Notes de cours Algorithmique de graphes : Magistère Informatique, Cachan

Michel Habib,
email: habib@liafa.jussieu.fr
http://www.liafa.jussieu.fr/~habib

18 décembre 2006

1

[1]Merci de me faire part de toute remarque, concernant ce texte, erreurs, passages trop rapides . . .

Chapitre 1

Introduction

Nous noterons un graphe $G = (X, E)$, où X est un ensemble **fini** de sommets et $E \subseteq X^2$ un ensemble d'arêtes. Une arête est donc constituée de deux sommets. Les graphes non orientés considérés dans ce cours seront, sauf mention contraire, **simples** : sans boucle (arête de type xx) et sans arête multiple (plusieurs arêtes entre deux sommets).

$H = (Y, F)$ est un **sous-graphe induit** de $G = (X, E)$, si $Y \subseteq X$ et si $F = \{e \in E |$ les deux extrémités de e sont dans $Y\}$.

$H = (Y, F)$ est un **sous-graphe partiel** de $G = (X, E)$, si $Y \subseteq X$ et si $F \subseteq E \cap Y^2$.

En général $|X| = n$ et $|E| = m$. Le **degré** d'un sommet noté $d(x)$ est le nombre d'arêtes adjacentes à x. Un sommet **pendant** dans un graphe, est un sommet x tel que : $d(x) = 1$.

Une **chaîne** de longueur k est un graphe :

$P = (\{x_0, x_1, \ldots, x_k\}, \{x_0x_1, x_1x_2, \ldots x_{k-1}x_k\})$ ayant $k + 1$ sommets et k arêtes, noté P_k. Ce graphe s'appelle une chaîne de longueur k joignant x_0 à x_k.

On confond souvent la chaîne avec l'une des deux séquences : $[x_0, x_1, \ldots, x_k]$ ou $[x_k, x_{k-1}, \ldots, x_0]$. Un graphe réduit à un sommet et sans arête est une chaîne de longueur 0.

Un **cycle** de longueur k est un graphe :

$C = (\{x_1, \ldots, x_k\}, \{x_1x_2, x_2x_3, \ldots x_{k-1}x_k, x_kx_1\})$ ayant k sommets et k arêtes, noté C_k. On confond souvent le cycle avec l'une des séquences : $[x_1, x_2, \ldots, x_k, x_1]$ (à une permutation circulaire près).

Avec de telles définitions, une chaîne (resp. un cycle) ne passe qu'une fois par un sommet, cela correspond à une chaîne (resp. cycle) élémentaire chez

certains auteurs.

On appelle **marche** dans $G = (X, E)$ une séquence non nulle $M = x_0 e_1 x_1 e_2 \dots e_k x_k$ alternant sommets et arêtes telle que :

$\forall 1 \leq i \leq k$, les extrémités de e_i sont x_{i_1} et x_i. On dit que la marche est de longueur k et qu'elle joint x_0 à x_k.

Lorsque $x_0 = x_k$ on dit que la marche est **fermée** ou un **pseudocycle**. Si le graphe est simple, nous pouvons omettre les arêtes et noter la marche par $[x_0, x_1, \dots, x_k]$.

Lorsque toutes les arêtes de la marche sont différentes on dit que la marche est un **sentier**[1], lorsque tous les sommets sont distincts la marche est une chaîne.

Proposition 1 *S'il existe une marche finie joignant x à y dans G, alors il existe une chaîne joignant x à y dans G, en outre les sommets et les arêtes de cette chaîne appartiennent à la marche.*

Preuve: Soit M la marche de G joignant x à y. Il existe donc une marche de longueur minimum joignant x à y dans G. Notons M', cette marche. Si elle n'est pas élémentaire, on peut extraire une marche de longueur strictement plus petite, d'où la contradiction. □

N.B. Cette preuve repose sur la finitude du graphe, car si M est de longueur infinie, il n'est pas sûr que M' soit de longueur strictement inférieure.

Proposition 2 *S'il existe un pseudo-cycle de longueur impaire dans G, alors il existe un cycle impair dans G.*

Un graphe $G = (X, E)$ est **connexe**, si $\forall x, y \in X$, il existe une chaîne allant de x à y.

Proposition 3 *On considère un graphe $G = (X, E)$ connexe, si $\forall x \in X$, $d(x) \leq 2$ alors G est soit une chaîne, soit un cycle.*

Preuve: Si G n'a aucune arête, vu l'hypothèse de connexité, G est réduit à un sommet et est donc une chaîne de longeur 0. Sinon G possède au moins une arête, donc au moins une chaîne et soit C une chaîne de longueur maximale de G. Seules les extrémités x, y de la chaîne peuvent être adjacentes à des

[1] Ces définitions de chaînes, marches, sentiers et autres cycles ne sont pas standard dans la littérature

arêtes n'appartenant pas à la chaîne (car les autres sommets de la chaîne sont déjà de degré 2). Mais si ces arêtes ont une extrémité hors de la chaîne, celle-ci ne serait pas de longueur maximale. Donc soit $d(x) = d(y) = 1$ et G est une chaîne, soit $xy \in E$ et G est un cycle. □

Théorème 1 *Les 6 conditions suivantes sont équivalentes et caractérisent les arbres :*

1. *G est connexe minimal (si on enlève une arête, le graphe n'est plus connexe)*
2. *G est sans cycle maximal (si on ajoute une arête, le graphe admet un cycle)*
3. *G est connexe sans cycle*
4. *G est connexe avec $n-1$ arêtes*
5. *G est sans cycle avec $n-1$ arêtes*
6. *$\forall x, y \in X$, il existe une chaîne unique joignant x à y.*

Preuve: Il est facile de vérifier que les conditions 1, 2, 3 et 6 sont équivalentes.

Montrons l'équivalence avec les conditions 4 et 5. Pour ce faire nous allons commencer par deux lemmes.

Lemme 1 *Un graphe connexe ayant $n-1$ arêtes possède nécessairement un sommet pendant.*

Preuve: G étant connexe, $\forall x \in X$, $d(x) \geq 1$ (il n'y a pas de sommet isolé). S'il n'existe pas de sommet pendant, alors $\forall x \in X$, $d(x) \geq 2$ et donc $\Sigma_{x \in X} d(x) \geq 2n > 2m = 2n - 2$, d'où la contradiction. □

Lemme 2 *Un graphe G sans cycle, ayant au moins une arête, possède nécessairement un sommet pendant.*

Preuve: Il existe au moins une chaîne dans G. Considérons une chaîne $[x = x_1, \ldots, x_k = y]$ de longueur maximale.

Si x n'est pas un sommet pendant alors il admet un autre voisin z différent de x_2. Si $z \in \{x_3, \ldots, x_k\}$, alors l'arbre possède un cycle ce qui est exclu.

Donc $z \notin \{x_2, \ldots, x_k\}$, mais alors la chaîne choisie n'était pas maximale. □

Revenons à la preuve du théorème. Le premier lemme permet de montrer par induction 4 implique 3.

En effet d'après ce lemme un graphe G, vérifiant la condition 4, admet au moins un sommet pendant x. Si l'on considère le graphe $G' = G - x$. G' est sans cycle ssi G est sans cycle. En outre G' a un sommet et une arête de moins que G et vérifie donc aussi la condition 4. On peut donc réitérer le procédé jusqu'à arriver sur le graphe vide réduit à un sommet qui est évidemment sans cycle. Donc 4 implique 3.

Un raisonnement analogue basé sur le lemme 2 permet de montrer 5 implique 3.

Montrons maintenant 3 implique 4 et 5. Le seul graphe à un sommet est connexe et sans cycle et il a bien $n - 1 = 0$ arête.

Suposons maintenant vraie jusqu'à l'ordre $n - 1$ que la condition 3 implique 4 et 5. On considère un graphe $G = (X, E)$ ayant n sommets. Soit $a \in X$. $G - a$ possède $k = d(a) \geq 1$ composantes connexes $G_i = (X_i, E_i)$ qui sont nécessairement connexes et sans cycle, vérifiant ainsi la condition 3. L'hypothèse d'induction sur les graphes G_i donne :

$|E_i| = |X_i| - 1$. D'où l'on déduit :

$|E| = \Sigma_{1 \leq i \leq k} |E_i| + k = \Sigma_{1 \leq i \leq k} (|X_i| - 1) + k = \Sigma_{1 \leq i \leq k} |X_i| = |X| - 1$. $\square$,

Corollaire 1 *Tout arbre non réduit à un sommet admet au moins deux sommets pendants (sommet de degré 1)*

Remarquons que les deux preuves des lemmes précédents permettent d'affirmer que si l'arbre n'est pas réduit à un sommet, il admet au moins **deux** sommets pendants. En outre lorsqu'on supprime un sommet pendant à un arbre, le graphe obtenu est encore un arbre, ce qui nous donne un schéma récursif simple pour manipuler les arbres.

Nous allons maintenant exhiber une propriété d'échange ainsi qu'une dualité.

Définition 1 *Un cocyle $\theta(A)$ (ou une coupe (A, X-A)) d'un graphe $G = (X, E)$ est un ensemble d'arêtes incidentes à un sous ensemble $A \subset X$, i.e. l'ensemble des arêtes n'ayant qu'une extrémité dans A l'autre extrémité étant dans X-A.*

Par définition $\theta(A) = \theta(X - A)$.

Lemme 3 *Si un cycle μ et un cocycle θ ont une arête en commun, ils en ont au moins une deuxième.*

Il est possible de préciser un peu le résultat : un cycle et un cocyle ont en commun un nombre pair d'arêtes.

Lemme 4 *Soit $G = (X, E)$ un graphe connexe et $T = (X, F)$ un arbre recouvrant de G, pour toute arête $e \in E - F$ il existe une arête $f \in F$ telle que $T' = T - f + e$ soit un arbre recouvrant de G.*

Une arête d'un graphe est appelée un **isthme** lorsque sa suppression disconnecte le graphe.

Lemme 5 *Soit $G = (X, E)$ un graphe connexe et $T = (X, F)$ un arbre recouvrant de G, pour toute arête $f \in F$, si $G - f$ est toujours connexe (i.e. f n'est pas un isthme de G), il existe une arête $e \in E - F$ telle que $T' = T - f + e$ soit un arbre recouvrant de G.*

Dans les deux cas, on dit alors avoir procédé à un échange. Il est facile d'en déduire que si T_1 et T_2 sont deux arbres recouvrants de G, il existe une suite finie d'échanges qui permet de passer de T_1 à T_2.

1.1 Espaces vectoriels des cycles-coycles

À chaque cycle (resp. cocycle) on associe le vecteur caractéristique de ses arêtes, un vecteur de $\{0, 1\}^m$.

On considère les coefficients dans $Z/2Z$.

Il est facile de vérifier que : $\theta(A) = \Sigma_{x \in A} \theta(x)$

Ce qui permet de définir la somme de cocycles :

$\theta(A) + \theta(B) = \theta(A \Delta B)$

Nous avons ainsi un sous espace vectoriel des cocycles.

Lorsqu'on considère la somme de cycles, les arêtes communes disparaissent, cependant la définition de la somme de cycles quelconques, nous oblige à considérer comme cycle une union disjointe de cycles. À ce prix on peut définir le sous espace vectoriel des cycles.

On vérifie bien que :

$\forall$ cycle μ et $\forall$ cocycle $\theta(A)$, on a bien

$\mu.\theta(A) = 0$. Les deux sous-espaces sont donc orthogonaux.

Théorème 2 *On considère un graphe $G = (X, E)$ connexe, la dimension de l'espace vectoriel des cycles (resp. cocycles) est $m - n + 1$ (resp. $n - 1$).*

Preuve: Soit $T = (X, F)$ un arbre recouvrant de G. Pour $\forall e \in E - F$, $T+e$ contient un cycle unique μ_e. Ceci nous permet de construire un ensemble de $m - n + 1$ cycles indépendants, car ils ont tous une arête qui les distingue de tous les autres. Donc $dim(Cycles) \geq m - n + 1$.

De même pour $\forall f \in F$, $T - f$ admet deux composantes connexes, et il existe donc un cocyle unique de G, noté θ_f. On peut ainsi construire $n - 1$ cocyles indépendants. Donc $dim(Cocycles) \geq n - 1$.

Comme les espaces vectoriels sont orthogonaux,

$dim(Cycles) + dim(Cocyles) \leq m$, d'où le résultat annoncé. □

On appelle base fondamentale de cycles d'un graphe, une base obtenue à l'aide d'un arbre recouvrant. Il est facile de voir en considérant le graphe 3-soleil, qu'il existe des bases de cycles qui ne sont pas fondamentales.

Ces bases de cycles sont utiles en chimie organique pour l'analyse des fonctions associées à une molécule.

Chapitre 2

Arbre recouvrant de poids minimum et algorithmes gloutons

On considère, un graphe non orienté connexe $G = (X, E)$ et une fonction de poids $\omega : E \to R_+$. Et l'on cherche donc un un sous-graphe $T = (X, F)$ connexe dont la somme des poids des arêtes est minimale.

Comme application on peut imaginer que le graphe G représente le réseau des rues d'une ville que l'on cherche à cabler. Il faut donc trouver un graphe **connexe** (chaque sommet doit avoir accès au réseau) et de coût minimum. En effet le coût d'installation d'un cable peut dépendre de la rue choisie (faire passer un cable sous un tramway ou un monument historique peut coûter cher) c'est ce que modélise la valuation ω.

Il est facile de voir, comme la valuation ω est positive qu'une solution optimale d'un tel problème est nécessairement un arbre (graphe connexe minimal, cf. le chapitre précédent). Ce problème est donc appelé **le problème de l'arbre recouvrant de poids minimum**.

Ce problème admet une solution, car l'ensemble des arbres recouvrants est un ensemble fini, donc il admet un élément de coût minimal. Tout le problème c'est de le trouver sans énumérer tous les arbres recouvrants car il peut y en avoir beaucoup (ex : n^{n-2} pour un graphe complet, d'après le célèbre théorème de Cayley publié en 1889).

Pour ce faire on va construire un schéma d'algorithme, basé sur une bi-coloration en rouge ou bleu des arêtes de G. Cette présentation très élégante est due à Tarjan [32].

Au départ les arêtes ne sont pas colorées et pour les colorier (de manière définitive) nous allons utiliser les deux règles suivantes :

Règle Bleue : Choisir un cocycle θ qui ne contient pas d'arête bleue, colorier en bleu une arête non colorée de coût minimal de θ.

Règle Rouge : Choisir un cycle μ qui ne contient pas d'arête rouge, colorier en rouge une arête non colorée de coût maximal de μ

On remarquera que ces règles sont duales l'une de l'autre, par l'échange cycle-cocyle et bleu-rouge.

Arbre recouvrant de poids minimum
tant que *une règle est applicable* **faire**
└ l'appliquer

Théorème 3 *Si G est connexe, l'algorithme précédent calcule un arbre recouvrant de poids minimum.*

Preuve:

Le graphe étant fini, il existe un nombre fini d'arbres recouvrants, parmi ceux-ci il en existe un de poids minimum, notons le $T = (X, F)$.

Nous allons prouver que l'invariant suivant est maintenu dans l'algorithme :

Invariant principal : Il existe un arbre de poids minimum qui contient toutes les arêtes bleues et aucune rouge.

La preuve se fait par récurrence sur le nombre d'application des règles. Initialement l'invariant est trivialement vrai.

Montrons tout d'abord que les arêtes bleues forment une forêt de G (i.e. un graphe sans cycle). La preuve se fait par induction sur le nombre d'applications de la règle bleue. Au départ il n'y a pas d'arêtes bleues donc pas de cycle et supposons que l'application de la ième règle bleue sur l'arête $ab \in \theta$ introduise un cycle entièrement bleu μ. Mais d'après le lemme 3 θ et μ ont une autre arête en commun, d'où la contradiction.

Supposons le vrai lorsque k arêtes ont déjà été coloriées par application des ces deux règles, il existe donc un arbre $T = (X, F)$ recouvrant de poids minimal qui contient toutes les arêtes bleues et aucune rouge.

Considérons une nouvelle application, par exemple de la règle bleue. On colore donc en bleu une arête f d'un cocyle θ. Dans le cas où $f \in F$ alors l'invariant est encore trivialement vérifié. Supposons donc le contraire, f n'appartient pas à F. Le graphe $T = (X, F \cup \{f\})$ admet donc un cycle μ. f appartient à μ et θ, or un cycle et un cocycle ont au moins deux arêtes en

commun, soit g une telle arête. Nécessairement g $\in \theta$ et d'après l'application de la règle bleue $\omega(f) \leq \omega(g)$.

Cependant $T' = (X, F \cup \{f\} - g)$ est aussi un arbre recouvrant (échange des arêtes f et g, on note $T' = T + f - g$. Nous avons $\omega(T') = \omega(T) + \omega(f) - \omega(g)$, la minimalité de T impose donc que $\omega(f) = \omega(g)$ et donc $\omega(T') = \omega(T)$, et l'arbre T' permet de montrer que l'invariant est encore vrai.

Le raisonnement est similaire dans le cas d'une application de la règle rouge.

Examinons ce que se passe lorsqu'on ne peut plus appliquer de règle. Supposons que les arêtes bleues forment un graphe partiel ayant plusieurs composantes connexes sur les ensembles de sommets $X_1, \ldots, X_k$, avec $k \geq 2$. Considérons le cocyle engendré par X_1, $\theta(X_1)$ qui ne contient par définition aucune arête bleue. Mais est-il possible que toutes ses arêtes soient rouges ? Montrons le contraire. L'ivariant étant toujours vrai, il existe un arbre de poids minimal $T_{final} = (X, F_{final})$ qui contient toutes les arêtes bleues et aucune rouge. Cet arbre utilise au moins une arête de $\theta(X_1)$, et cette arête ne peut donc être rouge, donc il existe au moins une arête de $\theta(X_1)$ incolore. On peut appliquer sur $\theta(X_1)$ la règle bleue et colorier en bleu l'arête incolore de poids minimal, d'où la contradiction.

Ceci implique qu'à la fin les arêtes bleues forment un graphe partiel connexe de G lorsque celui-ci est connexe. Ce graphe est nécessairement un arbre d'après l'invariant principal car les arêtes bleues sont incluses dans un arbre de poids minimum.

Montrons maintenant que toutes les arêtes sont bien coloriées. Pour ce faire supposons qu'il existe une arête $g \in E$ non coloriée. Nécessairement $g \notin F$, mais alors il existe un cycle unique dans $T + g$ sur lequel nous allons pouvoir appliquer la règle rouge, contradiction.

Conclusions : l'algorithme s'arrête donc et lorsqu'aucune règle n'est applicable, toutes les arêtes sont coloriées. Les arêtes bleues constituent un arbre recouvrant de poids minimum et les arêtes rouges un coarbre (un coarbre est par définition le complémentaire d'un arbre recouvrant de G).

□

L'algorithme générique proposé est dit **glouton** car il ne remet jamais en question la coloration d'une arête. Il existe plusieurs mises en application différentes de cet algorithme générique. Les plus célèbres sont dues à Kruskal [23] et à Prim [27].

Dans tous les algorithmes présentés ci-après pour la recherche d'un arbre recouvrant de poids minimum, le graphe initial G est supposé connexe.

Algorithme de Kruskal
Trier les arêtes de G par poids croissant en une liste L
$F \leftarrow \emptyset$
tant que $|F| < n-1$ *et* $L \neq \emptyset$ **faire**
  $e \leftarrow Premier(L)$
  Retrait(L, e) ; **si** $T = (X, F \cup \{e\})$ *n'a pas de cycle* **alors**
    $F \leftarrow F \cup \{e\}$

Cet algorithme revient à utiliser n-1 fois la règle bleue lorsqu'on ajoute une arête dans F, et un autant de fois la règle rouge que l'on détecte la présence d'un cycle en ajoutant l'arête e. En outre il est facile de montrer qu'à chaque itération la propriété : F *est sans cycle* est maintenue, et donc à la fin F est un arbre recouvrant de G.

Il n'est donc pas nécessaire de colorier explicitement les arêtes non considérées (les arêtes de L à la fin de l'algorithme) car elles sont nécessairement rouges et ne peuvent intervenir dans un arbre de poids minimal (cf. l'invariant).

L'algorithme de Prim revient aussi à toujours appliquer la règle bleue, en travaillant toujours sur un seul cocyle que l'on maintient au cours de l'algorithme. En outre la propriété F *engendre un graphe connexe est maintenue*, et à la fin F est nécessairement un arbre recouvrant de G.

Algorithme de Prim
$F \leftarrow \emptyset$
$A \leftarrow \{x_0\}$
tant que $|F| < n-1$ **faire**
  Calculer l'arête $e = (a, x)$ (avec $a \in A$) de coût minimal de $\theta(A)$
  $F \leftarrow F \cup \{e\}$
  $A \leftarrow A \cup \{x\}$

Une variante due à Boruvka (1926) d'après [19], est d'autant plus intéressante qu'elle précédait historiquement celles de Kruskal et de Prim.

Algorithme de Boruvka
$F \leftarrow \emptyset$
tant que G *n'est pas trivial* **faire**
    Détruire les boucles de G
    Remplacer les arêtes multiples entre deux sommets, par une seule dont le poids est le minimum des poids de ces arêtes multiples
    pour $\forall x \in G$ **faire**
        trouver l'arête e_x de poids minimum adjacente à x
        $F \leftarrow F \cup \{e_x\}$
        Contracter e_x

Montrer que cet algorithme se prête aisément à la parallélisation.

2.1 Quelques variantes

Kruskal à l'envers
Trier les arêtes de G par poids décroissant en une liste L
$F \leftarrow E$
tant que $|F| > n-1 et L \neq \emptyset$ **faire**
  $e \leftarrow Premier(L)$
  Retrait(L, e) ; **si** $T = (X, F - \{e\})$ *est connexe* **alors**
    $F \leftarrow F - \{e\}$

L'algorithme "Kruskal à l'envers" est intéressant à utiliser lorsque le nombre d'arêtes du graphe G est très proche de n-1, car le nombre de passages dans la boucle **tant que** est borné par $|E| - n + 1$.

Kruskal-dans-le-désordre
Construire une liste L avec les arêtes de G
$F \leftarrow \emptyset$
tant que $L \neq \emptyset$ **faire**
  $e \leftarrow Premier(L)$
  Retrait(L, e)
  si $T = (X, F \cup \{e\})$ *n'a pas de cycle* **alors**
    $F \leftarrow F \cup \{e\}$
  sinon
    si $\omega(e) < \omega(f)$, *où f est une arête de poids maximum du cycle de* $F \cup \{e\}$ **alors**
      $F \leftarrow (F \cup \{e\}) - \{f\}$

L'algorithme **Kruskal-dans-le-désordre** permet d'éviter la phase initiale de tri des arêtes du graphe suivant leur poids.

2.1.1 Ordre lexicographique sur les arbres

L'ensemble des arbres recouvrants d'un graphe peuvent être muni d'un ordre partiel $\preccurlyeq$, défini comme suit :

À chaque arbre T on associe le mot $m(T)$ constitué des poids des arêtes de l'arbre classées par ordre croissant.

Nous pouvons alors définir $T \preccurlyeq T'$ ssi $m(T) \leq_{lex} m(T')$.

L'ordre défini ci-dessus n'est qu'un ordre partiel, car un même mot peut-

être associé à deux arbres, qui ont alors le même poids.

Il est facile de constater que l'algorithme de Kruskal construit un arbre minimal pour l'ordre $\preccurlyeq$. Cependant nous avons un résultat beaucoup plus fort.

Propriété 1 *Un arbre T est de poids minimum ssi il est minimal pour l'ordre $\preccurlyeq$.*

Chapitre 3

Graphes orientés

3.1 Définitions de base sur les graphes orientés

Nous noterons un graphe orienté $G = (X, U)$, où X est un ensemble **fini** de sommets et $U \subseteq X^2$ un ensemble d'arcs. Un arc (x, y) est donc constituée de deux sommets : une origine x et une extrémité terminale y. Les graphes orientés considérés dans ce cours seront, sauf mention contraire, **simples** : sans boucle (arête de type xx) et sans arc multiple (plusieurs arcs entre deux sommets).

$H = (Y, V)$ est un **sous-graphe induit** de $G = (X, U)$, si $Y \subseteq X$ et si $V = \{(x, y) \in U \mid x, y \in Y\}$.

$H = (Y, V)$ est un **sous-graphe partiel** de $G = (X, U)$, si $Y \subseteq X$ et si $V \subseteq U \cap Y^2$.

En général $|X| = n$ et $|E| = m$. Le **degré** d'un sommet noté $d+(x)$ (resp. $d^-(x)$) est le nombre d'arcs sortant (resp. rentrant) de x. Un sommet **pendant** dans un graphe, est un sommet x tel que : $d(x) = 1$.

Une **chemin** de longueur k est un graphe :

$P = (\{x_0, x_1, \ldots, x_k\}, \{(x_0, x_1), (x_1, x_2), \ldots (x_{k-1}, x_k)\})$ ayant $k + 1$ sommets et k arcs, noté P_k. Ce graphe s'appelle un chemin de longueur k joignant x_0 à x_k.

On confond souvent le chemin avec la séquence : $[x_0, x_1, \ldots, x_k]$. Un graphe réduit à un sommet et sans arc est une chemin de longueur 0.

Un **circuit** de longueur k est un graphe :

$C = (\{x_1, \ldots, x_k\}, \{(x_1, x_2), (x_2, x_3), \ldots (x_{k-1}, x_k), (x_k, x_1)\})$ ayant k sommets et k arcs, noté C_k. On confond souvent le circuit avec l'une des sé-

quences : $[x_1, x_2, \ldots, x_k, x_1]$ (à une permutation circulaire près).

Avec de telles définitions, un chemin (resp. un circuit) ne passe qu'une fois par un sommet, cela correspond à une chaîne (resp. circuit) élémentaire chez certains auteurs.

On appelle **marche** dans $G = (X, U)$ une séquence non nulle $M = x_0 u_1 x_1 u_2 \ldots u_k x_k$ alternant sommets et arcs telle que :

$\forall 1 \leq i \leq k$, l'origine de u_i est x_{i_1} et son extrémité terminale x_i. On dit que la marche est de longueur k et qu'elle joint x_0 à x_k.

Lorsque $x_0 = x_k$ on dit que la marche est **fermée** ou un **pseudocycle**. Si le graphe est simple, nous pouvons omettre les arcs et noter la marche par $[x_0, x_1, \ldots, x_k]$.

Lorsque tous les arcs de la marche sont différents on dit que la marche est une **piste**[1], lorsque tous les sommets sont distincts la marche est un chemin.

Pour les graphes orientés, les choses se compliquent un peu, car nous retrouvons les objets du chapitre précédents dès que l'on considère le graphe non orienté sous-jacent, sur lequel on peut identifier des chaînes, cycles et autres marches.

Proposition 4 *S'il existe une marche finie joignant x à y dans G, alors il existe un chemin joignant x à y dans G, en outre les sommets et les arcs de ce chemin appartiennent à la marche.*

Preuve: Soit M la marche de G joignant x à y. Il existe donc une marche de longueur minimum joignant x à y dans G. Notons M', cette marche. Si elle n'est pas élémentaire, on peut extraire une marche de longueur strictement plus petite, d'où la contradiction. □

N.B. Cette preuve repose sur la finitude du graphe, car si M est de longueur infinie, il n'est pas sûr que M' soit de longueur strictement inférieure.

3.2 Arborescences

Une **racine** d'un graphe orienté $G = (X, U)$ est un sommet $r \in X$ tel que $\forall x \in X$, $x \neq r$, il existe un chemin de r à x.

Théorème 4 *Pour un graphe orienté $G = (X, U)$ avec $|X| \geq 2$, les 6 conditions suivantes sont équivalentes et caractérisent les arborescences :*

[1]Ces définitions de chemins, marches, pistes et autres circuits ne sont pas standard dans la littérature.

1. *G est connexe sans cycle et admet une racine*
2. *G possède une racine et a $n-1$ arcs*
3. *G possède une racine r et $\forall x \in X$, il existe un chemin unique de r à x*
4. *G possède une racine et est minimal avec cette propriété (tout retrait d'un arc de G supprime la propriété)*
5. *G est connexe et il existe un sommet $r \in X$, avec $d^-(r) = 0$ et $\forall x \in X$, $x \neq r$, $d^-(x) = 1$*
6. *G est sans cycle et il existe un sommet $r \in X$, avec $d^-(r) = 0$ et $\forall x \in X$, $x \neq r$, $d^-(x) = 1$*

Preuve: (2) $\rightarrow$ (3) :

L'existence d'une racine implique la connexité de G, le nombre d'arcs permet de conclure que le graphe non orienté sous-jacent est un arbre. D'après le théorème sur les arbres, $\forall x \in X$, $x \neq r$ le chemin de r à x est nécessairement unique.

(3) $\rightarrow$ (4) : trivial.

(4) $\rightarrow$ (5) :

□

Ainsi dans une arborescence, tout sommet sauf la racine admet un prédécesseur unique ($d^-(x) = 1$) et donc on peut représenter une arborescence à l'aide de la fonction Parent : $X \rightarrow X$.

Chapitre 4

Parcours et chemins dans les graphes

4.1 Parcours de graphes

Un algorithme de parcours dans un graphe peut se formaliser à l'aide de deux ensembles de sommets OUVERTS et FERMES, le premier contenant les sommets à explorer et le deuxième les sommets déjà explorés. L'exploration d'un sommet consistant à examiner tous les voisins d'un sommet : visite de tous les arcs sortants du sommet.

ParcoursGénérique(G, x_0)
Données: un graphe orienté $G = (X, U)$, une fonction de coût $\omega : U \to T$, un sommet x_0
Résultat: une arborescence de chemins issus de x_0
$OUVERTS \leftarrow \{x_0\}$
$FERMES \leftarrow \emptyset$
$Parent(x_0) \leftarrow NIL$
$\forall y \neq x_0\ ,\ Parent(y) \leftarrow y$
tant que $OUVERTS \neq \emptyset$ **faire**
  $z \leftarrow Choix(OUVERTS)$
  $Ajout(z, FERMES)$
  Explorer(z)
  $Virer(z, OUVERTS)$

Explorer(z)
pour *Tous les voisins y de z* **faire**
  si $y \in FERMES$ **alors**
    Ne rien faire
  si $y \in OUVERTS$ **alors**
    Ne rien faire
  $Ajout(y, OUVERTS)$
  $Parent(y) \leftarrow z$

Théorème 5 *L'algorithme précédent calcule en $O(n+m)$ une arborescence de racine x_0 recouvrant l'ensemble des sommets atteignables à partir de x_0.*

Preuve: Il est facile de vérifier que l'algorithme précédent vérifie bien les invariants suivants :

Invariants :

– La fonction Parent définit une arborescence de racine x_0,

– $\forall x \in OUVERTS$, il existe un chemin de x_0 à x.

Ainsi à la fin du parcours, l'ensembles des sommets FERMES est égal à l'ensemble des sommets atteignables dans le graphe G à partir de x_0. Cet ensemble est décrit à l'aide de la fonction Parent. Un sommet est exploré au plus une fois et donc l'algorithme est en $O(n+m)$ si le graphe est représenté par ses listes d'adjacence. □

Ce parcours s'applique aux graphes non orientés et orientés. Le déroulement d'un parcours permet d'ordonner (numéroter) les sommets d'un graphe. Ces ordres sont liés à la gestion de l'ensemble des sommets OUVERTS et à la fonction de choix du prochain sommet à explorer. Cet ensemble peut être géré comme une pile (Parcours en profondeur) ou une File (Parcours en largeur).

Lorsque tous les sommets du graphe ne sont atteignables à partir de x_0 si l'on veut parcourir tout le graphe, il suffit d'ajouter les instructions suivantes et de représenter l'ensemble des sommets FERMES à l'aide d'un tableau de booléens :

$Parcours(G)$:
forall $v \in X$ **do**
  $Ferme(x) \leftarrow Faux$
forall $v \in X$ **do**
  si $Ferme(x) = Faux$ **alors**
    ParcoursGénérique(G, x)

Dans ce cas l'algorithme calcule une forêt d'arborescences recouvrante de G.

Parcours en largeur

On obtient un parcours en largeur dès que l'on gère l'ensembles des sommets OUVERTS comme une file (premier entré, premier sorti).

Parcours en profondeur

On obtient un parcours en profondeur dès que l'on gère l'ensembles des sommets OUVERTS comme une pile (dernier entré, premier sorti). Cette gestion d'une pile se prête à une écriture récursive du programme, comme suit :

$DFS(G)$:
forall $v \in X$ **do**
  $Ferme(x) \leftarrow Faux$
forall $v \in X$ **do**
  si $Ferme(x) = Faux$ **alors**
    $Explorer(G, x)$

$Explorer(G, x)$:
$Ferme(x) \leftarrow Vrai$;
pre(x) ;
forall $xy \in U$ **do**
  si $Ferme(y) = Faux$ **alors**
    $Explorer(G, y)$
  post(x) ;

Et les deux fonctions :
$pre(x)$:
$pre(x) \leftarrow comptpre$;
$comptpre \leftarrow comptpre + 1$;

$post(x)$:
$post(x) \leftarrow comptpost$;
$comptpost \leftarrow comptpost + 1$;

En utilisant les propriétés des numérotations $pre(x)$ et $post(x)$, il est possible de reconstituer le fonctionnement de la pile qui a permis de gérer

l'ensemble de sommets OUVERTS. Ce parcours a permis d'écrire des algorithmes linéaires (en $O(n+m)$) pour la recherche des composantes fortement connexes d'un graphe orienté, de recherche des composantes 2-arête connexes d'un graphe non orienté, ainsi qu'un algorithme qu'un test de planarité.

Une **extension linéaire** d'un graphe sans circuit $G = (X, U)$ est un ordre total sur les sommet de G, noté τ, vérifiant :

$\forall x, y \in X$, $xy \in U$ implique $x \leq_\tau y$.

Théorème 6 *Si G est un graphe sans circuit, alors le parcours en profondeur appliqué sur G vérifie la propriété :*

L'ordre inverse de dépilement (noté $post^d$) est une extension linéaire de G.

Preuve: Considérons deux sommets $x, y \in X$ et supposons $xy \in U$. Deux cas sont possibles :

1. $pre(x) < pre(y)$. Mais alors l'exploration de y se terminera donc avant celle de x et nous avons bien : $post^d(x) < post^d(y)$.

2. $pre(y) < pre(x)$. Comme il n'existe pas de chemin de y à x, car sinon G aurait un circuit, nous avons que l'exploration de y se terminera avant celle de x. Et donc $post^d(x) < post^d(y)$.

□

Parcours en largeur lexicographique

Il suffit de prendre $T = \{1, 2, \ldots, n\}^*$ (l'ensemble des mots construits avec les nombres 1,2 jusqu'à n), et de choisir comme sommet à explorer celui dont l'étiquette est maximale lexicographiquement.

Cet algorithme s'utilise sur les graphes non orientés.

La fonction Explorer devenant :

Explorer(z)
$numero(z) \rightarrow numerocourant - 1$
$numerocourant \rightarrow numerocourant - 1$
pour *Tous les voisins y de z* **faire**
  si $y \in FERMES$ **alors**
    Ne rien faire
  si $y \in OUVERTS$ **alors**
    $d(y) \rightarrow d(y) \bullet numero(z)$
  sinon
    $d(y) \rightarrow d(y) \bullet numero(z)$
    $Ajout(y, OUVERTS)$

le symbole "$\bullet$" représentant l'opération de concaténation, on étiquette ainsi les sommets du graphe avec des mots sur $\{1, 2, \ldots, n\}^*$.

La variable globale numerocourant doit être initialisée à n dans les initialisations de la procédure principale.

4.1.1 Algorithmes de plus courts chemins

4.2 Introduction

On considère un graphe $G = (X, U)$ orienté et l'on suppose les arcs munis d'étiquettes, i.e. une valuation $\omega : U \rightarrow R$.

4.3 Les différents problèmes

Pour un graphe $G = (X, U)$ orienté dont les arcs sont valués

1. Etant donné deux sommets a et b, trouver la plus courte marchant allant de a jusqu'à b dans G.
2. Etant donné un sommet a trouver pour chaque $x \in X$ les plus courtes marches allant de a jusqu'à x dans G.
3. Pour tout $x, y \in G$, trouver une plus courte marche allant de x jusqu'à y.

Dans l'énumération ci-dessus, la complexité du problème est croissante. En effet un algorithme qui resout le problème 3, permet de résoudre le pro-

blème 2 et un algorithme qui résout le problème 2 permet de résoudre le problème 1.

Cependant lorsque les valuations choisies sont de signe quelconque, la solution au problème 1 peut-etre une marche infinie comprenant un circuit de valeur négative (circuit dit **absorbant**).

Il est tentant de transformer le problème en :

Etant donné deux sommets a et b, trouver le plus court chemin allant de a jusqu'à b dans G.

Mais alors le problème devient NP-difficile.

Algorithme de Dijkstra [16]

Algorithme de Dijkstra 1959
Données: un graphe orienté $G = (X, U)$, une fonction de coût
$\omega : U \to \mathcal{R}^+$
Résultat: une arborescence de chemins issus de x_0
$OUVERTS \leftarrow \{x_0\}$
$FERMES \leftarrow \emptyset$
$Parent(x_0) \leftarrow NIL$
$\forall y \neq x_0\ ,\ Parent(y) \leftarrow y$
$d(x_0) \leftarrow 0$
$\forall y \neq x_0\ ,\ d(y) \leftarrow \top$ (l'élément maximal de T)
tant que $OUVERTS \neq \emptyset$ **faire**
  Choisir un sommet $z \in OUVERTS$ tel que
  $d(z) = min_{y \in OUVERTS}\{d(y)\}$
  $Ajout(z, FERMES)$
  Explorer(z)
  $Virer(z, OUVERTS)$

Explorer(z)
pour *Tous les voisins y de z* **faire**
  si $y \in FERMES$ **alors**
    Ne rien faire
  si $y \in OUVERTS$ **alors**
  sinon
    si $d(z) + \omega(z, y) < d(y)$ **alors**
      $Parent(y) \leftarrow z$
      $d(y) \leftarrow d(z) + \omega(z, y)$
  $Ajout(y, OUVERTS)$
  $Parent(y) \leftarrow z$
  $d(y) \leftarrow d(z) + \omega(z, y)$

Théorème 7 *Lorsque la valuation ω est à valeurs positives, l'algorithme calcule bien une arborescence des plus courts chemins issus de x_0.*

Preuve:

Considérons les invariants principaux de l'algorithme.

Invariants

1. $\forall x \in OUVERTS$, il existe un chemin μ de x_0 à x tel que, $d(x) = \omega(\mu)$
2. À la fin de la ième exploration d'un sommet, $\forall x \in OUVERTS \cup FERMES$, $d(x)$ est égale à la valuation minimale d'un chemin de G de x_0 à x n'utilisant que des sommets $FERMES$ sauf éventuellement x.

L'invariant 1 se montre facilement par récurrence, montrons l'invariant 2 aussi par récurrence. Pour se faire indexons par i tous les objets de l'algorithme à la fin de l'exploration du ième sommet.

À l'initialisation lorsque $i = 0$, l'invariant est trivialement vrai. Supposons le maintenant vrai jusqu'à l'exploration du (i-1)ème sommet. Soit x le sommet exploré à la ième itération.

Considérons $z \in OUVERTS_i \cup FERMES_i$, il y a plusieurs cas possibles.

- Soit $xz \notin U$, mais alors il n'existe pas de chemin joignant x_0 à z n'utilisant que des sommets de $FERMES_i$ sauf éventuellement z. Comme $d_i(z) = d_{i-1}(z)$, l'hypothèse de récurrence permet de conclure.
- Soit $xz \in U$ et $z \notin OUVERTS_{i-1} \cup FERMES_{i-1}$. Dans ce cas le seul chemin joignant x_0 à z n'utilisant que des sommets de $FERMES_i$

sauf éventuellement z passe par x et l'algorithme affecte bien à $d_i(z)$ la valeur de ce chemin.

- Soit $xz \in U$ et $z \in OUVERTS_{i-1}$. Quand le sommet x est fermé à la ième étape, il existe un chemin joignant x_0 à z n'utilisant que des sommets de $FERMES_i$ sauf éventuellement z passant par x. L'algorithme prend en compte la valeur de ce chemin pour calculer $d_i(z)$, on peut conclure grâce à l'hypothèse de récurrence.
- Soit $xz \in U$ et $z \in FERMES_{i-1}$. Dans ce cas l'algorithme ne fait rien (i.e. $d_i(z) = d_{i-1}(z)$) montrons que c'est justifié. En effet supposons qu'il existe un chemin $\mu = [x_0, x_1, \ldots x_k, x_{k+1} = x, z]$ allant de x_0 à z tel que $\omega(\mu) < d_i(z)$.
  Soit $j \in [0, k]$ l'indice du dernier sommet de μ qui ait été exploré avant que z ne soit exploré à l'étape h. Par hypothèse de récurrence $d_h(x_{j+1}) = \omega(\mu[x_0, x_{j+1}])$. Comme les valuations des arcs sont positives, on en déduit : $d_h(x_{j+1}) < d_h(z)$ ce qui contredit l'hypothèse sur j, car l'algorithme aurait du explorer x_{j+1} avant z.

□

L'algorithme A*

Si la fonction choix fournit un sommet z vérifiant :

$d(z) + h(z) = min_{y \in OUVERTS}\{d(y) + h(y)\}$ où h(y) est une information "heuristique" sur la distance qui reste à parcourir.

et si l'instruction : "Ne rien faire" de l'algorithme de Dijkstra est remplacée par :

si $d(z) + \omega(z, y) < d(y)$ **alors**
  $Parent(y) \leftarrow z$
  $d(y) \leftarrow d(z) + \omega(z, y)$
  $Ajout(y, OUVERTS)$

On suppose que cette fonction $h(y)$ est une information disponible en chaque sommet du graphe. L'usage de cet algorithme est adapté au cas où l'on cherche un plus court chemin d'un sommet x_0 à un sommet t. La valeur de $h(x)$ pour un sommet x donné, étant une estimation de la distance qu'il reste à parcourir pour aller de x à t.

Dans les deux instanciations précédentes, le goulot d'étranglement de complexité provient de la gestion de l'ensembles des OUVERTS. Il faut utili-

ser une structure de données qui permette le calcul du minimum en $O(logn)$ ou mieux.

4.3.0.1 Algorithme de Prim

Le graphe est non orienté, les arêtes sont valuées et si le sommet à explorer est celui

$d(z) = min_{y \in OUVERTS}\{d(y)\}$

et si l'exploration devient :

Explorer(z)
pour *Tous les voisins y de z* **faire**
  si $y \in FERMES$ **alors**
    Ne rien faire
  si $y \in OUVERTS$ **alors**
  sinon
    si $\omega(z,y) < d(y)$ **alors**
      $Parent(y) \leftarrow z$
      $d(y) \leftarrow \omega(z,y)$
  $Ajout(y, OUVERTS)$
  $Parent(y) \leftarrow z$
  $d(y) \leftarrow \omega(z,y)$

L'algorithme ci-dessus calcule un arbre de poids minimum de G et c'est une implémentation de l'algorithme de Prim. En outre, il est facile de montrer que l'arborescence obtenue est celle des distances minimum (suivant la distance du max) issue de x_0.

4.4 Un cadre général

Tous les algorithmes de plus courts chemins précédemment présentés peuvent être vus comme des implémentations d'un algorithme générique, défini comme suit. On considère un graphe $G = (X, U)$ orienté et l'on suppose les arcs munis d'étiquettes, i.e. une valuation $\omega : U \rightarrow T$. où T est un ensemble muni d'une relation d'ordre total $\ll$, et de deux éléments distingués $\top$ (resp. $\bot$) un unique élément maximal (resp. minimal).

On étend cette valuation aux chemins comme suit :

$\mu = [x_1, \ldots, x_k]$, $\omega(\mu) = \oplus_{i=0}^{i=k-1} \omega(x_i, x_{i+1})$.

La relation $\oplus$ binaire associative sur T, étant interprétée suivant les exemples, comme :

- l'addition dans le cas usuel où T est l'ensemble des entiers, des rationnels ou des réels,
- le maximum,
- un produit ou la concaténation dans un monoide,
- ...

La relation d'ordre total $\ll$ s'interprétant comme :

- l'ordre usuel quand T est l'ensemble des entiers,
- l'ordre lexicographique lorsque T est un langage,
- ...

Algorithme de Parcours Générique
Données: un graphe orienté $G = (X, U)$, une fonction de coût
$\omega : U \to T$
Résultat: une arborescence de chemins issus de x_0
$OUVERTS \leftarrow \{x_0\}$
$FERMES \leftarrow \emptyset$
$Parent(x_0) \leftarrow NIL$
$\forall y \neq x_0$, $Parent(y) \leftarrow y$
$d(x_0) \leftarrow \perp$ (l'élément minimal de T)
$\forall y \neq x_0$, $d(y) \leftarrow \top$ (l'élément maximal de T)
tant que $OUVERTS \neq \emptyset$ **faire**
  $z \leftarrow Choix(OUVERTS)$
  $Ajout(z, FERMES)$
  Explorer(z)
  $Virer(z, OUVERTS)$

Explorer(z)
pour *Tous les voisins y de z* **faire**
  si $y \in FERMES$ **alors**
    Ne rien faire
  si $y \in OUVERTS$ **alors**
  sinon
  si $d(z) \oplus \omega(z,y) < d(y)$ **alors**
  $Parent(y) \leftarrow z$
  $d(y) \leftarrow d(z) \oplus \omega(z,y)$
  $Ajout(y, OUVERTS)$
  $Parent(y) \leftarrow z$
  $d(y) \leftarrow d(z) \oplus \omega(z,y)$

Chapitre 5

Connexités

Un graphe G est **k-connexe** avec $k \geq 1$ si pour $\forall x, y \in G$, il existe k chaînes sommet-disjointes allant de x à y.

De même un graphe G est **k-arête-connexe** avec $k \geq 1$ si pour $\forall x, y \in G$, il existe k chaînes arête-disjointes allant de x à y.

Un point **d'articulation** (resp. un **isthme**) de G, est un sommet x (resp. une arête e) tel que $G - x$ (resp. $G - e$) est non-connexe.

Théorème 8 *Pour un graphe non orienté G connexe, les quatre conditions suivantes sont équivalentes :*

1. *G est 2-connexe*
2. *G n'admet pas de point d'articulation*
3. *$\forall x, y, z \in G$, il existe une chaîne de x à y passant par z*
4. *$\forall x, y, z \in G$, il existe une chaîne de x à y évitant z*

Chapitre 6

Graphes planaires

6.1 Définitions de base

Les notions étudiées ci-après de planarité, sont parmi les notions les plus délicates à présenter de la théorie des graphes car elles utilisent de la topologie (notions de voisinage) de la géométrie (coordonnées, distances) et de la combinatoire (gestion de tous les associations possibles).

Cependant l'étude des graphes planaires est indispensable en imagerie, car toute image 2D peut être représentée par un graphe planaire.

Définition 2 *Un graphe $G = (X, E)$ non orienté est dit planaire s'il admet une représentation dans le plan dans laquelle les arêtes ne se croisent qu'aux sommets.*

Une représentation planaire d'un graphe est appelée un graphe planaire topologique ou plongement, on la notera $\phi(G)$. On posera $|X| = n$ et $|E| = m$.

Tous les graphes ne sont pas planaires, ainsi $K_{3,3}$ et K_5 ne le sont pas.

Un graphe planaire peut avoir plusieurs plongement planaires non isomorphes.

Dans un plongement planaire on associe à chaque sommet un point du plan et les arêtes sont des courbes continues du plan "arcs de Jordan". Cependant pour l'usage que nous en ferons nous pouvons faire l'hypothèse que les arêtes sont représentées par des lignes polygonales (suite finie de segments de droites).

Commençons par quelques définitions de topologie :

Définition 3 *U un ensemble de points de R^2 est appelé un **ouvert**, s'il vérifie :*

$\forall p \in U, \exists \epsilon > 0$ tel que $\forall p'$ vérifiant $d(p,p') \leq \epsilon$ alors $p' \in U$.

Les deux définitions suivantes utilisent un plongement planaire $\phi(G)$ de G.

Définition 4 *O un ouvert de R^2 est appelé une **région** de R^2 par rapport à $\phi(G)$, s'il vérifie :*

$\forall p, p' \in O$, il existe une ligne polygonale qui va de p à p' sans croiser d'arête de $\phi(G)$.

Définition 5 *Une région F maximale est appelée une **face** de $\phi(G)$.*

La version restreinte du théorème de Jordan :

Théorème 9 *Une ligne polygonale simple fermée partage le plan en 2 régions (l'intérieur et l'extérieur).*

Soit F l'ensemble des faces de $\phi(G)$, on notera $|F| = f$ et $f = f_0 + 1$ où f_0 représente le nombre de faces finies.

Théorème 10 *Si G est connexe, pour tout plongement planaire $\phi(G)$ de G vérifie la relation d'Euler :*

$\boxed{n - m + f = 2}$

Preuve: La preuve se fait à l'aide d'une induction simple sur le nombre de sommets.

- Pour $n = 1$.
  Lorsque $m = 0$, il n'y a qu'une face et la relation est vérifiée. On procède alors par induction sur le nombre d'arêtes en remarquant qu'un tel graphe est constitué de m boucles sur l'unique sommet. Si on considère une boucle, d'apres le théorème précédent elle sépare le plan en deux. On peut donc enlever une boucle et cela revient à enlever exactement une face et donc cela marche par induction.
- Dans le cas où $n > 1$.
  Comme G est connexe il admet au moins une arête $e = [a, b]$. On contracte cette arête, cela nous donne un nouveau graphe G' ayant une arête et un sommet de moins. Cette opération ne peut ni supprimer ni créer de face donc on peut conclure en utilisant l'hypothèse d'induction.
  ■

Il existe une autre preuve directe, basée sur la construction d'un arbre recouvrant T de $\phi(G)$. À chaque arête du coarbre de $\phi(G)$ on associe l'arête correspondante du dual. Cela nous permet de construire un arbre recouvrant de $\phi(G)^d$. Donc $m = n - 1 + f - 1$. Cette très jolie preuve est tirée de [4].

En conséquence le nombre de faces de $\phi(G)$ ne dépend pas du plongement de ϕ (ne dépend que des nombres de sommets et d'arêtes).

On peut aussi remarquer que les faces finies de G supposé connexe constituent une base de l'espace des cycles et l'on a :

$f_0 = m - n + 1 = \nu(G)$ formule du nombre cyclomatique d'un graphe, i.e. la dimension de l'espace vectoriel des cycles de G, ce qui engendre une autre preuve de la relation d'Euler.

Cette égalité n'implique pas que pour tout cycle μ de G, il existe un plongement $\phi(G)$ dans lequel ce cyle μ soit une face (pour s'en convaincre, il suffit de considérer un cycle à 4 sommets muni de deux diagonales K_4).

Remarque : En fait la relation d'Euler est valable pour un plongement sur une surface orientable S de genre $genre(S)$:

$\boxed{n - m + f = 2 - 2genre(S)}$

Rappelons que le genre de la sphère est 0, surface équivalente au plan, donc aussi de genre 0.

Corollaire 2 *Un polyèdre convexe P de R^3 ayant n sommets, f faces et m arêtes vérifie :$n - m + f = 2$.*

En effet on peut représenter sans croisement sur la sphère un tel polyèdre. En prenant une face quelconque du polyèdre comme face infinie on peut obtenir une représentation plane de P sur lequel on applique la formule d'Euler.

Cette formule n'est plus valable si le polyèdre admet des trous, il y a des contrexemples.

6.2 Dualité

Définition 6 *A tout plongement planaire $\phi(G)$ d'un graphe $G = (X, E)$, on peut associer son dual, i.e. un autre graphe $\phi(G)^d$ dont les sommets sont les faces de $\phi(G)$ et l'adjacence entre faces est définie par les arêtes communes aux deux faces dans $\phi(G)$. À chaque arête commune aux deux faces, on associe une arête dans $\phi(G)^d$. Ainsi même si $\phi(G)$ est un graphe simple, $\phi(G)^d$ peut être un multigraphe.*

Une représentation planaire de $\phi(G)^d$ s'obtient aisément, il suffit de mettre un point au milieu de chaque face de $\phi(G)$ et de les relier à toutes les faces adjacentes.

Exemple célèbre : le diagramme de Voronoi [34] et la triangulation de Delaunay (Charles Eugène Delaunay, mathématicien Français 1816-1872), sont deux graphes planaires duaux définis à partir d'un ensemble de points.

Ainsi $\phi(G)^d$ admet f sommets et m arêtes (car une arête de $\phi(G)^d$ correspond exactement à une arête de $\phi(G)$).

Il est facile de vérifier que $(\phi(G)^d)^d$ est isomorphe à $\phi(G)$ si et seulement si G est connexe. En effet le passage au dual préserve la connexité, mais si un graphe admet plusieurs composantes connexes son plongement $\phi(G)$ aussi. Cependant $(\phi(G)^d)$ est connexe (grâce à la face infinie) et donc aussi $(\phi(G)^d)^d$ qui ne peut donc pas être isomorphe à $\phi(G)$.

Cependant il existe des graphes (par exemple un cyle avec deux arêtes pendantes) pour lesquels tous les plongements bien que combinatoirement isomorphes ne sont pas topologiquement équivalents (sur l'exemple du cyle, suivant que l'on mette les deux arêtes pendantes dans la même face ou non). Ces deux plongements même s'ils sont combinatoirement isomorphes, ne sont pas topologiquement équivalents.

Ainsi il existe des graphes planaires G possédant deux plongements $\phi(G)$ et $\phi'(G)$ tels que $\phi(G)^d$ et $\phi'(G)^d$ ne sont pas combinatoirement isomorphes.

En outre on peut montrer :

Un graphe n'admet qu'un plongement planaire (à un isomorphisme près préservant la topologie) s'il est 3-connexe. Dans un tel cas on peut vraiment parler du graphe dual G^d d'un graphe planaire G.

6.3 Cas particulier des graphes simples

On dit qu'un graphe G est *simple* s'il n'admet ni boucle ni arête multiple. Pour un graphe planaire être simple implique que pour tout plongement les faces ont au moins trois arêtes :

$\forall \alpha \in F$, $degre(\alpha) \geq 3$

d'où : $2m = \sum_{\alpha \in F} degre(\alpha) \geq 3f$,

et donc $\boxed{f \leq \frac{2m}{3}}$.

En reportant dans la formule d'Euler, on obtient :

$m = n - 2 + f \leq n - 2 + \frac{2m}{3}$ d'où : $\boxed{m \leq 3n - 6}$

De même on a $f \leq \frac{2m}{3} \leq 2/3(3n-6)$, c'est à dire :
$\boxed{f \leq 2n-4}$.
On peut retenir que pour les graphes planaires simples $m, f \in O(n)$.

6.4 Exercices

1. On considère un graphe planaire G et un de ses plongement $\phi(G)$, montrer que pour toute face F de $\phi(G)$, il existe un plongement $\psi(G)$ dans lequel F est la face infinie.
2. Montrer en utilisant la formule d'Euler que $K_{3,3}$ et K_5 ne sont pas planaires. Proposer une autre preuve, sans utiliser la formule d'Euler.
3. $K_{3,3}$ et K_5 sont-ils plongeables planairement sur un tore, un ruban de Moebius ?
4. Montrer qu'un graphe planaire simple admet au moins un sommet de degré inférieur ou égal à 5.

   En déduire qu'un ballon de football ne peut être uniquement réalisé par coutures de pièces de cuir hexagonales.
5. Déduire aussi de cette propriété un codage efficace en mémoire d'un graphe planaire. Préciser le nombre de bits nécessaires pour coder un graphe planaire ayant n sommets.
6. Montrer que toute triangulation d'un polygone à n côtés utilise exactement $n-3$ diagonales pour créer $n-2$ triangles.
7. Les solides platoniciens. Montrer qu'il n'existe que 5 solides ou polyèdres qui soient réguliers de degré k et dont toute face soit un polygone régulier à d côtés.

6.5 Graphes planaires maximaux

Nous appelerons graphe planaire maximal un graphe simple planaire G, tel que l'ajout d'une arête lui fasse perdre cette planarité.

Il est facile de vérifier que dans un tel graphe toute face est un triangle (même la face infinie) et le nombre d'arêtes vérifie $m = 3n-6$. En fait, nous avons les équivalences suivantes pour un graphe simple et planaire G :

G est maximal ssi toute face est un triangle ssi $m = 3n-6$.

De tels graphes sont presque composés de 3 arbres recouvrants (ce qui ferait 3n-3 arêtes) et il y a eu de nombreux travaux de recherche sur ce sujet.

[]

Bibliographie

[1] J. R. Abrial. *The B-book, assigning programs to meanings.* Cambridge University Press, 1996.

[2] M. Agrawal, N. Kayal, and N. Saxena. Primes is in p. 2002.

[3] A. V. Aho, I. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms.* Addison-Welsey, 1983.

[4] M. Aigner and G.M. Ziegler. *Proofs for the BOOK.* Springer-Verlag, 1999.

[5] J.D. Boissonnat and M. Yvinec. *Géométrie Algorithmique.* Ediscience International, 1995.

[6] P.B. Borwein. On the complexity of calculating factorials. *Journals of Algorithms*, 6 :376–380, 1985.

[7] Catalan. Notes sur une équation aux différences finies. *J. Math. Pures et Appliquées*, 1838.

[8] L. Comtet. *Analyse Combinatoire, volumes I et II.* Presses Universitaires de France, 1970.

[9] S.A. Cook and R.A. Reckhow. Time bounded random machines. *Journal of Computer Science and Systems*, pages 354–375, 1973.

[10] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms.* MIT Press. En français chez InterEditions.

[11] J.P. D'Angelo and D.B. West. *Mathematical Thinking : problem solving and proofs.* Prentice-Hall, 1997.

[12] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms.* Mc Graw Hill, 2006.

[13] M.P. Delest and G. Viennot. Algebraic languages and polyominoes enumeration. *Theoretical Computer Science*, 1984.

[14] R. Diestel. *Graph Theory*. Springer-verlag, 1997.

[15] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. on Information Theory*, 22(6) :644–654, 1976.

[16] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, (1) :269–271, 1959.

[17] I. Dutour. *Grammaires d'objets : énumération, bijections et génération aléatoire*. PhD thesis, université Bordeaux, 1996.

[18] P.C. Gilmore and R.E. Gomory. Sequencing a one-state-variable machine : a solvable case of the traveling salesman problem. *Operation Research*, pages 655–679, 1964.

[19] R.L. Graham and P. Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7(1) :43–57, 1985.

[20] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete mathematics, a foundation for computer science*. Addison -Welsey.

[21] D.E. Knuth. Big omicron and big omega and big theta. *Sigact news*, April–June :18–24, 1976.

[22] C. Kozen. *The design and analysis of algorithms*. Texts and monographs in computer science. Springer-Verlag, 1991.

[23] J.B. Kruskal. On the shortest spanning tree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.*, pages 48–50, 1956.

[24] B.M.E. Moret and H.D. Shapiro. *Algorithms from P to NP, volume I*. Benjamin Cummings Publishing Company, 1991.

[25] R. Motvani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[26] G. Pick. Geometrisches zur zahlenlehre. *Sitzungsber lotos, Prague*, 1900.

[27] R.C. Prim. Shortest connections networks and some generalizations. *Bell Syst. Tech. J.*, pages 1389–1401, 1957.

[28] R. Rivest, A. Shamir, and Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.

[29] J.E. Savage. *Models of Computation : exploring the power of computing*. Addison-Welsey, 1998.

[30] D. Shasha and C. Lazere. *Out of their minds*. Copernicus, 1995.

[31] I. Stewart. *Visions géométriques*. Belin, 1994.

[32] R.E. Tarjan. *Data structures and network algorithms*, volume 44. SIAM Monography, 1983.

[33] A. Troesch. Interpretation géométrique de l'algorithme d'euclide et reconnaissance de segments. *Theoretical Computer Science*, 1993.

[34] G. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques ; deuxième mémoire : Recherche sur les parallèloèdres primitifs. *J. Reine Angew. Math.*, pages 198–287, 1908.

[35] D. West. *Introduction to Graph Theory*. Prentice Hall, 1996.