INF 431

F. Morain

Graphes IV: topologie

28 mars 2007

Où en est-on ?

Amphi 1: introduction.

Amphi 2: génie logiciel avec Java.

Amphi 3: analyse lexicale.

Amphi 4: analyse syntaxique.

Amphi 5: graphes I.

Amphi 6: graphes II (parcours).

Amphi 7: graphes III (optimisation combinatoire).

Amphi 8: graphes IV (topologie).

Amphi 9: ouverture (année 3+4; trajectoires).

Plan

I. Rappels sur les parcours

On se donne un parcours L d'un graphe orienté $\mathcal{G} = (\mathcal{S}, \mathcal{A})$, ainsi que la forêt couvrante $\mathcal{F}$.

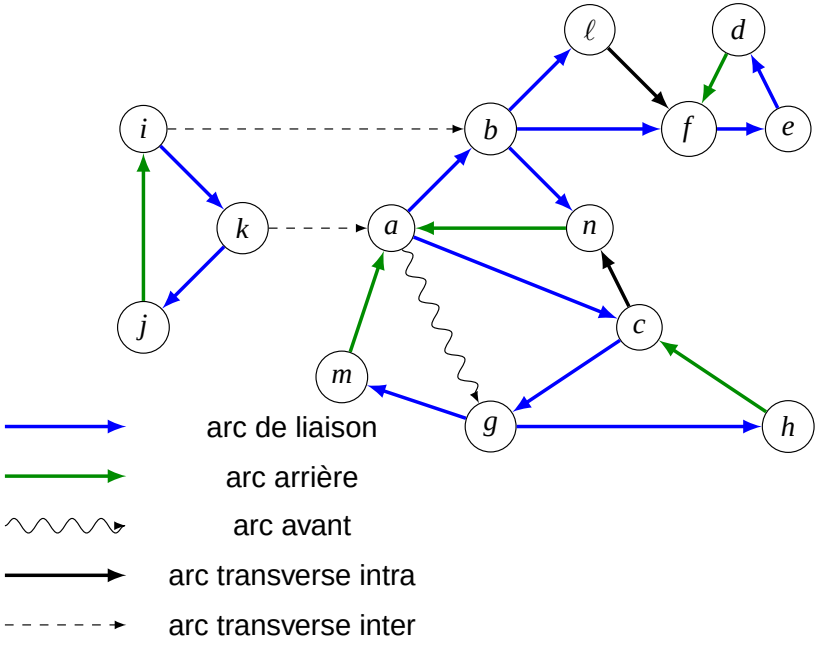
Chaque arbre de la forêt est une arborescence de Trémaux.

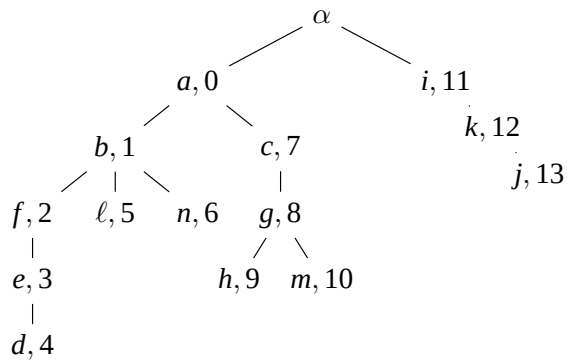
Prop. Les rangs des nœuds d'une arborescence de Trémaux forment un intervalle des entiers $[0, N]$. L'arborescence est préfixe, c'est-à-dire que pour tout sommet s, les rangs des descendants de s forment un sous-intervalle de $[0, N]$ dont le plus petit élément est $\mathrm{rang}(s)$.

Notations:
$s \to t$ signifie qu'il existe un chemin de s à t (dans $\mathcal{G}$).

$\mathcal{D}(s)$ est l'ensemble des descendants de s dans l'arborescence.

$L = (a, b, f, e, d, \ell, c, n, g, h, m, i, k, j)$





On a associé à chaque sommet son rang lors du parcours.

Identifi cation des arcs à l'aide du rang

Prop. Relativement à L (ou $\mathcal{F}$):

(i) Si t est un descendant (resp. ascendant) strict de s dans l'arborescence, alors $\text{rang}(s) < \text{rang}(t)$ (resp. $\text{rang}(s) > \text{rang}(t)$).

(ii) Si (s, t) est un arc arrière, alors $\text{rang}(t) < \text{rang}(s)$.

(iii) Si (s, t) est un arc avant, alors $\text{rang}(s) < \text{rang}(t)$.

(iv) Le sommet t appartient à l'arborescence de racine s si et seulement si $\text{rang}(s) \leqslant \text{rang}(t) \leqslant \text{rang}(s) + |\mathcal{D}(s)|$.

(v) Si (s, t) est un arc transverse, alors $\text{rang}(s) > \text{rang}(t)$. Autrement dit, s est visité après t.

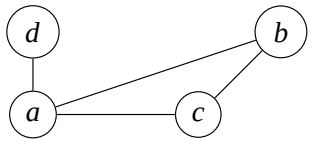
Identifi cation des arcs (suite)

```
dfsRec(etat, rang, rg, racine, s)
  // s est non exploré,
  // racine est la racine de l'arborescence courante
  etat[s] <- encours;
  rang[s] <- rg++;
  pour t voisin de s faire
    si etat[t] == inexploré alors
      rg <- dfsRec(etat, rang, rg, racine, t);
    sinon si etat[t] == encours alors
      écrire "(s, t) arrière";
    sinon // t est déjà exploré
      si rang[t] < rang[racine] alors
        écrire "(s, t) transverse inter-arbre";
      sinon si rang[s] < rang[t] alors
        écrire "(s, t) avant";
      sinon // rang[s] > rang[t]
        écrire "(s, t) transverse intra-arbre";
  etat[s] <- exploré;
  retourner rg.
```

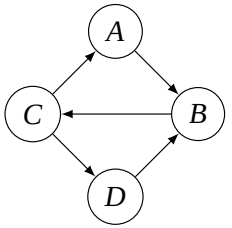
Un exemple animé

II. Composantes fortement connexes

$\mathcal{G}$ non orienté est connexe ssi $\forall (s,t) \in \mathcal{S}^2$, $s \to t$.



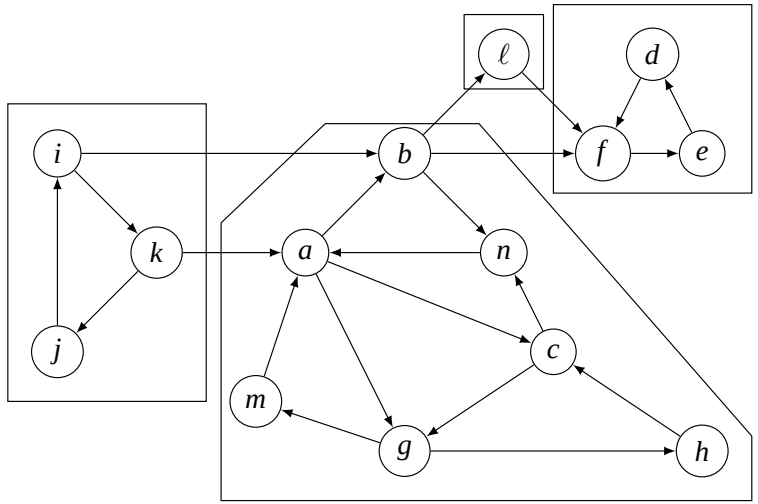
$\mathcal{G}$ orienté est fortement connexe ssi $\forall (s,t) \in \mathcal{S}^2$, $s \to t$ et $t \to s$.



Applications

- Dépendances entre fichiers pour la compilation ou le test, etc.
  $\Rightarrow$ prépare le terrain au tri topologique.
- Applications en combinatoire et optimisation: on découpe un problème en sous-problèmes.
- Représentation des relations d'ordre.
  Ex. $s \to t$ si s envoie du mail à t. Si $s \to t$ et $t \to s$, alors s et t sont des relations/copains/etc.

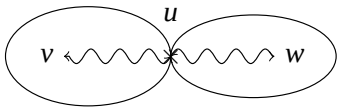
Composantes fortement connexes: exemple



Propriétés des composantes fortement connexes

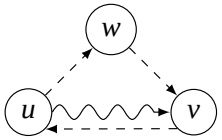
Prop. Tout $u \in \mathcal{S}$ appartient à une unique composante fortement connexe (cfc), que l'on notera $\mathcal{C}(u)$.

Dém.



Prop 1. Si $v \in \mathcal{C}(u)$ et $u \to w \to v$, alors $w \in \mathcal{C}(u)$.

Dém.



Utilisations de la dfs

On fixe un parcours dfs L, équipé de sa fonction rang**.**

Prop. Si x est la racine d'une arborescence de Trémaux de L, alors $\mathcal{C}(x) \subset \mathcal{D}(x)$.

Rem. La réciproque n'est pas vraie.

Algorithme naïf: on part de $x \in \mathcal{S}$ et on construit $\mathcal{D}(x)$. Pour $y \in \mathcal{D}(x)$, on fait une dfs à partir de y dans le graphe renversé et on regarde si on atteint x dans l'autre sens.

Algorithme de Tarjan: un parcours suffit $\Rightarrow$ complexité $O(|\mathcal{S}| + |\mathcal{A}|)$.

CFC et arborescence

Prop. Soit (Y, T) une arborescence de Trémaux de racine x. Soit $u \in Y$ et u_0 le sommet de plus petit rang dans $\mathcal{C}(u)$.

(i) pour tout $v \in \mathcal{C}(u)$, tous les sommets du chemin de (Y, T) joignant u_0 à v sont dans $\mathcal{C}(u)$;

(ii) $\mathcal{C}(u) \subset \mathcal{D}(u_0)$.

Dém. (i) Prop 1.

(ii) Par l'absurde: $v \in \mathcal{C}(u)$ et $c = (u_0, \ldots, w', w, \ldots, v)$ avec w le premier sommet $\notin \mathcal{D}(u_0)$.
$(w', w) \notin T$ et n'est pas un arc avant (sinon on pourrait le remplacer par une suite d'arcs de T).
$\Rightarrow (w', w)$ est arrière ou transverse: $\text{rang}(w) < \text{rang}(w')$.
D'après (i), $w \in \mathcal{C}(u)$ et par minimalité: $\text{rang}(u_0) < \text{rang}(w)$.
T préfixe $\Rightarrow w \in \mathcal{D}(u_0)$: contradiction. $\square$

Points d'attache

Remarque préliminaire: les arcs avant ne servent à rien dans la recherche, ils peuvent être remplacés par des arcs de liaison.

Déf. Soit $x \in \mathcal{G}$ racine de l'arborescence de Trémaux (Y, T). Le point d'attache $at(y)$ de $y \in Y$ est le sommet de plus petit rang extrémité d'un chemin de $\mathcal{G}$, d'origine y et contenant au plus un arc (u, v) tel que $\text{rang}(u) > \text{rang}(v)$ (i.e., un arc arrière ou un arc transverse intra arbre).

On suppose que le chemin vide d'origine et extrémité y est un tel chemin et donc

$$\text{rang}(at(y)) \leqslant \text{rang}(y).$$

Le rang du point d'attache est appelé rang d'attache.

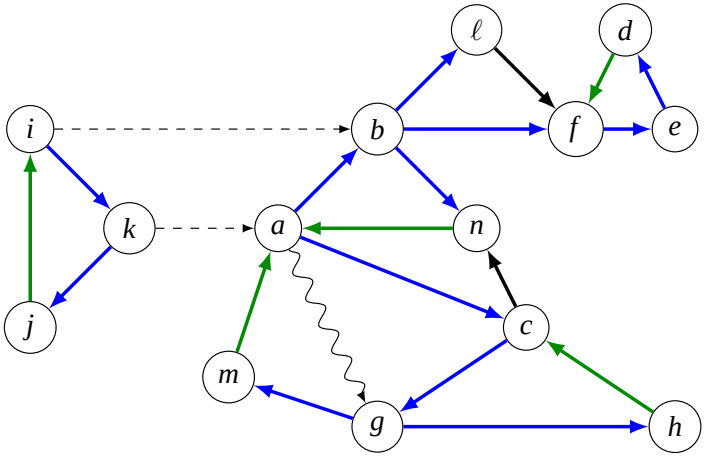
Un chemin de y à son point d'attache dans $\mathcal{G}$ est soit vide (et $at(y) = y$), ou bien une suite d'arcs **dans** T suivis par un arc arrière ou un arc transverse intra-arbre.

Proposition

Prop. Le point d'attache $at(y)$ est le sommet de plus petit rang parmi les sommets suivants:

- y lui-même;
- les points d'attache des fils de y dans (Y,T);
- les extrémités des arcs transverses (intra arbres) ou arrière d'origine y.

$L = (a,b,f,e,d,\ell,c,n,g,h,m,i,k,j)$



s	a	b	c	d	e	f	g	h	i	j	k	ℓ	m	n
rang(s)	0	1	7	4	3	2	8	9	11	13	12	5	10	6
rat(s)	0	0	0	2	2	2	0	7	11	11	11	2	0	0

Application à la forte connexité

Théorème fondamental. Si $u \in Y$ satisfait:
(i) $u = at(u)$;
(ii) Pour tout descendant $v \in \mathcal{D}(u)$, $\text{rang}(at(v)) < \text{rang}(v)$.

Alors $\mathcal{D}(u) = \mathcal{C}(u)$.

Le sommet u est appelé point d'entrée du parcours dans la composante fortement connexe.

Démonstration

a) $\mathcal{D}(u) \subset \mathcal{C}(u)$.

Par l'absurde: soit $v \in \mathcal{D}(u)$ le sommet de plus petit rang pour lequel on ne puisse pas rejoindre u.
$c_1 = v \to at(v)$
$c_2 = u \to v$
$c_3 = c_2 \cup c_1 = (u, \ldots, v, \ldots, at(v))$
Par définition de $at(v)$, c_3 contient au plus un arc arrière ou transverse:

$$\text{rang}(u) = \text{rang}(at(u)) \leqslant \text{rang}(at(v)) < \text{rang}(v).$$

(Y,T) est préfixe $\Rightarrow at(v) \in \mathcal{D}(u)$.
Minimalité de $v \Rightarrow$ il existe $c_4 = at(v) \to u$ dans $(Y,T) \Rightarrow c_1 \cup c_4$ est un chemin de v à u, contradiction.

b) $\mathcal{C}(u) \subset \mathcal{D}(u)$.

Lemme. Tout arc dont l'origine est dans $\mathcal{D}(u)$ a aussi son extrémité dans $\mathcal{D}(u)$.

Dém. Soit $(v_1, v_2) \in \mathcal{A}$ tel que $v_1 \in \mathcal{D}(u)$.

Si $\text{rang}(v_2) > \text{rang}(v_1)$, $v_2 \in \mathcal{D}(v_1) \subset \mathcal{D}(u)$.

Si $\text{rang}(v_2) < \text{rang}(v_1)$, le chemin de u à v_2 contient exactement un arc arrière ou transverse, donc

$$\text{rang}(u) = \text{rang}(at(u)) \leqslant \text{rang}(v_2) < \text{rang}(v_1),$$

donc $v_2 \in \mathcal{D}(u)$ par préfixité de (Y, T). □

Principes de l'algorithme de Tarjan

- On fait une seule dfs en calculant le rang et les composantes fortement connexes.
- Deux types d'états pour un sommet:
  - Les états classiques: inexploré, encours, exploré.
  - Trois états cfc: libre, empilé (on cherche $\mathcal{C}(s)$), exclus (on a trouvé $\mathcal{C}(s)$).

  On a les propriétés:
  $libre \Leftrightarrow inexplore, encours \Rightarrow empile, exclus \Rightarrow explore$.

Principes (suite)

- On parcourt une arborescence jusqu'à trouver un point d'entrée. Une fois celui-ci trouvé, on en déduit la composante fortement connexe, qu'on retire du graphe, et on continue.
- On utilise une pile: pour stocker les sommets empilés. Le point d'attache d'un sommet du graphe courant se trouve dans cette pile.
  Subtilité: le rang d'attache qu'on calcule dépend du graphe courant. *Ce n'est pas le rang d'attache du graphe original.*
  Regarder le dessin en mouvement au moment du traitement du sommet ℓ.

Pseudocode

```
cfc(G)
1. etat <- tableau de taille n; // classique
   etat_cfc <- tableau de taille n;
   rang <- tableau de taille n;
   rat <- tableau de taille n; // rang d'attache
2. pourtout sommet s faire
     etat[s] <- inexploré;
     etat_cfc[s] <- libre;
3. rg <- 0;
4. pile <- NIL;
5. tantqu'il reste un sommet s inexploré faire
     rg <- dfsCfc(rg, s, s);
```

Rem. Pour alléger, les tableaux sont "globaux".

```
dfsCfc(rg, racine, s)
  rang[s] <- rg++; etat[s] <- encours;
  pile <- s # pile; etat_cfc[s] <- empilé;
  rat[s] <- rang[s];
  pour t voisin de s faire
    si etat[t] == inexploré alors // t ∈ D(s)
      rg <- dfsCfc(rg, racine, t);
      rat[s] <- min(rat[s], rat[t]);
    sinon
      si etat_cfc[t] == empilé alors
        si rang[t] > rang[s] alors
          // (s, t) arc avant
        sinon // (s, t) arriere ou intra-arbre
          si etat[t] == encours alors
            // (s, t) est arriere
          sinon
            // (s, t) est intra-arbre
          // t est dans C(s) car t ∈ D(s)
          // et t est successeur de s
          rat[s] <- min(rat[s], rang[t]); // [1]
      sinon // t est exclus
          [...]
  etat[s] <- exploré;
```

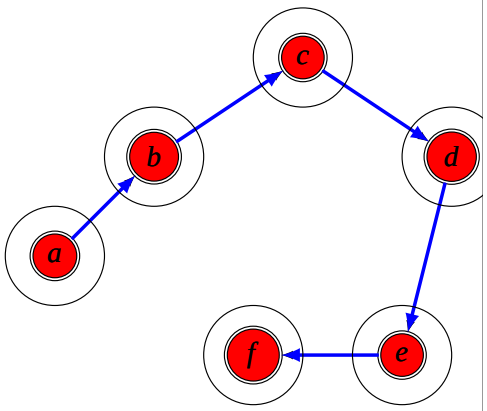


```
  si rat[s] == rang[s] alors
    // s est point d'entrée, C(s) est
    // dans la pile jusqu'à s
    dépiler(etat_cfc, pile, s);
  retourner rg.

dépiler(etat_cfc, pile, s)
    répéter
      t <- tête(pile);
      etat_cfc[t] <- exclus;
      écrire t;
    jusqu'à ce que t == s;
```

Rem. Dans la vraie vie, on n'a pas vraiment besoin de l'état classique, mais ça aide à comprendre ce qui se passe.

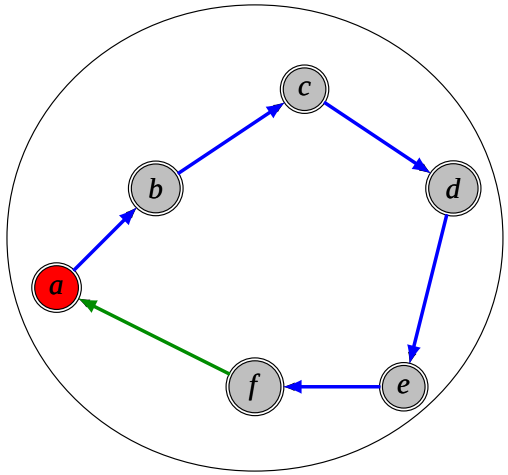
Exemple 1



*f*f
*eeeee*e
*ddddddddd*d
*ccccccccccccc*c
*bbbbbbbbbbbbbbbbb*b
*aaaaaaaaaaaaaaaaaaaa*a

s	a	b	c	d	e	f
rang(s)	0	1	2	3	4	5
rat(s)	0	1	2	3	4	5

Exemple 2



*ffffff*f
*eeeeeeee*e
*ddddddddddd*d
*ccccccccccccc*c
*bbbbbbbbbbbbbbb*b
*aaaaaaaaaaaaaaaa*a

s	a	b	c	d	e	f
rang(s)	0	1	2	3	4	5
rat(s)	0	10	20	30	40	50

Quelques exemples animés

Preuve de la correction de l'algorithme (esquisse)

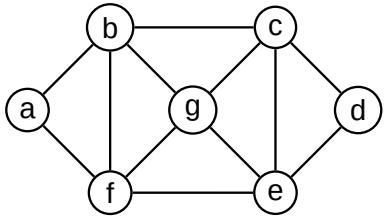
On raisonne par récurrence sur le nombre de composantes fortement connexes: quand on trouve la première composante, on a calculé les rangs d'attache et on applique le théorème fondamental.

Quand le graphe ne contient qu'une composante, le point d'entrée est le sommet de rang minimum r.

Si on passe dans [1], alors $rat(s) < \text{rang}(s)$: ou bien c'était déjà le cas, ou de toute façon $\text{rang}(t) < \text{rang}(s)$. Et donc s ne sera pas point d'entrée.

Rem. Il existe un chemin élémentaire $(s_1 = r, s_2, \ldots, s_k = r)$ tel que $rat(s_i) = rat(r)$.

III. Euler et Hamilton



Circuit Eulérien: on passe par toutes les arêtes une fois et seule (mais peut-être plusieurs fois par le même sommet).

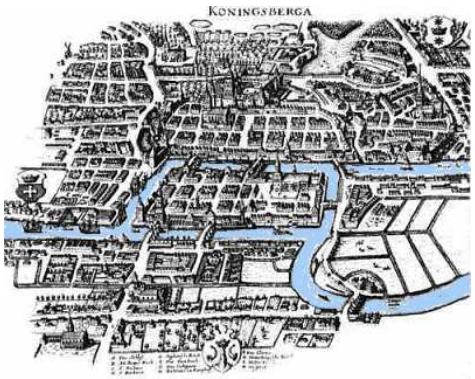
Ex. `abcdefbgcegfa` ou **`afgcdegbcefba`**.

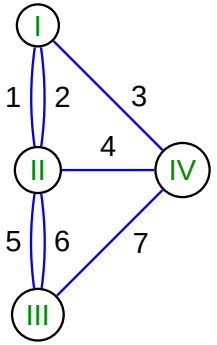
Circuit Hamiltonien: on passe par tous les sommets une fois et seule.

Ex. `abcdegfa` ou **`afedcgba`**.

A) L'acte fondateur de la théorie

Ponts de Königsberg (Kaliningrad)





Rappel. Le degré d'un sommet est le nombre de ses voisins.

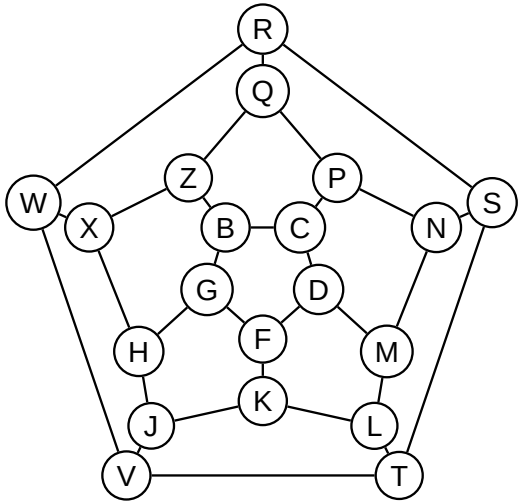
Prop. (hand-shaking lemma) $\sum_s \deg(s) = 2m$.
Dém. Chaque arête contribue pour deux sommets dans la somme. □

Th. (Euler, 1736) Un graphe $\mathcal{G}$ possède un circuit Eulérien si et seulement si chaque sommet est de degré pair.

Dém. cf. poly. □

Coro. Il n'existe pas de chemin passant une fois et une seule par les 7 ponts de Königsberg.

B) Hamilton (1805–1865)



BCPNMDFKLTSRQZXWVJHGB
BCPNMDFGHXWVJKLTSRQZB

Rem. Pas de critère simple d'existence!

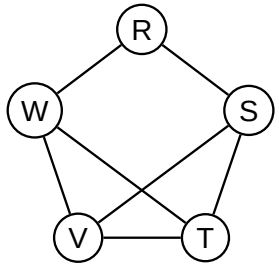
Ex. circuit hamiltonien pour K_n: $0, 1, 2, \ldots, n-1, 0$.

Th. (Ore, 1960) Si $n \geqslant 3$, et

$$\deg(v) + \deg(w) \geqslant n$$

pour tout $(v, w) \notin A$, alors $\mathcal{G}$ est Hamiltonien.

Ex.

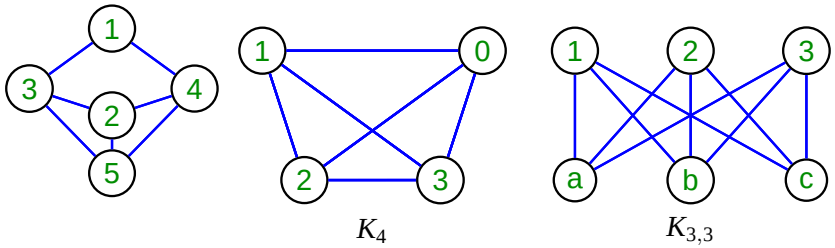


IV. Planarité

Déf. Un graphe est planaire si on peut le dessiner sans que les arcs se croisent.

Hyp. $\mathcal{G}$ est simple (pas d'arêtes multiples, pas de boucles).

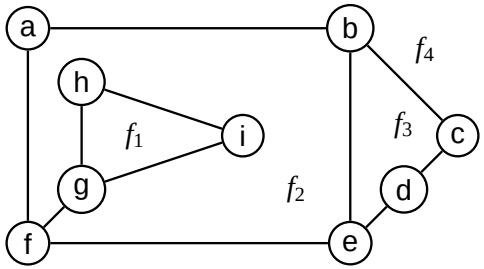
Ex.



Applications : dessin de circuits imprimés, cablage, etc.

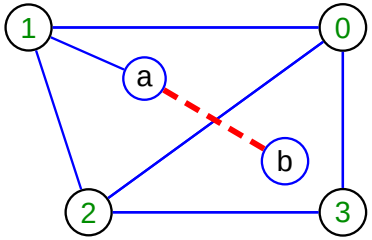
Faces

Soit $\mathcal{G}$ un (dessin d'un) graphe ayant n sommets et m arêtes. Les arêtes de $\mathcal{G}$ découpent l'espace en régions ou faces.
Déf. Le degré d'une face est le nombre d'arêtes rencontrées dans un chemin qui suit la frontière de f.
Ex. Dans un arbre à n sommets, la face infinie a degré $2n-2$.



$\deg(f_1) = 3$, $\deg(f_2) = 9$, $\deg(f_3) = 4$, $\deg(f_4) = 6$.

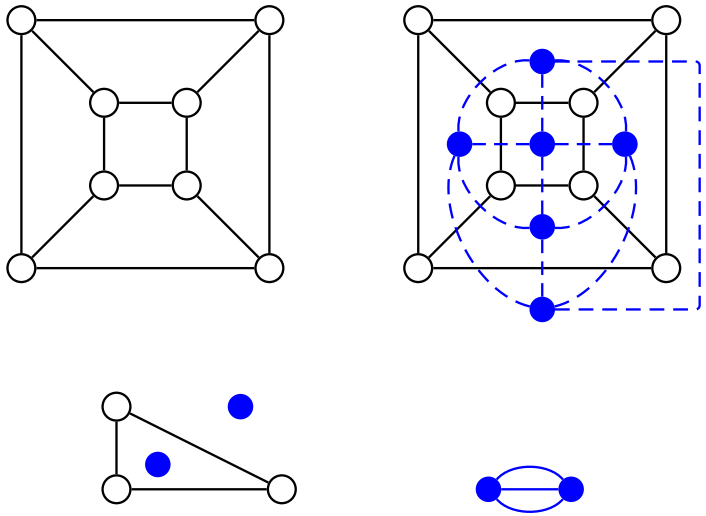
Le théorème de Jordan



Th. (très informel) Pour passer de a à b, on doit nécessairement couper une autre arête du graphe.

Graphe dual

Déf. $\mathcal{G}^*$ est obtenu à partir de $\mathcal{G}$ en créant pour toute face de $\mathcal{G}$, un sommet de $\mathcal{G}^*$; si deux faces sont contiguës, on crée une arête entre les deux sommets de $\mathcal{G}^*$.



Prop. Si $\mathcal{G}$ a n sommets, m arêtes, f faces, alors $\mathcal{G}^*$ a f sommets, m arêtes, n faces.

Prop. $\mathcal{G}$ est planaire ssi $\mathcal{G}^*$ est planaire.

Prop. $(\mathcal{G}^*)^*$ est isomorphe à $\mathcal{G}$.

Th. Si $\mathcal{G}$ est planaire, alors pour tout dessin planaire de $\mathcal{G}$, on a $\sum_f \deg(f) = 2m$.

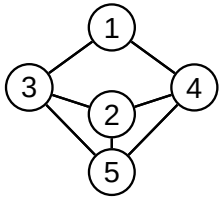
Dém. on applique le hand-shaking lemma au graphe dual. □

Relation d'Euler

Th. On note f le nombre de régions de $\mathcal{G}$. Alors

$$f = m - n + 2.$$

Ex.

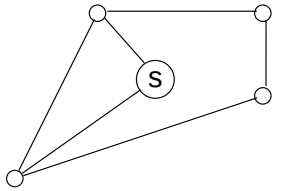


$n = 5, m = 7, f = 4 = 7 - 5 + 2$

Rem. Le résultat est vrai pour les arbres, puisque $m = n - 1$, et $f = 1 = (n-1) - n + 2$ (une seule face infinie).

Dém. On va raisonner par récurrence à m ou n fixé.
$H_{m,n}$: la relation d'Euler est pour tous $(i, j), i \leqslant m$ et $j \leqslant n$.
$H_{3,3}$ est vraie:

Récurrence sur n, avec m fixé: Soit $\mathcal{G}$ un graphe connexe planaire avec m arêtes et $n+1$ sommets. Plusieurs cas se présentent :
1. Le graphe $\mathcal{G}$ possède un sommet intérieur s de degré $\nu \geqslant 1$

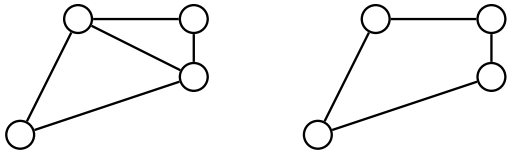


Si on enlève ce sommet et ces ν arcs ($\nu \geqslant 1$), le graphe restant $\mathcal{G}'$ est toujours connexe planaire et

$$m' = m - \nu, n' = (n+1) - 1, f' = f - (\nu - 1)$$

et donc : $f - m + (n+1) = f' - m' + n' = 2$ par $H_{n,m}$.

2. Le graphe $\mathcal{G}$ n'a pas de point intérieur. Supposons que $\mathcal{G}$ contienne une corde. En l'enlevant, on crée un graphe $\mathcal{G}'$, qui est toujours connexe planaire et qui a une arête de moins, mais également une face de moins :



$$m' = m - 1, n' = n + 1, f' = f - 1.$$

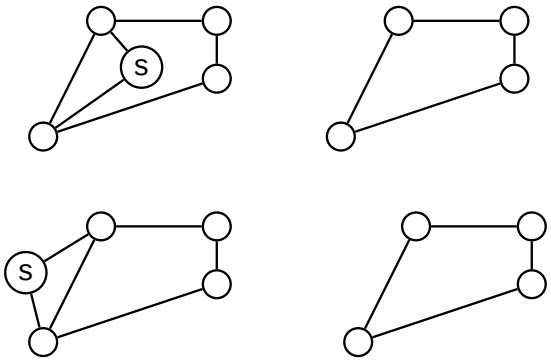
On itère ce raisonnement tant qu'il reste une corde dans $\mathcal{G}$. On s'est donc ramené au cas où $\mathcal{G}$ ne contient que des points sur le périmètre (et donc de chaque sommet partent deux arêtes). Dans ce cas, on enlève un point quelconque en fusionnant les deux arêtes, ce qui donne $\mathcal{G}'$ avec

$$m' = m - 1, n' = n + 1 - 1 = n, f' = f$$

et on utilise $H_{n,m}$.

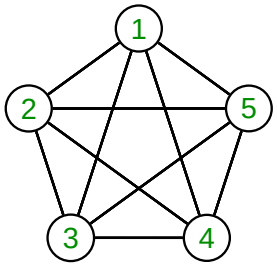
Récurrence sur m, avec n fixé: $\mathcal{G}$ avec $m+1$ arêtes et n sommets. Soit s un sommet quelconque et $\nu \geqslant 1$ son degré. Le fait de l'enlever donne un nouveau graphe $\mathcal{G}'$ qui est connexe planaire pour lequel :

$$m' = m - \nu, n' = n - 1, f' = f - (\nu - 1).$$



Applications

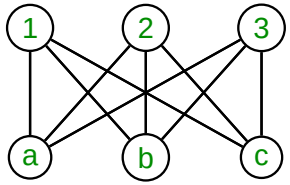
Coro. Si $\mathcal{G}$ est planaire avec $n \geqslant 3$, alors $m \leqslant 3n - 6$.
Dém. Le degré de chaque face est au moins 3, d'où $3f \leqslant 2m$ et on utilise Euler. □
Coro. K_5 n'est pas planaire.
Dém.



K_5

$n = 5$, $m = 10$, mais $10 > 3 \times 5 - 6$. □

Coro. Si $\mathcal{G}$ a $n \geqslant 3$ et n'a pas de triangle, alors $m \leqslant 2n - 4$.
Dém. chaque face a degré au moins 4, donc $4f \leqslant 2m$ d'où le résultat. □
Coro. $K_{3,3}$ n'est pas planaire.



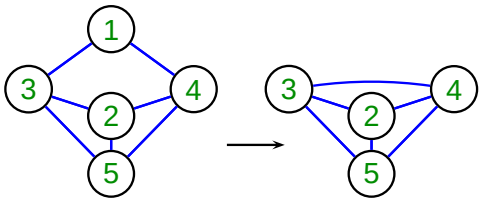
$K_{3,3}$

$9 > (2 \times 6) - 4$. □

Un critère théorique de planarité

Déf. Deux graphes sont **homéomorphes** ssi on peut passer de l'un à l'autre par fusion ou scission d'arcs passant par un nœud de degré 2.

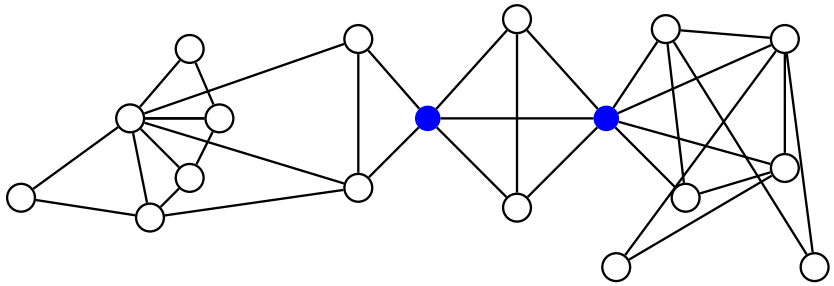
Ex.

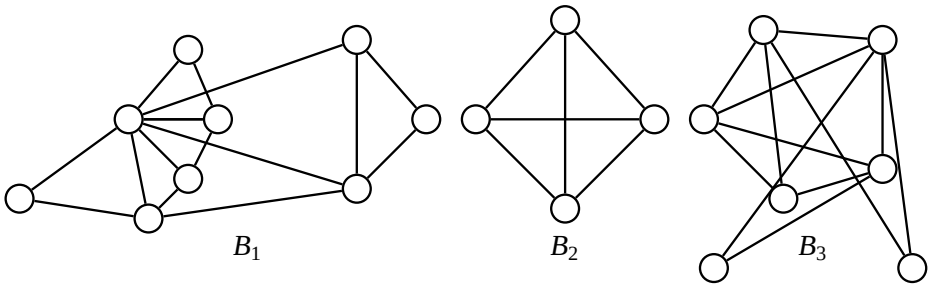


Th. (Kuratowski, 1930) Un graphe $\mathcal{G}$ est planaire ssi il ne contient aucun sous-graphe homéomorphe à K_5 ou $K_{3,3}$.

Vers un algorithme de test de planarité

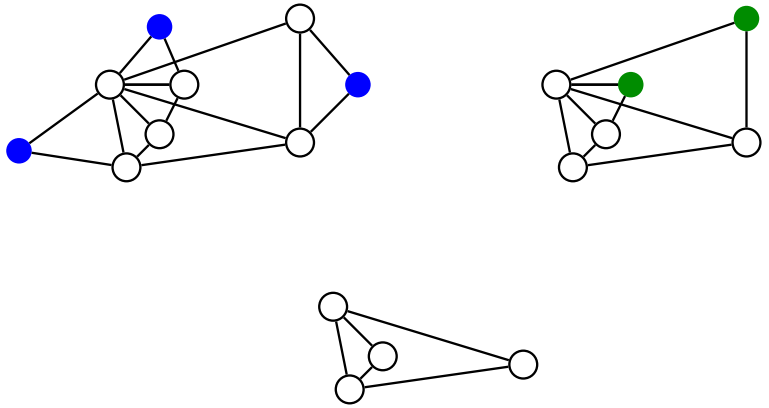
On coupe le graphe en blocs à partir des points d'articulation:



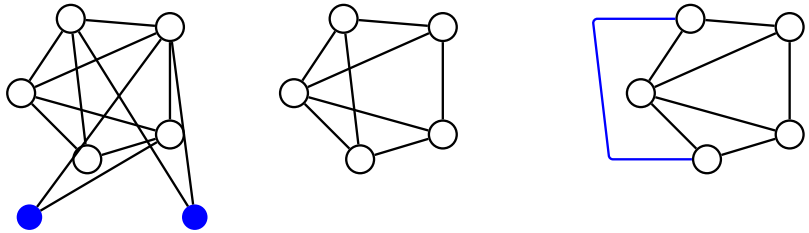

B_1 B_2 B_3

Le cas de B_2: planaire par inspection (ou en remarquant que tout graphe avec $n < 5$ est planaire).

Le cas de B_1:



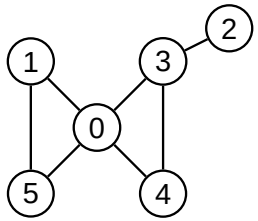
Le cas de B_3:



On peut aussi montrer que $m < 9$ implique $\mathcal{G}$ planaire.

Parenthèse: les points d'articulation

Déf. Un point d'articulation est un sommet dont la disparition supprime la connexité de $\mathcal{G}$.



0 et 3 sont des points d'articulations

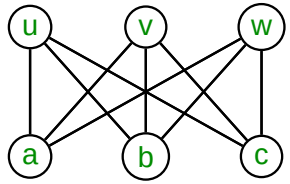
Solution brutale: pour chaque sommet s, on considère le graphe $\mathcal{G}' = (\mathcal{S}', \mathcal{A}')$ avec $\mathcal{S}' = \mathcal{S} - \{s\}$ et $\mathcal{A}' = \{(u, v) \in \mathcal{A}, u \neq s, v \neq s\}$; si $\mathcal{G}'$ n'est pas connexe, s est un point d'articulation. D'où un coût en $O(|\mathcal{S}||\mathcal{A}|)$.

On peut faire mieux avec un algorithme très proche de l'algorithme de recherche de composantes fortement connexes (cf. poly), d'où $O(|\mathcal{S}| + |\mathcal{A}|)$.

L'idée du test de planarité

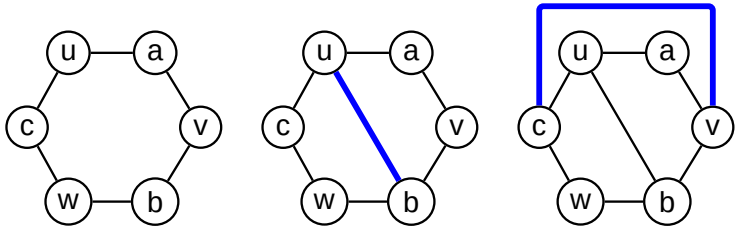
Hyp. Soit $\mathcal{G}$ un graphe hamiltonien: ou bien c'est un arbre (donc planaire), ou bien les sommets peuvent être mis sur un polygone fermé (un cercle).

L'idée est de répartir les arêtes à l'extérieur ou à l'intérieur du cycle. Si on y arrive sans que les arêtes ne se coupent, $\mathcal{G}$ est planaire, sinon, il ne l'est pas.
De nouveau, l'exemple de $K_{3,3}$.



$K_{3,3}$

On sélectionne le circuit hamiltonien **uavbwc**. S'il existe un dessin planaire, alors ce circuit formera nécessairement un polygone:

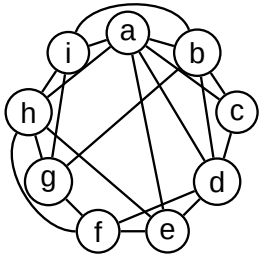


Arêtes non utilisées: **ub**, **vc**, **wa**.

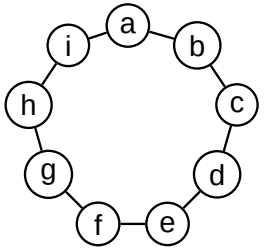
Impossible de tracer **wa** sans couper de fil.

Idem si on permute, etc.

Deuxième exemple

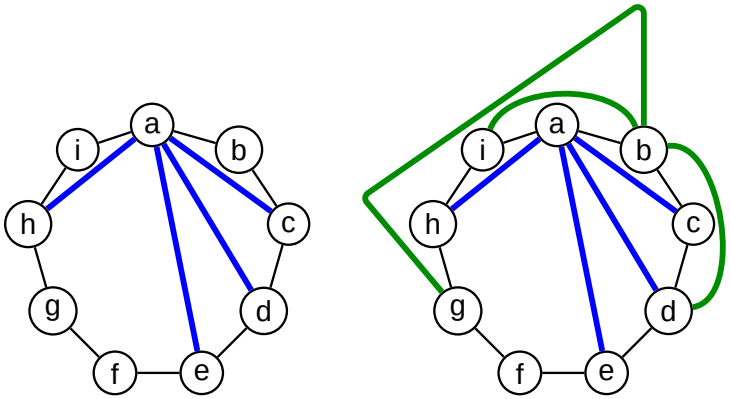


On choisit

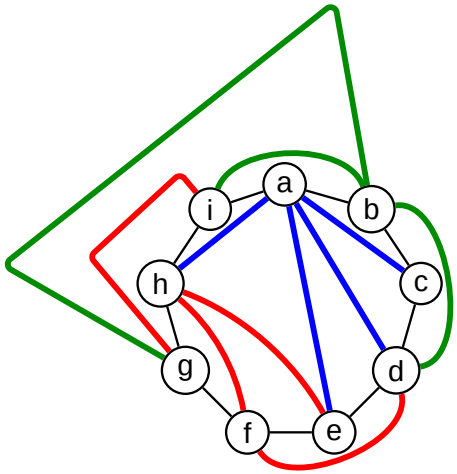


Il reste: **ac, ad, ae, ah, bd, bg, bi, df, eh, fh, gi**.

On place d'abord **ac, ad, ae, ah** à l'intérieur.
On place ensuite **bd, bg, bi** à l'extérieur.



Finalement, on place **df, eh, fh, gi**:



Mieux: algorithme de Tarjan en $O(n)$.

Conclusions (provisoires)

Théorie des graphes:

- On a vu quelques problèmes, dont les solutions sont trouvables en temps polynomial.
- Ce n'est pas toujours le cas (voyageur de commerce): la plupart des problèmes concernant les graphes sont difficiles (au sens de la théorie de la complexité). Cf. Majeure 2.

Prochains rendez-vous: PC cet après-midi; le flambeau passe à J.-M. Steyaert.