ensuite la boucle **foreach** utilise un temps constant pour chaque sommet et est donc globalement en $O(n)$. La boucle principale **while** coûte $O(n + m)$ tandis que l'étape finale est en temps constant.

La complexité totale de l'algorithme est donc $O(n + m)$. Cet algorithme effectue en fait 2 parcours du graphe, un pour les degrés internes, un pour les calculs.

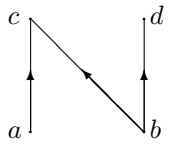
4.2 Tris topologiques

Définition (Tri topologique). Soit $G = (X, U)$ un graphe orienté. Un ***tri topologique*** est une permutation $\sigma = (x_1, \ldots, x_n)$ des sommets telle que $(x_i, x_j) \in U \Rightarrow i < j$.

Théorème 4.4. *G admet un tri topologique si et seulement si G est sans circuit.*

Démonstration. Ce théorème se déduit immédiatement du précédent. ■

Exemple. On veut déterminer tous les tris topologiques possibles commençant par a ou b. Pour a, on obtient $a\ b\ c\ d$ ou $a\ b\ d\ c$. Pour b, on obtient $b\ a\ c\ d$, $b\ a\ d\ c$ ou $b\ d\ a\ c$. Ce graphe possède 5 numérotations compatibles.



Dans un graphe, s'il y a un chemin, il y a un seul ordre compatible. S'il n'y aucun arc, il y a $n!$ ordres compatibles.

Comment engendrer tous les tris topologiques ? On note $\Pi(G)$ l'ensemble des tris topologiques de G et $S(G) = \{x \in X \mid d^-(x) = 0\}$ (sources de G).

$$\Pi(G) = \cup_{s \in S(G)} \{s.\sigma \mid \sigma \in \Pi(G \setminus \{s\})\}$$

C'est en fait la méthode que l'on a utilisé sur l'exemple. Cela donne une méthode récursive.

```
foreach x ∈ X do calculer d⁻(x)
S := ∅
foreach x ∈ X do
    if d⁻(x)=0 then
        S := {x}
        σ := ∅
        tri_topologique(G, S, σ)
```

Algorithme 5: calcul des tris topologiques

4.3 Décomposition en niveaux

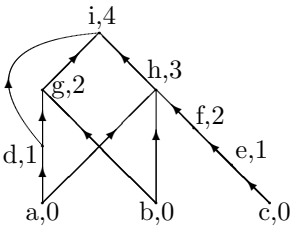
Définition (Hauteur). Soit $G = (X, U)$ un graphe orienté sans circuit. On définit une fonction ***hauteur*** $h : X \to \mathbb{N}$ par :

$$h : \quad x \longmapsto \begin{cases} 0 & \text{si} \quad d^-(x) = 0 \\ 1 + \max\limits_{\{y \mid (y,x) \in U\}} h(y) & \text{sinon} \end{cases}$$

```
Fonction : tri_topologique(G, S, σ)
if S=∅ then écrire(σ)
else
    foreach s ∈ S do
        S' := S ∖ {s}
        foreach y ∈ Γ+(s) do
            d−(y)--
            if d−(y)=0 then S' ∪= {y}
            tri_topologique(G, S', σ.s)
            foreach y ∈ Γ+(s) do d−(y)++
```

Exemple. On a représenté ici un graphe dont les sommets sont étiquetés par des lettres. Les chiffres représentent les hauteurs des sommets.

On veut calculer $nv[i]$, l'ensemble des sommets de hauteur $i-1$. Si on veut le faire de manière efficace ($O(n+m)$), au lieu de gérer par sommet, il faut gérer par niveau.




```
calculer les degrés internes
nb_sommets := 0
foreach x ∈ X tel que d−(x) = 0 do
    ajouter(x, nv[1])
    nb_sommets++
i := 1
while nv[i] ≠ ∅ do
    foreach x ∈ nv[i] do  foreach y ∈ Γ+(x) do
        d−(y)--
        if d−(y) = 0 then
            ajouter(y, nv[i + 1])
            nb_sommets++
    i++
if nb_sommets = n then G sans circuit else G a un circuit
```

Algorithme 6: décomposition en niveaux

La complexité de cet algorithme est en $O(n+m)$. En effet, on n'a fait que restructurer l'algorithme précédent en ajoutant la construction des niveaux, qui se fait en temps constant.

Proposition 4.5. *En notant* $h(G) = \max_{x \in X} h(x)$*, il existe un chemin de* G *de longueur* $h(G)$*.*

Démonstration. Il suffit de remarquer que tout $x \in nv[i]$ a un prédécesseur dans $nv[i-1]$. On part du niveau maximum et on descend pour construire le chemin. ■

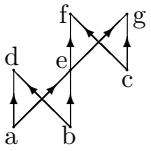
4.4 Ordres gradués

On considère un graphe $G = (X, U)$ orienté sans circuit et sans arcs de transitivité.

$(x, y) \in U$ est un ***arc de transitivité*** s'il existe un chemin $\mu = (x_1, \ldots, x_k)$ avec $x_1 = x$, $x_k = y$ et $\forall i, (x_i, x_{i+1}) \in U \setminus \{(x, y)\}$ (un chemin allant de x à y sans passer par l'arc (x, y)).

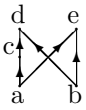
Définition (Graphe gradué). Un graphe G est ***gradué*** s'il existe une fonction $h : X \longmapsto \mathbb{Z}$ telle que $\forall (x, y) \in U, h(y) = h(x) + 1$.

Exemple.



Supposons qu'il existe une graduation h. Elle doit vérifier $h(g) = h(f) = h(e) + 1 = h(c) + 1$. Donc $h(c) = h(e)$. Elle doit aussi vérifier $h(d) = h(e) = h(a)+1 = h(b)+1$ donc $h(a) = h(b)$. On fixe donc $h(a) = h(b) = h(c) - 1$ et on obtient ainsi un arbre gradué (mais la hauteur n'est pas une graduation).

Exemple. Ici, la graduation doit vérifier $h(c) = h(a) + 1$, $h(d) = h(c) + 1 = h(b) + 1$ et $h(e) = h(a) + 1 = h(b) + 1$. On a donc d'une part $h(a) = h(b)$, et d'autre part $h(a) + 1 = h(b)$. C'est impossible. On ne peut graduer cet arbre.



Définition (Longueur compensée). On définit la ***longueur compensée*** d'une chaîne comme étant la différence des nombres d'arcs dans le bon et le mauvais sens :

Soit $\mu = (x_1, \ldots, x_k)$ une chaîne. $lc(\mu) = l - l'$ où $l = Card\{(x_i, x_{i+1}) \in U\}$ et $l' = Card\{(x_{i+1}, x_i) \in U\}$.

Proposition 4.6. *G est gradué si et seulement, $\forall x, y \in X$ les chaînes allant de x à y ont toutes la même longueur compensée.*

Démonstration.

- $\Rightarrow$ Supposons G gradué et appelons h la fonction de graduation. On raisonne par récurrence sur la longueur de la chaîne.
  Soit $\mu = (x_0, \ldots, x_p)$. Si $p = 0$, la chaîne ne contient qu'un sommet. La propriété est donc forcément vérifiée.
  Si $p \geqslant 1$, $lc(x_0, \ldots, x_p) = lc(x_0, \ldots, x_{p-1}) + lc(x_{p-1}, x_p) = h(x_{p-1}) - h(x_0) + lc(x_{p-1}, x_p)$ car l'hypothèse de récurrence s'applique à la sous-chaîne $(x_0, \ldots, x_{p-1})$.
  Si $(x_{p-1}, x_p) \in U$, on obtient $h(x_{p-1}) - h(x_0) + 1$ c'est à dire $h(x_p) - h(x_0)$.
  Sinon, $(x_p, x_{p-1}) \in U$. Alors on obtient $h(x_{p-1}) - h(x_0) + (0 - (-1))$, ce qui donne le même résultat.
- $\Leftarrow$ On peut supposer G connexe. Soit $x_0 \in X$. On pose $h(x_0) = 0$. Soit $x \in X$, $x \neq x_0$. il existe une chaîne μ de G reliant x_0 à x (G est supposé connexe). On pose $h(x) = lc(\mu)$. h est bien indépendant du choix de μ, par hypothèse.
  Montrons que h est bien une graduation : considérons un arc $(x, y) \in U$. Il existe une chaîne μ de G reliant x_0 à x. La chaîne $\mu' = \mu + (x, y)$ est une chaîne de G reliant x_0 à y. Et $lc(\mu') = lc(\mu) + 1$ soit $h(y) = h(x) + 1$. ■

Étude de la graduabilité dans le cas à une source unique

On cherche un algorithme permettant de déterminer en temps linéaire si un graphe à source unique (un seul sommet de degré entrant nul) est gradué. Les graphes seront par ailleurs supposés sans circuit et sans arc de transitivité.

On utilise le fait qu'un tel graphe à une seule source est gradué si et seulement si tous les chemins (orientés) reliant deux points x et y quelconques ont le même nombre d'arcs. Ce résultat se démontre facilement en adaptant la démonstration de la propriété ci-dessus.

Pour étudier la graduabilité du graphe, on le parcourra donc en largeur à partir de la source. S'il y a égalité de la hauteur à chaque niveau, on continue. Dans le cas contraire, on sort de l'algorithme.

Chapitre 5

Arbre couvrant de poids minimum

Définition (Poids). Étant donnés un graphe $G = (X, E)$ et une application p de E dans $\mathbb{R}$, on appelle ***poids*** d'un graphe partiel $G' = (X', E')$ de G la somme $p(E') = \sum_{e \in E'} p(e)$.

Définition (Arbre recouvrant). On appelle ***arbre recouvrant*** un graphe G, un graphe partiel qui est un arbre.

On va s'intéresser dans ce chapitre à la recherche d'arbres recouvrants minimisant un poids.

Théorème 5.1. *Un graphe $G = (X, E)$ admet un arbre recouvrant si et seulement si G est connexe.*

Démonstration.

- $\Rightarrow$ Si $T = (X, F)$ est un arbre couvrant G, la connexité de T entraîne celle de G car $F \subset E$.
- $\Leftarrow$ Réciproquement, si G est connexe, on considère $T = (X, F)$ un graphe partiel de G, sans cycle et ayant un nombre maximal d'arêtes. On va montrer que T est connexe.
  Supposons le contraire, il existe deux sommets x et y non reliés (situés dans deux composantes connexes disjointes). On note C_x la composante connexe contenant x. Dans G, il existe une chaîne $\mu = (x_1, \ldots, x_k)$ avec $x = x_1$ et $y = x_k$.
  Soit i le plus petit indice tel que x_{i+1} ne soit pas dans C_x. L'arête $(x_i x_{i+1})$ n'est dans aucune composante connexe et $T \cup \{x_i x_{i+1}\}$ est sans cycle (par construction). Cela constitue une contradiction. ■

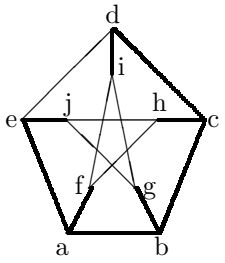
5.1 Caractérisation des arbres couvrants de poids minimal

Définition (Cycle et cocycle fondamentaux). Soient $G = (X, E)$ un graphe et $T = (X, F)$ un arbre recouvrant G.

Le ***cycle fondamental*** $\mu_T(e)$ associé à $e \in E \setminus F$ est l'unique cycle de $T \cup \{e\}$.

Le ***cocycle fondamental*** associé à $e \in F$ est $\omega_T(e) = \{xy \in E | x \in T_1 | y \in T_2\}$ où T_1 et T_2 sont les deux composantes connexes de $T \setminus \{e\}$.

Exemple.



Il s'agit de l'arbre de Petersen. On a représenté le graphe ainsi que l'arbre T le recouvrant en trait plus épais.

L'arête if n'est pas dans T et $\mu_T(if) = (i, f, a, b, c, d, i)$.

L'arête ab est dans T et $T \setminus \{ab\}$ possède deux composantes connexes : $T_1 = \{a, f, e, j\}$ et $T_2 = \{b, c, d, g, h, i\}$. Alors $\omega_T(ab) = \{ab, fh, fi, jh, jg, ed\}$.

Théorème 5.2. *Soient $G = (X, E)$ connexe, $p : E \to \mathbb{R}$ et $T = (X, F)$ un arbre couvrant G.*

T est de poids minimal si et seulement si :

$$\forall e \in F, \forall f \in \omega_T(e), p(e) \leqslant p(f)$$

Démonstration.

- $\Rightarrow$ Supposons qu'il existe $e \in F$ et $f \in \omega_T(e)$ avec $p(f) < p(e)$. On considère $T' = T \setminus \{e\} \cup \{f\}$. On a alors $p(T') < p(T)$.
  Or T' est un arbre car il a $n-1$ arêtes et il est connexe : $T \setminus \{e\}$ a 2 composantes connexes et comme $f \in \omega_T(e)$, f relie ces deux composantes.
- $\Leftarrow$ Supposons que $T = (X, F)$ vérifie les conditions du théorème et considérons $T^* = (X, F^*)$ un arbre recouvrant de poids minimum tel que $|F \cap F^*|$ soit maximal (c'est-à-dire, T^* possède un nombre maximal d'arêtes en commun avec T). Montrons que $T = T^*$.
  Supposons le contraire et considérons $e \in F \setminus F^*$. $T^* \cup \{e\}$ contient un cycle unique $\mu_{T^*}(e)$.
  $T \setminus \{e\}$ a deux composantes connexes T_1 et T_2. e est une arête entre $x \in T_1$ et $y \in T_2$. Donc il existe une arête $f \neq e$ du cycle $\mu_{T^*}(e)$ reliant $z \in T_1$ à $t \in T_2$. On a donc $f \in \omega_T(e)$ et, par hypothèse, $p(e) \leqslant p(f)$ (car e est l'arête de poids minimum sur le cocycle).
  On considère $T^{*'} = T^* \setminus \{f\} \cup \{e\}$. $T^{*'}$ est un arbre car il a $n-1$ arêtes et est sans cycle ($T^* \cup \{e\}$ contient un unique cycle et ce cycle passe par f). Comme T^* est de poids minimal et que $p\left(T^{*'}\right) \leqslant p\left(T^*\right)$, $T^{*'}$ est de poids minimal. Et comme $\left|F \cap F^{*'}\right| = |F \cap F^*| + 1$ (car on a échangé e et f alors que f n'est pas dans F^*), on a donc une contradiction sur la minimalité du poids de $T = T^*$. ■

Théorème 5.3. *Soient $G = (X, E)$ un graphe connexe, $p : E \to \mathbb{R}$ et $T = (X, F)$ un arbre couvrant G.*

T est de poids minimal si et seulement si :

$$\forall f \in E \setminus F, \forall e \in \mu_T(f), p(e) \leqslant p(f)$$

Démonstration.

⇒ Supposons qu'il existe $f \in E \setminus F$ et $e \in \mu_T(f)$ tel que $p(f) < p(e)$. On échange e et f. Soit $T' = T \setminus \{e\} \cup \{f\}$. T' est un arbre car possède $n-1$ arêtes et est sans cycle ($T \cup \{f\}$ contient un cycle unique et ce cycle passe par e). T' est alors de poids inférieur à T, ce qui constitue une contradiction sur la minimalité du poids de T.

⇐ Soit $T = (X, F)$ un arbre couvrant vérifiant les conditions du théorème. Montrons qu'il vérifie les conditions du théorème 5.2. Soit $e \in F$ et $f \in \omega_T(e)$ avec $f \neq e$. Par définition du cocycle f n'est pas dans F. $F \cup \{f\}$ contient un cycle et ce cycle passe par e. On a donc $p(e) \leqslant p(f)$. ■

5.2 Algorithme de Kruskal (1956)

```
trier les arêtes en ordre de poids croissant et les ranger dans une liste L.
nb_sommets := 0 ; T := ∅ ; poids := 0
while nb_sommets < n − 1 do
    e := première arête de L
    L \= {e}
    if T ∪ {e} ne contient pas de cycle then
        T ∪= {e}
        nb_sommets++
        poids += p(e)
```

Algorithme 7: algorithme de Kruskal

5.2.1 Correction

Théorème 5.4. *L'algorithme de Kruskal calcule un arbre couvrant de poids minimal.*

Démonstration. L'arbre construit vérifie les conditions du théorème 5.3.

Soit $f \in E \setminus F$. Lorsque l'algorithme rencontre l'arête, il connaît $T' \subset T$. Puisque f est injectée, c'est que $T' \cup \{f\}$ contient un cycle μ et $p(e) \leqslant p(f), \forall e \in \mu$.

En fin d'algorithme, on a T et $T \cup \{f\}$ contient un cycle unique, c'est forcément μ. ■

5.2.2 Complexité

Le tri des arêtes se fait en $O(m \log m) = O(m \log n)$. L'initialisation se fait en temps constant.

La boucle **while** est effectuée au pire m fois. Pour tester si $T \cup \{e\}$ contient un cycle (avec $e = xy$), on fait un parcours en largeur dans T à partir de x. Si on rencontre y, $T \cup \{e\}$ contient un cycle. Sinon, il n'en contient pas. Le coût du parcours est $O(n)$.

La complexité totale est donc $O(nm)$.

On va maintenant essayer d'améliorer notre algorithme en changeant la méthode de test de la présence de cycles.

Remarque. Il existe une structure de données plus efficace pour tester si l'on crée un cycle. En revanche sa complexité est difficile à démontrer. Il s'agit des *tas de Tarjan*, appelée aussi structure *union find*.

On obtient alors une complexité $O(m \log n)$, qui correspond au tri des arêtes.

5.3 Algorithme de Prim (1957)

schéma de l'algorithme de Prim

$T := \varnothing$
$A := x$
while $|T| < n - 1$ **do**
  $e := xy$ telle que $x \in A$, $y \notin A$, xy de poids minimal
  $T \cup= \{e\}$
  $A \cup= \{y\}$

5.3.1 Correction

Théorème 5.5. *L'algorithme de Prim calcule un arbre recouvrant de poids minimum.*

Démonstration. Soit $T = (X, F)$ un arbre couvrant de poids minimal. Soit A une partie connexe de T. Soit $e \in \omega(A)$ de poids minimal, où on a noté $\omega(A) = \{xy \in E \mid x \in A, y \notin A\}$.

Il existe un arbre recouvrant de poids minimal T' contenant $F|_A \cup \{e\}$.

Si $e \in T$, T couvrant. Si $e \notin T$, $T \cup \{e\}$ contient un cycle unique $\mu_T(e)$. Ce cycle passe par une arête $f \in \omega(A)$. On a donc $p(e) \leqslant p(f)$.

$T' = T \setminus \{f\} \cup \{e\}$ est un arbre car possède $n - 1$ arêtes et est sans cycle (car $T \cup \{e\}$ contient un unique cycle, et ce cycle passe par f).

On a donc $p(T') = p(T)$ et T' couvrant. ■

5.3.2 Complexité

Pour chaque sommet x de $X \setminus A$, il faut connaître l'arête f de poids minimal qui le relie aux autres sommets, et son poids $\mathsf{poids}_A(x) = \mathsf{poids}(f)$.

Chaque étape de sélection coûte $O(n)$. Le traitement coûte $O(d^+(\mathsf{à\_traiter}))$.

Au total on a donc $O(n^2)$ pour la sélection et $\sum_{x \in X} O(d^+(x))$ c'est-à dire $O(n + m)$.

La complexité de l'algorithme est donc $O(n^2)$.

```
foreach x ∈ X do
    poids_A(x) := +∞
    père(x) :=⊥
    état(x) := non_atteint
à_traiter := x ∈ X
état(à_traiter) := atteint
nb_sommets := 1
foreach x ∈ Γ+(à_traiter) do
    poids_A(x) := p(à_traiter, x)
    père(x) := à_traiter
while nb_sommets < n do
    (* Sélection de l'arête *)
    poidsmin:= +∞
    foreach x ∈ X do
        if état(x) = non_atteint et poids(x) <poidsmin then
            poidsmin:= poids_A(x)
            à_traiter := x
    (* Traitement de l'arête *)
    état(à_traiter) := atteint
    nb_sommets++
    foreach x ∈ Γ+(à_traiter) do
        if état(x) = non_atteint et poids(à_traiter, x) < poids_A(x) then
            poids_A(x) := poids(à_traiter, x)
            père(x) := à_traiter
```

Algorithme 8: algorithme de Prim

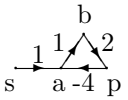
Chapitre 6

Chemin de coût minimum

Soient $G = (X, U)$ un graphe orienté et $c : U \to \mathbb{R}$. Si $\mu = (x_1, x_2, \ldots, x_n)$ est un chemin de G, son coût est $c(\mu) = \sum_{i=1}^{n} c(x_i x_{i+1})$.

Étant donné une source s et un puits p, on va chercher un chemin μ de s à p minimisant $c(\mu)$.

Remarque. Le graphe ci-contre pose problème car il y a un *cycle absorbant*, c'est-à-dire de coût négatif.



6.1 Algorithme de Dijkstra

On ne considère ici que des poids positifs $c : U \to \mathbb{R}^+$. On va calculer une arborescence de plus courts chemins de s à tous les sommets accessibles à partir de s.

6.1.1 Explication de la technique

A chaque étape, on sélectionne un sommet x et on fixe la valeur du plus court chemin de s à x.

On dispose de deux ensembles de sommets :

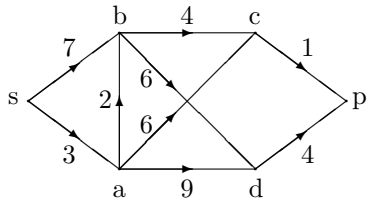
examinés C'est l'ensemble des sommets pour lesquels on connaît la valeur du plus court chemin issu de s.

à_traiter C'est l'ensemble des sommets qui ont au moins un prédécesseur dans examinés.

Pour les sommets x dans à_traiter, on connaîtra la valeur d'un plus court chemin $\mu = (x_1, \ldots, x_k)$ avec $x_1 = s$, $x_k = x$ et $\{x_1, \ldots, x_k\} \subseteq$ examinés.

Notre critère de sélection sera le suivant : on choisira dans à_traiter le sommet x minimisant le coût de $\mu(s, x)$.

Exemple.



On part de s. Le sommet le plus proche est a, son coût est définitivement 3. On mémorise l'arc sa.

Pour atteindre b, on peut y aller directement depuis s (coût total 7) ou en passant par a (coût total 3+2=5). On attribue 5 à b et on mémorise l'arc ab.

On mémorise ensuite l'arc bc et on attribue 5+4=9 à c. Atteindre c depuis a coûte 3+6=9 mais on privilégie les arcs les moins coûteux dans le choix du nouvel arc. Donc bc ici et non ac.

On mémorise bd et on attribue 5+6=11 à d. On mémorise cp et on attribue 9+1=10 à p.

6.1.2 Algorithme

foreach $x \in X$ **do** x$c(x) := +\infty$
à_traiter $:= \{s\}$
$c(s) := 0$
examinés $:= \varnothing$
while à_traiter $\neq \varnothing$ **do**
  $x :=$ élément de à_traiter minimisant $c(x)$
  examinés $\cup= \{x\}$
  à_traiter $\setminus= \{x\}$
  foreach $y \in \Gamma^+(x)$ **do**
    if not $y \in$ examinés **then** à_traiter $\cup= \{y\}$
    if $y \in$ à_traiter et $c(x)+$ coût$(x, y) < c(y)$ **then**
      $c(y) := c(x)+$coût(x, y)
      père$(y) := x$

Algorithme 9: algorithme de Dijkstra

6.1.3 Correction

Théorème 6.1. *L'algorithme de Dijkstra calcule un plus court chemin.*

Démonstration. On montre ce résultat par récurrence sur le nombre k de passages dans la boucle **while**.

On vérifie que :

1. $\forall x \in$ examinés$, c(x) = \mu(s, x)$ (valeur du plus court chemin de s à x).
2. $\forall y \in$ à_traiter$, c(y) = \overline{\mu}(s, y) = \min\{\mu(s, x)+$ coût$(x, y)\}$ avec $x \in$ examinés$et(x, y) \in U$.

Cas $k = 1$.

- examinés $= \{s\}$
- $c(s) = 0 = \mu(s, s)$
- à_traiter $= \Gamma^+(s)$
- $\forall x \in$ à_traiter, $c(x) = cout(s, x) = \overline{\mu}(s, x)$

Cas $k > 1$. Notons x le sommet sélectionné à la k^{e} étape. Supposons que $c(x) > \mu(s, x)$. Il existe un chemin $\nu = (x_1, \ldots, x_n)$ avec $x_1 = s$, $x_n = x$ et $cout(\nu) = \mu(s, x)$ (on a choisi ν, un plus court chemin entre s et x).

A l'étape $k - 1$, le sommet $x_{n-1} \notin$ examinés car sinon à l'étape $k - 1$, la valeur de $c(x)$ serait $\mu(s, x)$ (on a supposé $c(x) > \mu(s, x)$).

Appelons i le plus petit indice tel que $y_i \notin$ examinés à l'étape $k - 1$. Donc, le sommet $y_{i-1} \in$ examinés et donc $y_i \in$ à_traiter.

$c(y_i) = \min \{\mu(s, x) + cout(s, y_i) \mid z \in \text{examinés}, (s, y_i) \in U\} = \mu(s, y_i)$ et $\mu(s, y_i) \leqslant \mu(s, x) < c[x]$.

On obtient une contradiction car on aurait dû choisir y_i à la place de x. On a donc montré 1).

Prouvons maintenant 2) : Pour $y \in \Gamma^+(x)$, à l'étape $k - 1$, on connaît $c(y) = \min \{\mu(s, z) + \text{coût}(z, y) \mid z \in \text{examinés} \setminus \{x\}, (z, y) \in U\}$.

Si on modifie la valeur de $c(y)$ à l'étape k, c'est que $c(x)+\text{coût}(x, y) < c(y)$ et la nouvelle valeur est donc
$c(y) = \min \{\mu(s, z) + cout(z, y) \mid z \in \text{examinés}\}$. ■

6.1.4 Complexité

L'initialisation est en $O(n)$.

Pour trouver x de à_traiter tel que $c(x)$ minimum, on parcourt un tableau jusqu'à trouver la plus petite valeur. C'est en $O(n)$.

Le traitement d'un sommet x impose de regarder tous ses voisins, ce qui se fait en $O(d^+(x))$.

Le coût total est donc $\sum_{x \in X} O(n + d^+(x)) = O(n^2 + m) = O(n^2)$.

C'est la sélection d'un sommet dans à_traiter qui est coûteuse. Pour obtenir quelque chose de plus efficace, on pourrait utiliser un tas au lieu du tableau.

6.2 Algorithme de Bellman

L'algorithme de Dijkstra fonctionne sur les graphes avec circuit mais le coût doit être positif.

On suppose ici que $G = (X, U)$ est un graphe *sans circuit* mais c est à valeurs réelles, de signes quelconques.

Le plus court chemin de s à x passe forcément par un prédécesseur y de x.

$$\mu(s, x) = \min \left\{\mu(s, y) + \text{coût}(y, x) \;\middle|\; y \in \Gamma^-(x)\right\}$$

En calculant les $\mu(s, x)$ dans l'ordre d'un tri topologique, lorsqu'on veut évaluer $\mu(s, x)$, on connaît toutes les valeurs de $\mu(s, y)$ pour $y \in \Gamma^-(x)$. Il faut restreindre l'espace de recherche à $\uparrow s = \{x \in X \mid \exists \text{ un chemin de } s \text{ à } x\}$.

6.2.1 Algorithme

Il faut tout d'abord calculer $d^-_{\uparrow s}(x)$ pour tout $x \in \uparrow s$.

Ensuite, il faut inverser $\uparrow s$ (on va calculer le minimum sur les prédécesseurs).

Enfin, on exécute une boucle tant que tous les sommets n'ont pas été examinés. On sélectionne un sommet dont les voisins ont été examinés. On calcule le plus court chemin de s à x. On décrémente le degré interne du successeur de x. Cette boucle permet de suivre un tri topologique.

```
(* inversion du graphe et calcul des d⁻_{↑s}. *)
foreach x ∈ X do
    d⁻(x) := 0
    Γ⁻(x) := ∅
    état(x) := non_atteint
à_traiter := {s}
état(s) := atteint
while à_traiter ≠ ∅ do
    x := premier(à_traiter)
    à_traiter ⧵= {x}
    foreach y ∈ Γ⁺(x) do
        d⁻(y)++
        Γ⁻(y) ∪= {x}
        if état(y) = non_atteint then
            état(y) := atteint
            à_traiter ∪= {y}
(* calcul des plus courts chemins *)
à_traiter := ∅
c(s) := 0
foreach y ∈ Γ⁺(s) do
    d⁻(y)--
    if d⁻(y) = 0 then à_traiter := à_traiter ∪ {y}
while à_traiter ≠ ∅ do
    x := premier(à_traiter)
    à_traiter ⧵= {x}
    c(x) := ∞
    foreach y ∈ Γ⁻(x) do
        if c(y)+coût(y, x) < c(x) then
            c(x) := c(y)+coût(y, x)
            père(x) := y
    foreach y ∈ Γ⁺(x) do
        d⁻(y)--
        if d⁻(y) = 0 then à_traiter := à_traiter ∪ {y}
```

Algorithme 10: algorithme de Bellman

La validité de l'algorithme est évidente. La description vue juste avant le

justifie.

6.2.2 Complexité

On se déplace à partir de s. On balaie le graphe en ne passant qu'une fois et une seule par chaque arête. La première partie de l'algorithme est donc en $O(n+m)$.

Dans la boucle **while** de l'autre partie, on sélection un sommet et on l'initialise (temps constant). Puis pour chaque sommet, on examine tous ses prédécesseurs, et tous ses successeurs subissent des opérations en temps constant. On obtient donc également $O(n+m)$ pour cette partie.

Cet algorithme a une complexité de $O(n+m)$ et est donc plus efficace que l'algorithme de Dijkstra.

Chapitre 7

Facteurs d'un graphe

On s'intéresse ici à l'existence d'un ensemble d'arêtes deux à deux disjointes recouvrant tous les sommets d'un graphe.

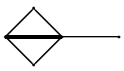
Définitions (Graphe factorisable). Un graphe $G = (X, E)$ non orienté est ***k*-factorisable** si on peut partitionner $E = \{E_1, \ldots, E_p\}$ de sorte que le graphe partiel $G_i = (X, E_i)$ soit ***k*-régulier**, c'est-à-dire $\forall x \in X, d_{G_i}(x) = k$ (les G_i sont les k-facteurs de G).

Définitions (Couplage). Un ***couplage*** de G est un ensemble d'arêtes $C \subset E$ tel que deux arêtes de C n'ont pas d'extrémités communes. Autrement dit, dans le graphe partiel $G' = (X, C)$, on a $d_{G'}(x) \leqslant 1, \forall x \in X$.

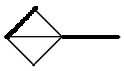
Les sommets x tel que $d_{G'}(x) = 1$ sont dits ***C*-saturés**. Les sommets x tel que $d_{G'}(x) = 0$ sont dits ***C*-insaturés**.

Un couplage C est ***maximal*** si $\forall e \in E \setminus C$, $C \cup \{e\}$ n'est pas un couplage. Un couplage C est ***maximum*** si $\forall C'$ couplage, $|C'| \leqslant |C|$ (maximal au sens du cardinal). Un couplage est ***parfait*** s'il sature tous les sommets : c'est un ***1*-facteur**.

Exemple.



couplage maximal



couplage maximum

Remarque. Si G admet un couplage parfait, G est d'ordre pair.

Définition (Chaîne améliorante). Soit $G = (X, E)$ non orienté, C un couplage. Une chaîne $\mu = (x_1, x_2, \ldots, x_{2k})$ est ***C*-améliorante** si :

- x_1 et x_{2k} sont C-insaturés ;
- $(x_{2i-1}, x_{2i}) \in E \setminus C$, $\forall i$;
- $(x_{2i}, x_{2i+1}) \in C$, $\forall i$;

Théorème 7.1 (Berge, 1957). *Soient $G = (X, E)$ un graphe et C un couplage.*

C est maximum si et seulement si il n'existe pas de chaîne C-améliorante.

Démonstration.

$\Rightarrow$ On utilise la contraposée.
Soit $\mu = (x_1, \ldots, x_{2k})$ une chaîne C-améliorante. On note $\oplus$ la différence symétrique.
Montrons que $C \oplus \mu$ est un couplage : $d_C(x_1) = d_C(x_{2k}) = 0$ car la chaîne est C-améliorante. $d_C(y) = 1$ pour tout $y \in \{x_2, \ldots, x_{2k-1}\}$. Donc $d_{C\oplus\mu}(y) = 1$ pour tout $y \in \mu$.
Pour $y \notin \{x_1, \ldots, x_{2k}\}$, $d_C(y) = d_{C\oplus\mu}(y)$.
Donc, en passant de C à $C \oplus \mu$, on a incrémenté le degré de 2 sommets. On en déduit que $C \oplus \mu$ est un couplage et comme il sature 2 sommets de plus que C, on a $|C \oplus \mu| = |C| + 1$.
C n'est donc pas maximum.

$\Leftarrow$ Supposons que C n'est pas maximum. On choisit un couplage C' maximum. On veut montrer qu'il existe une chaîne C-améliorante.
On pose $G' = (X, C \oplus C')$. Alors $\forall x \in G'$, $d(x) \leqslant 2$ (une arête dans C ou dans C').
G' est constitué de sommets isolés et de chaînes alternant des arêtes de C et de C'.
Puisque $|C| \leqslant |C'|$, l'une des chaînes a au moins une arête dans C' de plus que dans C. Cette chaîne est C-améliorante. ■

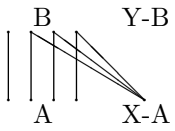
Définition. On appelle graphe ***biparti*** un graphe $G = (X, E)$ tel que X peut-être partitionné en deux ensembles X_1 et X_2 de façon à ce que toute arête e de E relie un sommet de X_1 à un sommet de X_2. On note alors $G = (X_1, X_2, E)$.

Définition (Chaîne C-alternée). Soit C un couplage. Une chaîne est ***C-alternée*** si ses arêtes sont alternativement dans C et dans $E \setminus C$.

Proposition 7.2. *Tout graphe biparti k-régulier admet un couplage parfait (i.e. un 1-facteur).*

Démonstration. Considérons un couplage maximum
$C = \{a_i b_i \mid i \in I, a_i \in X, b_i \in Y\}$ et supposons que C n'est pas parfait. On note $A = \{a_i \mid i \in I\}$ et $B = \{b_i \mid i \in I\}$. On a $|A| = |B|$.

Puisque C n'est pas parfait, $\exists x \in X$ C-insaturé. On a $\Gamma(x) \subseteq B$ sinon $\exists y \in Y \setminus B$ avec $xy \in E$ et $C' = C \cup \{xy\}$ est un couplage tel que $|C'| = |C| + 1$. On obtient une contradiction.



Soit un sommet x. On définit $A' = \{a_j \mid j \in J\}$ l'ensemble des points reliés à un sommet x par un chaîne C-alternée, et B' l'ensemble des b_j correspondants.

On a $\Gamma(x) \subseteq B'$ car xb_ja_j est une chaîne C-alternée.

Remarque. Toute chaîne C-alternée issue de x est de la forme $xb_{i_1}a_{i_1}\ldots b_{i_k}$.

Pour tout $i \in I$ et $j \in J$, $a_jb_i \in E \Rightarrow i \in J$. En effet, si $j \in J$, il existe une chaîne C-alternée $\mu = (x, \ldots, a_j)$. D'autre part, $a_jb_i \in E$ et $b_ia_i \in C \subseteq E$ d'où $\mu' = (x, \ldots, a_j, b_i, a_i)$ est une chaîne C-alternée, ce qui entraîne $a_i \in A$ ou encore $i \in J$.

J'affirme maintenant que $\exists a \in A' \quad \Gamma(a) \cap (Y \setminus B) \neq \varnothing$.

Si ce n'est pas le cas, pour tout $a \in A'$, $\Gamma(a) \subseteq B$ mais d'après le lemme précédent, on a $\Gamma(a) \subseteq B'$. Comptons les arêtes incidentes à B'. On prend $p = |B'| = |A'|$. Puisque G est k-régulier, il y en a kp. Les arêtes provenant de x et de A' sont toutes incidentes à B', il y a au moins $k(p+1)$ arêtes incidentes à B, une contradiction.

On en déduit qu'il existe $a \in A'$ avec $ay \in E$ et $y \in Y \setminus B$. Mais alors $\mu = (x, \ldots, a, y)$ est C-améliorante. D'après le théorème de Berge, il s'en suit que C n'est pas un couplage de cardinal maximum, ce qui constitue une contradiction. ■

Théorème 7.3. *Tout graphe biparti k-régulier est 1-factorisable.*

Démonstration. Nous avons montré que tout graphe biparti k-régulier admet un 1-facteur. On fait alors une récurrence facile sur k. ■

Définition (Graphe connexe eulérien). Un graphe connexe est ***eulérien*** s'il existe un cycle passant une fois et une seule par chaque arête.

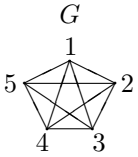
Théorème 7.4 (Euler). *Un graphe connexe est eulérien si et seulement si le degré de chacun de ses sommets est pair.*

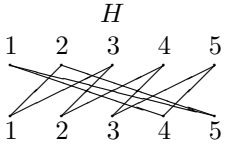
Théorème 7.5 (Petersen, 1891). *Un graphe est 2-factorisable si et seulement si il est $2k$-régulier.*

Démonstration.

- $\Rightarrow$ Si G est 2-factorisable, il existe q 2-facteurs qui partitionnent l'ensemble des arêtes. Chaque sommet dans chaque 2-facteur est de degré 2. Il s'en suit que chaque sommet est de degré $2q$.
- $\Leftarrow$ Réciproquement, supposons G $2k$-régulier et montrons qu'il possède un 2-facteur. On peut supposer G connexe (sinon, on travaille de manière indépendante sur chaque composante connexe) et puisque tous les degrés sont pairs, le graphe G est eulérien. On note $V = (x_{i_1}, \ldots, x_{i_{kn}}, x_{i_1})$ un cycle eulérien. Quand k décrit $[\![1, kn]\!]$, i_k décrit exactement k fois $[\![1, n]\!]$.
  On définit le graphe biparti $H = (A, B, F)$ par $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$ et $F = \{a_p b_q \mid p = i_r, q = i_s, s = r+1\}$.
  Ainsi, un sommet a_i est relié à k sommets b_j. Ces k sommets sont tous différents car chaque arête considérée ici correspond à une arête et une seule dans le graphe original.
  Le graphe H est biparti k-régulier. D'après le théorème précédent, il admet un couplage parfait C. Le graphe $G' = (X, E')$ défini par $E' = \{x_{i_r} x_{i_s} \mid i_r = p, i_s = q, a_p b_q \in C\}$ est un 2-facteur du graphe G.
  Puisque G admet un 2-facteur, par récurrence triviale, G est 2-factorisable. ■

Exemple.





V

$(1, 2, 3, 4, 5, 1, 3, 5, 2, 4, 1)$

Dans le cycle eulérien, le sommet 1 est suivi par le sommet 2 puis, la seconde fois par le sommet 3. On relie donc a_1 à b_2 et b_3. De même, on relie a_3 à b_4 et b_5.

Notation. On note $i(G)$ le nombre de composantes connexes d'ordre impair d'un graphe G.

Théorème 7.6 (Tutte, 1947). *Un graphe $G = (X, E)$ admet un couplage parfait (1-facteur) si et seulement si $\forall S \subseteq X i(G \setminus S) \leqslant |S|$.*

Démonstration.

$\Rightarrow$ Supposons que G admet un 1-facteur et considérons $S \subseteq X$.
On s'intéresse désormais au graphe $G \setminus S$. Ce couplage parfait peut saturer en interne toutes les composantes connexes d'ordre pair.
Pour les composantes d'ordre impair, le couplage ne peut pas saturer en interne. Il reste donc un sommet insaturé, qui est forcément relié à un sommet de S. Pour chaque composante connexe d'ordre impaire, il existe une arête allant vers un sommet de S.
Donc $i(G \setminus S) \leqslant |S|$.

$\Leftarrow$ Soit $G = (X, E)$ un graphe vérifiant $\forall S \subseteq X$, $i(G \setminus X) \leqslant |S|$.
En prenant $S = \varnothing$, on déduit $|X|$ pair. Si on ajoute une arête e au graphe G, il vérifie toujours l'hypothèse.
Supposons que $G = (X, E)$ vérifie cette hypothèse et n'admet pas de couplage parfait. Parmi les graphes vérifiant cette propriété, on en choisit un maximisant le nombre d'arêtes, c'est-à-dire $\forall e \notin E$, $G + \{e\}$ admet un couplage parfait.
On pose $Y = \{x \in X \mid d(x) = n - 1\}$ (sommets dominants). On va montrer que les composantes connexes de $G \setminus Y$ sont des graphes complets.
Supposons qu'il existe une composante connexe qui ne soit pas une clique. Il existe alors deux sommets à une distance 2, c'est-à-dire $\exists a, b, c \in X \setminus Y$ tel que $ab, ac \in E$ et $bc \notin E$.
Comme $a \notin Y$, il existe $d \in X \setminus Y$ tel que $ad \notin E$. On considère les deux graphes $G_1 = G + \{bc\}$ et $G_2 = G + \{ad\}$. Ils admettent tous deux un couplage parfait (par maximalité de G) F_1 et F_2. On a $bc \in F_1$ et $ad \in F_2$.
Le graphe partiel $H = (X, F_1 \oplus F_2)$ vérifie $\forall x \in X$, $d_H(x) = 0$ ou 2. Soit e_1 l'arête de F_1 incidente à x, e_2 celle de F_2. Si $e_1 = e_2$, $d_H(x) = 0$. Sinon, $d_H(x) = 2$.
H est donc constitué de sommets isolés et de cycles (de longueur paire car alternant les arêtes de F_1 et de F_2). Remarquons que ad et bc appartiennent à $F_1 \oplus F_2$.

- Si ad et bc sont sur des cycles différents : On construit un couplage parfait de G de la manière suivante : sur le cycle de bc, on prend des arêtes de F_2. Sur celui de ad, on prend celui de F_1. Partout ailleurs, on prend F_1. On obtient donc une contradiction.
- Si ad et bc sont sur le même cycle : Ce cycle possède deux cordes ac et ab, qui le découpent en deux cycles μ_1 et μ_2, le premier contenant l'arête ad. On construit alors le couplage parfait de G comme suit : Sur le premier cycle, on prend les arêtes de F_1. Sur le second cycle, on prend les arêtes de F_2. On ajoute ac ou ab en choisissant celle qui se trouve sur un cycle de longueur paire. Ailleurs, on prend les arêtes de F_1. On obtient donc une contradiction.

Les composantes connexes de $G \setminus Y$ sont donc des cliques.
On construit alors un couplage parfait de G comme suit : On choisit un couplage parfait dans chaque composante connexe d'ordre pair. On fait de même dans chaque composante d'ordre impair en laissant un sommet insaturé. ■

Chapitre 8

Couplage maximum dans les bipartis

8.1 Couplage dans les graphes quelconques

Théorème 8.1. *Soient un graphe $G = (X, E)$, C_1 et C_2 deux couplages de cardinaux respectifs r et s avec $r < s$. Alors $C_1 \oplus C_2$, donc G, contiennent au moins $s - r$ chaînes C_1-améliorantes.*

Démonstration. On considère $G' = (X, C_1 \oplus C_2)$. Les composantes connexes de G' sont des sommets isolés, des cycles de longueur paire alternant des arêtes de C_1 et C_2, ou enfin des chaînes alternant des arêtes de C_1 et C_2.

A toute chaîne μ_i de G', on associe $\delta(\mu_i) = |\mu_i \cap C_2| - |\mu_i \cap C_1|$. $\delta(\mu_i) \in \{-1, 0, 1\}$.

$\sum_i \delta(\mu_i) = s - r$ donc il y a au moins $s - r$ chaînes avec $\delta(\mu_i) = 1$ donc C_1-améliorantes. ■

Corollaire. *Soit C un couplage de G avec $|C| = r$. On note $\nu(G) = s$ le cardinal d'un couplage maximal.*

Si $r < s$, il existe une chaîne C-améliorante de longueur inférieure ou égale à $2\left\lfloor \frac{r}{s-r} \right\rfloor + 1$.

Démonstration. Il y a au moins $s - r$ chaînes C-améliorantes, ces chaînes se partagent les r arêtes de C. La plus courte en a au plus $\left\lfloor \frac{r}{s-r} \right\rfloor$ (c'est la moyenne). La longueur totale est donc au plus $2\left\lfloor \frac{r}{s-r} \right\rfloor + 1$. ■

Théorème 8.2. *Soient C un couplage de G, μ une plus courte chaîne C-améliorante et μ' une chaîne $C \oplus \mu$ améliorante.*

Alors $|\mu'| \geqslant |\mu| + 2|\mu \cap \mu'|$ ($\cap$ au sens des arêtes).

Démonstration. Soit $C' = C \oplus \mu \oplus \mu'$. C'est un couplage et $|C'| = |C| + 2$. $C \oplus C'$ contient au moins 2 chaînes C-améliorantes μ_1 et μ_2 (d'après le théorème précédent).

Or $$C \oplus C' = C \oplus C \oplus \mu \oplus \mu' = \varnothing \oplus \mu \oplus \mu' = \mu \oplus \mu'.$$

Donc $$|\mu \oplus \mu'| = |C \oplus C'| \geqslant |\mu_1| + |\mu_2| \geqslant 2|\mu|$$

$$|\mu \oplus \mu'| = |\mu| + |\mu'| - 2|\mu \cap \mu'|$$

D'où $$|\mu'| \geqslant |\mu| + 2|\mu \cap \mu'|$$ ■

Ceci suggère l'algorithme suivant :

On calcule une suite de couplages C_0, C_1, ... C_s telle que $C_0 = \varnothing$, $C_{i+1} = C_i \oplus \mu_i$ où μ_i est une plus courte chaîne C_i-améliorante. Alors $s = \nu(G)$ et $|C_i| = i$.

Corollaire 1. $|\mu_{i+1}| \geqslant |\mu_i|$

Corollaire 2. *$\forall i < j$, si $|\mu_i| = |\mu_j|$ alors μ_i et μ_j n'ont pas de sommet en commun.*

Démonstration. Supposons que $\exists i < j$ tels que $|\mu_i| = |\mu_j|$ et μ_i et μ_j ont un sommet en commun. On choisit k, l tels que $i \leqslant k < l \leqslant j$, μ_k et μ_l ne sont pas disjointes et $\forall m, k < m < l$, μ_m disjointe de μ_k et μ_l.

Par définition, μ_l est une chaîne $C_k \oplus \mu_k \oplus \cdots \oplus \mu_{l-1}$ améliorante. Par choix de k et l, μ_l est aussi $C_k \oplus \mu_k$-améliorante.

D'après le théorème précédent, $|\mu_l| \geqslant |\mu_k| + 2|\mu_k \cap \mu_l| \Rightarrow |\mu_k \cap \mu_l| = 0$ (elles n'ont pas d'arêtes en commun).

Soit x un sommet commun à μ_k et μ_l. Comme $x \in \mu_k$, il est saturé par une arête e du couplage $C_k \oplus \mu_k$. L'arête e se trouve aussi sur μ_l parce que μ_l est $C_k \oplus \mu_k$-améliorante et que μ_l passe par x.

On obtient donc une contradiction. ■

Théorème 8.3. *Le nombre d'entiers différents dans la suite $|\mu_1|$, $|\mu_2|$, ..., $|\mu_s|$ est au plus $2\lfloor\sqrt{s}\rfloor + 2$.*

Démonstration. On prend $r = \lfloor s - \sqrt{s}\rfloor$. $|C_r| = r$ donc (par un corollaire précédent) :

$$\begin{aligned} |\mu_r| &\leqslant 2\left\lfloor \frac{r}{s-r} \right\rfloor + 1 = 2\left\lfloor \frac{\lfloor s - \sqrt{s}\rfloor}{\lceil \sqrt{s} \rceil} \right\rfloor + 1 \\ &\leqslant 2\frac{s-\sqrt{s}}{\sqrt{s}} + 1 \;=\; 2\left(\sqrt{s} - 1\right) + 1 \;=\; 2\sqrt{s} - 1 \\ |\mu_r| &\leqslant 2\lfloor\sqrt{s}\rfloor + 1 \end{aligned}$$

Les longueurs des chaînes C-améliorantes entre μ_1 et μ_r sont des entiers impairs compris entre 1 et $2\lfloor\sqrt{s}\rfloor + 1$. Il y en a au plus $\lfloor\sqrt{s}\rfloor + 1$.

Les chaînes μ_{r+1}, ..., μ_s contribuent pour au plus $s - r = \lceil\sqrt{s}\rceil$ valeurs. Donc en tout, au plus $2(\lfloor\sqrt{s}\rfloor + 1)$. ■

On utilise l'algorithme suivant :

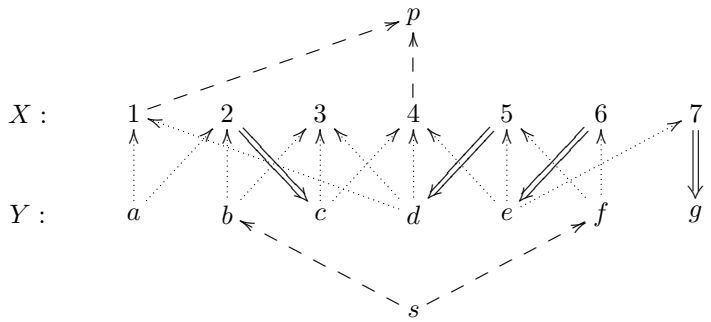
8.2 Couplage des bipartis

8.2.1 Graphe des plus courtes chaînes C-améliorantes

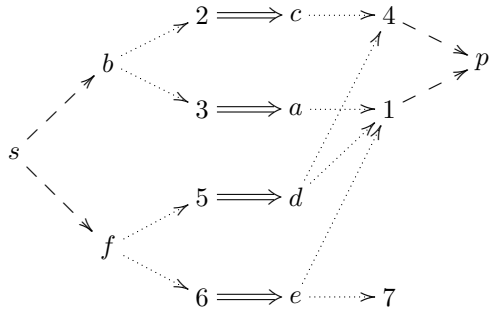
On va construire un graphe orienté $\vec{G}$ dans lequel les chemins maximaux seront en bijection avec les plus courtes chaînes C-améliorantes de G. On dispose de $G = (X, Y, E)$ et d'un couplage C.

$C := \varnothing$
while $\exists$ une chaîne C-améliorante **do**
  trouver $\{\mu_1, \ldots, \mu_k\}$ un ensemble maximal de plus courtes chaînes C-améliorantes 2 à 2 disjointes
  $C := C \oplus \mu_1 \oplus \cdots \oplus \mu_k$

On construit $\vec{G}$. Pour cela, on oriente les arêtes de C de X vers Y, celles de $E \setminus C$ de Y vers X. Puis on ajoute deux sommets s et p, et les arcs $(s, y), \forall y \in$ Insaturé(Y) et $(x, p), \forall x \in$ Insaturé(X).



Puis on fait un parcours en largeur de s vers p (on ne met pas les arcs qui «reculent»).



Dès qu'on rencontre p, on arrête tout. Les chaînes de s à p sont les plus courtes chaînes C-améliorantes (en enlevant s et p).

8.2.2 Description plus formelle

$L_0 = \text{Insaturé}(X)$ $\qquad \vec{E}_i = \{(z,t) \in E \mid z \in L_i, t \notin L_0 \cup \cdots \cup L_{i-1}\}$

$L_{i+1} = \left\{ t \in X \cup Y \;\middle|\; (z,t) \in \vec{E}_i \right\}$ $\qquad i^* = \min\{i \mid L_i \cap \text{Insaturé}(Y) \neq \varnothing\}$

$\vec{G} = (\vec{X}, \vec{E})$ avec

$\vec{X} = \{s, p\} \cup L_0 \cup \cdots \cup L_{i^*-1} \cup (L_{i^*} \cap \text{Insaturé})$

$\vec{E} = \{(s,x) \mid x \in \text{Insaturé}(X)\} \cup \vec{E}_0 \cup \vec{E}_{i^*-1} \cup \{(y,p) \mid y \in L_{i^*} \cap \text{Insaturé}(Y)\}$

Complexité : $O(n+m)$.

Complexité : une étape de la boucle coûte $O(n+m)$. Le total est donc $O(\sqrt{n}(n+m))$ ou $O(n^{\frac{5}{2}})$ (car $m = O(n^2)$).

Algorithme : construction de $\vec{G}$

améliorante:= $faux$
foreach $x \in X \cup Y$ **do** utilisé(x) := $faux$
foreach $y \in$ Insaturé(Y) **do**
  $\Gamma_{\vec{G}}(y) := \{p\}$
  $\Gamma_{\vec{G}}(s) := \varnothing$
  $nvsommets := \varnothing$
foreach $x \in$ Insaturé(X) **do**
  $\Gamma_{\vec{G}}(s) := \Gamma_{\vec{G}}(s) \cup \{x\}$
  $nvsommets := nvsommets \cup \{x\}$
  utilisé(x) := $vrai$
$couplage := faux$
repeat
  $nvsomm := \varnothing$
  if couplage
  then
    foreach $x \in nvsommets$ **do**
      $\Gamma_{\vec{G}}(x) := \{couple[x]\}$
      $nvsomm \cup= \{couple[x]\}$
  else
    foreach $x \in nvsommets$ **do**
      $\Gamma_{\vec{G}}(x) := \varnothing$
      foreach $y \in \Gamma_{\vec{G}}(x)$ **do**
        if non(utilisé(y)) et $y \neq couple[x]$ **then**
          $\Gamma_{\vec{G}}(x) := \Gamma_{\vec{G}}(x) \cup \{y\}$
          $nvsomm \cup= \{y\}$
        if $y \in$ Insaturé(Y) **then** améliorante:= $vrai$
  $couplage := non(couplage)$
  foreach $x \in nvsomm$ **do**
    utilisé(x) := $vrai$
  $nvsommets := nvsomm$
until ($nvsommets = \varnothing$) ou améliorante

```
foreach x ∈ X ∪ Y do couple[x] := 0
Insaturé(X) := X
Insaturé(Y) := Y
repeat
    Construction_G⃗
    if améliorante then
        pile := ∅
        foreach x ∈ X ∪ Y do interdit(x) := faux
        empiler(s)
        while pile ≠ ∅ do
            while Γ_G⃗(haut_pile) ≠ ∅ do
                x := pop(Γ_G⃗(haut_pile))
                if non(interdit(x)) then
                    if x ≠ p then
                        interdit(x) := vrai
                        empiler(x)
                    else
                        (*échange sur le couplage*)
                        Insaturé(Y) := Insaturé(Y) ∖ {haut_pile}
                        while haut_pile ≠ s do
                            y := haut_pile
                            dépiler
                            x := haut_pile
                            dépiler
                            couple[x] := y
                            couple[y] := x
                        Insaturé(X) := Insaturé(X) ∖ {x}
            dépiler
until non améliorante
```

Algorithme 11: couplage maximum dans les bipartis

Chapitre 9

Connexité

La connexité a d'importantes applications en informatique, notamment dans l'étude de la fiabilité des réseaux.

Définition (AB-chaîne, AB-séparateur). Soit $G = (X, E)$ un graphe. On considère A et B, deux sous ensembles de X. Une ***AB-chaîne*** est une chaîne $(a, \dots, b)$ avec $a \in A$ et $b \in B$.

Un ***AB-séparateur*** est un ensemble de sommets $S \subseteq X$ tel que $G \setminus S$ ne contient plus de AB-chaînes.

Exemple. A, B et X sont des AB-séparateurs.

Remarque. Comme $\forall x \in A \cap B$, x est une AB-chaîne, tout AB-séparateur contient $A \cap B$.

Notation. On note $p(A, B, G)$ le nombre maximal de AB-chaînes 2 à 2 disjointes (sans sommets communs). On note $q(A, B, G)$ le cardinal minimal d'un AB-séparateur.

Théorème 9.1 (Menger, 1927). *Dans un graphe $G = (X, E)$, on a $p(A, B, G) = q(A, B, G)$.*

Démonstration. Cette preuve est due à Nash-Williams et Tutte en 1977.

$p \leqslant q$ On considère un ensemble de cardinal maximal de AB-chaînes 2 à 2 disjointes. Il y en a p. Un AB-séparateur S doit contenir au moins un sommet de chacune des chaînes.

Donc $p \leqslant |S| = q$.

$p \geqslant q$ Supposons qu'il existe un graphe $G = (X, E)$ et A et B deux sous-ensembles de X tels que $p < q$. On choisit un tel graphe avec $|E|$ minimum. $E \neq \varnothing$ sinon $p = q = |A \cap B|$.

Soit $e = xy \in E$. On considère les deux graphes $G \setminus \{e\}$ et G/e On définit le graphe G/e en supprimant l'arête $e = xy$ puis en fusionnant les sommets x et y. On obtient un sommet z dont le voisinage est l'union des voisinages de x et y, privée bien-sûr de x et y.

Les deux nouveaux graphes ont une arête de moins que G, et par minimalité de $|E|$, ne vérifient pas $p < q$.

Donc, $p(A, B, G \setminus \{e\}) = q(A, B, G \setminus \{e\}) \leqslant p(A, B, G) < q(A, B, G)$ et $p(A, B, G/e) = q(A, B, G/e) \leqslant p(A, B, G) < q(A, B, G)$.

Attention : A et B ne sont pas les mêmes dans G et dans G/e.

Notons $q = q(A, B, G)$. Soit I un AB-séparateur de cardinal minimal de $G \setminus \{e\}$. $|I| = q(A, B, G \setminus \{e\}) < q$. Soit J' défini de même pour G/e. $|J'| < q$.

On a $z \in J'$. Supposons le contraire et considérons $\mu = (a, \ldots, b)$ une AB-chaîne. La trace μ/e de μ dans G/e est une AB-chaîne, elle coupe donc J' en un sommet t, et $t \in X$ ($t \neq z$ par hypothèse). On en déduit que J' est un AB-séparateur de G car μ coupe J' en t. Mais $|J'| < q$. Contradiction.

On pose $J = J' \setminus \{z\} \cup \{x, y\}$. J est un AB-séparateur de G. En effet, soit $\mu = (a, \ldots, b)$ une AB-chaîne. μ/e coupe J' en un sommet t. Si $t \neq z$, μ coupe J en t. Si $t = z$, μ coupe J en x ou y. J est un AB-séparateur implique $|J| \geqslant q$, $|J| = |J'| + 1$ et $|J'| < q$, ce qui implique $|J| = q$.

On va construire deux AB-séparateurs de G, H_A et H_B vérifiant $H_A \cap H_B \subseteq I \cap J$ et $H_A \cup H_B \subseteq I \cup J$, ce qui mènera à la contradiction :

$$2q \leqslant |H_A| + |H_B| = |H_A \cup H_B| + |H_A \cap H_B| \leqslant |I \cup J| + |I \cap J| = |I| + |J| < 2q$$

On pose $H_A = \{t \in I \cup J | \exists$ une At-chaîne qui n'intersecte pas $I \cup J \setminus \{t\}\}$ et $H_B = \{t \in I \cup J | \exists$ une Bt-chaîne qui n'intersecte pas $I \cup J \setminus \{t\}\}$

Alors $H_A \cup H_B \subseteq I \cup J$.

Montrons que H_A est un AB-séparateur. J étant un AB-séparateur, $I \cup J$ est aussi un AB-séparateur. Si $\mu = (a, \ldots, b)$ est une AB-chaîne, elle coupe $I \cup J$. Soit t l'élément de μ le plus à gauche appartenant à $I \cup J$. $t \in H_A$.

De même, H_B est un AB-séparateur.

Montrons que $H_A \cap H_B \subseteq I \cap J$. Si leur intersection est vide, c'est évident. Sinon, soit u un élément de l'intersection. Il existe une Au-chaîne $\mu = (a, \ldots, u)$ et une Bu-chaîne $\mu' = (b, \ldots, u)$ qui ne coupent pas $I \cup J \setminus \{u\}$.

Soit λ une chaîne élémentaire extraite de $\mu\overline{\mu'}$ ($\overline{\mu}$ est le miroir de μ). Par construction, λ est une AB-chaîne, elle coupe donc J, ce ne peut-être qu'en u (donc $v = u$) d'où $u \in J$.

La chaîne λ ne contient pas l'arête e sinon elle intersecterait J en x, y et u, soit au moins deux sommets. Puisque λ est une AB-chaîne de $G \setminus \{e\}$, elle coupe I, ce ne peut-être qu'en u d'où $u \in I$. ■

Définition (Séparateur). Soit un graphe $G = (X, E)$. On dit que $Y \subseteq X$ ***sépare*** x et y si x et y sont dans 2 composantes connexes différentes de $G[X \setminus Y]$. On dit aussi que Y est un xy***-séparateur***.

Remarque. S est un xy-séparateur si et seulement si S est un $\{x\}\{y\}$-séparateur ne contenant ni x ni y.

Définitions.

- $q(\{x\}, \{y\}, G) = \begin{cases} 1 & \text{si } x \text{ et } y \text{ sont dans une même composante connexe} \\ 0 & \text{sinon .} \end{cases}$
- $k(x, y, G)$ est le cardinal minimal d'un xy-séparateur si $xy \notin E$. Si $e = xy \in E$, $k(x, y, G) = k(x, y, G \setminus \{e\}) + 1$.
- $p(x, y, G)$ est le nombre maximal de xy-chaînes 2 à 2 disjointes sauf en x et y.

Théorème 9.2 (Menger). *Dans un graphe $G = (X, E)$, on a $p(x, y, G) = k(x, y, G)$.*

Démonstration. Supposons que $xy \notin E$.

On pose $A = \Gamma(x)$ et $B = \Gamma(y)$. Montrons qu'on a $p(x, y, G) = p(A, B, G)$:

Considérons un ensemble de xy-chaînes 2 à 2 disjointes de cardinal $p(x, y, G)$. En supprimant x et y aux extrémités de ces chaînes, on obtient autant de AB-chaînes 2 à 2 disjointes. D'où $p(x, y, G) \leqslant p(A, B, G)$.

Considérons un ensemble de AB-chaînes 2 à 2 disjointes de cardinal $p(A, B, G)$. Soit μ l'une de ces chaînes. Si μ ne contient ni x ni y, on ajoute x et y aux extrémités. Si μ contient x, $\mu = (a, \ldots, x, a', \ldots, b)$. Alors $(x, a', \ldots, b, y)$ est une xy-chaîne. Les autres cas sont semblables. On obtient autant de xy-chaînes 2 à 2 disjointes d'où $p(A, B, G) \leqslant p(x, y, G)$.

D'après le premier théorème, il suffit de prouver que $k(x, y, G) = q(A, B, G)$.

Soit Y un xy-séparateur de cardinal minimum, c'est-à-dire $|Y| = k(x, y, G)$. Montrons que Y est un AB-séparateur. Considérons $\mu = (a, \ldots, b)$ une AB-chaîne. On la transforme en xy-chaîne μ' (voir plus haut). μ' intersecte Y donc μ intersecte Y. Donc, $q(A, B, G) \leqslant k(x, y, G)$.

Soit S un AB-séparateur de cardinal minimum. $|S| = q(A, B, G)$.

Montrons que S ne contient ni x ni y. Supposons que $x \in S$. $S \setminus \{x\}$ n'est pas un AB-séparateur. Dans $G \setminus (S \setminus \{x\})$, il existe une AB-chaîne μ. μ coupe S. Cela ne peut-être qu'en x. On a donc $\mu = (a, \ldots, x, a', \ldots, b)$ avec $a, a' \in A$ et $b \in B$. La sous-chaîne $\mu' = (a', \ldots, b)$ est une AB-chaîne qui ne coupe pas S. Contradiction.

Montrons que S est un xy-séparateur. Soit μ une xy-chaîne, soit μ' la sous-chaîne de μ obtenue en effaçant x et y. μ' est une AB-chaîne, elle coupe donc S. Il en est de même pour μ. On a donc $k(x, y, G) \leqslant q(A, B, G)$.

On a donc démontré le résultat recherché dans le cas $xy \notin E$.

Supposons maintenant que $xy \in E$.
$k(x, y, G) = k(x, y, G \setminus \{e\}) + 1$ par définition.
$p(x, y, G \setminus \{e\}) = k(x, y, G \setminus \{e\})$
$p(x, y, G) = p(x, y, G \setminus \{e\}) + 1$ (il suffit de prendre la chaîne réduite à l'arête e). ■

Définition (Connexité). Soit $G = (X, E)$ un graphe.

La ***connexité*** du graphe G est $k(G) = \min_{x,y} k(x, y, G)$.

Un graphe est ***h-connexe*** si $k(G) \geqslant h$.

Théorème 9.3. *Le graphe $G = (X, E)$ est h-connexe si et seulement si $\forall x, y \in X$, il existe au moins h xy-chaînes 2 à 2 disjointes.*

Démonstration. Cette preuve est due à Menger.

G est h-connexe
$\iff$ $k(G) \geqslant h$
$\iff$ $\forall x, y \in X, k(x, y, G) \geqslant h$
$\iff$ $\forall x, y \in X, p(x, y, G) \geqslant h$
$\iff$ $\forall x, y \in X$, il y a au moins h xy-chaînes 2 à 2 disjointes. ■

Définition (étoile). Soient $G = (X, E)$ un graphe, $Y \subseteq X$ et $x \in X \setminus Y$.

Une xY***-étoile*** est un ensemble de chaînes 2 à 2 sommet-disjointes (sauf en x) reliant x à tous les sommets de y.

Théorème 9.4 (Dirac, 1960). *Un graphe $G = (X, E)$ est h-connexe (avec $h \geqslant 1$) si et seulement si $|X| \geqslant h + 1$ et $\forall Y \subseteq X$ tel que $|Y| = h$, $\forall x \in X \setminus Y$, il existe une xY-étoile.*

Démonstration.

$\Rightarrow$ Supposons G h-connexe. Considérons deux sommets x et y. Il existe h xy-chaînes 2 à 2 disjointes, ce qui entraîne $|\Gamma(x)| \geqslant h$ d'où $|X| \geqslant h + 1$.
Posons $Y = \{y_1, \ldots, y_h\}$. Soit $x \in X \setminus Y$. On ajoute un sommet z que l'on relie à y_1, ... et y_h. On appelle G' le nouveau graphe.
On va montrer que G' est h-connexe. Pour cela, il suffit de prouver que tout tz-séparateur, $t \in X$, contient au moins h sommets. Soit S un tz-séparateur, supposons que $|S| < h$. Alors il existe $y \in Y \setminus S$. Dans $G \setminus S$, t et y sont toujours connectés (car G est h-connexe) et donc dans $G' \setminus S$, t et z sont toujours connectés. Contradiction.
Puisque G' est h-connexe, il existe h xz-chaînes 2 à 2 disjointes. En effaçant z en bout de chaîne, on obtient la xY-étoile.

$\Leftarrow$ Si $\forall Y \subseteq X$, $|Y| = h$, $\forall x \in X \setminus Y$, il existe une xY-étoile et $|X| \geqslant h + 1$. Alors $\forall x \in X, d(x) \geqslant h$ (par une xY-étoile).
Si $x \notin \Gamma(y)$, on considère $Y \subseteq \Gamma(y)$ avec $|Y| = h$: il existe une xY-étoile, on prolonge chaque chaîne par y et on obtient h xy-chaînes 2 à 2 disjointes.
Si $x \in \Gamma(y)$, on prend $Y \subseteq (\{y\} \cup \Gamma(y)) \setminus \{x\}$ et $|Y| = h$. Même conclusion.

■

Chapitre 10

Flot maximum dans un réseau

Définition (Réseau). Un ***réseau*** est un triplet $R = (X, U, c)$ où $G = (X, U)$ est un graphe orienté et $c : X \times X \longrightarrow \mathbb{R}_+$ est nulle hors de U. $c(u)$ est la ***capacité*** de l'arc $u \in U$.

On distingue deux sommets s et p qui seront appelés ***source*** et ***puits***.

Définition (Flot). Soit $R = (X, U, c)$ un réseau, s et p les source et puits. Un ***flot*** φ est une application de $X \times X$ dans $\mathbb{R}$ vérifiant :

i) ***pseudo-symétrie*** : $\forall x, y \in X, \quad \varphi(x, y) = -\varphi(y, x)$;

ii) ***Contraintes de capacité*** : $\forall u \in X \times X, \quad \varphi(u) \leqslant c(u)$;

iii) ***Loi de conservation*** : $\forall x \in X \setminus \{s, p\}, \quad \varphi(x, X) = 0 = \sum_{y \in X} \varphi(x, y)$.

($1^{\text{ère}}$ loi de Kirschhoff)

Par convention, on pose $f(A, B) = \sum_{x \in A} \sum_{y \in B} f(x, y)$.

Remarque. La seconde propriété implique en particulier que $\varphi(x, y) = 0$ si $(x, y) \notin U$ et $(y, x) \notin U$.

Concrètement, on dispose d'un graphe orienté dont les arêtes sont étiquetées par des nombres. On cherche à étiqueter par des nombres inférieurs ou égaux à C en assurant la nullité de leur somme orientée en tout point n'étant pas la source ou le puits.

La valeur du flot φ est $|\varphi| = \varphi(s, X)$. On va s'intéresser à la recherche d'un flot φ maximisant $|\varphi|$.

Remarque. Décider de l'existence d'un flot compatible est NP-complet. En revanche, trouver un flot maximum à partir d'un flot compatible est dans P.

10.1 Méthode de Ford-Fulkerson

«méthode» car il existe plusieurs implantations de complexités différentes (polynomiale ou exponentielle).

On va utiliser le *graphe des écarts*, les *chemins améliorants* et les *coupes*.

10.1.1 Graphe des écarts

Soit $R = (X, U, c)$ un réseau, s et p les source et puits, φ un flot sur R. La capacité résiduelle d'un arc u est $c_\varphi(u) = c(u) - \varphi(u)$.

Définition (Réseau des écarts). Le réseau des écarts est le graphe $R_\varphi = (X, U_\varphi, c_\varphi)$ où $U_\varphi = \{(x, y) \in X \times X, c_\varphi(xy) > 0\}$.

Concrètement, on remplace chaque arc u par un arc étiqueté par $c_\varphi(u)$ si $c_\varphi(u)$ est non nul, on le supprime sinon.

Lemme 10.1. *Soit un réseau $R = (X, U, c)$ et φ un flot sur R. Soit φ' un flot de s à p dans R_φ. Alors $\varphi + \varphi'$ est un flot de valeur $|\varphi + \varphi'| = |\varphi| + |\varphi'|$.*

C'est globalement trivial.

10.1.2 Chemins améliorants

Définition (Chemin améliorant). Étant donnés un réseau $R = (X, U, c)$ et un flot φ, un ***chemin améliorant*** μ relativement à φ est un chemin de s à p dans R_φ.

Définition (Capacité résiduelle). La ***capacité résiduelle*** de μ est $c_\varphi(\mu) = \min\{c_\varphi(u) \mid u \in \mu\}$

Lemme 10.2. *Soient $R = (X, U, c)$ un réseau, φ un flot sur R et μ un chemin améliorant. Alors $\varphi_\mu : X \times X \longrightarrow \mathbb{R}$ défini par :*

$$\varphi_\mu(u) = \begin{cases} c_\varphi(\mu) & si \quad u \in \mu \\ -c_\varphi(\mu) & si \quad \overline{u} \in \mu \\ 0 & sinon \end{cases}$$

est un flot de R_φ de valeur $c_\varphi(\mu)$.

Corollaire. *Avec les mêmes notations, $\varphi' = \varphi + \varphi_\mu$ est un flot de valeur $|\varphi| + |\varphi_\mu| > |\varphi|$.*

10.1.3 Coupes dans les réseaux

Définition (Coupe). Une ***coupe*** dans $R = (X, U, c)$ est une partition de X en $(Y, \overline{Y})$ avec $s \in Y$ et $p \in \overline{Y}$. La capacité de la coupe est $c(Y, \overline{Y}) = \sum_{y \in Y} \sum_{z \in \overline{Y}} c(y, z)$. Le flot à travers la coupe est $\varphi(Y, \overline{Y})$.

Lemme 10.3. $\forall (Y, \overline{Y})$ *coupe d'un réseau R, $|\varphi| = \varphi(Y, \overline{Y})$.*

Démonstration. Soit $(Y, \overline{Y})$ une coupe.

$$\begin{aligned} \varphi(Y, \overline{Y}) &= \varphi(Y, X) - \varphi(Y, Y) \\ &= \varphi(Y, X) \qquad \text{(car chaque arc de Y est vu dans les deux sens)} \\ &= \varphi(s, X) + \varphi(Y \setminus \{s\}, X) \\ &= \varphi(s, X) \qquad \text{(loi de conservation)} \\ \varphi(Y, \overline{Y}) &= |\varphi| \end{aligned}$$

■

Corollaire. *Soient φ un flot sur R et $(Y, \overline{Y})$ une coupe. Alors $|\varphi| \leqslant c(Y, \overline{Y})$.*

Théorème 10.4 (Ford et Fulkerson). *Soit φ un flot un un réseau $R = (X, U, c)$. Il y a équivalence entre les trois propositions suivantes :*

i) φ est un flot maximum ;

ii) Il n'existe pas de chemin améliorant ;

iii) Il existe une coupe $(Y, \overline{Y})$ telle que $|\varphi| = c(Y, \overline{Y})$.

Démonstration.

i⇒ii Démontré précédemment.

iii⇒i Idem.

ii⇒iii Posons $Y = \{x \in X \mid \exists \text{ un chemin de } s \text{ à } x \text{dans} R_\varphi\}$. ii) implique alors $p \in \overline{Y}$. Montrons que $c(Y, \overline{Y}) = |\varphi|$. Soit $(y, z) \in Y \times \overline{Y}$. Si $\varphi(y, z) < c(y, z)$ alors $yz \in U_\varphi$. Donc $y \in Y$. Contradiction. ■

La méthode de Ford-Fulkerson consiste alors à partir du flot nul et à ajouter φ_μ tant qu'il existe un chemin améliorant μ.

Remarque. Nous n'avons pas démontré l'existence d'un flot maximum. On voit qu'il existe si les capacités sont entières car cette méthode termine ($|\varphi|$ croît strictement). On en déduit l'existence pour les capacités rationnelles. La méthode des préflots montrera l'existence dans le cas réel.

10.2 Algorithme d'Edmonds Karp

```
foreach arc (x, y) ∈ U do
    φ(x, y) := 0
    φ(y, x) := 0
while ∃ chemin μ de s à p dans R_φ do
    c_φ(μ) := min {c_φ(x, y) | (x, y) ∈ μ}
    foreach (x, y) ∈ μ do
        φ(x, y) := φ(x, y) + c_φ(μ)
        φ(y, x) := −φ(x, y)
```

Algorithme 12: algorithme d'Edmonds Karp

Le principe de cet algorithme est de choisir dans R_φ, à chaque étape de la méthode de Ford-Fulkerson, un plus court chemin en nombre d'arcs.

On note $d_\varphi(s, x)$ la distance (en nombre d'arcs) de s à x dans R_φ.

Lemme 10.5. *Soient $R = (X, U, c)$ un réseau, φ un flot sur R, μ un plus court chemin et φ' le flot $\varphi + \varphi_\mu$. Alors $\forall x \in X \quad d_\varphi(s, x) \leqslant d_{\varphi'}(s, x)$.*

Démonstration. Supposons le contraire. Soit x un sommet vérifiant $d_\varphi(s, x) > d_{\varphi'}(s, x)$ et minimisant $d_{\varphi'}(s, x)$. Soit ν un plus court chemin de s à x dans R_φ. Soit y le prédécesseur de x dans ce chemin. y existe car $x \neq s$.

Alors $d_{\varphi'}(s, x) = d_{\varphi'}(s, y) + 1 \geqslant d_\varphi(s, y) + 1$. Il y a deux cas possibles :

Si $\varphi(y,x) < c(y,x)$, Alors $(y,x) \in U_\varphi$ et donc $d_\varphi(s,x) \leqslant d_\varphi(s,y)+1 \leqslant d_{\varphi'}(s,y)+1 = d_{\varphi'}(s,x)+1$. Impossible.

Sinon $\varphi(y,x) = c(y,x)$ donc $(y,x) \notin U_\varphi$. Or $yx \in \nu$ donc $(y,x) \in U_{\varphi'}$. Ainsi $xy \in \mu$ et $d_\varphi(s,x) = d_\varphi(s,y)-1 \leqslant d_{\varphi'}(s,y)-1 = d_{\varphi'}(s,x)-2$. Impossible. ■

Théorème 10.6. *La boucle **while** dans l'algorithme d'Edmonds-Karp est exécutée $O(nm)$ fois.*

Démonstration. Un arc (x,y) est dit critique sur le plus court chemin μ si $c_\varphi(x,y) = c_\varphi(\mu)$. Alors l'arc xy va disparaître. Lorsqu'il réapparaîtra, c'est qu'on utilisera l'arc yx sur un chemin améliorant. Alors $d_{\varphi'}(s,x) = d_{\varphi'}(s,y)+1 \geqslant d_\varphi(s,y)+1 = d_\varphi(s,x)+2$. Donc un arc ne peut pas être utilisée plus de $\frac{n}{2}$ fois. ■

Corollaire. *La complexité de l'algorithme est en $O(nm^2)$.*

On raisonne en fait sur un graphe non-orienté et ne calcule pas le graphe des écarts : on ne parcourt que les arcs de capacité résiduelle strictement positive.

10.3 Méthode des préflots — Algorithme de Goldberg

Définition (Préflot). Soit $R = (X,U,c)$ un réseau de source s et de puits p. Un ***préflot*** est une application $\varphi : U \to \mathbb{R}$ qui vérifie :

1. ***Pseudo-symétrie*** : $\varphi(x,y) = -\varphi(y,x)$;
2. Contraintes de ***capacité*** : $\varphi(u) \leqslant c(u) \quad \forall u \in U$;
3. Relaxation de la ***loi de conservation*** : $\varphi(X,x) \geqslant 0 \quad \forall x \in X \setminus \{s\}$.

Le flot en ***excès*** en x est défini par $e(x) = \varphi(X,x)$. Un sommet x sera ***excédentaire*** si $e(x) > 0$.

10.3.1 Les opérations de base

Définition (Fonction de hauteur). Soir $R = (X,U,c)$ un réseau de source s, de puits p. Soit φ un préflot.

Une fonction $h : X \to \mathbb{N}$ est une ***fonction de hauteur*** si :

- $h(s) = |X|$
- $h(p) = 0$
- $h(x) \leqslant h(y)+1 \quad \forall (x,y) \in U_\varphi$

Remarque. Il n'existe pas toujours de fonction de hauteur.

Lemme 10.7. $h(x) > h(y)+1 \quad \Longrightarrow \quad (x,y) \notin U_\varphi$

10.3.2 Les opérations

L'opération Pousser

L'opération `pousser` (x,y) (poussée du flot à travers l'arc (x,y)) peut être appliquée si :

· $e(x) > 0$ (x est excédentaire)
· $c_\varphi(x, y) > 0$ (pas saturé par le flot)
· $h(x) = h(y) + 1$ (on pousse de haut en bas)

Fonction : `pousser` (x, y)
$d_\varphi := \min(e(x), c_\varphi(x, y))$
$\varphi(x, y) := \varphi(x, y) + d_\varphi$
$\varphi(y, x) := -\varphi(x, y)$
$e(x) := e(x) - d_\varphi$
$e(y) := e(y) + d_\varphi$

La première ligne correspond à la conservation des contraintes ; les trois dernières correspondent à la conservation de l'excédentarité. d_φ est le flot qu'on a poussé.

La procédure `pousser` s'effectue en temps constant. Si on avait un préflot avant de l'exécuter, on a toujours un préflot après.

La poussée est ***saturante*** si, après la poussée, l'arc (x, y) est saturé, c-à-d $c_\varphi(x, y) = 0$.

L'opération Élever

L'opération `élever` (x) peut être appliquée si :
· $e(x) > 0$ (x est excédentaire)
· $\forall (x, y) \in U_\varphi, h(x) \leqslant h(y)$
· $x \neq p$

Fonction : `élever` (x)
$h(x) := 1 + \min \{h(y) \mid (x, y) \in U_\varphi\}$

Cette procédure s'effectue en temps linéaire en n.

Remarque. L'opération min ne s'applique jamais à $\varnothing$. En effet, $e(x) > 0$ implique $\varphi(X, x) > 0$ donc $\exists y \in X$ tel que $\varphi(y, x) > 0$, d'où $\varphi(x, y) < 0$. Et $c_\varphi(x, y) = c(x, y) - \varphi(x, y) > 0$ implique $(x, y) \in U_\varphi$.

10.3.3 L'algorithme générique

La fonction init_préflot sature tous les arcs issus de la source.

Lemme 10.8. *Soit $R = (X, U, c)$ un réseau de source s et de puits p. Soit φ un préflot et h la fonction de hauteur.*

Si x est un sommet excédentaire, on peut lui appliquer une opération de poussée ou d'élévation.

Démonstration. Pour tout arc résiduel (x, y), on a $h(x) \leqslant h(y) + 1$. Si on ne peut pas appliquer de poussée, c'est que $h(x) \leqslant h(y)$. On peut donc appliquer une élévation. ■

Fonction : init_préflot(R, s, p)
foreach $x \in X$ **do** $h(x) := 0$ et $e(x) := 0$
foreach $(x, y) \in U$ **do**
  $\varphi(x, y) := 0$
  $\varphi(y, x) := 0$
$h(s) := |X|$
foreach $y \in \Gamma^+(s)$ **do**
  $\varphi(s, y) := c(s, y)$
  $\varphi(y, s) := -\varphi(s, y)$
  $e(y) := \varphi(s, y)$

init_préflot **while** $\exists$ une opération de poussée ou d'élévation applicable **do**
  sélectionner une opération valide et l'appliquer

Algorithme 13: méthode du préflot

10.3.4 Validité de la méthode du préflot

Lemme 10.9. *Durant l'algorithme, les hauteurs ne décroissent jamais. Chaque fois qu'un sommet x est élevé, sa hauteur augmente au moins de 1.*

Lemme 10.10. *A chaque étape de l'algorithme générique, la fonction h est une fonction de hauteur.*

Démonstration. On raisonne par récurrence sur le nombre de passages dans la boucle **while**.

Après l'initialisation, c'est vrai.

Supposons le résultat vrai à une certaine étape et montrons qu'il est vrai à l'étape suivante. Il y a deux cas :

- On fait une opération `élever` (x). $x \neq p$ et $x \neq s$ car s ne sera jamais excédentaire. Si $(x, y) \in U_\varphi$, $h(x) \leqslant h(y)+1$ après élévation. Si $(y, x) \in U_\varphi$, avant élévation on avait $h(y) \leqslant h(x) + 1$ (par hypothèse de récurrence). L'élévation de modifie pas U_φ. Après élévation, on a $h(y) < h(x)+1$. C'est une hauteur.
- On fait une opération `pousser` (x, y). Avant l'opération, on a $h(x) = h(y) + 1$ (sinon, on aurait pas pu pousser). Après l'opération, le graphe des écarts est le même sauf apparition éventuelle de l'arc (y, x).
  $h(y) = h(x)-1$ et on a $h(y) \leqslant h(x)+1$ c'est donc toujours une hauteur. ■

Lemme 10.11. *Soit $R = (X, U, c)$ un réseau, s une source, p un puits, φ un préflot et h une fonction de hauteur.*

Alors il n'existe pas de chemin de s à p dans R_φ, le graphe des écarts.

Démonstration. Supposons le contraire et considérons $\mu = (x_0, \ldots, x_k)$ un chemin élémentaire de $x_0 = s$ à $x_k = p$ dans R_φ.

On a $h(x_i) \leqslant h(x_{i+1}) + 1 \qquad \forall 0 \leqslant i \leqslant h - 1$ puisqu'on est dans R_φ. $h(x_0) = h(s) = |X|$ et $h(x_k) = h(p) = 0$.

Et comme μ est élémentaire $k < |X|$. On a donc $h(s) \leqslant h(p) + k < |X|$. Contradiction. ■

Théorème 10.12. *Si l'algorithme termine, il calcule un flot maximum.*

Démonstration. Si l'algorithme termine, c'est que l'on ne peut plus appliquer l'opération de poussée ou d'élévation et donc d'après le Lemme 10.19, aucun sommet n'est excédentaire : φ est un flot.

D'autre part, comme il n'y a pas de chemin de s à p dans R_φ, d'après le théorème de Ford-Fulkerson, φ est maximum. ■

10.3.5 Analyse de la méthode du préflot

Lemme 10.13. *Soit $R = (X, U, c)$ un réseau, s une source, p un puits, φ un préflot et h une fonction de hauteur.*

Pour tout sommet x excédentaire, il existe un chemin de x à s dans R_φ (le graphe des écarts).

Démonstration. Posons $U = \{y \in X \mid \exists \text{ chemin } x \to y \in R_\varphi\}$ et $\overline{U}$ son complémentaire. Montrons que $s \in U$:

Supposons que $s \in \overline{U}$. Pour tout arc $(y, z) \in U \times \overline{U}$, on a $\varphi(z, y) \leqslant 0$. En effet, si $\varphi(z, y) > 0$, $\varphi(y, z) < 0$ et $c_\varphi(y, z) = c(y, z) - \varphi(y, z) > 0$ donc $(y, z) \in U_\varphi$ et comme $y \in U$ on aurait $z \in U$. Contradiction.

On a donc $\varphi(z, y) \leqslant 0$ et $\varphi(\overline{U}, U) \leqslant 0$.

$$e(U) = \sum_{y \in U} e(y) = \sum_{y \in U} \varphi(X, y) = \varphi(X, U) = \varphi(U, U) + \varphi(\overline{U}, U) \leqslant 0$$

Les sommets de $X \setminus \{s\}$ vérifient $e(x) \geqslant 0$ et comme $U \subseteq X \setminus \{s\}$, on en déduit $\forall y \in U$, $e(y) = 0$, ce qui contredit $e(x) > 0$. ■

Lemme 10.14. *Soit un réseau $R = (X, U, c)$, s une source, p un puits, φ un préflot et h une hauteur.*

On a toujours $h(x) \leqslant 2|X| - 1$.

Démonstration. Considérons un sommet x après une opération d'élévation. Le sommet x est excédentaire (puisqu'on a pu lui appliquer l'opération d'élévation). Il existe donc un chemin $\mu = (x_0, \ldots, x_k)$ élémentaire dans R_φ avec $x_0 = x$ et $x_k = s$. On a donc $k < |X|$ et $h(x_0) = h(x) \leqslant h(s) + k \leqslant 2|X| - 1$ car $h(s) = |X|$. ■

Corollaire. *Il y a au plus $2|X|^2$ opérations d'élévation.*

Démonstration. Chaque fois qu'un sommet est élevé, sa hauteur augmente au moins de un et est toujours inférieure ou égale à $2|X| - 1$. Tous les sommets peuvent être élevés sauf s et p. On élève au plus $|X| - 2$ sommets. Donc, le nombre d'opérations est inférieur à $(2|X| - 1)(|X| - 2) \leqslant 2|X|^2$. ■

Lemme 10.15. *Il y a au plus $2|X|.|U|$ poussées saturantes.*

Démonstration. Lorsqu'on applique une poussée saturante, l'arc sur lequel on pousse disparaît du graphe des écarts.

Soit $x, y \in X$, on veut compter le nombre maximum de poussées saturantes sur (x, y) et (y, x).

Si de telles poussées existent, c'est que $(x, y) \in U$ ou $(y, x) \in U$. Supposons que l'on ait effectué une poussée saturante sur (x, y), l'arc (x, y) disparaît de R_φ. Pour qu'il y revienne, il faut effectuer une poussée sur l'arc (y, x).

Comme on a effectué une poussée saturante sur l'arc (x, y) on avait $h(x) = h(y) + 1$, puis on a fait une poussée sur l'arc (y, x), cela veut dire que $h(y) = h(x) + 1$ et donc, entre deux poussées saturantes sur (x, y) la hauteur de y a augmenté au moins de 2. De même, entre 2 poussées saturantes sur (y, x), la hauteur de x a augmenté de au moins 2.

On considère la suite (finie) s des entiers $\{h(x) + h(y)\}$ entre deux poussées saturantes sur (x, y) ou (y, x). À la première poussée saturante, $h(x)+h(y) \geqslant 1$.

À la dernière poussée saturante, $h(x) + h(y) \leqslant (2|X| - 1) + (2|X| - 2) = 4|X| - 3$.

Or, entre 2 poussées saturantes, $h(x) + h(y)$ augmente de 2, on en déduit que $|s| \leqslant \frac{4|X|-3-1}{2} + 1 = 2|X| - 1 \leqslant 2|X|$. Pour l'ensemble des arcs, on obtient $2|X|.|U|$. ■

Lemme 10.16. *Il y a au plus* $4|X|^2(|X| + |U|)$ *poussées non saturantes.*

Démonstration. On définit la fonction potentiel : $\Theta = \sum_{y \in S} h(y)$ où S est l'ensemble des sommets excédentaires.

Au départ $\Theta = 0$. Une opération d'élévation fait augmenter Θ d'au plus $2|X|$ (R_φ ne varie pas donc il n'y a qu'une hauteur modifiée, or $2|X|$ est la hauteur maximum qu'un sommet puisse atteindre).

Une opération de poussée saturante sur (x, y) fait augmenter Θ d'au plus $2|X|$ (les hauteurs ne varient pas mais il y a éventuellement un sommet supplémentaire dans S).

Une opération de poussée non saturante sur (x, y) fait baisser Θ d'au moins 1 (la poussée est non-saturante donc (x, y) reste dans U_φ donc x n'est plus excédentaire puisque l'on n'a pas pu saturer l'arc. En revanche y est éventuellement excédentaire).

Donc Θ a baissé d'au moins $h(x) - h(y) = 1$.

La valeur maximum que peut atteindre Θ est majorée par :

$$2|X|.\underbrace{\text{nb élévations}}_{2|X|^2} + 2|X|.\underbrace{\text{nb poussées saturantes}}_{2|X|.|U|} = 4|X|^2(|X| + |U|)$$

On en déduit qu'il y a au plus $4|X|^2(|X| + |U|)$ poussées non saturantes. ■

D'où le :

Théorème 10.17. *Si* $n = O(m)$*, l'algorithme générique réalise au plus* $O(n^2m)$ *opérations de base.*

Théorème 10.18. *L'algorithme de Goldberg a une complexité* $O(n^2m)$*.*

Démonstration. On peut réaliser une poussée en $O(1)$ et une élévation en $O(n)$. Le résultat découle des lemmes précédents. ■

Chapitre 11

Coloration de graphes

11.1 Nombre chromatique

11.1.1 deux gros théorèmes

Définition (Nombre chromatique). Le ***nombre chromatique*** $\chi(G)$ d'un graphe G est le plus petit nombre de couleurs tel que qu'on puisse colorier les sommets de G de façon à ce que deux sommets adjacents ne portent pas la même couleur.

C'est aussi une partition de X en un nombre minimum de stables (2 sommets de la même couleur ne sont pas reliés donc les ensembles monochromes sont des stables).

On note $\Delta(G)$ le degré maximum d'un sommet de G.

Proposition 11.1. *Soit $G = (X, E)$ un graphe. On note $\alpha(G)$ le **nombre de stabilité**, c'est à dire la taille d'un stable maximum.*

On a :

$$\left\lceil \frac{n}{\alpha(G)} \right\rceil \leqslant \chi(G) \leqslant n - \alpha(G) + 1$$

Démonstration. On démontre tout d'abord la première inégalité :

Soit $P = \{s_1, .., s_k\}$ une partition minimale en stables de G. On a $k = \chi(G)$. De plus, $|s_i| \leqslant \alpha(G)$ par définition de α, pour $1 \leqslant i \leqslant k$.

Or $n = \sum_1^k |s_i| \leqslant \sum_1^k \alpha(G) = \chi(G).\alpha(G)$.

D'où $\frac{n}{\alpha(G)} \leqslant \chi(G)$ qui est un entier. Donc $\left\lceil \frac{n}{\alpha(G)} \right\rceil \leqslant \chi(G)$.

Démontrons maintenant la seconde inégalité :

On considère un stable S de taille $\alpha(G)$. Il suffit d'une couleur pour colorier S. Il reste alors $n - \alpha(G)$ sommets à colorier. On utilise une couleur différente pour chacun.

D'où $\chi(G) \leqslant n - \alpha(G) + 1$. ■

Corollaire. *Le graphe complet K_n d'ordre n est n-coloriable.*

Démonstration. Pour un graphe complet G, $\alpha(G) = 1$. Le théorème précédent donne alors directement le résultat. ■

Théorème 11.2. *Soit un graphe $G = (X, E)$.*

$$\chi(G) = 1 \Leftrightarrow G \text{ est stable (n'a pas d'arêtes)}$$
$$\chi(G) = 2 \Leftrightarrow G \text{ est biparti}$$

Définitions (Point d'articulation, isthme, bloc). Soit $G = (X, E)$ un graphe connexe.

Un sommet $x \in X$ est un ***point d'articulation*** si $G \setminus \{x\}$ n'est pas connexe.

Une arête $e \in E$ est un ***isthme*** si $G \setminus \{e\}$ n'est pas connexe.

Un ***bloc*** de G est soit un isthme, soit une composante 2-connexe (c'est à dire un sous graphe 2-connexe maximal).

Le ***graphe des blocs*** $G_B = (X_B, E_B)$ est défini par :

$X_B = \{B_i \mid B_i \text{ bloc}\} \cup \{a_j \mid a_j \text{ point d'articulation}\}$

$E_B = \{a_j B_i \mid a_j \in B_i\}$

Remarque. Dans ce graphe, on place un sommet par bloc, et un sommet par point d'articulation. Les blocs sont reliés aux points d'articulations qu'ils contiennent dans le graphe d'origine. On voit clairement à travers sa définition que le graphe des blocs est biparti.

Proposition 11.3. *Soit $G = (X, E)$ un graphe connexe.*

L'intersection des ensembles des sommets de deux blocs distincts, est vide, ou réduit à un point d'articulation.

G_B est un arbre.

Rappel. Un graphe est ***k-connexe*** s'il reste connexe après retrait de $k - 1$ sommets quelconques.

Théorème 11.4 (Brooks, 1941). *Soit G un graphe tel que $\Delta(G) \geqslant 3$ et G ne contient pas de clique à $\Delta(G) + 1$ sommets.*

Alors $\chi(G) \leqslant \Delta(G)$.

Remarque. Si G est connexe, on peut numéroter les sommets de G de sorte que x_i à un voisin dans $\{x_1, \dots x_{i-1}\}$. En effet, on choisit x_1 quelconque. Si on a déjà numéroté convenablement $x_1, \dots, x_{i-1}$, on choisir pour x_i l'extrémité d'une arête dans le cocycle $\omega(\{x_1, \dots, x_{i-1}\})$ et qui n'est pas dans cet ensemble $(\{x_1, \dots, x_{i-1}\})$. On a $\omega(\{x_1, \dots, x_{i-1}\}) \neq \varnothing$ sinon G ne serait pas connexe.

On va commencer par montrer le résultat dans un cas très particulier (le lemme), puis on généralisera.

Lemme 11.5. *S'il existe dans G deux sommets a et b tel que $d(a, b) = 2$ et si $G \setminus \{a, b\}$ connexe, alors $\chi(G) \leqslant \Delta(G)$.*

Démonstration. Notons x_1 le voisin commun de a et b. D'après la remarque précédente, on peut numéroter les sommets de $X \setminus \{a, b\}$ sous la forme $\{x_1, \dots, x_{n-2}\}$ de sorte que x_i a un voisin dans $\{x_1, \dots, x_{i-1}\}$.

Montrons que l'on peut colorier le graphe avec $\Delta(G)$ couleurs : on colorie a et b avec la couleur 1. On colorie les sommets de x_{n-2} à x_1 en utilisant la plus petite couleur disponible. Pour tout sommet x_i (pour $2 \leqslant i \leqslant n-2$), x_i a un voisin dans $\{x_1, \dots, x_{i-1}\}$, la partie non encore coloriée. Comme $d(x_i) \leqslant \Delta(G)$, il reste une couleur disponible dans $\{1, \dots, \Delta(G)\}$. Lorsqu'on colorie x_1, tous ses voisins sont déjà coloriés et a et b ont la même couleur : il reste une couleur disponible pour x_1 dans $\{1, \dots, \Delta(G)\}$. ■

Intéressons nous maintenant au cas général :

Démonstration. On raisonne sur la connexité de G.

Si G est 3-connexe : le graphe n'étant pas une clique (par hypothèse), il existe deux sommets a et b à distance 2. Comme G est 3-connexe, $G \setminus \{a, b\}$ est connexe et on peut appliquer le lemme.

Si G est 2-connexe : G ne peut pas avoir plus de 2 sommets universels (de degré $n-1$) puisque tant que l'on ne supprime pas tous les sommets universels, on ne déconnecte pas le graphe.

⋆ Si G possède 1 ou 2 sommets universels : appelons x l'un d'entre eux. G n'étant pas complet (par hypothèse), il existe deux sommets a et b (différents de x, à distance 2 l'un de l'autre. Un chemin de longueur de 2 entre a et b passe par x car x est universel. $G \setminus \{a, b\}$ est connexe puisque l'on conserve le sommet universel x. On peut donc encore appliquer le lemme.

⋆ Si G ne possède pas de sommet universel : choisissons un sommet $x \in X$ tel que $3 \leqslant d(x) < n-1$ (possible car $\Delta(G) \geqslant 3$). Le graphe $G \setminus \{x\}$ est soit 2-connexe, soit 1-connexe.

□ Si $G \setminus \{x\}$ est 2-connexe : $\{x\} \cup \Gamma(x) \neq X$ car $d(x) < n-1$. G étant connexe, $\omega(\{x\} \cup \Gamma(x)) \neq \varnothing$. Soit y l'extrémité d'une arête de ce cocycle, à l'extérieur de $\{x\} \cup \Gamma(x)$. x et y ne sont pas reliés donc y est relié à un voisin de x et $d(x, y) = 2$. De plus, $G \setminus \{x, y\}$ est connexe car $G \setminus \{x\}$ est 2-connexe. On peut donc appliquer le cas vu plus haut.

□ Si $G \setminus \{x\}$ est 1-connexe : $G \setminus \{x\}$ admet au moins un point d'articulation. On considère $(G \setminus \{x\})_B$ l'arbre des blocs, il a au moins deux blocs pendants B_1 et B_2 (correspondant aux feuilles) puisque c'est un arbre, et il existe un point d'articulation. Ces deux blocs sont reliés au reste de $G \setminus \{x\}$ par des points d'articulation b_1 et b_2. En fait, x a un voisin $a \neq b_1$ dans B_1 car sinon, en enlevant b_1 à G, on déconnecterait le graphe initial (absurde car G est 2-connexe). De même x a un voisin b dans B_2 autre que b_2.

a et b ne sont pas reliés car sinon, on aurait deux chaînes disjointes les reliant (l'arête ab et la chaîne passant par b_1 puis b_2), et on contredirait le fait que a et b sont dans 2 blocs différents. De plus ils ont x comme voisin commun donc sont à distance 2 l'un de l'autre. Le graphe $G \setminus \{a, b, x\}$ est connexe car a et b ne sont pas les points d'articulation de B_1 et B_2

$G \setminus \{a, b\}$ est connexe car $d(x) \geqslant 3$ donc x est connecté à $G \setminus \{a, b\}$. Donc on peut appliquer à $G \setminus \{a, b\}$ le résultat vu plus haut.

Si G est 1-connexe : on considère B un bloc pendant de G. Soit b le point d'articulation de B.

⋆ Si B est une clique à $\Delta(B) + 1$ sommets, $\Delta(B) + 1 \leqslant \Delta(G)$ puisque G ne contient pas de clique de taille $\Delta(G) + 1$. On peut donc colorier B à l'aide d'au plus $\Delta(G)$ couleurs.

⋆ Si B n'est pas une clique à $\Delta(B) + 1$ sommets, on peut toujours colorier B en $\Delta(B) \leqslant \Delta(G)$ couleurs.

On a montré que l'on peut colorier B avec au plus $\Delta(G)$ couleurs. Reste à montrer le résultat pour tout G :

Par récurrence, on peut colorier le graphe $(G \setminus B) \cup \{b\}$ avec $\Delta(G)$ couleurs. En faisant coïncider les deux colorations sur b, on obtient une coloration de G en $\Delta(G)$ coloris.

Si G est non-connexe : il suffit de travailler sur les composantes connexes. ■

Théorème 11.6 (Tutte, 1954). *Pour tout entier $k \geqslant 2$, il existe un graphe k-chromatique sans triangles (sans cliques à 3 sommets K_3).*

Démonstration. On distingue trois cas.

- **$k = 2$** C'est trivial car un graphe 2-chromatique est biparti et ne peut donc pas contenir de triangle.
- **$k = 3$** Un cycle de longueur impaire convient (sans triangle).
- **$k \geqslant 4$** On construit une famille de graphes $G_i = (X_i, E_i)$ telle que G_i est i-chromatique et sans triangle. Pour G_3, on prend C_5 le cycle à 5 sommets. On note $n_i = |X_i|$.

  Supposons G_i construit et construisons G_{i+1} : on prend $p = C_{in_i - i + 1}^{n_i}$ copies de G_i notées $G_{i_1}, \ldots, G_{i_p}$. On ajoute $in_i - i + 1$ sommets appelés centraux (ils forment un ensemble noté A). On note $A_1, \ldots, A_p$ les sous-ensembles de A à n_i sommets.

  Pour tout j tel que $1 \leqslant j \leqslant p$, on ajoute un couplage parfait entre G_{i_j} et A_j. Cela forme G_{i+1}. Montrons que cela convient :

  G_i sans triangle $\Rightarrow$ G_{i+1} sans triangle : car s'il y a un triangle dans G_{i+1}, il contient un arête ab avec $a \in A$ et $b \in G_{i_j}$. Mais un sommet de A ne peut pas avoir 2 voisins dans une même copie de G_i (à cause du couplage parfait). Contradiction.

  G_i i-coloriable $\Rightarrow$ G_{i+1} $(i+1)$-coloriable : il suffit de reporter la i-coloration de G sur chaque copie, et d'ajouter une couleur pour a.

  G_{i+1} i-coloriable $\Rightarrow$ G_i $(i-1)$-coloriable : considérons une i-coloration de G_{i+1}. C'est en particulier une i-coloration de A. Notons $\{s_1, \ldots, s_i\}$ la partition de A en classes de couleurs. L'un des s_j a au moins n_i sommets. En effet, supposons que $|s_j| < n_i$ (où $1 \leqslant j \leqslant i$), on aurait $|A| = \sum_1^i |s_j| \leqslant \sum_1^i (n_i - 1) = i(n_i - 1) < in_i - i + 1 = |A|$. Contradiction.

  Il existe donc un sous-ensemble A_k de A de cardinal n_i monochromatique. Le graphe G_{i_k} couplé avec A_k ne peut pas utiliser cette couleur. On en déduit que G_{i_k} est $(i-1)$-colorié donc G_i est $(i-1)$-coloriable.

  Par récurrence, on en déduit que G_{i+1} est $(i+1)$-chromatique. ■

11.1.2 Un problème dû à Berge

Définition (Graphe parfait). G est ***parfait*** si et seulement si $\chi(G') = \omega(G')$ pour tout sous-graphe G' de G, où $\omega(G')$ désigne la taille de la plus grande clique de G'.

On cherche à caractériser cette classe de graphes.

Théorème 11.7 (ex-conjecture faible des graphes parfaits). *Un graphe est parfait si et seulement si son complémentaire est parfait.*

Admis.
Les cycles sans cordes de longueur impaire ne sont pas parfaits.

Définition (Trou, anti-trou). Un ***trou*** est un cycle sans cordes. Un ***anti-trou*** est le complémentaire d'un trou.

Théorème 11.8 (mai 2002, ex-conjecture des graphes parfaits).
Un graphe est parfait si et seulement si il ne contient ni trou ni anti-trou impair.

11.1.3 Des problèmes plus basiques

Théorème 11.9 (Roy, 1967 et Gallai, 1968). *Soit $G = (X, E)$ un graphe non-orienté q-chromatique.*

1. *Pour toute orientation de G, il existe un chemin élémentaire de longueur au moins $q - 1$,*
2. *Il existe une orientation de G pour laquelle tout chemin est de longueur au plus $q - 1$.*

Démonstration. Une orientation de G consiste à choisir pour chaque arête $ab \in E$ l'arc (a, b) ou l'arc (b, a). Il y a 2^m orientations possibles.

1 On va construire une suite de partitions $\mathcal{P}_k = \{C_0^k, \ldots, C_{p_k}^k\}$ de X vérifiant $\forall x \in C_i^k$, il existe un chemin d'extrémité terminale x, et de longueur i. La dernière partition de cette suite aura chaque C_i^k stable, ce qui impliquera $p_k \geqslant \chi(G)$ et donc le résultat recherché.
On pose $\mathcal{P}_0 = X$ et on suppose avoir construit $\mathcal{P}_k = \{C_0^k, \ldots, C_{p_k}^k\}$. S'il existe i tel que C_i^k n'est pas stable, c'est-à-dire C_i^k contient un arc (x, y), on pose $C_i^{k+1} = C_i^k \setminus \{y\}$, $C_{i+1}^{k+1} = C_i^k \cup \{y\}$ et $C_j^{k+1} = C_j^l$ pour tout j différent de i et $i + 1$.
Clairement, la limite de cette suite est une partition en stables.

2 On colorie G avec q couleurs numérotées de 1 à q. Si $c(a) < c(b)$, on oriente avec l'arc (a, b), sinon avec (b, a). Conclusion, $(x_1, \ldots, x_k)$ vérifie $1 \leqslant c(x_1) < \cdots < c(x_k) \leqslant q$ donc $k \leqslant q$ donc la longueur du chemin est inférieure ou égale à $q - 1$. ■

Définition (Tournoi). Un ***tournoi*** est un graphe complet orienté.

Définition (Graphe hamiltonien). Un graphe est dit ***hamiltonien*** s'il existe un chemin passant une fois et une seule par chaque sommet.

Corollaire. *Un tournoi est hamiltonien.*

Démonstration. Un tournoi d'ordre n est un graphe complet donc est n-coloriable. Il existe un chemin élémentaire de longueur au moins $n-1$. Ce chemin passe forcément par chaque sommet donc est un chemin hamiltonien. ■

Théorème 11.10 (Nordhaus et Gaddum, 1960). *Soit $G = (X, E)$ un graphe, son complémentaire est $\overline{G} = (X, \overline{E})$ où $\overline{E} = \mathcal{P}_2(X) \setminus E$.*

1. $\chi(G).\chi(\overline{G}) \geqslant n$
2. $\chi(G) + \chi(\overline{G}) \leqslant n + 1$
3. $\chi(G) + \chi(\overline{G}) \geqslant 2\sqrt{n}$
4. $\chi(G).\chi(\overline{G}) \leqslant \left(\frac{n+1}{2}\right)^2$

Démonstration. **1** Soit $\{s_1, \ldots, s_k\}$ une partition minimale de $\overline{G}$ en stables c'est-à-dire $k = \chi(\overline{G})$. Les s_i sont des cliques dans G d'où $|s_i| \leqslant \chi(G)$.

Donc $\quad n = \sum_1^k |s_i| \leqslant k.\chi(G) = \chi(G).\chi(\overline{G})$.

2 Le résultat est trivialement vrai pour $n = 1$ ou 2 (voire 3). Supposons $n > 2$.
Soit x_0 un sommet quelconque de G. Notons $G_0 = G \setminus \{x\}$.
On a $\chi(G) \leqslant \chi(G_0) + 1$ et $\chi(\overline{G}) \leqslant \chi(\overline{G_0}) + 1$.
Si ces deux inégalités sont des égalités, on a en fait $d_G(x_0) \geqslant \chi(G_0)$ et $d_{\overline{G}}(x_0) \geqslant \chi(\overline{G_0})$. On a alors :

$$\chi(G) + \chi(\overline{G}) = \chi(G_0) + \chi(\overline{G_0}) + 2 \leqslant d_G(x_0) + d_{\overline{G}}(x_0) + 2 = n + 1$$

Si l'une des deux n'est pas une égalité, on a :

$$\chi(G) + \chi(\overline{G}) \leqslant \chi(G_0) + \chi(\overline{G_0}) + 1 \leqslant n + 1$$

3 $\quad 4xy \leqslant (x-y)^2 + 4xy = (x+y)^2$

Donc $\quad 4n \leqslant 4\chi(G)\chi(\overline{G}) \leqslant \big(\chi(G) + \chi(\overline{G})\big)^2$

D'où $\quad (\chi(G) + \chi(\overline{G})) \geqslant 2\sqrt{n}$

4 Le résultat se montre de la même façon que le 2. ■

11.2 Indice chromatique

Définition (Indice chromatique). L'***indice chromatique*** $\chi'(G)$ d'un graphe G est le cardinal minimal d'une partition en couplages de l'ensemble des arêtes.

C'est également le nombre minimum de couleurs nécessaires pour colorier les arêtes de sorte que deux arêtes adjacentes ne sont jamais de la même couleur.

Proposition 11.11. *Soit $G = (X, E)$ un graphe. On note $\nu(G)$ la taille maximale d'un couplage.*

On a alors :

$$\Delta(G) \leqslant \chi'(G) \qquad \left\lceil \frac{|E|}{\nu(G)} \right\rceil \leqslant \chi'(G)$$

Démonstration. Tout d'abord, il existe un sommet x tel que $d(x) = \Delta(G)$. En x, on a besoin de $\Delta(G)$ couleurs différentes donc $\Delta(G) \leqslant \chi'(G)$.

Ensuite, soit $\{C_1, \ldots, C_k\}$ une partition de E minimale, en couplage. On a $k = \chi'(G)$.

$$|E| = \sum_1^k |C_i| \leqslant \sum_1^k \nu(G) = \chi'(G).\nu(G)$$

Donc $\chi'(G) \geqslant \frac{|E|}{\nu(G)}$. Le fait que $\chi'(G)$ soit un entier permet d'ajouter la partie entière supérieure. ■

Proposition 11.12.

$$\chi'(K_{2n}) = 2n - 1 \qquad \chi'(K_{2n+1}) = 2n + 1$$

Démonstration. **Le cas de K_{2n}** $m(K_{2n}) = \frac{(2n)(2n-1)}{2} = n(2n-1)$ et on voit facilement que $\nu(K_{2n}) = n$. La proposition précédente donne alors $\chi'(K_{2n}) \geqslant 2n - 1$.

Pour montrer l'inégalité contraire, on colorie K_{2n} avec $2n - 1$ couleurs. On numérote tout d'abord les sommets de 0 à $2n - 1$. On place $2n - 1$ au centre tandis que les $2n - 1$ autres forment un polygone régulier tout autour.

On relie alors $2n - 1$ à 0, puis 1 à $2n - 2$, 2 à $2n - 3$, ..., et $n - 1$ à n. On forme ainsi un couplage parfait C_0. On décale les sommets du polygone pour former le couplage parfait C_1. $2n - 1$ est relié à 1, puis 2 à 0, 3 à $2n - 2$, 4 à $2n - 3$, ..., n à $n + 1$.

De la façon, on forme successivement les couplages parfaits C_0 à C_{2n-2}. Le couplage C_k est en fait $\{(k, 2n-1)\} \cup \{(i + k, 2n - 1 - i + k) \mid 1 \leqslant i \leqslant n - 1\}$ (on compte les sommets modulo $2n - 1$).

Il est clair que ces $2n-1$ couplages forment une partition de l'ensemble des arêtes. En associant une couleur à chaque couplage, on obtient le coloriage recherché.

Le cas de K_{2n+1} $m(K_{2n+1}) = \frac{(2n)(2n+1)}{2} = n(2n+1)$ et on voit que $\nu(K_{2n+1}) = n$. La proposition précédente donne alors $\chi'(K_{2n+1}) \geqslant 2n + 1$.

K_{2n+1} est inclus dans K_{2n+2} donc $\chi'(K_{2n+1}) \leqslant \chi'(K_{2n+2}) = 2n + 1$. ■

Théorème 11.13 (Vizing, 1964). *Pour tout graphe G, $\Delta(G) \leqslant \chi'(G) \leqslant \Delta(G) + 1$.*

Remarque. Déterminer si $\chi'(G)$ vaut $\Delta(G)$ ou $\Delta(G) + 1$ est un problème NP-complet.

Démonstration. Il nous reste à prouver la seconde inégalité. On va raisonner par récurrence sur $|E|$. Si $|E| \leqslant \Delta + 1$, il est clair qu'on peut colorier le graphe avec $\Delta + 1$ couleurs.

Si $|E| > \Delta + 1$, on peut supposer que tout le graphe sauf une arête se colorie avec $\Delta + 1$ couleurs. Notons xy_1 cette arête non-coloriée.

La définition de $\Delta(G)$ impose l'absence d'au moins une couleur en chaque sommet. En effet, les degrés des sommets sont au maximum Δ et on utilise ici $\Delta + 1$ couleurs.

S'il manque une même couleur en x et y_1, on colorie l'arête avec cette couleur et le problème est résolu. Tout le problème réside donc dans le cas où les couleurs manquantes en x et en y_1 sont différentes.

On construit la suite (xy_1, C_1), (xy_2, C_2), ... et (xy_h, C_h) vérifiant C_i est la couleur manquante en y_i et l'arête y_{i+1} est coloriée en C_i. On prend cette suite maximale. Cela impose soit l'absence d'une arête incidente à x de couleur C_h, soit la présence de C_h parmi les éléments précédents de la suite.

On note C la couleur absente en x. Par construction, $C \neq C_i$ pour tout $1 \leqslant i \leqslant k-1$. Deux cas se présentent :

Si C_h est absente en x, on décale toutes les couleurs : xy_1 est colorié en C_1, ..., xy_h en C_h. Le problème est alors résolu.

Sinon, la couleur C_h est présente en x. Par maximalité de la suite, il existe j tel que xy_j est colorié en C_h ($C_{j-1} = C_h$).

On décale également les couleurs : xy_i est colorié en C_i pour $i < j$, xy_i rest en C_{i-1} pour $i > j$. xy_j n'a plus de couleur.

On considère le graphe partiel H de G réduit aux couleurs C et C_h. On a $d_H(x) = 1$ car C est absente mais C_h est présente. $d_H(y_h) \leqslant 1$ car C_h est absente. $d_H(y_j) \leqslant 1$ car on a supprimé la couleur C_h.

Si x et y_j sont dans des composantes connexes de H différentes. Si $d_H(y_j) = 0$, c'est-à-dire C absente en y_j, il suffit de colorier xy_j en C pour résoudre le problème.

Sinon, $d_H(y_j) = 1$ et C est présente en y_j. La composante connexe de x est une chaîne alternée de C et C_h finissant en x (grâce au degré) par une arête C_h. La composante de y_j finit en y_j par une arête C. On échange donc C et C_h dans cette dernière et on colorie xy_j en C pour résoudre le problème.

Si x et y_j sont dans la même composante connexe de H. On a fini de décaler les couleurs dans le graphe initial. xy_i est colorié en C_i pour tout $1 \leqslant i \leqslant h-1$ et xy_h n'est pas coloriée. On considère de nouveau le graphe partiel H. Il n'a pas changé car on a décalé des couleurs différentes de C et C_h. On a toujours une chaîne entre y_j et x, alternée en C et C_h, finissant par une arête C_h en x. La composante connexe de y finit en y par une arête coloriée en C. On échange C et C_h dans cette composante pour résoudre le problème. ■

Liste des Algorithmes