École normale supérieure de Lyon
1ère Année de Magistère d'Informatique et Modélisation

Graphes et algorithmique des graphes

par Brice GOGLIN

d'après le cours de M. Vincent Bouchitté

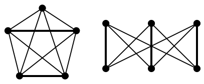
Sommaire

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.

Définitions

Définition (1.1) — 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 (1.2) — Graphe orienté

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

On autorise également parfois les boucles ou arcs multiples.

Définitions (1.3) — Adjacence, incidence, voisinage, degré

Deux sommets x et y d'un graphe $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *sont* **adjacents** *si* $xy \in \mathrm{E}$.
Deux arêtes e et f d'un graphe $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *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* $\Gamma(x) = \{y \in \mathrm{X} | xy \in \mathrm{E}\}$. *Le* **degré** *du sommet x est alors* $d(x) = |\Gamma(x)| = Card\ \Gamma(x)$.

Pour un graphe orienté, les définitions peuvent être précisées :

Définitions (1.4) — Voisinages extérieurs et intérieurs

Soit $\mathrm{G} = (\mathrm{X}, \mathrm{U})$ *un graphe orienté et x un sommet.*
Le **voisinage extérieur** *de x est défini par* $\Gamma^+(x) = \{y \in \mathrm{X} | (x, y) \in \mathrm{U}\}$.
Son **voisinage intérieur** *est* $\Gamma^-(x) = \{y \in \mathrm{X} | (y, x) \in \mathrm{U}\}$.
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 (1.5) — 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 $d(x) = |\mathrm{X}| - 1$, *le sommet x est dit* **dominant**.

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

Propriétés (1.6)

Soit $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *un graphe non-orienté.*

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

<u>*Preuve :*</u>

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

Propriété (1.7)

Soit $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *un graphe. Il existe deux sommets différents ayant le même degré.*

<u>*Preuve :*</u>

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 (1.8) — Sous-structures

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

Définitions (1.9) — 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 (1.10) — Parcours, chaîne, cycle, longueur

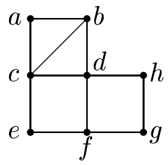
Un **parcours** *de* G *est une suite d'arêtes* $\mu = (x_1, ..., x_k)$ *telle que* $x_i x_{i+1} \in \mathrm{E}$ *pour tout* $i \in \{1, ..., k-1\}$.
Un parcours $(x_1, ..., x_k)$ *est* **fermé** *si* $x_i = x_k$.
Une **chaîne** *de* $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *est un parcours sans répétition 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.



Propriété (1.11)

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

Preuve :

Soit $\mu = (x_1, ..., x_k)$ une chaîne. Si μ n'est pas élémentaire, il existe $i < j$ tel que $x_i = x_j$. Alors $\mu' = (x_1, ..., x_i, x_{j+1}, ...x_k)$ 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 (1.12) — Sommets connectés, relation d'équivalence

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 possèdant qu'une seule composante connexe est dit* **connexe**.

Chapitre 2

Arbres

Définition (2.1) — Arbre

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

Lemme (2.2)

Soit $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *un graphe non-orienté.*

1. *Si* G *est sans cycle et* $|\mathrm{E}| \geqslant 1$ *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.*

Preuve :

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

   $|\mathrm{E}| \geqslant 1$ 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 $y \in \{x_3, ..., x_k\}$, 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 (2.3)

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.*

<u>*Preuve :*</u>

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 $\mathrm{G} + xy$ possède deux cycles distincts $\nu_1 = (x, ..., z, t_1, ..., y)$ et $\nu_2 = (x, ..., z, t_2, ..., y)$ avec $(x, ..., z)$ le plus long préfixe commun à ν_1 et ν_2. Notons $\nu_1' = (z, t_1, ..., y)$ et $\nu_2' = (z, t_2, ..., y)$. 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.

$\boxed{\mathbf{4)} \Rightarrow \mathbf{5)}}$ 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, $\mathrm{G} + xy$ contient un cycle $\nu = (\underbrace{x, ..., y}_{\mu}, x)$. μ est une chaîne connectant x et y. Donc G est connexe.

Montrons sa minimalité. Soit $xy \in$ E. Si $\mathrm{G} \backslash \{xy\}$ est connexe, il existe une chaîne μ reliant x et y dans $\mathrm{G} \backslash \{xy\}$. 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.

$\boxed{\mathbf{5)} \Rightarrow \mathbf{6)}}$ 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 $\mathrm{G} \backslash \{zz'\}$ 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.

$\boxed{\mathbf{6)} \Rightarrow \mathbf{1)}}$ 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 $(x, ..., y, x)$, les chaînes $(x, ..., y)$ et (y, x) relient x à y de deux façons différentes. On peut extraire une chaîne élémentaire de $(x, ..., y)$ 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

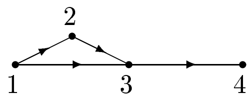
Parcours dans les Graphes

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 :** $\mathrm{M} = (a_{ij})_{1 \leqslant i,j \leqslant n}$ 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.

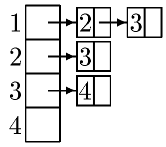
Exemple :



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 $\mathrm{O}(n+m)$.

Existence d'un arc

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

Degré d'un sommet

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

3.2.3 Analyse de l'algorithme

1) <u>pour tout</u> $x \in X$ <u>faire</u> $etat[x] \leftarrow non\_atteint$
2) $T \leftarrow \varnothing$; $A\_traiter \leftarrow \varnothing$
3) <u>pour tout</u> $x \in X$ faire
    <u>si</u> $etat[x] = non\_atteint$ <u>alors</u>
4) $A\_traiter \leftarrow A\_traiter \cup \{x\}$
5) $etat[x] \leftarrow atteint$
6)   <u>tant que</u> $A\_traiter \neq \varnothing$ <u>faire</u>
7) $y \leftarrow choix(A\_traiter)$
            $A\_traiter \leftarrow A\_traiter \backslash \{y\}$
8)   <u>pour tout</u> $z \in \Gamma^+(y)$<u>faire</u>
9)   <u>si</u> $etat[z] = non\_atteint$ <u>alors</u>
10) $T \leftarrow T \cup \{(y, z)\}$
11) $etat[z] \leftarrow atteint$
12) $A\_traiter \leftarrow A\_traiter \cup \{z\}$
13) $etat[y] \leftarrow examine$

Remarque :

Chaque sommet entre une fois et une seule dans *A_traiter* (ligne 4 et 12). Quand il entre dedans, il change d'état, ce qui l'empêche d'y entrer une seconde fois.

La boucle de la ligne 6 garantit que tout sommet sortira de l'algorithme en étant passé à l'état examiné.

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

Dans T, on aura une forêt orientée. En effet, on part d'un sommet x pour réaliser un parcours, ce sommet x (ligne 3) sera racine d'un des arbres de la forêt T, les autres sommets visités après s ne seront pas racine.

3.2.4 Complexité

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

Faisons une description ligne-à-ligne du coût :

1) $\mathrm{O}(n)$ car $|\mathrm{X}| = n$.

2) T sera codé comme un tableau. L'intialisation coutera donc O(1) ou $\mathrm{O}(n)$.

3) $\mathrm{O}(n)$ (on ne compte pas l'intérieur de la boucle).

4 et 5) Effectués au plus une fois par sommet. Coût total de $\mathrm{O}(n)$ sur l'algorithme.

7) Effectué une fois par sommet. $\mathrm{O}(n)$ sur le total de l'algorithme.

8) Une fois par sommet. $\mathrm{O}(n)$.

9) Effectué $d^+(y)$ fois pour un sommet y. La somme est donc en $\mathrm{O}(m)$.

10 à 12) Effectués au plus une fois par sommet. $\mathrm{O}(n)$.

Avec ces hypothèses, l'algorithme a donc un coût total inférieur à $\mathrm{O}(n+m)$.

Par contre, si on avait eu une fonction de choix en $\mathrm{O}(n)$, comme on l'appelle n fois, on aurait un coût en $\mathrm{O}(n^2)$.

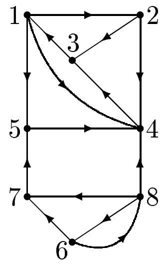
3.3 Parcours en largeur

En fait, c'est une généralisation des parcours hiérarchiques des arbres (on regarde la racine puis les voisins, puis les voisins des voisins, ...).

On utilisera comme critère de choix le fait qu'un sommet atteint sera toujours un sommet examiné. On va donc gérer la variable "*A_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.

3.3.1 Description du parcours en largeur

A_traiter est un tableau avec 2 pointeurs : un marqueur de début i et un marqueur d'insertion j. La partie active dans *A_traiter* est donc située entre i et $j-1$.

pour tout $x \in \mathrm{X}$ faire
    $etat[x] \leftarrow non\_atteint$
    $pere[x] \leftarrow \perp$ // Valeur indéterminée
$i \leftarrow 1$; $j \leftarrow 1$
pour tout $x \in \mathrm{X}$ faire
si $etat[x] = non\_atteint$ alors
    $etat[x] \leftarrow \mathrm{A}\_traiter[i]$; $i{+}{+}$
    $\mathrm{A}\_traiter[j] \leftarrow atteint$
    tant que $i < j$ faire
        $y \leftarrow \mathrm{A}\_traiter[i]$; $i{+}{+}$
        pour tout $z \in \Gamma^+(y)$ faire
            si $etat[z] = non\_atteint$ alors
                $etat[z] \leftarrow atteint$
                $\mathrm{A}\_traiter[j] \leftarrow z$
                $j{+}{+}$
                $pere[z] \leftarrow y$

Le choix et la suppression se font bien en temps constant (incrémentation d'indice). On vérifie donc bien les hypothèses du cas général étudié plus haut. La complexité de l'algorithme est donc $\mathrm{O}(n+m)$.

Le plus simple est de représenter la file dans un tableau à l'aide de 2 indices.

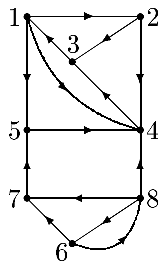
3.4 Parcours en profondeur

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 structure de donnée associée est la pile.

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

On va utiliser un programme auxiliaire dont le but est de rechercher un élément.

<u>procedure</u> $recherche(x)$
  $etat[x] = atteint$
  $\mathrm{OE}[x] \leftarrow i$; i++ // Inutile ici
  <u>pour tout</u> $y \in \Gamma^+(x)$ <u>faire</u>
    <u>si</u> $etat[y] = non\_atteint$ <u>alors</u>
      $pere[y] \leftarrow x$
      $recherche(y)$
  $etat[x] = examine$ // Inutile ici
  $\mathrm{OD}[x] \leftarrow j$; ++ // Inutile ici

Dans l'exemple cité plus haut, on utilise en fait $recherche(1)$ et $recherche(6)$.
Donnons maintenant le code du programme principal :

$i \leftarrow 1$; $j \leftarrow 1$
<u>pour tout</u> $x \in \mathrm{X}$ <u>faire</u>
  $etat[x] \leftarrow non\_atteint$
  $pere[x] \leftarrow \perp$ // Valeur indéterminée
<u>pour tout</u> $x \in \mathrm{X}$ <u>faire</u>
<u>si</u> $etat[x] = non\_atteint$ <u>alors</u>
  $recherche(x)$

3.4.2 Retour à l'exemple

Dans le parcours en profondeur, il y a quatre types d'arcs :

▷ Les <u>arcs de la forêt</u> (comme $(1,2)$ ou $(2,4)$). Si on les note $arc(x,y)$, ils vérifient $\mathrm{OE}[x] < \mathrm{OE}[y]$ et $\mathrm{OD}[x] > \mathrm{OD}[y]$.

▷ 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 $\mathrm{OE}[x] < \mathrm{OE}[y]$ et $\mathrm{OD}[x] > \mathrm{OD}[y]$.

▷ 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 $\mathrm{OE}[x] > \mathrm{OE}[y]$ et $\mathrm{OD}[x] < \mathrm{OD}[y]$.

▷ Les arcs traversiers (comme $(4,3)$ et $(7,5)$). Ce sont les autres arcs. Ils vérifient $\mathrm{OE}[x] > \mathrm{OE}[y]$ et $\mathrm{OD}[x] > \mathrm{OD}[y]$.

Sur notre dessin, les arcs traversiers vont toujours de gauche à droite.

3.5 Plus court chemin

Définition (3.1) — Distance

Soit $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *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 (3.2) — Diamètre, rayon, excentricité

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

Etant donné un sommet $s \in \mathrm{X}$, 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 qui conserve la distance à partir de s.

```
pour tout x ∈ X faire d[x] ← +∞
d[s] ← 0 ; exc ← 0
i ← 1 ; j ← 2
A_traiter[i] ← s
tant que i < j faire
    x ← A_traiter[i] ; i++
    pour tout y ∈ Γ+(x) faire
        si d[y] = +∞ alors
            A_traiter[j] ← y ; j++
            d[y] ← d[x] + 1 ; exc ← d[y]
```

Chapitre 4

Graphes orientés sans circuit

4.1 Reconnaissance d'un graphe orienté sans circuit

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

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

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

<u>*Preuve :*</u>

$\boxed{\Rightarrow}$ Supposons G sans circuit. D'après le lemme précédent, $\exists x \in \mathrm{X}$ tel que $d^-(x) = 0$ et tout sous-graphe d'un graphe sans circuit est sans circuit.

$\boxed{\Leftarrow}$ Considérons $x \in \mathrm{X}$ tel que $d^-(x) = 0$ (c'est possible par hypothèse). $\mathrm{G} \backslash \{x\}$ 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

<u>tant que</u> $\exists x \in \mathrm{X}$ <u>tel que</u> $d^-(x) = 0$ <u>faire</u> $\mathrm{G} \leftarrow \mathrm{G} \backslash \{x\}$
<u>si</u> $\mathrm{G} \neq \varnothing$ <u>alors</u> G sans circuit
<u>sinon</u> 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 si il existe une permutation* $\sigma = (x_1, ..., x_n)$ *des sommets de* G *tel que* $d^-_{G_i}(x_i) = 0$*, où* $G_i = G[\{x_i, ..., x_n\}]$ *(on supprime les* $i-1$ *premiers sommets).*

Preuve :

$\boxed{\Rightarrow}$ Supposons G sans circuit. D'après le lemme, $\exists x \in X$ tel que $d^-(x) = 0$. D'après le théorème précédent, $G \backslash \{x\}$ est sans circuit. Par hypothèse de récurrence, il existe une permutation $\sigma = (y_1, ..., y_{n-1})$ de $G \backslash \{x\}$ satisfaisant aux conditions du théorème. La permutation $(x, y_1, ..., y_{n-1})$ convient.

$\boxed{\Leftarrow}$ 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 donc que $G_1 = G$ est sans circuit.

Utilisons cette caractéristique pour obtenir un algorithme efficace.

Algorithme

```
1)    pour tout x ∈ X faire d(x) ← 0
2)    pour tout x ∈ X faire
          pour tout y ∈ Γ⁺(x) faire d(y)++
3)    A_traiter ← ∅
      compteur ← 0
4)    pour tout x ∈ X faire
          si d⁻(x) = 0 alors
              A_traiter ← A_traiter ∪ {x}
              compteur++
5)    tant que A_traiter ≠ ∅ faire
          x ← premier(A_traiter)
          A_traiter ← A_traiter\{x}
          pour tout y ∈ Γ⁺(x) faire
              d⁻(y)--
              si d⁻(y) = 0 alors
                  A_traiter ← A_traiter ∪ {y}
                  compteur++
6)    si compteur = n alors G sans circuit
      sinon G a un circuit
```

La première étape de l'algorithme coûte $O(n)$. La seconde est en $O(n + m)$ tandis que la suivante est constante ($O(1)$). La quatrième étape utilise un temps constant pour chaque sommet et est donc globalement en $O(n)$. La boucle principale explicitée à la $5^{\text{ème}}$ étape 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 (4.4) — Tri topologique

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

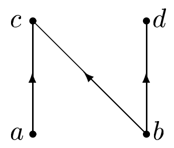
Théorème (4.5)

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

Preuve :

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 $S(G) = \{x \in X | d^-(x) = 0\}$ (sources de G).

$$\boxed{\Pi(G) = \cup_{s \in S(G)} \{s.\sigma | \sigma \in \Pi(G \backslash \{s\})\}}$$

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

Algorithme

Voici tout d'abord l'algorithme principal :

<u>pour tout</u> $x \in X$ <u>faire</u> calculer $d^-(x)$
$S \leftarrow \varnothing$
<u>pour tout</u> $x \in X$ <u>faire</u>
    <u>si</u> $d^-(x)$=0 <u>alors</u> $S \leftarrow \{x\}$
$\sigma \leftarrow \varnothing$
tri_topo(G, S, σ)

On explicite maintenant l'algorithme récursif *tri_topo* :

<u>si</u> S=∅ <u>alors</u> $ecrire(\sigma)$
<u>sinon</u> <u>pour tout</u> $s \in \mathrm{S}$ <u>faire</u>
  $\mathrm{S}' \leftarrow \mathrm{S} \backslash \{s\}$
  $\sigma \leftarrow \sigma.s$
  <u>pour tout</u> $y \in \Gamma^+(s)$ <u>faire</u>
    $d^-(y)$--
    <u>si</u> $d^-(y)$=0 <u>alors</u>
      $\mathrm{S}' \leftarrow \mathrm{S}' \cup \{y\}$
    $tri\_topo(\mathrm{G}, \mathrm{S}', \sigma)$
    <u>pour tout</u> $y \in \Gamma^+(s)$ <u>faire</u> $d^-(y)$++
    $\sigma \leftarrow \sigma \backslash \{s\}$

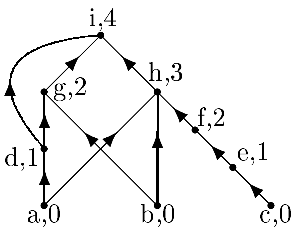
4.3 Décomposition en niveaux

Définition (4.6) — Hauteur

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

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

<u>*Exemple :*</u>



On a representé 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 ($\mathrm{O}(n + m)$), au lieu de gérer par sommet, il faut gérer par niveau.

Algorithme

$calcul\_degres\_internes$
$nbsom \leftarrow 0$
<u>pour tout</u> $x \in \mathrm{X}$ tel que $d^-(x) = 0$ <u>faire</u>
  $ajouter(x, nv[1])$
  $nbsom$++
$i \leftarrow 1$
<u>tant que</u> $nv[i] \neq \varnothing$ <u>faire</u>
  <u>pour tout</u> $x \in nv[i]$ <u>faire</u>
    <u>pour tout</u> $y \in \Gamma^+(x)$ <u>faire</u>
      $d^-(y)$--
      <u>si</u> $d^-(y) = 0$ <u>alors</u>
        $ajouter(y, nv[i + 1])$
        $nbsom$++
  i++
$circuit \leftarrow (n \neq nbsom)$

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

Propriété (4.7)

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

Preuve :

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

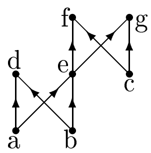
On considère un graphe $\mathrm{G} = (\mathrm{X}, \mathrm{U})$ orienté sans circuit et sans arcs de transitivité.

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

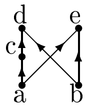
Définition (4.8) — Graphe gradué

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

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).



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

Propriété (4.10)

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

Preuve :

$\boxed{\Rightarrow}$ Supposons G gradué et appelons h la fonction de graduation. On raisonne par récurrence sur la longueur de la chaîne.

Soit $\mu = (x_0, ..., x_p)$. Si $p = 0$, la chaîne ne contient qu'un sommet. La propriété est donc forcément vérifiée.

Si $p \geqslant 1$, $lc(x_0, ..., x_p) = lc(x_0, ..., x_{p-1}) + lc(x_{p-1}, x_p) = h(x_{p-1}) - h(x_0) + lc(x_{p-1}, x_p)$ car l'hypothèse de récurrence s'applique à la sous-chaîne $(x_0, ..., x_{p-1})$.

Si $(x_{p-1}, x_p) \in \mathrm{U}$, on obtient $h(x_{p-1}) - h(x_0) + 1$ c'est à dire $h(x_p) - h(x_0)$.

Sinon, $(x_p, x_{p-1}) \in \mathrm{U}$. Alors on obtient $h(x_{p-1}) - h(x_0) + (0 - (-1))$, ce qui donne le même résultat.

$\boxed{\Leftarrow}$ On peut supposer G connexe. Soit $x_0 \in \mathrm{X}$. On pose $h(x_0) = 0$. Soit $x \in \mathrm{X}$, $x \neq x_0$. 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 $(x, y) \in \mathrm{U}$. 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 $h(y) = h(x) + 1$.

Etude 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 parcourera 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 recouvrant de poids minimum

Définition (5.1) — Poids

Etant 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 (5.2) — 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 (5.3)

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

<u>*Preuve :*</u>

$\boxed{\Rightarrow}$ Si $T = (X, F)$ est un arbre couvrant G, la connexité de T entraîne celle de G car $F \subset E$.

$\boxed{\Leftarrow}$ 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 $\mu = (x_1, ..., x_k)$ avec $x = x_1$ et $y = x_k$.

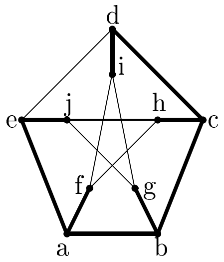
Soit i le plus petit indice tel que x_{i+1} ne soit pas dans C_x. L'arête $(x_i x_{i+1})$ n'est dans aucune composante connexe et $T \cup \{x_i x_{i+1}\}$ est sans cycle (par construction). Cela constitue une contradiction.

5.1 Caractérisation des arbres recouvrants de poids minimal

Définition (5.4) — Cycle et cocycle fondamentaux

Soient $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *un graphe et* $\mathrm{T} = (\mathrm{X}, \mathrm{F})$ *un arbre recouvrant* G.
Le **cycle fondamental** $\mu_{\mathrm{T}}(e)$ *associé à* $e \in \mathrm{E} \backslash \mathrm{F}$ *est l'unique cycle de* $\mathrm{T} \cup \{e\}$.
Le **cocycle fondamental** *associé à* $e \in \mathrm{F}$ *est* $\nu_{\mathrm{T}}(e) = \{xy \in \mathrm{E} | x \in \mathrm{T}_1 | y \in \mathrm{T}_2\}$ *où* T_1 *et* T_2 *sont les deux composantes connexes de* $\mathrm{T} \backslash \{e\}$.

Exemple :



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

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

L'arête ab est dans T et $\mathrm{T} \backslash \{ab\}$ possède deux composantes connexes : $\mathrm{T}_1 = \{a, f, e, j\}$ et $\mathrm{T}_2 = \{b, c, d, g, h, i\}$. Alors $\omega_{\mathrm{T}}(ab) = \{ab, fh, fi, jh, jg, ed\}$.

Théorème (5.5)

Soient $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *connexe,* $p : \mathrm{E} \to \mathbb{R}$ *et* $\mathrm{T} = (\mathrm{X}, \mathrm{F})$ *un arbre couvrant G.*
T *est de poids minimal si et seulement si :*

$$\forall e \in \mathrm{F}, \forall f \in \omega_{\mathrm{T}}(e), p(e) \leqslant p(f)$$

Preuve :

$\boxed{\Rightarrow}$ Supposons qu'il existe $e \in \mathrm{F}$ et $f \in \omega_{\mathrm{T}}(e)$ avec $p(f) < p(e)$. On considère $\mathrm{T}' = \mathrm{T} \backslash \{e\} \cup \{f\}$. On a alors $p(\mathrm{T}') < p(\mathrm{T})$.

Or T' est un arbre car il a $n - 1$ arêtes et il est connexe : $\mathrm{T} \backslash \{e\}$ a 2 composantes connexes et comme $f \in \omega_{\mathrm{T}}(e)$, f relie ces deux composantes.

$\boxed{\Leftarrow}$ Supposons que $\mathrm{T} = (\mathrm{X}, \mathrm{F})$ vérifie les conditions du théorème et considérons $\mathrm{T}^* = (\mathrm{X}, \mathrm{F}^*)$ un arbre recouvrant de poids minimum tel que $|\mathrm{F} \cap \mathrm{F}^*|$ soit maximal (c'est-à-dire, T^* possède un nombre maximal d'arêtes en commun avec T). Montrons que $\mathrm{T} = \mathrm{T}^*$.

Supposons le contraire et considérons $e \in \mathrm{F} \backslash \mathrm{F}^*$. $\mathrm{T}^* \cup \{e\}$ contient un cycle unique $\mu_{\mathrm{T}^*}(e)$.

$\mathrm{T} \backslash \{e\}$ a deux composantes connexes T_1 et T_2. e est une arête entre $x \in \mathrm{T}_1$ et $y \in \mathrm{T}_2$. Donc il existe une arête $f \neq e$ du cycle $\mu_{\mathrm{T}^*}(e)$ reliant $z \in \mathrm{T}_1$ à $t \in \mathrm{T}_2$. On a donc $f \in \omega_{\mathrm{T}}(e)$ et, par hypothèse, $p(e) \leqslant p(f)$ (car e est l'arête de poids minimum sur le cocycle).

On considère $\mathrm{T}^{*'} = \mathrm{T}^* \backslash \{f\} \cup \{e\}$. $\mathrm{T}^{*'}$ est un arbre car il a $n - 1$ arêtes et est sans cycle ($\mathrm{T}^* \cup \{e\}$ contient un unique cycle et ce cycle passe par f). Comme T^* est de poids minimal et que $p\left(\mathrm{T}^{*'}\right) \leqslant p(\mathrm{T}^*)$, $\mathrm{T}^{*'}$ est de poids minimal. Et comme $\left|\mathrm{F} \cap \mathrm{F}^{*'}\right| = |\mathrm{F} \cap \mathrm{F}^*| + 1$ (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 (5.6)

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 :*

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

Preuve :

$\boxed{\Rightarrow}$ Supposons qu'il existe $f \in E\backslash F$ et $e \in \mu_T(f)$ tel que $p(f) < p(e)$. On échange e et f. Soit $T' = T\backslash\{e\} \cup \{f\}$. 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.

$\boxed{\Leftarrow}$ Soit $T = (X, F)$ un arbre couvrant vérifiant les conditions du théorème. Montrons qu'il vérifie les conditions du **théorème précédent (5.5)**. 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

trier les arêtes en ordre de poids croissant
et les ranger dans une liste L.
$m \leftarrow 0$; $T \leftarrow \varnothing$; $poids \leftarrow 0$
tant que $m < n-1$ faire
    $e \leftarrow$ première arête de L
    $L \leftarrow L\backslash\{e\}$
    si $T \cup \{e\}$ ne contient pas de cycle alors
        $T \leftarrow T \cup \{e\}$
        $m{+}{+}$
        $poids \leftarrow poids + p(e)$

5.2.1 Correction

Théorème (5.7)

L'algorithme de Kruskal calcule correctement un arbre couvrant de poids minimal.

Preuve :

L'arbre construit vérifie les conditions du **théorème précédent (5.6)**.

Soit $f \in E\backslash F$. 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 $\mathrm{O}(m \log m) = \mathrm{O}(m \log n)$. L'initialisation se fait en temps constant.

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

La complexité totale est donc $\mathrm{O}(nm)$.

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

5.2.3 Comment tester si l'on crée un cycle ?

Définissons tout d'abord quelques variables que nous allons utiliser par la suite :

`CNX` : C'est un tableau de listes. CNX[i] contient la liste des sommets de la i-ème composante connexe du graphe considéré.

`CCX` : C'est un tableau d'entiers. CCX[x] $= i$ si et seulement si le sommet x est dans la i-ème composante connexe.

`taille` : C'est un tableau d'entiers. *taille*[i] contient le nombre de sommets de la i-ème composante connexe.

Soit une arête $e = xy$. Si les composantes connexes des sommets x et y sont différentes, c'est à dire $\mathrm{CCX}[x] \neq \mathrm{CCX}[y]$, alors $\mathrm{T} \cup \{e\}$ ne contient pas de cycle. On va utiliser cette propriété pour écrire un algorithme plus précis que précédemment :

Après l'initialisation, on ajoute ce qui suit :

pour tout $x \in \mathrm{X}$ faire
  $\mathrm{CCX}[x] \leftarrow x$
  $taille[x] \leftarrow 1$
  $\mathrm{CNX}[x] \leftarrow x$

La fin de l'algorithme devient quant à elle :

<u>tant que</u> $m < n - 1$ <u>faire</u>
  $e \leftarrow$ première arête de L
  $\mathrm{L} \leftarrow \mathrm{L} \setminus \{e\}$
  /* On note x et y les extrémités de e */
  <u>si</u> $\mathrm{CCX}[x] \neq \mathrm{CCX}[y]$ <u>alors</u>
    $\mathrm{T} \leftarrow \mathrm{T} \cup \{e\}$
    <u>si</u> $taille[\mathrm{CCX}[x]] \geqslant taille[\mathrm{CCX}[y]]$ <u>alors</u>
      $t_1 \leftarrow \mathrm{CCX}[x]$
      $t_2 \leftarrow \mathrm{CCX}[y]$
    <u>sinon</u>
      $t_1 \leftarrow \mathrm{CCX}[y]$
      $t_2 \leftarrow \mathrm{CCX}[x]$
    <u>pour tout</u> $z \in \mathrm{CNX}[t_2]$ <u>faire</u>
      $\mathrm{CNX}[t_1] \leftarrow \mathrm{CNX}[t_1] \cup \{z\}$
      $\mathrm{CCX}[z] \leftarrow t_1$
    $taille[t_1] \leftarrow taille[t_1] + taille[t_2]$
    $m{+}{+}$
    $poids \leftarrow poids + p(e)$

On peut remarquer que la complexité de la boucle <u>tant que</u> est proportionnelle au nombre de sommets z qui changent de composante connexe.

Un sommet qui change de composante connexe se retrouve dans une composante connexe de taille au moins double (puisque l'on a choisi de mettre les plus petits dans la plus grande). Donc un sommet change au plus $\log_2 n$ fois de composante connexe.

Sur l'ensemble des sommets, on a donc une complexité de $\mathrm{O}(n \log n)$. La complexité de l'agorithme de Kruskal est donc $\mathrm{O}(m \log n)$.

5.3 Algorithme de Prim

$\mathrm{T} \leftarrow \varnothing$
$\mathrm{A} \leftarrow x$
<u>tant que</u> $|\mathrm{T}| < n - 1$ <u>faire</u>
  $e \leftarrow xy$ <u>tels que</u> $x \in \mathrm{A}$, $y \notin \mathrm{A}$, xy de poids minimal
  $\mathrm{T} \leftarrow \mathrm{T} \cup \{e\}$
  $\mathrm{A} \leftarrow \mathrm{A} \cup \{y\}$

5.3.1 Correction

Théorème (5.8)

L'algorithme de Prim calcule correctement un arbre recouvrant de poids minimum.

<u>*Preuve :*</u>

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

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

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

$T' = T \backslash \{f\} \cup \{e\}$ 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 de A, il faut connaître l'arête de poids minimal qui le relie aux autres sommets.

```
pour tout x ∈ X faire
    poids_A(x) ← +∞
    ancetre(x) ←⊥
    etat(x) ← non_atteint
A_traiter ← choix(x)
etat(A_traiter) ← atteint
nbsommet ← 1
pour tout x ∈ Γ+(A_traiter) faire
    poids(A(x)) ← p(A_traiter, x)
    ancetre(x) ← A_traiter
tant que nbsommet < n faire
    /* Sélection de l'arête */
    poidsmin ← +∞
    pour tout x ∈ X faire
        si etat(x)non_atteint faire
            si poids(x) < poidsmin alors
                poidsmin ← poids_A(x)
                A_traiter ← x
    /* Traitement de l'arête */
    etat(A_traiter) ← atteint
    nbsommet++
    pour tout x ∈ Γ+(A_traiter) faire
        si etat(x)non_atteint alors
            si poids(A_traiter, x) < poids_A(x) alors
                poids_A(x) ← poids(A_traiter, x)
                ancetre(x) ← A_traiter
```

Chaque étape de sélection coûte $O(n)$. Le traitement coûte $O(d^+(A\_traiter))$.

Sur l'ensemble de l'algorithme, 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 $O(n^2)$.

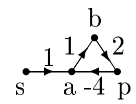
Chapitre 6

Chemin de coût minimum

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

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

Exemple :



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 les **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 :

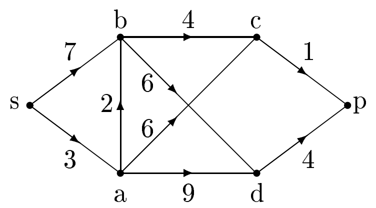
`Examine` : C'est l'ensemble des sommets pour lesquels on connaît la valeur du plus court chemin issu de s.

`A_traiter` : C'est l'ensemble des sommets qui ont au moins un prédecesseur dans E*xamine*.

Pour les sommets x dans A*_traiter*, on connaîtra la valeur d'un plus court chemin $\mu = (x_1, ..., x_k)$ avec $x_1 = s$, $x_k = x$ et $\{x_1, ...x_k\} \subseteq$ E*xamine*.

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

Exemple :



On part de s. Le sommet le plus proche est a. On attribue 3 à a et 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 couteux 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

<u>pour tout</u> $x \in$ X <u>faire</u>
  $c(x) \leftarrow +\infty$
  $aretes(x) \leftarrow 1$
A_*traiter* $\leftarrow \{s\}$
$c(s) \leftarrow 0$
E*xamine* $\leftarrow \varnothing$
<u>tant que</u> A_*traiter* $\neq \varnothing$ <u>faire</u>
  $x \leftarrow$élément de A_*traiter* minimisant $c[x]$
  E*xamine* $\leftarrow$ E*xamine* $\cup \{x\}$
  A_*traiter* $\leftarrow$ A_*traiter*$\setminus\{x\}$
  <u>pour tout</u> $y \in \Gamma^+(x)$ <u>faire</u>
    <u>si</u> $y \notin$ E*xamine* $\cup$ A_*traiter* <u>alors</u>
      A_*traiter* $\leftarrow$ A_*traiter* $\cup \{y\}$
    <u>si</u> $y \in$ A_*traiter* <u>alors</u>
      <u>si</u> $c(x) + cout(x,y) < c(y)$ <u>alors</u>
        $c(y) \leftarrow c(x) + cout(x,y)$
        $ancetre(y) \leftarrow x$

6.1.3 Correction

Théorème (6.1)

L'algorithme de Dijkstra calcule correctement les plus courts chemins.

Preuve :

On montre ce résultat par récurrence sur le nombre k de passages dans la boucle <u>tant que</u>.

On vérifie que :

1. $\forall x \in$ E*xamine*$, c(x) = \mu(s,x)$ (valeur du plus court chemin de s à x).

2. $\forall y \in$ A_*traiter*$, c(y) = \overline{\mu}(s,y) = \min\{\mu(s,x) + cout(x,y)\}$ avec $x \in$ E*xamine* et $(x,y) \in$ U.

$\boxed{k=1}$ E*xamine* $= \{s\}$
$c(s) = 0 = \mu(s,s)$
A_*traiter* $= \Gamma^+(s)$
$\forall x \in$ A_*traiter*$, c(x) = cout(s,x) = \overline{\mu}(s,x)$

$\boxed{k>1}$ Notons x le sommet sélectionné à la k-ième étape. Supposons que $c(x) > \mu(s,x)$. Il existe un chemin $\nu = (x_1, ..., x_n)$ 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 $x_{n-1} \notin$ E*xamine* 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 $y_i \notin$ E*xamine* à l'étape $k-1$. Donc, le sommet $y_{i-1} \in$ E*xamine* et donc $y_i \in$ A_*traiter*.

$c(y_i) = \min\{\mu(s,x) + cout(s,y_i) | z \in$ E*xamine*$, (s,y_i) \in$ 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 $c(y) = \min\{\mu(s,z) + cout(z,y) | z \in$ E*xamine*$\backslash\{x\}, (z,y) \in$ U$\}$.

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

6.1.4 Complexité

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

Pour trouver x de A_*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 A_*traiter* qui est coûteuse. Pour obtenir quelquechose 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 et mais le coût ne doit pas être quelconque.

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 prédecesseur y de x.

$$\mu(s,x) = \min\{\mu(s,y) + cout(y,x) | y \in \Gamma^-(x)\}$$

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 $\uparrow s = \{x \in \mathrm{X} | \exists$ un chemin de s 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. */
pour tout x ∈ X faire
    d⁻(x) ← 0
    Γ⁻(x) ← ∅
    etat(x) ← non_atteint
A_traiter ← {s}
etat(s) ← atteint
tant que A_traiter ≠ ∅ faire
    x ← premier(A_traiter)
    A_traiter ← A_traiter\{x}
    pour tout y ∈ Γ⁺(x) faire
        d⁻(y)++
        Γ⁻(y) ← Γ⁻(y) ∪ {x}
        si etat(y) = non_atteint alors
            etat(y) ← atteint
            A_traiter ← A_traiter ∪ {y}
/* calcul des plus courts chemins */
A_traiter ← ∅
c(s) ← 0
pour tout y ∈ Γ⁺(s) faire
    d⁻(y)--
    si d⁻(y) = 0 alors A_traiter ← A_traiter ∪ {y}
tant que A_traiter ≠ ∅ faire
    x ← premier(A_traiter)
    A_traiter ← A_traiter\{x}
    c(x) ← ∞
    pour tout y ∈ Γ⁻(x) faire
        si c(y) + cout(y, x) < c(x) alors
            c(x) ← c(y) + cout(y, x)
            ancetre(x) ← y
    pour tout y ∈ Γ⁺(x) faire
        d⁻(y)--
        si d⁻(y) = 0 alors A_traiter ← A_traiter ∪ {y}
```

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 $\mathrm{O}(n+m)$.

Dans la boucle tant que 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 $\mathrm{O}(n+m)$ pour cette partie.

Cet algorithme a une complexité de $\mathrm{O}(n+m)$ 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.

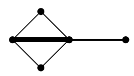
Définitions (7.1) — Graphe factorisable

Un graphe $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *non orienté est* $\boldsymbol{k}$**-factorisable** *si on peut partitionner* $\mathrm{E} = \{\mathrm{E}_1, ..., \mathrm{E}_p\}$ *de sorte que le graphe partiel* $\mathrm{G}_i = (\mathrm{X}, \mathrm{E}_i)$ *soit* $\boldsymbol{k}$**-régulier**, *c'est-à-dire* $\forall x \in \mathrm{X}, d_{\mathrm{G}_i}(x) = k$ *(les* G_i *sont les* $\boldsymbol{k}$*-facteurs de* G*).*

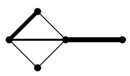
Définitions (7.2) — Couplage

Un **couplage** *de* G *est un ensemble d'arêtes* $\mathrm{C} \subset \mathrm{E}$ *tel que deux arêtes de* C *n'ont pas d'extrémités communes. Autrement dit, dans le graphe partiel* $\mathrm{G}' = (\mathrm{X}, \mathrm{C})$, *on a* $d_{\mathrm{G}'}(x) \leqslant 1, \forall x \in \mathrm{X}$.
Les sommets x *tel que* $d_{\mathrm{G}'}(x) = 1$ *sont dits* **C-saturés**. *Les sommets* x *tel que* $d_{\mathrm{G}'}(x) = 0$ *sont dits* **C-insaturés**.
Un couplage C *est* **maximal** *si* $\forall e \in \mathrm{E} \backslash \mathrm{C}$, $\mathrm{C} \cup \{e\}$ *n'est pas un couplage.*
Un couplage C *est* **maximum** *si* $\forall \mathrm{C}'$ *couplage,* $|\mathrm{C}'| \leqslant |\mathrm{C}|$ *(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 (7.3) — Chaîne améliorante

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

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

Théorème (7.4) — (Barge, 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.*

<u>*Preuve :*</u>

$\boxed{\Rightarrow}$ On utilise la contraposée.

Soit $\mu = (x_1, ..., x_{2k})$ 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 $y \in \{x_2, ..., x_{2k-1}\}$. Donc $d_{C\oplus\mu}(y) = 1$ pour tout $y \in \mu$.

Pour $y \notin \{x_1, ..., x_{2k}\}$, $d_C(y) = d_{C\oplus\mu}(y)$.

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.

$\boxed{\Leftarrow}$ 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 cycles de longueur paire 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 (7.5) — Graphe biparti

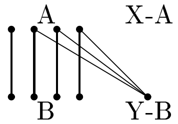
On appelle graphe **biparti** *un graphe* $G = (X, E)$ *tel que* X *peut-être partitionné en deux ensembles* X_1 *et* X_2 *de façon à ce que toute arête* e *de* E *relie un sommet de* X_1 *à un sommet de* X_2*. On note alors* $G = (X_1, X_2, E)$.

On va maintenant prouver que tout biparti $G = (X, Y, E)$ k-régulier admet un couplage parfait (1-facteur).

Pour cela, on va utiliser différents lemmes.

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

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



Définition (7.6) — Chaîne C-alternée

Une chaîne est **C-alternée** *si ses arêtes sont alternativement dans* C *et dans* $E \backslash C$.

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

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

Remarque :

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

Lemme (7.7)

Pour tout $i \in \mathrm{I}$ et $j \in \mathrm{J}$, $a_jb_i \in \mathrm{E} \Rightarrow i \in \mathrm{J}$.

Preuve :

Si $j \in \mathrm{J}$, il existe une chaîne C-alternée $\mu = (x, ..., a_j)$. D'autre part, $a_jb_i \in \mathrm{E}$ et $b_ia_i \in \mathrm{C} \subseteq \mathrm{E}$ d'où $\mu' = (x, ..., a_j, b_i, a_i)$ est une chaîne C-alternée, ce qui entraîne $a_i \in \mathrm{A}$ ou encore $i \in \mathrm{J}$.

Lemme (7.8)

$\exists a \in \mathrm{A}'$ tel que $\Gamma(a) \cap (\mathrm{Y} \backslash \mathrm{B}) \neq \varnothing$.

Preuve :

Si ce n'est pas le cas, pour tout $a \in \mathrm{A}'$, $\Gamma(a) \subseteq \mathrm{B}$ mais d'après le lemme précédent, on a $\Gamma(a) \subseteq \mathrm{B}'$.

Comptons les arêtes incidentes à B'. On prend $p = |\mathrm{B}'| = |\mathrm{A}'|$. 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, ce qui constitue une contradiction.

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

Théorème (7.9)

Tout graphe biparti k-régulier est 1-factorisable, c'est-à-dire admet un 1-facteur.

Preuve :

Ce corollaire se démontre trivialement par récurrence sur l'entier k.

Définition (7.10) — Graphe connexe eulérien

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

Théorème (7.11) — (Euler)

Un graphe connexe est eulérien si et seulement si le degré de chacun de ses sommets est pair.

Théorème (7.12) — (Peterson)

Un graphe est 2-factorisable si et seulement si il est $2k$-régulier.

Preuve :

$\boxed{\Rightarrow}$ 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 $2q$.

$\boxed{\Leftarrow}$ 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 $\mathrm{V} = (x_{i_1}, ..., x_{i_{kn}}, x_{i_1})$ un cycle eulérien. Quand k décrit $\{1...kn\}$, i_k décrit exactement k fois $\{1...n\}$.

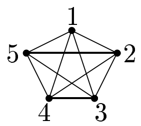
On définit le graphe biparti $\mathrm{H} = (\mathrm{A}, \mathrm{B}, \mathrm{F})$ par $\mathrm{A} = \{a_1, ..., a_n\}$, $\mathrm{B} = \{b_1, ..., b_n\}$ et $\mathrm{F} = \{a_p b_q | p = i_r, q = i_s, s = r + 1\}$.

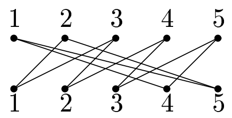
Ainsi, un sommet a_i est relié à k sommets b_j. Ces k sommets sont tous différents car chaque arête considérée ici correspond à une arête et une seule dans le graphe original.

Le graphe H est biparti k-régulier. D'après le théorème précédent, il admet un couplage parfait C. Le graphe $\mathrm{G}' = (\mathrm{X}, \mathrm{E}')$ défini par $\mathrm{E} = \{x_{i_r} x_{i_s} | i_r = p, i_s = q, a_p b_q \in \mathrm{C}\}$ est un 2-facteur du graphe G.

Puisque G admet un 2-facteur, par récurrence triviale, G est 2-factorisable.

Exemple :





$\mathrm{V} = (1, 2, 3, 4, 5, 1, 3, 5, 2, 4, 1)$

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.

Théorème (7.13) — (Tutte, 1947)

Un graphe $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *admet un couplage parfait (1-facteur) si et seulement si* $\forall \mathrm{S} \subseteq \mathrm{X}$, *le nombre de composantes connexes de* $\mathrm{G} \backslash \mathrm{S}$ *d'ordre impair est inférieur ou égal à* $|\mathrm{S}|$

Dans toute la suite, on notera $i(\mathrm{G})$ le nombre de composantes connexes d'ordre impair d'un graphe G.

Preuve :

$\boxed{\Rightarrow}$ Supposons que G admet un 1-facteur et considérons $\mathrm{S} \subseteq \mathrm{X}$.

On s'intéresse désormais au graphe $\mathrm{G} \backslash \mathrm{S}$. 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 $i(\mathrm{G} \backslash \mathrm{S}) \leqslant |\mathrm{S}|$.

$\boxed{\Leftarrow}$ Soit $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ un graphe vérifiant $\forall \mathrm{S} \subseteq \mathrm{X}$, $i(\mathrm{G} \backslash \mathrm{X}) \leqslant |\mathrm{S}|$.

En prenant $\mathrm{S} = \varnothing$, on déduit $|\mathrm{X}|$ pair. Si on ajoute une arête e au graphe G, il vérifie toujours l'hypothèse.

Suppsons que $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ 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 $\forall e \notin \mathrm{E}$, $\mathrm{G} + \{e\}$ admet un couplage parfait.

On pose $\mathrm{Y} = \{x \in \mathrm{X} | d(x) = n-1\}$ (sommets dominants). On va montrer que les composantes connexes de $\mathrm{G} \backslash \mathrm{Y}$ 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 $\exists a, b, c \in \mathrm{X} \backslash \mathrm{Y}$ tel que $ab, ac \in \mathrm{E}$ et $bc \notin \mathrm{E}$.

Comme $a \notin \mathrm{Y}$, il existe $d \in \mathrm{X} \backslash \mathrm{Y}$ tel que $ad \notin \mathrm{E}$. On considère les deux graphes $\mathrm{G}_1 = \mathrm{G} + \{bc\}$ et $\mathrm{G}_2 = \mathrm{G} + \{ad\}$. Ils admettent tous deux un couplage parfait (par maximalité de G) F_1 et F_2. On a $bc \in \mathrm{F}_1$ et $ad \in \mathrm{F}_2$.

Le graphe partiel $\mathrm{H} = (\mathrm{X}, \mathrm{F}_1 \oplus \mathrm{F}_2)$ vérifie $\forall x \in \mathrm{X}$, $d_{\mathrm{H}}(x) = 0$ ou 2. Soit e_1 l'arête de F_1 incidente à x, e_2 celle de F_2. Si $e_1 = e_2$, $d_{\mathrm{H}}(x) = 0$. Sinon, $d_{\mathrm{H}}(x) = 2$.

H est donc constitué de sommets isolés et de cycles de longueur paire alternant les arêtes de F_1 et de F_2. Remarquons que ad et bc appartiennent à $\mathrm{F}_1 \oplus \mathrm{F}_2$.

- 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 $\mathrm{G} \backslash \mathrm{Y}$ 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* G *contient au moins* $s - r$ *chaînes* C_1*-améliorantes.*

Preuve :

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 (8.2)

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* $2\left\lfloor \frac{r}{s-r} \right\rfloor + 1$.

Preuve :

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 (8.3)

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

Alors $|\mu'| \geqslant |\mu| + 2|\mu \cap \mu'|$.

Preuve :

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 :

$$|\mu'| \geqslant |\mu| + 2|\mu \cap \mu'|$$

Schéma de calcul :

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 (8.4)

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

Corollaire (8.5)

$\forall i < j$, si $|\mu_i| = |\mu_j|$ alors μ_i et μ_j n'ont pas de sommets de commun.

Preuve :

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 $C_k \oplus \mu_k \oplus ... \oplus \mu_{l-1}$ 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 (8.6)

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

Preuve :

On prend $r = \lfloor s - \sqrt{s} \rfloor$.

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

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 $2\left(\lfloor\sqrt{s}\rfloor + 1\right)$.

On utilise l'algorithme suivant :

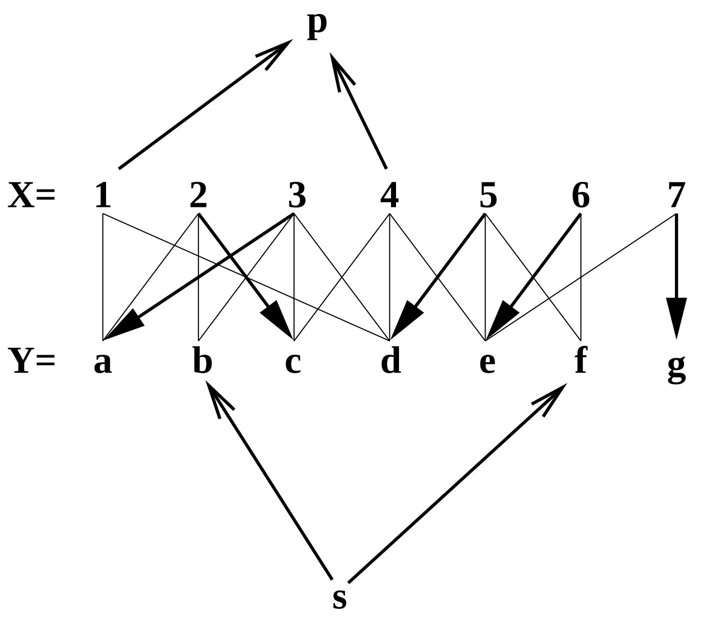
```
C ← ∅
tant que ∃ une chaîne C-améliorante faire
      trouver {μ1, ..., μk} un ensemble maximal de plus courtes
         chaînes C-améliorantes 2 à 2 disjointes
      C ← C ⊕ μ1 ⊕ ... ⊕ μk
```

8.2 Couplage des bipartis

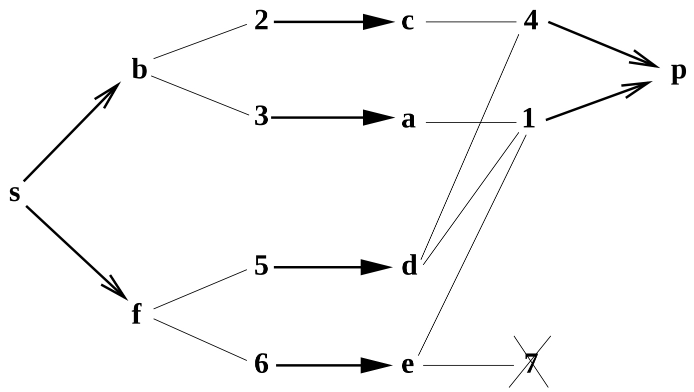
8.2.1 Graphe des plus courtes chaînes C-améliorantes

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.

On construit $\vec{G}$. Pour cela, on oriente les arêtes de C de X vers Y, celles de E\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

$\mathrm{L}_0 = \text{Insaturé}(\mathrm{X})$ $\qquad \vec{\mathrm{E}}_i = \{(z,t) \in \mathrm{E} \mid z \in \mathrm{L}_i, t \notin \mathrm{L}_0 \cup ... \cup \mathrm{L}_{i-1}\}$

$\mathrm{L}_{i+1} = \{t \in \mathrm{X} \cup \mathrm{Y} \mid (z,t) \in \vec{\mathrm{E}}_i\}$ $\qquad i^* = \min\{i \mid \mathrm{L}_i \cap \text{Insaturé}(\mathrm{Y}) \neq \varnothing\}$

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

$\vec{\mathrm{X}} = \{s,p\} \cup \mathrm{L}_0 \cup ... \cup \mathrm{L}_{i^*-1} \cup (\mathrm{L}_{i^*} \cap \text{Insaturé})$

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

On explicite maintenant la procédure `construction_`$\vec{\mathrm{G}}$.

améliorante$\leftarrow$ *faux*
<u>pour tout</u> $x \in \mathrm{X} \cup \mathrm{Y}$ <u>faire</u> utilisé$(x) \leftarrow$ *faux*
<u>pôur tout</u> $y \in$ Insaturé(Y) <u>faire</u> $\Gamma_{\vec{\mathrm{G}}}(y) \leftarrow \{p\}$
$\Gamma_{\vec{\mathrm{G}}}(s) \leftarrow \varnothing$
nvsommets $\leftarrow \varnothing$
<u>pour tout</u> $x \in$ Insaturé(X) <u>faire</u>
    $\Gamma_{\vec{\mathrm{G}}}(s) \leftarrow \Gamma_{\vec{\mathrm{G}}}(s) \cup \{x\}$
    nvsommets $\leftarrow$ *nvsommets* $\cup \{x\}$
    utilisé$(x) \leftarrow$ *vrai*
couplage $\leftarrow$ *faux*
<u>répéter</u>
    nvsomm $\leftarrow \varnothing$
    <u>si</u> couplage <u>alors</u> <u>pour tout</u> $x \in$ *nvsommets* <u>faire</u>
        $\Gamma_{\vec{\mathrm{G}}}(x) \leftarrow \{couple[x]\}$
        nvsomm $\leftarrow$ *nvsomm* $\cup \{couple[x]\}$
    <u>sinon</u> <u>pour tout</u> $x \in$ *nvsommets* <u>faire</u>
        $\Gamma_{\vec{\mathrm{G}}}(x) \leftarrow \varnothing$
        <u>pour tout</u> $y \in \Gamma_{\vec{\mathrm{G}}}(x)$ <u>faire</u>
            <u>si</u> *non*(utilisé(y)) <u>et</u> $y \neq couple[x]$ <u>alors</u>
                $\Gamma_{\vec{\mathrm{G}}}(x) \leftarrow \Gamma_{\vec{\mathrm{G}}}(x) \cup \{y\}$
                nvsomm $\leftarrow$ *nvsomm* $\cup \{y\}$
            <u>si</u> $y \in$ Insaturé(Y) <u>alors</u> améliorante$\leftarrow$ *vrai*
couplage $\leftarrow$ *non*(*couplage*)
<u>pour tout</u> $x \in$ *nvsomm* <u>faire</u> utilisé$(x) \leftarrow$ *vrai*
nvsommets $\leftarrow$ *nvsomm*
<u>jusqu'à</u> (*nvsommets* $= \varnothing$) <u>ou</u> améliorante

Complexité en $\mathrm{O}(n+m)$.

Procédure `couplage_maximum`.

```
pour tout x ∈ X ∪ Y faire couple[x] ← 0
Insaturé(X) ← X
Insaturé(Y) ← Y
Répéter
    Construction_G⃗
    si améliorante alors
        pile ← ∅
        pour tout x ∈ X ∪ Y faire interdit(x) ← faux
        empiler(s)
        tant que pile ≠ ∅ faire
            tant que Γ_G⃗(haut_pile) ≠ ∅ faire
                x ← pop(Γ_G⃗(haut_pile))
                si non(interdit(x)) alors
                    si x ≠ p alors
                        interdit(x) ← vrai
                        empiler(x)
                    sinon
                        Insaturé(Y) ← Insaturé(Y)\{haut_pile}
                        tant que haut_pile ≠ s faire
                            y ← haut_pile
                            dépiler
                            x ← haut_pile
                            dépiler
                            couple[x] ← y
                            couple[y] ← x
                        Insaturé(X) ← Insaturé(X)\{x}
            dépiler
jusqu'à non(améliorante)
```

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 $m = O(n^2)$).

Chapitre 9

Connexité

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

Définition (9.1) — 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* $(a, ..., b)$ *avec* $a \in A$ *et* $b \in B$.
Un **AB-séparateur** *est un ensemble de sommets* $S \subseteq X$ *tel que* $G\backslash S$ *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 (9.2)

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 (9.3) — (Menger, 1927)

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

Preuve :

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

$\boxed{p \leqslant q}$ 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 $p \leqslant |S| = q$.

$\boxed{p \geqslant q}$ 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 $G\backslash\{e\}$ 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, $p(A, B, G\backslash\{e\}) = q(A, B, G\backslash\{e\}) \leqslant p(A, B, G) < q(A, B, G)$ 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/e. $|I| = q(A, B, G/e) < q$. Soit J' défini de même pour G/e. $|J'| < q$.

On a $z \in J'$. Supposons le contraire et considérons $\mu = (a, ..., b)$ 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 $J = J'\backslash\{z\} \cup \{x, y\}$. J est un AB-séparateur de G. En effet, soit $\mu = (a, ..., b)$ 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 $I \cup J\backslash\{t\}\}$ et $H_B = \{t \in I \cup J | \exists$ une Bt-chaîne qui n'intersecte pas $I \cup J\backslash\{t\}\}$

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 $\mu = (a, ..., b)$ 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 $\mu = (a, ..., u)$ et une Bu-chaîne $\mu' = (b, ..., u)$ qui ne coupent pas $I \cup J\backslash\{u\}$.

Appelons v le sommet de μ le plus proche de a qui soit aussi dans μ'. On appelle λ la sous-chaîne de μ allant de a à v et λ' celle de μ' allant de b à v. Par construction, $\lambda\overline{\lambda}'$ 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 $\lambda\overline{\lambda}' = \mu\overline{\mu}'$ ne contient pas l'arête e sinon elle intersecterait J en x, y et u, soit au moins deux sommets. Puisque $\mu\overline{\mu}'$ est une AB-chaîne de $G\backslash\{e\}$, elle coupe I, ce ne peut-être qu'en u d'où $u \in I$.

cqfd

Définition (9.4) — 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* $G[X\backslash Y]$. *On dit aussi que* Y *est un* xy-**séparateur**.

Remarque :

Il faut faire attention au fait que xy-séparateur ne signifie pas la même chose que $\{x\}\{y\}$-séparateur.

Un xy-séparateur ne peut contenir ni x ni y.

$q(\{x\},\{y\},\mathrm{G}) = 1$ si x et y sont dans une même composante connexe, $= 0$ sinon.

$k(x,y,\mathrm{G})$ est le cardinal minimal d'un xy-séparateur si $xy \notin \mathrm{E}$. Si $e = xy \in \mathrm{E}$, $k(x,y,\mathrm{G}) = k(x,y,\mathrm{G}\backslash\{e\}) + 1$.

On note $p(x,y,\mathrm{G})$ le nombre maximal de xy-chaînes 2 à 2 disjointes sauf en x et y.

Théorème (9.5) — (Menger)

Dans un graphe $\mathrm{G} = (\mathrm{X},\mathrm{E})$, *on a* $p(x,y,\mathrm{G}) = k(x,y,\mathrm{G})$.

Preuve :

Supposons que $xy \notin \mathrm{E}$.

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

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

Considérons un ensemble de AB-chaînes 2 à 2 disjointes de cardinal $p(\mathrm{A},\mathrm{B},\mathrm{G})$. 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, $\mu = (a,...,x,a',...,b)$. Alors $(x,a',...,b,y)$ est une xy-chaîne. Les autres cas sont semblables. On obtient autant de xy-chaînes 2 à 2 disjointes d'où $p(\mathrm{A},\mathrm{B},\mathrm{G}) \leqslant p(x,y,\mathrm{G})$.

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

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

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

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

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

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

cqfd

Définition (9.6) — 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 (9.7)

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.*

Preuve :

Cette preuve est due à Menger.

$$\begin{aligned} G \text{ est } h\text{-connexe} &\iff k(G) \geqslant h \\ &\iff \forall x, y \in X, k(x, y, G) \geqslant h \\ &\iff \forall x, y \in X, p(x, y, G) \geqslant h \\ &\iff \forall x, y \in X, \text{il y a au moins } h \ xy\text{-chaînes 2 à 2 disjointes.} \end{aligned}$$

Définition (9.8) — Etoile

Soient $G = (X, E)$ *un graphe,* $Y \subseteq X$ *et* $x \in X \backslash Y$.
Une xY-**étoile** *est un ensemble de chaînes 2 à 2 disjointes reliant* x *à tous les sommets de* y*, sauf en* x.

Théorème (9.9) — (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* $|Y| = h$, $\forall x \in X \backslash Y$*, il existe une* xY*-étoile.*

Preuve :

$\boxed{\Rightarrow}$ 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 $Y = \{y_1, ..., y_h\}$. Soit $x \in X \backslash Y$. 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 $y \in Y \backslash S$. Dans $G \backslash S$, t et y sont toujours connectés et donc dans $G' \backslash S$, 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.

$\boxed{\Leftarrow}$ Si $\forall Y \subseteq X$, $|Y| = h$, $\forall x \in X \backslash Y$, 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 $Y \subseteq (\{y\} \cup \Gamma(y)) \backslash \{x\}$ et $|Y| = h$. Même conclusion.

Chapitre 10

Flot maximum dans un réseau

Définition (10.1) — Réseau

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

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

Définition (10.2) — Flot

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

i) **pseudo-symétrie** *:* $\forall x, y \in \mathrm{X}, \quad \phi(x, y) = -\phi(y, x)$ *;*

ii) **Contraintes de capacité** *:* $\forall u \in \mathrm{U}, \quad \phi(u) \leqslant c(u)$ *;*

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

(1ère loi de Kirschhoff)

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

Remarque :

A la seconde propriété s'ajoute le fait que $\phi(x, y) = 0$ si $(x, y) \notin \mathrm{U}$ et $(y, x) \notin \mathrm{U}$.

Concrètement, on dispose d'un graphe orienté dont les arêtes sont etiquetées 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 $|\phi| = \phi(s, \mathrm{X})$. On va s'intéresser à la recherche d'un flot ϕ maximisant $|\phi|$.

10.1 Méthode de Ford-Fulkerson

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

10.1.1 Graphe des écarts

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

Définition (10.3) — Réseau des écarts

Le réseau des écarts est le graphe $\mathrm{R}_\phi = (\mathrm{X}, \mathrm{U}_\phi, c_\phi)$ *où* $\mathrm{U}_\phi = \{(x,y) \in \mathrm{X} \times \mathrm{X}, c_\phi(xy) > 0\}$.

Concrètement, on remplace chaque arc u par un arc étiqueté par $c_\phi(u)$ (si c'est non nul).

Lemme (10.4)

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

<u>*Preuve :*</u>

C'est globalement trivial.

10.1.2 Chemins améliorants

Défintion (10.5) — Chemin améliorant

Etant donnés un réseau $\mathrm{R} = (\mathrm{X}, \mathrm{U}, c)$ *et un flot* ϕ, *un* **chemin améliorant** μ *relativement à* ϕ *est un chemin de* s *à* p *dans* R_ϕ.

Défintion (10.6) — Capacité résiduelle

La **capacité résiduelle** *de* μ *est* $c_\phi(\mu) = \min\{c_\phi(u) | u \in \mu\}$

Lemme (10.7)

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

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

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

Corollaire (10.8)

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

10.1.3 Coupes dans les réseaux

Définition (10.9) — Coupe

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

Lemme (10.10)

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

<u>*Preuve :*</u>

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

$$\begin{aligned}
\phi(\mathrm{Y},\overline{\mathrm{Y}}) &= \phi(\mathrm{Y},\mathrm{X}) - \phi(\mathrm{Y},\mathrm{Y}) \\
&= \phi(\mathrm{Y},\mathrm{X}) \\
&= \phi(s,\mathrm{X}) + \phi(\mathrm{Y}\backslash\{s\},\mathrm{X}) \\
&= \phi(s,\mathrm{X}) \\
\phi(\mathrm{Y},\overline{\mathrm{Y}}) &= |\phi|
\end{aligned}$$

Corollaire (10.11)

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

Théorème (10.12) — Ford & Fulkerson

Soit ϕ un flot un un réseau $\mathrm{R} = (\mathrm{X},\mathrm{U},c)$. 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 $(\mathrm{Y},\overline{\mathrm{Y}})$ telle que $|\phi| = c(\mathrm{Y},\overline{\mathrm{Y}})$.

<u>*Preuve :*</u>

$\boxed{\text{i)}\Rightarrow\text{ii)}}$ Démontré précédemment.

$\boxed{\text{iii)}\Rightarrow\text{i)}}$ Idem.

$\boxed{\text{ii)}\Rightarrow\text{iii)}}$ Posons $\mathrm{Y} = \{x \in \mathrm{X} | \exists$ un chemin de s à x dans $\mathrm{R}_\phi\}$. ii) implique alors $p \in \overline{\mathrm{Y}}$. Montrons que $c(\mathrm{Y},\overline{\mathrm{Y}}) = |\phi|$. Soit $(y,z) \in \mathrm{Y}\times\overline{\mathrm{Y}}$. Si $\phi(y,z) < c(y,z)$ alors $yz \in \mathrm{U}_\phi$. Donc $y \in \mathrm{Y}$. Contradiction.

<u>*Remarque :*</u>

Nous n'avons pas démontré l'existence d'un flot maximum. On sait qu'il existe si les capacités sont entières ou rationnelles. On ne sait rien si les capacités sont réelles (*ie* l'algorithme peut ne pas finir et même converger vers un flux différent du flux maximum). On verra plus loin l'existence dans le cas réel.

10.2 Algorithme d'Edmonds Karp

<u>pour tout</u> arc $(x,y) \in \mathrm{U}$ <u>faire</u> $\phi(x,y) \leftarrow c(x,y)$ et $\phi(y,x) \leftarrow 0$
<u>tant que</u> $\exists$ chemin μ de s à p dans R_ϕ <u>faire</u>
  $c_\phi(\mu) \leftarrow \min\{c_\phi(x,y) | (x,y) \in \mu\}$
  <u>pour tout</u> $(x,y) \in \mathrm{U}$
    $\phi(x,y) += c_\phi(\mu)$
    $\phi(y,x) \leftarrow -\phi(x,y)$

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

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

Lemme (10.13)

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

<u>*Preuve :*</u>

Supposons le contraire. Soit x le sommet vérifiant $d_\phi(s,x) > d_{\phi'}(s,x)$ et minimisant $d_{\phi'}(s,x)$. Soit μ' 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 $d_{\phi'}(s,x) = d_{\phi'}(s,y) + 1 \geqslant d_\phi(s,y) + 1$. Il y a deux cas possibles :

Si $\phi(x,y) < c(y,x)$. Alors $(y,x) \in \mathrm{U}_\phi$ et donc $d_\phi(s,x) \leqslant d_\phi(s,y) + 1$. Impossible.

Sinon, $\phi(x,y) = c(y,x)$. Alors $(y,x) \notin \mathrm{U}_\phi$ et $(y,x) \in \mathrm{U}_{\phi'}$. Donc $xy \in \mu$. Alors $d_\phi(s,x) = d_\phi(s,y) - 1 \leqslant d_{\phi'}(s,y) - 1 = d_{\phi'}(s,x) - 2$. Impossible.

Théorème (10.14)

La boucle <u>*tant que*</u> *dans l'algorithme d'Edmonds-Karp est exécutée* $\mathrm{O}(nm)$ *fois.*

<u>*Preuve :*</u>

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

Corollaire (10.15)

La complexité de l'algorithme est en $\mathrm{O}(nm^2)$.

On raisonne en fait sur un graphe non-orienté et on ne parcourt que les arcs de capacité résiduelle strictement positive.

10.3 Méthode des préflots — Algorithme de Goldberg

Définition (10.16) — Préflot

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

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

Le flot en **excès** *en* x *est défini par* $e(x) = \phi(\mathrm{X}, x)$. *Un sommet* x *sera* **excendentaire** *si* $e(x) > 0$.

10.3.1 Les opérations de base

Définition (10.17) — Fonction de hauteur

Soir $\mathrm{R} = (\mathrm{X}, \mathrm{U}, c)$ *un réseau de source* s*, de puits* p*. Soit* ϕ *un préflot.*
Une fonction $h : \mathrm{X} \to \mathbb{N}$ *est une* **fonction de hauteur** *si :*

- $h(s) = |\mathrm{X}|$
- $h(p) = 0$
- $h(x) \leqslant h(y) + 1 \quad \forall (x, y) \in \mathrm{U}_\phi$ *(graphes de écarts).*

Lemme (10.18)

$$h(x) > h(y) + 1 \quad \Longrightarrow \quad (x, y) \notin \mathrm{U}_\phi$$

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)
- $c_\phi(x, y) > 0$ (pas saturé par le flot)
- $h(x) = h(y) + 1$ (on pousse de haut en bas)

Procédure `Pousser`(x, y)

1) $d_\phi \leftarrow \min(e(x), c_\phi(x, y))$
2) $\phi(x, y) \leftarrow \phi(x, y) + d_\phi$
3) $\phi(y, x) \leftarrow -\phi(x, y)$
4) $e(x) \leftarrow e(x) - d_\phi$
5) $e(y) \leftarrow e(y) + d_\phi$

1) correspond à la conservation des contraintes. 3),4) et 5) correspondent à la conservation de l'excendentarité. 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 $c_\phi(x, y) = 0$.

L'opération Elever :

L'opération **Elever**(x) s'applique si :

- $e(x) > 0$ (x est excédentaire)
- $\forall (x, y) \in \mathrm{U}_\phi, h(x) \leqslant h(y)$

Procédure `Elever`(x)

$$h(x) \leftarrow 1 + \min\{h(y) | (x, y) \in \mathrm{U}_\phi\}$$

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 $\phi(\mathrm{X}, x) > 0$ implique $\exists y \in \mathrm{X}$ tel que $\phi(y, x) > 0$ d'où $\phi(x, y) < 0$. Et $c_\phi(x, y) = c(x, y) - \phi(x, y) > 0$ implique $(x, y) \in \mathrm{U}$.

10.3.3 L'algorithme générique

Procédure `init_préflot`(R, s, p)

    pour tout $x \in \mathrm{X}$ faire $h(x) \leftarrow 0$ et $e(x) \leftarrow 0$
    pour tout $(x, y) \in \mathrm{U}$ faire $\phi(x, y) \leftarrow 0$ et $\phi(y, x) \leftarrow 0$
    $h(s) \leftarrow |\mathrm{X}|$
    pour tout $y \in \Gamma^+(s)$ faire
        $\phi(s, y) \leftarrow c(s, y)$
        $\phi(y, s) \leftarrow -\phi(s, y)$
        $e(y) \leftarrow c(s, y)$

Ainsi, on a saturé tous les arcs issus de la source

Procédure `préflot`

    tant que $\exists$ une opération de poussée ou d'élévation applicable
    faire sélectionner une opération valide et l'appliquer

Lemme (10.19)

Soit $\mathrm{R} = (\mathrm{X}, \mathrm{U}, c)$ *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.

Preuve :

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.

10.3.4 Validité de la méthode du préflot

Lemme (10.20)

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

Lemme (10.21)

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

Preuve :

On raisonne par récurrence sur le nombre de passages dans la boucle tant que.

Après l'initialisation, c'est vrai.

Supposons le résultat vrai à une certaine étape et montrons qu'il est vrai à l'étape suivante.

Supposons qu'on fait une opération `Elever`(x). Si $(x, y) \in \mathrm{U}_\phi$, $h(x) \leqslant h(y)+1$ après élévation. Si $(y, x) \in \mathrm{U}_\phi$, 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.

Supposons qu'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 (10.22)

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.*

Preuve :

Supposons le contraire et considérons $\mu = (x_0, ..., x_k)$ 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 (10.23)

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

Preuve :

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 (10.24)

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,* $\exists$ *un chemin de* x *à* s *dans* G_ϕ *(le graphe des écarts).*

Preuve :

Posons $U = \{y \in X | \exists$ chemin de x à y dans $R_\phi\}$ 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 $\phi(z, y) \leqslant 0$. En effet, si $\phi(z, y) > 0$, $\phi(y, z) < 0$ et $c_\phi(y, z) = c(y, z) - \phi(y, z) > 0$ donc $(y, z) \in U_\phi$ et comme $y \in U$ on aurait $z \in U$. Contradiction.

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

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

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

Lemme (10.25)

Soit un réseau $\mathrm{R} = (\mathrm{X}, \mathrm{U}, c)$*,* s *une source,* p *un puits,* ϕ *un préflot et* h *une hauteur.*
On a toujours $h(x) \leqslant 2|\mathrm{X}| - 1$*.*

Preuve :

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 $\mu = (x_0, ..., x_k)$ élémentaire dans G_ϕ avec $x_0 = x$ et $x_k = s$. On a donc $k < |\mathrm{X}|$ et $h(x_0) = h(x) \leqslant h(s) + k \leqslant 2|\mathrm{X}| - 1$ car $h(s) = |\mathrm{X}|$.

Corollaire (10.26)

Il y a au plus $2|\mathrm{X}|^2$ *opérations d'élévation.*

Preuve :

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

Lemme (10.27)

Il y a au plus $2|\mathrm{X}|.|\mathrm{U}|$ *poussées saturantes.*

Preuve :

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

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

Si de telles poussées existent, c'est que $(x, y) \in \mathrm{U}$ ou $(y, x) \in \mathrm{U}$. Supposons que l'on ait effectué une poussée saturante sur (x, y), l'arc (x, y) disparaît de G_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 A d'entiers $\{h(x)+h(y)\}$ entre deux poussées saturantes sur (x, y) ou (y, x). A la poussée $h(x) + h(y) \geqslant 1$.

A la dernière poussée, $h(x)+h(y) \leqslant (2|\mathrm{X}|-1)+(2|\mathrm{X}|-2)$ donc $h(x)+h(y) \leqslant 4|\mathrm{X}| - 3$.

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

Lemme (10.28)

Durant l'algorithme générique, il y a au plus $4|\mathrm{X}|^2(|\mathrm{X}| + |\mathrm{U}|)$ *poussées saturantes.*

Preuve :

On définit la fonction potentiel : $\Theta = \sum_{y \in \mathrm{S}} h(y)$ où S est un ensemble de sommets excédentaires.

Au départ $\Theta = 0$. Une opération d'élévation fait augmenter Θ d'au plus $2|X|$ (car ne modifie pas G_ϕ donc il n'y a qu'une hauteur qui est modifiée et $2|X|$ 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 au plus un sommet supplémentaire dans S).

Une opération de poussée non saturante sur (x, y) fait baisser Θ 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 excédentaire).

Donc Θ a baissé de $h(y) - h(x) = 1$ (car comme on a pu effectuer une poussée, $h(y) = h(x) + 1$).

La valeur maximum que peut atteindre Θ est majorée par :

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

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

Théorème (10.29)

L'algorithme générique réalise au plus $O(mn^2)$ *opérations de base.*

Remarque :

On a supposé $n = O(m)$.

Corollaire (10.30)

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

Preuve :

On peut réaliser une poussée en $O(1)$ et une élévation en $O(n)$.

Chapitre 11

Coloration de graphes

11.1 Nombre chromatique

Défintion (11.1) — Nombre chromatique

Le **nombre chromatique** $\chi(\mathrm{G})$ *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 $\Delta(\mathrm{G})$ le degré maximum d'un sommet de G.

Propriété (11.2)

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

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

Preuve :

On démontre tout d'abord la première inégalité :

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

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

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

Démontrons maintenaint la seconde inégalité :

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

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

Corollaire (11.3)

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

Preuve :

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

Théorème (11.4)

Soit un graphe $\mathrm{G} = (\mathrm{X}, \mathrm{E})$.

$$\chi(\mathrm{G}) = 1 \Leftrightarrow \mathrm{G} \textit{ est stable (n'a pas d'arêtes)}$$

$$\chi(\mathrm{G}) = 2 \Leftrightarrow \mathrm{G} \textit{ est biparti}$$

Définitions (11.5) — Point d'articulation, isthme, bloc

Soit $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *un graphe connexe.*
Un sommet $x \in \mathrm{X}$ *est un* **point d'articulation** *si* $\mathrm{G} \backslash \{x\}$ *n'est pas connexe.*
Une arête $e \in \mathrm{E}$ *est un* **isthme** *si* $\mathrm{G} \backslash \{e\}$ *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** $\mathrm{G_B} = (\mathrm{X_B}, \mathrm{E_B})$ *est défini par :*

$$\mathrm{X_B} = \{\mathrm{B}_i | \mathrm{B}_i \textit{ bloc}\} \cup \{a_j | a_j \textit{ point d'articulation}\}$$

$$\mathrm{E_B} = \{a_j \mathrm{B}_i | a_j \in \mathrm{B}_i\}$$

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 (11.6)

Soit $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *un graphe connexe.*
L'intersection des ensembles des sommets de deux blocs distincts, est vide, ou réduit à un point d'articulation.
$\mathrm{G_B}$ *est un arbre.*

Définition — Rappel (11.7) — *k*-connexité

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

Théorème (11.8) — (Brooks, 1941)

Soit $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *un graphe tel que* $\Delta(\mathrm{G}) \geqslant 3$ *et* G *ne contient pas de clique à* $\Delta(\mathrm{G}) + 1$ *sommets.*
Alors $\chi(\mathrm{G}) \leqslant \Delta(\mathrm{G})$.

Remarque :

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

On a $\omega(\{x_1, ...x_{i-1}\}) \neq \varnothing$ sinon G ne serait pas connexe.

Preuve :

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

Cas particulier

Supposons qu'il existe dans G deux sommets a et b tel que $d(a,b) = 2$ et supposons $\mathrm{G}\backslash\{a,b\}$ connexe. Alors $\chi(\mathrm{G}) \leqslant \Delta(\mathrm{G})$. En effet : notons x_1 le voisin commun de a et b. D'après la remarque précédente, on peut numéroter les sommets de $\mathrm{X}\backslash\{a,b\}$ sous la forme $\{x_1, ..., x_{n-2}\}$ de sorte que x_i a un voisin dans $\{x_1, ..., x_{i-1}\}$.

Montrons que l'on peut colorier le graphe avec $\Delta(\mathrm{G})$ 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 $\{x_1, ..., x_{i-1}\}$, la partie non encore coloriée. Comme $d(x_i) \leqslant \Delta(\mathrm{G})$, il reste une couleur disponible dans $\{1, ..., \Delta(\mathrm{G})\}$. 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 $\{1, ..., \Delta(\mathrm{G})\}$.

Cas général

Intéressons nous maintenant au cas général : 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, $\mathrm{G}\backslash\{a,b\}$ est connexe et on peut appliquer le résultat du cas particulier.

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.

$\star$ 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. $\mathrm{G}\backslash\{a,b\}$ est connexe puisque l'on conserve le sommet universel x. On peut donc encore appliquer le résultat du cas particulier.

$\star$ Si G ne possède pas de sommet universel : choisissons un sommet $x \in \mathrm{X}$ tel que $3 \leqslant d(x) < n-1$. C'est possible car $\Delta(\mathrm{G}) \geqslant 3$ donc G admet forcément un sommet de degré au moins 3. Le graphe $\mathrm{G}\backslash\{x\}$ est soit 2-connexe, soit 1-connexe.

$\square$ Si $\mathrm{G}\backslash\{x\}$ est 2-connexe : $\{x\} \cup \Gamma(x) \neq \mathrm{X}$ car $d(x) < n-1$. G étant connexe, $\omega(\{x\} \cup \Gamma(x)) \neq \varnothing$. Soit y l'extrémité d'une arête de 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, $\mathrm{G}\backslash\{x,y\}$ est connexe car $\mathrm{G}\backslash\{x\}$ est 2-connexe. On peut donc appliquer le cas vu plus haut.

$\square$ Si $\mathrm{G}\backslash\{x\}$ est 1-connexe : $\mathrm{G}\backslash\{x\}$ admet en fait des points d'articulation. On considère $(\mathrm{G}\backslash\{x\})_\mathrm{B}$ l'arbre des blocs, il a au moins deux blocs pendants B_1 et

B_2 (correspondant aux feuilles) puisque c'est un arbre. Ces deux blocs sont reliés au reste de $G \backslash \{x\}$ par des points d'articulation b_1 et b_2. En fait, x a un voisin a dans B_1 car sinon, en enlevant b_1 à G, on déconnecterait le graphe initial (absurde car G n'est pas 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), on contredirait le fait que a et b sont dans 2 blocs différents. a et b sont non reliés et ont x comme voisin commun donc sont à distance 2 l'un de l'autre. Le graphe $G \backslash \{a, b, x\}$ est connexe car a et b ne sont pas les points d'articulation de B_1 et B_2 (a et b sont soit des extrémités d'isthmes, soit dans des composantes connexes, les enlever laisse donc la connexité).

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

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

⋆ Si B contient uen clique à $\Delta(B)+1$ sommets. Cela signifie que B est une clique à $\Delta(B) + 1$ sommets. Et $\Delta(B) + 1 \leqslant \Delta(G)$ 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 ne contient pas de clique à $\Delta(B) + 1$ sommets, on peut toujours colorier B en $\Delta(B) \leqslant \Delta(G)$ couleurs.

On a montré aue 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 $(G \backslash B) \cup \{b\}$ 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.

cqfd

Théorème (11.9) — (Tutte, 1954)

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

Preuve :

Pour $k = 2$

C'est trivial car un graphe 2-chromatique est biparti et ne peut donc pas contenir de triangle.

Pour $k = 3$

Un cycle de longueur impaire convient (sans triangle).

Pour $k \geqslant 4$

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 $p = C^{n_i}_{in_i-i+1}$ copies de G_i notées $G_{i_1}, ..., G_{i_p}$. On ajoute $in_i - i + 1$ sommets appelés centraux (ils forment un ensmble noté A). On note $A_1, ..., A_p$ 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 $\{s_1, ..., s_i\}$ 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.

cqfd

On s'intéresse maintenant à un problème dû à Berge.

Définition (11.10) — 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, pour l'instant, on ne sait pas.

Conjecture des graphes parfaits (11.11) — (faible)

Un graphe est parfait si et seulement si son complémentaire est parfait.

Cette conjecture a été montrée.

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

Définition (11.12) — Trou, anti-trou

Un **trou** *est un cycle sans cordes, de longueur impaire. Un* **anti-trou** *est le complémentaire d'un trou.*

Conjecture des graphes parfaits (11.13) — (forte)

Un graphe est parfait si et seulement si il ne contient ni trou ni anti-trou.

Revenons maintenant à des problèmes plus basiques.

Théorème (11.14) — (Roy, 1967 et Gallai, 1968)

Soit $\mathrm{G} = (\mathrm{X}, \mathrm{E})$ *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* $q - 1$*.*

Preuve :

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

Montrons 1) : On va construire une suite de partitions $\mathcal{P}_k = \{\mathrm{C}_0^k, ..., \mathrm{C}_{p_k}^k\}$ de X vérifiant $\forall x \in \mathrm{C}_i^k$, il existe un chemin d'extrémité terminale x, et de longueur i. On va construire une telle partition avec chaque C_i^k stable, ce qui impliquera $p_k \geqslant \chi(\mathrm{G})$ et donc le résultat recherché.

On pose $\mathcal{P}_0 = \mathrm{X}$ et on suppose avoir construit $\mathcal{P}_k = \{\mathrm{C}_0^k, ..., \mathrm{C}_{p_k}^k\}$. 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 $\mathrm{C}_i^{k+1} = \mathrm{C}_i^k \backslash \{y\}$, $\mathrm{C}_{i+1}^{k+1} = \mathrm{C}_i^k \cup \{y\}$ et $\mathrm{C}_j^{k+1} = \mathrm{C}_j^l$ pour tout j différent de i et $i + 1$.

Clairement, la limite de cette suite est une partition en stables.

Montrons 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, $(x_1, ..., x_k)$ vérifie $1 \leqslant c(x_1) < ... < c(x_k) \leqslant q$ donc $k \leqslant q$ donc la longueur du chemin est inférieure ou égale à $q - 1$.

Définition (11.15) — Tournoi

Un **tournoi** *est un graphe complet orienté.*

Définition (11.16) — Graphe hamiltonien

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

Corollaire (11.17)

Un tournoi est hamiltonien.

Preuve :

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 (11.18) — (Nordhaus et Gaddum, 1960)

Soit $G = (X, E)$ *un graphe et son complémentaire* $\overline{G} = (X, \overline{E})$ *où* $\overline{E} = \mathcal{P}_2(X)\backslash E$.

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

Preuve :

1) Soit $\{s_1, ..., s_k\}$ 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 $$n = \sum_1^k |s_i| \leqslant k.\chi(G) = \chi(G).\chi(\overline{G}).$$

2) Le résultat est trivialement vrai pour $n = 1$ ou 2 (voire même 3). Supposons $n > 2$. Soit x_0 un graphe quelconque de G. Notons $G_0 = G\backslash\{x\}$.

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 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 une inégalité 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)

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

Donc :

$$4n \leqslant 4\chi(G)\chi(\overline{G}) \leqslant \left(\chi(G) + \chi(\overline{G})\right)^2$$

D'où :

$$(\chi(G) + \chi(\overline{G})) \geqslant 2\sqrt{n}$$

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

11.2 Indice chromatique

Définition (11.19) — Indice chromatique

L' **indice chromatique** $\chi'(G)$ *d'un graphe* G *est le cardinal minimal d'une partition de l'ensemble d'arêtes en couplages.*
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 (11.20)

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

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

Preuve :

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 $\{C_1, ..., C_k\}$ 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 (11.21)

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

Preuve :

Le cas de K_{2n} : $m(K_{2n}) = \dfrac{(2n)(2n-1)}{2} = n(2n-1)$ 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 et 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 polyèdre pour former le couplage parfait C_1. $2n - 1$ est alors 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 $\{(k, 2n - 1)\} \cup \{(i + k, 2n - 1 - i + k) | 1 \leqslant i \leqslant n - 1\}$ (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} : $m(K_{2n+1}) = \dfrac{(2n)(2n+1)}{2} = n(2n + 1)$ 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 inclu dans K_{2n+2} donc $\chi'(K_{2n+1}) \leqslant \chi'(K_{2n+2}) = 2n + 1$.

Théorème (11.22) — (Vizing, 1964)

Pour tout graphe $G = (X, E)$, $\Delta(G) \leqslant \chi'(G) \leqslant \Delta(G) + 1$.

Remarque :

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

Preuve :

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 le fait que 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 éléménts 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 $i < j$, et 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 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 dans la même composante connexe de H On va à nouveau décaler les couleurs dans le graphe initial. xy_i est alors 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_i, alternée en C et C_h, finissant par une arête C_h en x. La composante connexe de y_H finit en y_H par une arête coloriée en C. On échange C et C_h dans cette composante pour résoudre le problème.

<u>cqfd</u>

Table des matières