Master Sciences, Technologies, Santé
Mention Mathématiques, spécialité Enseignement des mathématiques

N1MA0011

Algorithmique et graphes, thèmes du second degré

ÉLÉMENTS DE THÉORIE DES GRAPHES

Éric SOPENA
Eric.Sopena@labri.fr

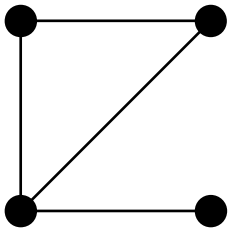
SOMMAIRE

Chapitre 1. Introduction, définitions

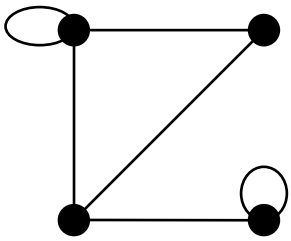
Nous donnons dans ce premier chapitre les principales définitions concernant les graphes, orientés ou non orientés, ainsi que quelques propriétés élémentaires. Nous verrons également de quelle façon on peut représenter un graphe afin de pouvoir écrire des algorithmes résolvant des problèmes de graphes.

1.1. Graphes non orientés

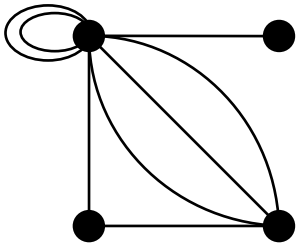
Un *graphe non orienté* G, est la donnée d'un couple (V(G),E(G)) où V(G) est un ensemble (fini) de *sommets* et E(G) un ensemble (fini) d'*arêtes*, chaque arête ayant deux sommets pour *extrémités*. Une arête est une *boucle* si elle relie un sommet à lui-même. On parle de *multigraphe* lorsque plusieurs arêtes relient les mêmes paires de sommets. Dans le cas contraire, on dit que le graphe est *simple*.



un graphe simple sans boucles



un graphe simple ayant deux boucles

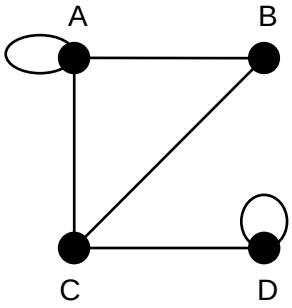


un multigraphe

Un graphe simple correspond ainsi au graphe d'une relation *symétrique* entre des objets, représentés par les sommets.

Deux sommets reliés par une arête sont dits *adjacents* (on dit également qu'ils sont *voisins*). Une arête reliant deux sommets est dite *incidente* à ces deux sommets.

Le *degré* d'un sommet est le nombre d'arêtes incidentes à ce sommet, les boucles comptant pour deux.



degré de A : 4

degré de B : 2

degré de C : 3

degré de D : 3

Tout graphe vérifie la propriété élémentaire suivante (qui se montre aisément par récurrence) :

> **Propriété (lemme des poignées de mains).** La somme des degrés des sommets d'un graphe vaut deux fois le nombre de ses arêtes.

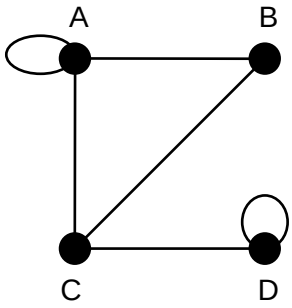
Sur l'exemple précédent, on vérifie en effet que 4 + 2 + 3 + 3 = 12 = 2 x 6.

Un *chemin de longueur k* est une suite de k+1 sommets u_0, …, u_k telle que deux sommets consécutifs sont reliés par une arête. On dit qu'un tel chemin *relie* le sommet u_0 au sommet u_k. Un chemin est *simple* s'il n'emprunte pas deux fois la même arête, il est *élémentaire* s'il ne passe pas deux fois par le même sommet. Un *cycle* est un chemin reliant un sommet à lui-même. Dans le graphe ci-dessus, DCABCA est un chemin (ni simple, ni élémentaire), ABCD est un chemin (simple et élémentaire), ABCA est un cycle.

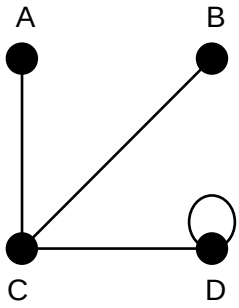
La *distance* entre deux sommets u et v est la longueur d'un plus court chemin reliant u à v (cette distance est infinie si aucun chemin ne relie ces deux sommets). Le *diamètre* d'un graphe est le

maximum des distances entre ses sommets. Le graphe précédent a ainsi pour diamètre 2 (qui correspond aux distances entre les sommets D et A, ou D et B).

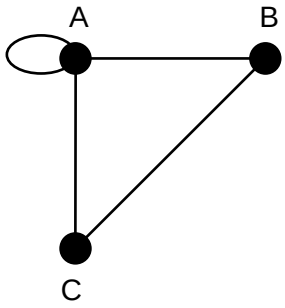
Un *sous-graphe partiel* G' d'un graphe G est un graphe vérifiant $V(G') \subseteq V(G)$ et $E(G') \subseteq E(G)$. Soit X un sous-ensemble de V(G). Le sous-graphe de G *induit par X*, noté G[X], est le graphe donné par $V(G[X]) = X$ et $E(G') = E(G) \cap X \times X$ (on garde ainsi toutes les arêtes reliant des sommets de X). Un *sous-graphe induit* d'un graphe G est un graphe G' tel qu'il existe un sous-ensemble X de V(G) vérifiant G' = G[X].



un graphe G

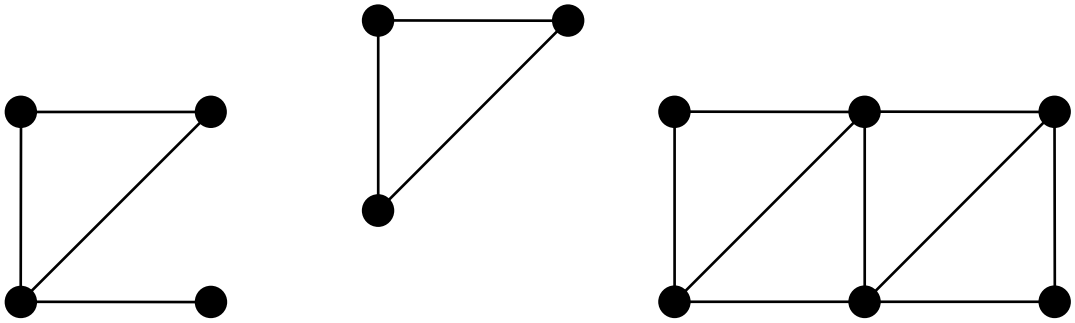


un sous-graphe partiel de G



un sous-graphe induit de G (par X = {A, B, C})

Un graphe est *connexe* si toute paire de sommets est reliée par un chemin. Un graphe est donc connexe si et seulement si son diamètre est fini. Une *composante connexe* d'un graphe est un sous-graphe induit connexe maximal.



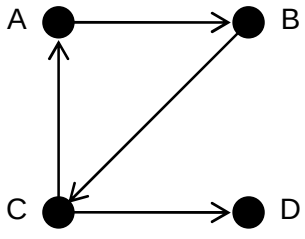
Un graphe non connexe ayant trois composantes connexes

1.2. Graphes orientés

Un *graphe orienté* G, est la donnée d'un couple (V(G),A(G)) où V(G) est un ensemble (fini) de *sommets* et A(G) un ensemble (fini) d'*arcs*, chaque arc ayant un sommet pour *extrémité initiale* et un sommet pour *extrémité finale*. Un arc est une *boucle* si il a le même sommet pour extrémités.

Un graphe orienté correspond ainsi au graphe d'une relation *non nécessairement symétrique* entre des objets, représentés par les sommets.

Deux sommets reliés par un arc sont dits *adjacents* (on dit également qu'ils sont *voisins*). Si deux sommets u et v sont reliés par un arc (u,v), v est un *successeur* de u et u est un *prédécesseur* de v.



un graphe orienté sans boucles

Le *degré sortant* d'un sommet est le nombre de successeurs de ce sommet. Le *degré entrant* d'un sommet est le nombre de prédécesseurs de ce sommet. Dans l'exemple précédent, le sommet C a pour degré entrant 1 et pour degré sortant 2.

Tout graphe orienté vérifie la propriété élémentaire suivante (qui se montre aisément par récurrence) :

> **Propriété (lemme des coups de pied).** La somme des degrés sortants des sommets d'un graphe est égale à la somme des degrés entrants de ces sommets et au nombre d'arcs du graphe.

Sur l'exemple précédent, on vérifie en effet que 1 + 1 + 2 + 0 = 1 + 1 + 1 + 1 = 4.

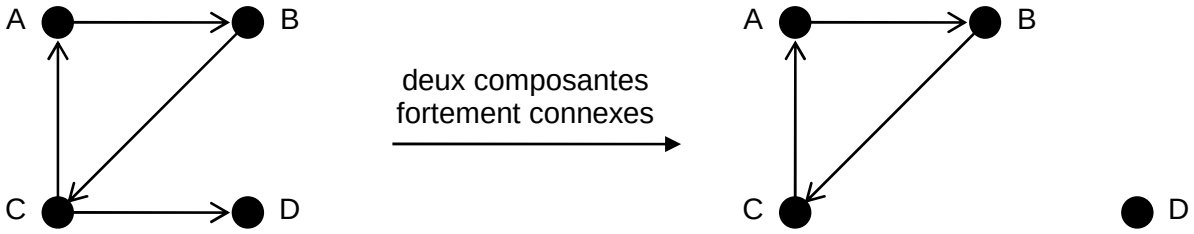
Un *chemin de longueur k* est une suite de k+1 sommets u_0, …, u_k telle que pour tout i, $0 \le i \le k$, (u_i,u_{i+1}) est un arc. Une *chaîne de longueur k* est une suite de k+1 sommets u_0, …, u_k telle que pour tout i, $0 \le i \le k$, (u_i,u_{i+1}) ou (u_{i+1},u_i) est un arc (dans le cas d'une chaîne, le parcours se fait donc indépendamment de la direction des arcs). On dit qu'un tel chemin (ou une telle chaîne) *relie* le sommet u_0 au sommet u_k. Un chemin est *simple* s'il n'emprunte pas deux fois le même arc, il est *élémentaire* s'il ne passe pas deux fois par le même sommet. Un *circuit* est un chemin reliant un sommet à lui-même. Dans le graphe précédent, ABCD est un chemin (simple et élémentaire), ABCA est un circuit.

La *distance* entre deux sommets u et v est la longueur d'un plus court chemin reliant u à v (cette distance est infinie si aucun chemin ne relie ces deux sommets). Le *diamètre* d'un graphe orienté est le maximum des distances entre ses sommets. Le graphe précédent a ainsi un diamètre infini, car il n'existe aucun chemin reliant le sommet D aux autres sommets.

Les notions de sous-graphe partiel et de sous-graphe induit se définissent comme dans le cas non orienté.

Le *graphe sous-jacent* à un graphe orienté est le graphe non orienté obtenu en « oubliant » la direction des arcs. Ainsi, chaque arc est transformé en une arête. Le graphe orienté symétrique associé à un graphe non orienté est le graphe orienté obtenu en remplaçant chaque arête {u,v} par les deux arcs (u,v) et (v,u).

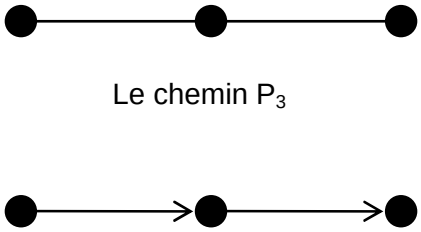
Un graphe orienté est *connexe* si toute paire de sommets est reliée par une chaîne (en d'autres termes, un graphe orienté est connexe si et seulement si son graphe sous-jacent est connexe). Un graphe orienté est *fortement connexe* si toute paire de sommets est reliée par un chemin. Un graphe orienté est donc fortement connexe si et seulement si son diamètre est fini. Une *composante fortement connexe* d'un graphe orienté est un sous-graphe induit connexe maximal.



1.3. Quelques familles de graphes

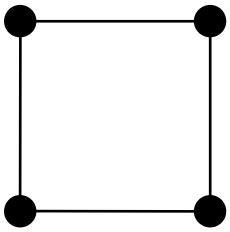
Le *chemin* à n sommets P_n est le graphe défini par $V(P_n) = \{0, 1, \ldots, n-1\}$ et $E(P_n) = \{ \{i,i+1\}, 0 \le i < n-1 \}$. Le *cycle* à n sommets C_n est le graphe défini par $V(P_n) = \{0, 1, \ldots, n-1\}$ et $E(P_n) = \{ \{i,i+1 \pmod n\}, 0 \le i \le n-1 \}$.

Le *chemin dirigé* à n sommets PP_n est le graphe orienté défini par $V(PP_n) = \{0, 1, \ldots, n-1\}$ et $A(PP_n) = \{ \{i,i+1\}, 0 \le i < n-1 \}$. Le *circuit* à n sommets CC_n est le graphe orienté défini par $V(CC_n) = \{0, 1, \ldots, n-1\}$ et $A(CC_n) = \{ \{i,i+1 \pmod n\}, 0 \le i \le n-1 \}$.

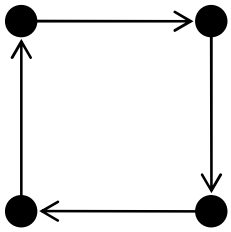


Le chemin P_3

Le chemin dirigé PP_3



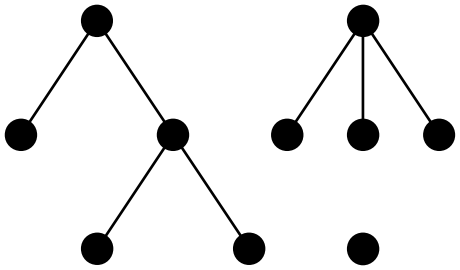
Le cycle C_4



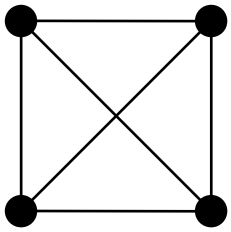
Le circuit CC_4

Un *arbre* est un graphe connexe sans cycle (un arbre à n sommets contient alors nécessairement n-1 arêtes). Une *forêt* est un graphe dont chaque composante connexe est un arbre.

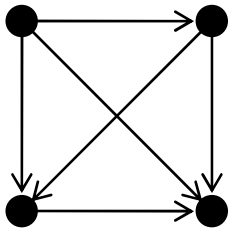
Le *graphe complet* à n sommets K_n est le graphe défini par $V(K_n) = \{0, 1, \ldots, n-1\}$ et $E(K_n) = V(K_n) \times V(K_n)$. Un *tournoi* à n sommets est une orientation quelconque du graphe complet K_n. Un tournoi *transitif* est un tournoi sans circuits.



Une forêt



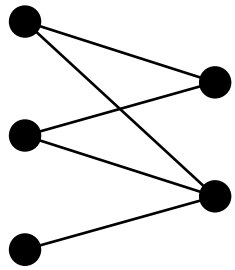
Le graphe complet K_4



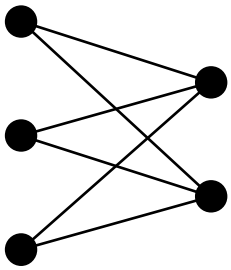
Un tournoi transitif

Un graphe G est *biparti* s'il existe une partition de ses sommets en deux classes V_1 et V_2 telle que toute arête relie un sommet de V_1 à un sommet de V_2. Ainsi, V_1 et V_2 sont deux ensembles *stables* (on dit également *indépendants*), c'est-à-dire que les sous-graphes induits $G[V_1]$ et $G[V_2]$ sont sans arêtes. Un graphe est dit *k-parti* s'il existe une telle partition en k sous-ensembles.

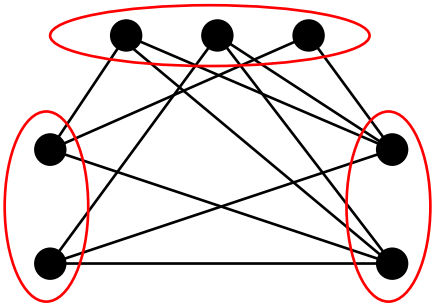
Le graphe *biparti complet* $K_{n,m}$ est le graphe biparti à n+m sommets, partitionné en un ensemble V_1 à n sommets et un ensemble V_2 à m sommets.



Un graphe biparti

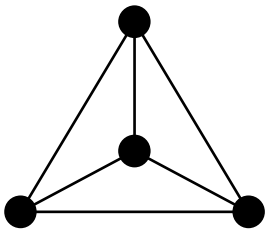


Le graphe complet biparti $K_{3,2}$



Un graphe 3-parti

Un graphe est dit *planaire* s'il *peut* être dessiné dans le plan sans croisement d'arêtes. On vérifie aisément que le graphe complet K_n est planaire si et seulement si $n \leq 4$. On peut également vérifier que le graphe biparti $K_{3,2}$ est planaire alors que $K_{3,3}$ ne l'est pas.

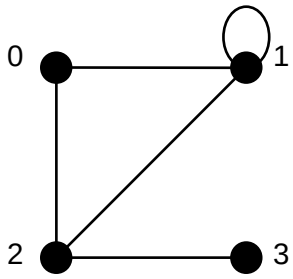


Une représentation planaire du graphe complet K_4

1.4. Représentation des graphes

1.4.1. Matrices d'adjacence

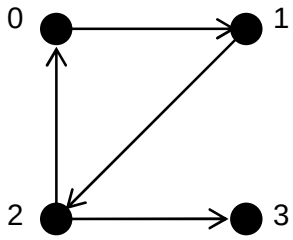
À tout graphe G, orienté ou non, on peut associer sa *matrice d'adjacence* M = M(G). Cette matrice est une matrice carrée dont les éléments, indicés par V(G) × V(G), sont définis par M[i,j] = 1 s'il existe un arc (ou une arête) allant de i vers j et M[i,j] = 0 sinon. Si le graphe G est non orienté, la matrice M(G) est symétrique.



Un graphe non orienté G

	0	1	2	3
0	0	1	1	0
1	1	1	1	0
2	1	1	0	1
3	0	0	1	0

La matrice M(G)



Un graphe orienté H

	0	1	2	3
0	0	1	0	0
1	0	0	1	0
2	1	0	0	1
3	0	0	0	0

La matrice M(H)

Une telle matrice se représente aisément au niveau algorithmique par un tableau à deux dimensions. En Python, on utilisera une liste de listes. La matrice M(H) ci-dessus sera alors représentée ainsi :

[[0,1,0,0], [0,0,1,0], [1,0,0,1], [0,0,0,0]]

Dans le cas d'un graphe non orienté, la matrice étant symétrique, il est plus efficace (en terme d'espace occupé) de représenter la matrice par un tableau à une seule dimension (ou une liste en Python), en ne stockant que la partie inférieure de la matrice, ligne par ligne. Pour la matrice M(G) ci-dessus, on obtiendrait la liste suivante en Python :

[0, 1, 1, 1, 1, 0, 0, 0, 1, 0]

Pour un graphe à n sommets, ce tableau n'utilise donc que n(n+1)/2 cases. La fonction suivante permet d'accéder à l'élément M[i,j] :

```
fonction accesElement ( E M : tableau de CMAX entiers,
                        E i, j : entiers ) : entier
# retourne l'élément M[i,j]
début
   si i > j
   alors  retourner ( M[ i*(i+1)/2 + j + 1 ] )
   sinon  retourner ( M[ j*(j+1)/2 + i + 1 ] )
   fin_si
fin
```

Notons que dans le cas d'un graphe non orienté sans boucles, il n'est pas nécessaire de stocker la diagonale. Ainsi, n(n-1)/2 cases suffisent dans ce cas (il est par contre nécessaire de modifier la fonction `accesElement`...).

La propriété suivante est une propriété remarquable de la matrice d'incidence d'un graphe (orienté ou non) :

> **Propriété.** Soit G un graphe, orienté ou non, et M sa matrice d'adjacence. La puissance k-ième M^k de la matrice M est telle que l'élément $M^k[i,j]$ correspond au nombre de chemins de longueur k allant de i vers j.

Preuve.

La preuve se fait par récurrence sur k. Si k=1, $M^k = M$ est deux sommets sont reliés par un chemin de longueur 1 si et seulement si ils sont reliés par une arête. La propriété est donc satisfaite.

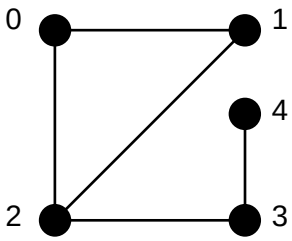
Supposons la propriété vraie jusqu'au rang k-1. Nous avons alors :

$$M^k[i,j] = M[i,0] * M^{k-1}[0,j] + M[i,1] * M^{k-1}[1,j] + ... + M[i,n-1] * M^{k-1}[n-1,j]$$

Par hypothèse de récurrence, $M^{k-1}[p,j]$ correspond au nombre de chemins de longueur k-1 reliant p à j. De plus, $M[i,p] * M^{k-1}[p,j]$ vaut $M^{k-1}[p,j]$ si i et p sont reliés par une arête et 0 sinon.

Ainsi, $M[i,p] * M^{k-1}[p,j]$ correspond au nombre de chemins de longueur k allant de i à j « en arrivant par le sommet p ». La somme globale correspond alors bien au nombre de chemins de longueur k allant de i à j. €

L'exemple suivant permet de constater cette propriété :



Un graphe non orienté G

	0	1	2	3	4
0	0	1	1	0	0
1	1	0	1	0	0
2	1	1	0	1	0
3	0	0	1	0	1
4	0	0	0	1	0

La matrice M(G)

	0	1	2	3	4
0	2	1	1	1	0
1	1	2	1	1	0
2	1	1	3	0	1
3	1	1	0	1	0
4	0	0	1	0	1

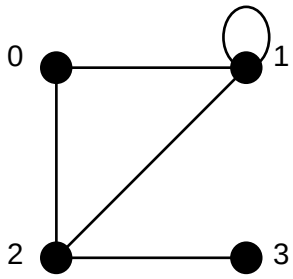
La matrice $M^2(G)$

Regardons par exemple les chemins de longueur 2 issus du sommet 2 (3ème ligne de la matrice $M^2(G)$) :

- 1 chemin de 2 vers 0 : 210
- 1 chemin de 2 vers 1 : 201
- 3 chemins de 2 vers 2 : 202, 212 et 232
- aucun chemin de 2 vers 3
- 1 chemin de 2 vers 4 : 234

1.4.2. Listes d'adjacence

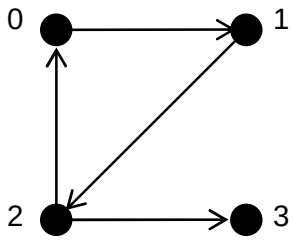
Une autre façon classique de représenter un graphe est d'associer à chaque sommet la liste de ses voisins (graphe non orienté) ou de ses successeurs (graphe orienté). Généralement, ces listes sont triées par ordre croissant.



Un graphe non orienté G

0 → 1, 2
1 → 0, 1, 2
2 → 0, 1, 3
3 → 2

Listes d'adjacence



Un graphe orienté H

0 → 1
1 → 2
2 → 0, 3
3 →

Listes d'adjacence

En algorithmique, cette représentation correspond à un tableau de listes. En Python, on utilisera une liste de listes. La représentation en Python du graphe G ci-dessus sera alors :

[[1, 2], [0, 1, 2], [0, 1, 3], [2]]

Notons que dans le cas des graphes non orientés, chaque arête est « codée » deux fois, dans chacune des listes d'adjacence de ses deux extrémités.

1.5. Modélisation de problèmes à l'aide de graphes

De nombreux problèmes peuvent être modélisés à l'aide de la théorie des graphes. Le principe général est le suivant :

- on modélise le « support » du problème à l'aide d'un graphe (il faut alors choisir les objets que représentent les sommets et les relations que représentent les arcs ou les arêtes),
- on reformule le problème en termes de graphes,
- on tente de résoudre le problème en question : on retombe parfois sur un problème bien connu en théorie des graphes, pour lequel on dispose d'algorithmes de résolution, et parfois sur un nouveau problème, source possible de nouvelles recherches en théorie des graphes…

Des domaines d'application courants sont par exemple :

- La théorie des jeux : les sommets représentent les positions de jeux, les arcs les « coups » faisant passer d'une position à une autre ; le problème consiste généralement à trouver un (plus court) chemin menant de la position initiale à une position gagnante.
- Les problèmes de réseau routier : le graphe (orienté) modélise le réseau et on cherche généralement des chemins optimaux selon des critères de coût spécifiques, distance, temps, … (exemple classique : le GPS).
- Les problèmes de réseaux (informatique, électrique, …) : diffusion ou collecte d'information, résistance aux pannes, optimisation de débit, …
- etc.

Chapitre 2. Coloration de graphes

Tous les graphes considérés dans ce chapitre sont des graphes non orientés, simples et sans boucles.

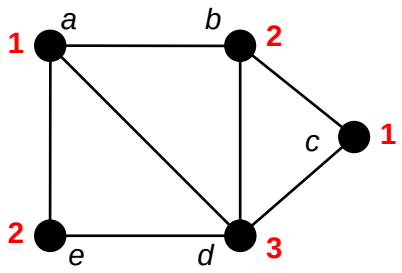
2.1. Définitions

Une *k-coloration* d'un graphe G est une application c qui associe à chaque sommet de G une couleur de l'ensemble {1, …, k} de façon telle que les sommets voisins ont des couleurs distinctes :

$$\{u, v\} \in E(G) \Rightarrow c(u) \neq c(v).$$

Une k-coloration d'un graphe G peut également être vue comme une partition de l'ensemble des sommets V(G) en k sous-ensembles disjoints V_1, …, V_k correspondant aux ensembles de sommets ayant même couleur. Chacun de ces sous-ensembles est un ensemble *stable* (on dit aussi *indépendant*), c'est-à-dire tel qu'aucune arête ne relie deux de ses sommets. Un graphe est donc k-coloriable si et seulement si il est k-parti.

L'exemple suivant montre une 3-coloration d'un graphe ainsi que la partition associée :



V1 = { a, c }
V2 = { b, e }
V3 = { 3 }

Le *nombre chromatique* d'un graphe G, noté $\chi(G)$, est le plus petit k pour lequel G admet une k-coloration. On peut aisément se convaincre que le nombre chromatique du graphe précédent est 3.

Les propriétés suivantes sont aisément vérifiables (et démontrables) :

- si G est un arbre, $\chi(G) = 2$,
- si $G = K_n$ (graphe complet à k sommets), $\chi(G) = n$,
- si G est un cycle pair, $\chi(G) = 2$,
- si G est un cycle impair, $\chi(G) = 3$.

2.2. Algorithme glouton de coloration

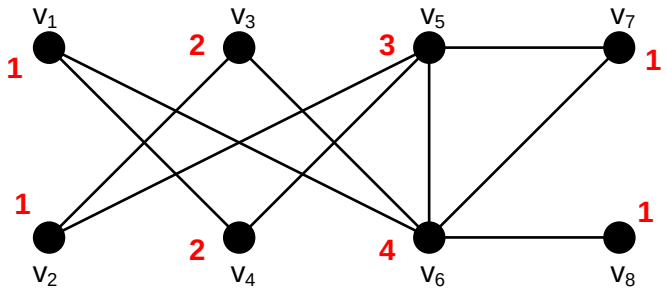
Considérons un graphe G à n sommets et ordonnons ses sommets de façon quelconque : $V(G) = \{v_1, \ldots, v_n\}$. Un algorithme simple de coloration, glouton (dit algorithme *First-Fit*), est alors le suivant :

- colorions les sommets dans l'ordre v_1, …, v_n,
- pour colorier le sommet v_i, utilisons la plus petite couleur possible (en d'autres termes, affectons à v_i la plus petite couleur non encore utilisée par les voisins de v_i déjà coloriés).

Une deuxième version de cet algorithme (on peut facilement se convaincre que ces deux versions produisent le même résultat) est la suivante :

- prenons les couleurs dans l'ordre 1, 2, …
- pour chaque couleur *c*, parcourons les sommets dans l'ordre v_1, …, v_n, et affectons la couleur *c* aux sommets chaque fois que cela est possible.

L'exemple suivant montre une coloration obtenue à l'aide de l'algorithme First-Fit :



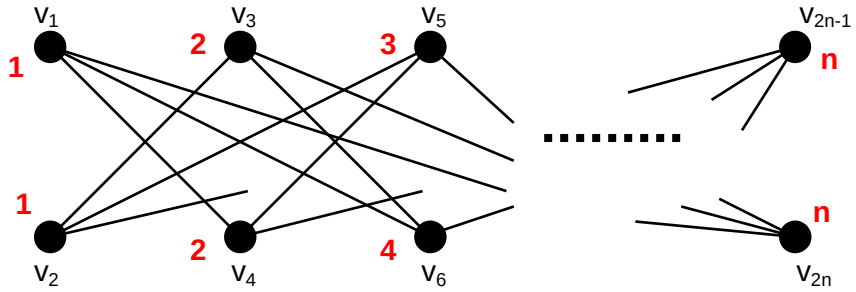
Clairement, cet algorithme produit une coloration de G utilisant au maximum $\Delta(G) + 1$ couleurs, où $\Delta(G)$ désigne le *degré maximum* de G, c'est-à-dire le maximum des degrés de ses sommets. En effet, lorsqu'on attribue une couleur à un sommet, le nombre de couleurs interdites est au plus le degré de ce sommet. Ainsi :

Proposition. Pour tout graphe G, $\chi(G) \leq \Delta(G) + 1$.

Les graphes pour lesquels l'inégalité précédente est une égalité ont été caractérisés par Brooks en 1941 :

Théorème [Brooks, 1941]. Soit G un graphe connexe de degré maximal $\Delta(G)$. Nous avons alors $\chi(G) = \Delta(G) + 1$ si et seulement si $\Delta(G) = 2$ et G est un cycle impair ou $\Delta(G) \neq 2$ et G est le graphe complet à $\Delta(G) + 1$ sommets. (Dans le cas contraire, nous avons $\chi(G) \leq \Delta(G)$).

Remarquons cependant que l'algorithme First-Fit peut utiliser un nombre de couleurs arbitrairement supérieur au nombre chromatique du graphe. Le graphe suivant est biparti (chaque sommet est relié à tous les sommets de l'autre partie, à l'exception du sommet « en vis-à-vis »), d'où $\chi(G) = 2$, et l'algorithme First-Fit utilise n couleurs pour l'ordre indiqué :



On s'aperçoit ainsi que le nombre de couleurs utilisées par l'algorithme First-Fit dépend fortement de l'ordre choisi. On peut facilement se convaincre que, pour tout graphe G, il existe un ordre de ses sommets pour lequel l'algorithme First-Fit produit une coloration optimale, c'est-à-dire utilisant $\chi(G)$ couleurs (sauriez-vous le prouver ?...).

Le problème consistant à déterminer le nombre chromatique d'un graphe est un problème difficile dans le cas général[1] : on ne connait pas d'algorithme polynomial permettant de déterminer le nombre chromatique d'un graphe quelconque. Le nombre chromatique peut seulement être déterminé en considérant « toutes les colorations possibles », ce qui, naturellement, est de complexité exponentielle.

Ainsi, déterminer un ordre optimal pour l'algorithme First-Fit ne peut se faire en temps polynomial. Il existe cependant certaines heuristiques permettant de produire des ordres *a priori satisfaisants* (mais on peut toujours construire des graphes pour lesquels cet ordre produit une très mauvaise coloration…). Par exemple, l'algorithme de Welsh et Powell consiste à ordonner les sommets du graphe *par ordre décroissant* avant d'appliquer l'algorithme First-Fit.

2.3. Bornes sur le nombre chromatique d'un graphe

2.3.1. Minorants

Une *clique* à k sommets est un sous-graphe isomorphe au graphe complet K_k à k sommets. Le *nombre de clique* (*clique number)* d'un graphe G, noté $\omega(G)$, est l'ordre maximum d'une clique dans G. Comme il est nécessaire d'utiliser k couleurs pour colorier une clique à k sommets, nous avons :

Proposition. Pour tout graphe G, $\chi(G) \geq \omega(G)$.

Notons cependant que l'écart entre $\chi(G)$ et $\omega(G)$ peut être arbitrairement grand. Ce résultat a été prouvé par Zykov :

Théorème [Zykov, 1949]. Pour tout k, il existe des graphes de nombre chromatique k ne contenant pas de triangle *(triangle-free graphs)*.

En effet, le nombre de clique d'un graphe sans triangle et contenant au moins une arête vaut 2…

Un *stable* (ou *ensemble indépendant*) dans un graphe G est un sous-ensemble S des sommets de G tel que le sous-graphe induit par S ne contient aucune arête (en d'autres termes, aucune arête ne joint deux sommets de S). Le cardinal maximum d'un tel ensemble est le *nombre de stabilité* de G, noté $\alpha(G)$.

Si G est un graphe colorié, chaque *classe de couleur* (ensemble de sommets ayant même couleur) est nécessairement un ensemble stable, donc de taille au plus $\alpha(G)$. Ainsi, nous avons :

Proposition. Pour tout graphe G, $\chi(G) \geq |V(G)| / \alpha(G)$.

Remarquons enfin que l'on peut facilement trouver des graphes pour lesquels les deux bornes précédentes sont atteintes. Ainsi, l'intérêt de ces deux bornes (au moins en classe de Terminale ES) est le suivant : il suffit de produire une coloration utilisant $\omega(G)$ ou $|V(G)| / \alpha(G)$ couleurs pour conclure que le nombre de couleurs utilisées par cette coloration correspond au nombre chromatique du graphe.

2.3.2. Majorants

Le Théorème de Brooks vu précédemment fournit un premier majorant du nombre chromatique d'un graphe.

Ce théorème a été amélioré par Halin en 1967 :

[1] Il existe par contre des algorithmes polynomiaux résolvant ce problème pour certaines classes de graphes. Par exemple, dans le cas des graphes *triangulés* (tout cycle de longueur au moins 4 possède une corde), le nombre chromatique peut être déterminé en temps linéaire.

Théorème [Halin, 1967]. Soit G un graphe et $k = \max_H \{ \delta(H) : H$ sous-graphe induit de $G \}$. Nous avons alors $\chi(G) \leq k + 1$ (en d'autres termes, si G est k-dégénéré, $\chi(G) \leq k + 1$).

Ici, $\delta(H)$ désigne le *degré minimum* de H, c'est-à-dire le minimum des degrés de ses sommets. Intuitivement, un graphe G est *k-dégénéré* s'il est possible de « l'éplucher complètement » en supprimant itérativement des sommets de degré au plus k. Ainsi, tout arbre est 1-dégénéré (on l'épluche par ses feuilles…).

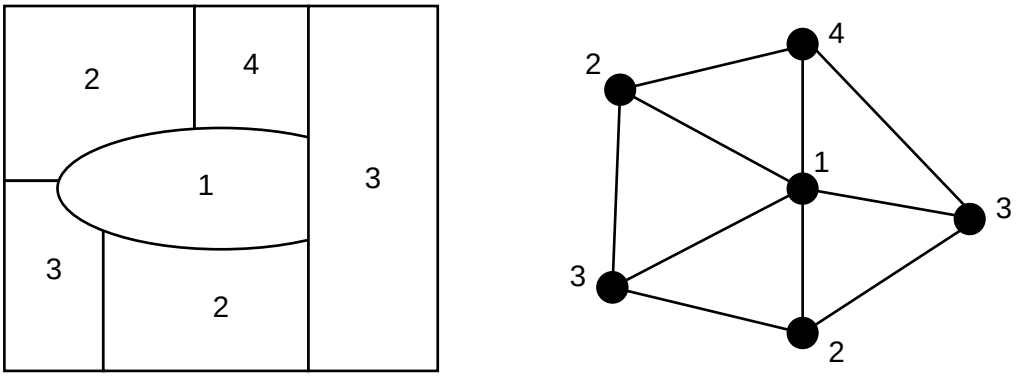
Dans le cas des arbres justement, le Théorème de Halin donne une borne de 2, donc optimale, alors que le Théorème de Brooks donne un majorant arbitrairement grand (le degré maximum de l'arbre).

2.4. Coloration des graphes planaires

En 1852, Francis Guthrie posa le fameux « problème des quatre couleurs » : est-il possible de colorier n'importe quelle carte en quatre couleurs de façon telle que les pays ayant une frontière commune aient des couleurs distinctes ? Ce problème fut communiqué à quelques mathématiciens célèbres de l'époque (De Morgan, Hamilton, Cayley, etc.). En 1879, Kempe en proposa une solution… qui fut démontrée fausse en 1890 par Heawood (dans son premier article).

Ce n'est qu'en 1976 que Appel et Haken produisirent une preuve de cette conjecture, preuve utilisant massivement l'ordinateur.

Ce problème de coloration de carte peut facilement être transformé en un problème de coloration des sommets d'un graphe : on associe un sommet à chaque pays de la carte (sa capitale), deux sommets étant reliés si et seulement si les pays correspondants ont une frontière commune. Toute coloration correcte de la carte correspond alors à une coloration propre du graphe associé, et réciproquement :



Une carte coloriée et le graphe colorié associé

On s'aperçoit aisément que le graphe associé à une telle carte est nécessairement planaire. Ainsi, le problème des quatre couleurs peut également être formulé ainsi :

Tout graphe planaire est-il coloriable avec quatre couleurs ?

Une *face* d'un graphe planaire dessiné est une portion connexe du plan délimitée par des arêtes. La *formule d'Euler* fournit un invariant des graphes planaires très utile :

Formule d'Euler. Si G est un graphe planaire dessiné ayant n sommets, m arêtes et f faces, alors $n - m + f = 2$.

À l'aide de cette formule, on peut notamment montrer les deux propriétés suivantes :

Propriété. Tout graphe planaire à n sommets, $n \geq 3$, possède au plus $3n - 6$ arêtes.

Preuve. Chaque face est bordée par au moins 3 arêtes, d'où $f \leq 2m/3$. Par la formule d'Euler, nous avons alors $m + 2 = n + f \leq n + 2m/3$, soit $m \leq 3n - 6$. €

Propriété. Tout graphe planaire contient un sommet de degré au plus 5.

Preuve. Raisonnons par l'absurde et supposons que G est un graphe planaire dont tous les sommets sont de degré au moins 6. Par le Lemme des poignées de main, nous avons alors $2m \geq 6n$. Par ailleurs, toute face étant bordée par au moins trois arêtes, nous avons $2m \geq 3f$. En injectant les deux inégalités $n \leq m/3$ et $f \leq 2m/3$ dans la Formule d'Euler, nous obtenons :

$$2 = n - m + f \leq m/3 - m + 2m/3 = 0$$

d'où la contradiction. €

Ainsi, les graphes planaires sont 5-dénégérés, et, par le Théorème de Halin, nous obtenons :

Corollaire. Tout graphe planaire est 6-coloriable.

La technique utilisée par Kempe dans sa preuve fausse du théorème des quatre couleurs permettait cependant de démontrer que tout graphe planaire est 5-coloriable :

Théorème (Kempe, 1879). Tout graphe planaire est 5-coloriable.

Preuve. Supposons le résultat faux et soit G un plus petit contre-exemple (en termes de sous-graphe). Nous savons que G contient un sommet x de degré au plus 5 et que le graphe G-x est 5-coloriable. Le fait que G soit un contre-exemple entraîne que les couleurs 1,2,…,5 soient toutes utilisées sur les voisins de x pour toute 5-coloration de G-x (dans le cas contraire, on pourrait étendre la coloration à G car il y aurait une couleur libre pour x).

Soit donc c une 5-coloration de G-x. Sans perte de généralité (quitte à renommer les couleurs) nous pouvons supposer que les voisins de x, notés dans l'ordre cyclique $x_1, x_2, \ldots, x_5$, sont respectivement coloriés 1,2,…,5. Notons G(i,j), pour $1 \leq i < j \leq 5$, le sous-graphe de G-x engendré par les sommets de couleur i ou j.

Supposons que x_1 et x_3 n'appartiennent pas à la même composante connexe de G(1,3). Dans ce cas, en échangeant les couleurs des sommets de la composante de x_1 (1 devient 3, 3 devient 1), on obtient une 5-coloration de G-x qui peut être étendue à G (car la couleur 1 est libre pour x), contredisant le fait que G est un contre-exemple.

Ainsi, il existe un chemin bicolore 1-3 reliant x_1 à x_3 dans G-x (de tels chemins sont appelés des *chaînes de Kempe*). En raisonnant de la même façon, on montre qu'il existe un chemin bicolore 2-4 reliant x_2 à x_4. On obtient alors une contradiction car ces deux chemins doivent nécessairement se couper (le sommet intersection ne peut avoir une couleur dans $\{1,3\} \cap \{2,4\} = \emptyset$). €

La conjecture des quatre couleurs a été démontrée par Appel et Haken en 1976 :

Théorème [Appel and Haken[2], 1976]. Tout graphe planaire est 4-coloriable.

La preuve de ce résultat est basée sur l'idée suivante (il suffit de montrer le résultat pour les graphes triangulés, i.e. toutes les faces sont des triangles) :

- il existe un ensemble de 1482 configurations inévitables, c'est-à-dire tel que tout graphe planaire triangulé contient nécessairement l'une de ces configurations comme sous-graphe,
- ces configurations sont réductibles, c'est-à-dire qu'une 4-coloration d'un graphe planaire contenant l'une de ces configurations peut toujours être obtenue à partir d'une 4-coloration d'un graphe planaire « plus petit »… (en d'autres termes, un contre-exemple minimal pour la conjecture des quatre couleurs ne peut pas contenir l'une de ces configurations).

Remarque. On peut observer que cette idée est similaire à celle utilisée dans la preuve de la 5-colorabilité des graphes planaires. Nous avons alors un ensemble de 5 configurations (sommet de degré 1, 2, … ou 5), inévitables (par la formule d'Euler) et réductibles (de façon triviale pour les 4 premières, grâce à l'argument des chaînes de Kempe pour la cinquième).

Cette preuve du « Théorème des Quatre Couleurs » a été simplifiée par Robertson, Sanders, Seymour et Thomas[3] en 1994 qui ont ramené à moins de 700 le nombre de configurations nécessaires (cette nouvelle preuve utilise cependant toujours un programme informatique…).

Le graphe K_4 étant planaire, la borne de 4 pour le nombre chromatique des graphes planaires est naturellement optimale.

[2] K. Appel and W. Haken, Every planar map is four-colorable, Part I : Discharging, *Illinois J. of Math.* **21** (1977), 429-490, et K. Appel, W. Haken and J. Koch, Every planar map is four-colorable, Part II : Reducibility, *Illinois J. of Math.* **21** (1977), 491-567.

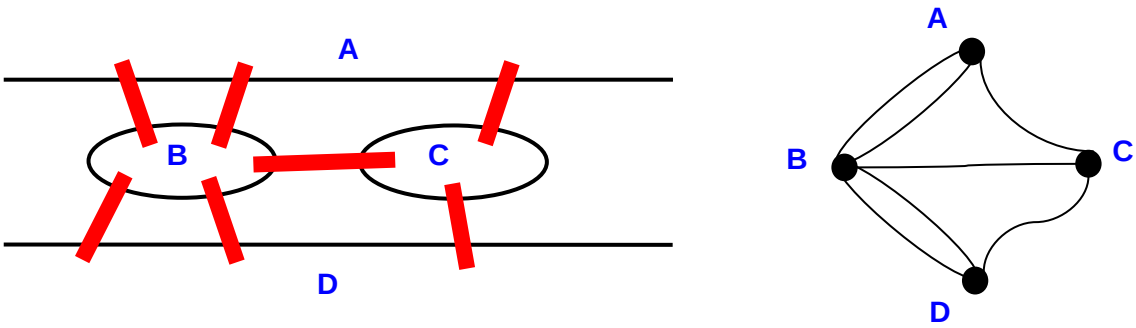
[3] N. Robertson, D. Sanders, P. Seymour and R. Thomas, The four-colour theorem, *J. Combinatorial Theory, Ser. B* **70** (1997), 2-44.

Chapitre 3. Graphes eulériens

3.1. Les ponts de Königsberg

La ville de Königsberg (aujourd'hui Kaliningrad) est construite sur la rivière Pregel et comporte deux îles reliées entre elles et aux berges de la rivière par sept ponts (voir ci-dessous). Les habitants se demandaient s'il était possible de réaliser un parcours de la ville empruntant une et une seule fois chacun de ces sept ponts en revenant au point de départ.

Ce problème fut résolu, en toute généralité, par Leonhard Paul Euler (1707-1783), qui le modélisa en termes de graphes (on attribue généralement à ces travaux l'origine de la théorie des graphes). Euler associa à ce problème un multigraphe dont les sommets représentaient les berges et les îles et les arêtes les ponts :



On appelle ***chemin eulérien*** un chemin empruntant exactement une fois chaque arête d'un graphe. On appelle ***cycle eulérien*** un chemin eulérien fermé.

3.2. Théorème d'Euler

Euler prouva le théorème suivant :

> **Théorème (Euler, 1736).** Un multigraphe connexe G possède un cycle eulérien si et seulement si tous ses sommets sont de degré pair. Il possède un chemin eulérien (non cycle) si et seulement si exactement deux de ses sommets sont de degré impair (dans ce cas, ces deux sommets constituent nécessairement les extrémités de ce chemin).

Preuve.

(1) On prouve d'abord le cas du cycle eulérien.

($\Rightarrow$) Un cycle eulérien emprunte toutes les arêtes du graphe. Ainsi, chaque fois que l'on arrive sur un sommet par une arête, on repart par une autre. Les arêtes sont donc « associées » ainsi deux à deux et leur nombre est bien pair. Dans le cas d'un chemin eulérien (non cycle), la remarque précédente s'applique sauf pour les deux sommets extrémités (première et dernière arête du chemin). Ces deux sommets sont alors les seuls de degré impair.

($\Leftarrow$) (1) On prouve d'abord le cas du cycle eulérien.

Montrons que tout graphe G dont les sommets sont tous de degré pair admet un cycle eulérien. On raisonne par récurrence sur le nombre d'arêtes de G. Si G ne contient qu'une seule arête, le résultat

est immédiat : cette arête est une boucle, nous n'avons qu'un seul sommet, de degré pair et elle constitue un cycle eulérien. Supposons maintenant la propriété vraie pour tous les graphes ayant au plus m arêtes et soit G un graphe à m+1 arêtes dont tous les sommets sont de degré pair.

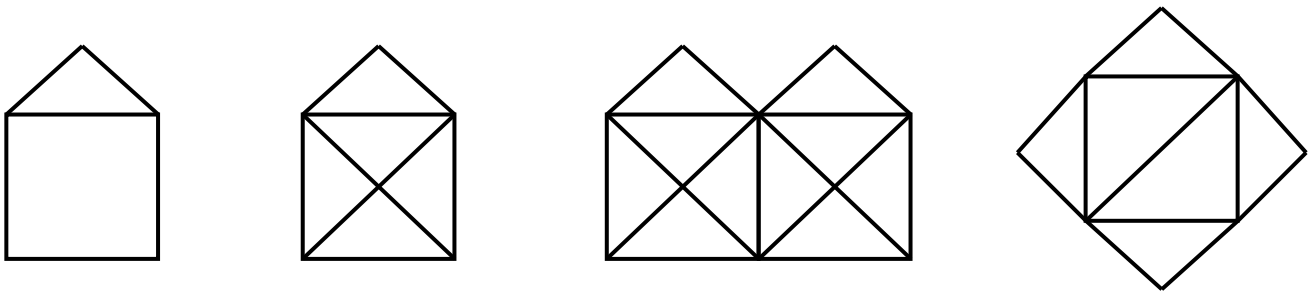
Remarquons que G contient nécessairement un cycle : dans le cas contraire, G est un arbre mais dans ce cas ses feuilles sont de degré 1, donc impair. Soit donc C un cycle de G. Si C est eulérien, c'est terminé. Dans le cas contraire, considérons le graphe G' obtenu à partir de G en supprimant les arêtes de G. Le graphe G' n'est pas nécessairement connexe mais, par hypothèse de récurrence, chacune de ses composantes connexes possède un cycle eulérien (en supprimant les arêtes de C, la parité des degrés des sommets a été préservée). Le graphe G étant connexe, chacune des composantes connexes de G' intersecte le cycle C sur au moins un sommet. Il est alors aisé de « construire » un cycle eulérien de G à partir de C et des cycles des composantes de G' : pour chaque composante G'_i intersectant C en un sommet v_i, on augmente C ainsi :

… u_i <cycle de G'_i> u_i …

(2) Le cas des graphes ayant deux sommets de degré impair se déduit aisément de la preuve précédente : soient u et v ces deux sommets ; en rajoutant l'arête uv, on obtient un graphe dont tous les sommets sont de degré pair ; ce graphe possède donc un cycle eulérien ; en supprimant l'arête uv de ce cycle, on obtient un chemin eulérien. €

Il est ainsi facile d'obtenir un algorithme vérifiant si un graphe donné est ou non eulérien. En se basant sur la preuve du théorème, il est également facile de proposer un algorithme construisant un cycle (ou un chemin) eulérien lorsqu'il en existe un, en s'inspirant de la preuve précédente (pour trouver le cycle C, il suffit de choisir un sommet de départ et d'avancer tant qu'on ne retombe pas sur ce même sommet…).

Les graphes possédant un chemin eulérien sont exactement les graphes « que l'on peut dessiner sans lever le crayon ». Ainsi, parmi les graphes ci-dessous, seul le troisième (qui contient quatre sommets de degré impair) ne peut être dessiné sans lever le crayon…

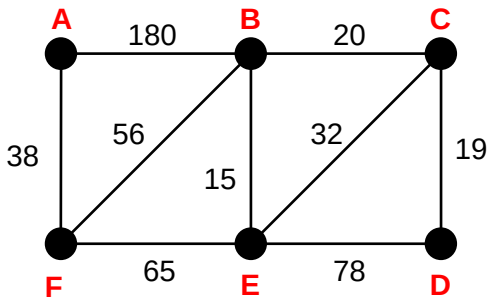


Chapitre 4. Chemins de moindre coût

4.1. Graphes valués et chemins de moindre coût

On considère dans ce chapitre des graphes *valués* c'est-à-dire des graphes (orientés ou non) pour lesquels une valeur est associée à chaque arête ou à chaque arc. Cette valeur est généralement numérique. Les graphes valués sont parfois qualifiés de *pondérés* (un *poids* est associé à chaque arête).

Les graphes valués permettent de modéliser des situations où un certain *coût* (valeur) peut être associé à chaque arête :



Le *coût* d'un chemin est la somme des coûts des arêtes qui le composent. Ainsi, dans l'exemple ci-dessus, le coût du chemin ABED est 180 + 15 + 78 = 273.

On doit ainsi plutôt parler de *chemin de moindre coût* plutôt que de *plus court chemin* (terminologie fréquemment rencontrée), la distance n'étant que l'une des multiples possibilités de la fonction de coût.

Les problèmes de calcul de chemin de moindre coût sont généralement classés en trois catégories :

1. trouver un chemin de moindre coût entre deux sommets donnés S et T,
2. trouver les chemins de moindre coût allant de S à chacun des autres sommets,
3. trouver les chemins de moindre coût reliant toutes les paires possibles de sommets.

Notons que dans le cas d'un GPS embarqué, c'est surtout le problème 1 qui est considéré…

4.2. Algorithme de Dijkstra « simplifié »

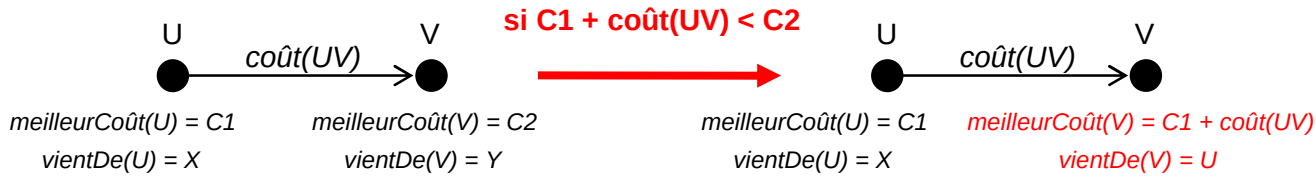
L'algorithme de Dijkstra permet de résoudre le problème 2 ci-dessus. Notons cependant qu'il peut être également utilisé pour résoudre le problème 1 : il suffit alors de stopper son exécution lorsque le chemin de moindre coût reliant S à T a été trouvé…

Le principe de cet algorithme est le suivant :

- les données de l'algorithme sont un graphe orienté G dont les arcs sont valués par une fonction *coût* positive ou nulle (pour les graphes non orientés, il suffit de remplacer arc par arête dans l'algorithme), et un sommet S (le sommet à partir duquel on souhaite déterminer les chemins de moindre coût),
- à chaque sommet U de G, on associe deux informations :

- un entier *meilleurCoût(U)* : la valeur du chemin de S à U de moindre coût connu à ce stade de l'algorithme (initialisé à 0 pour S, à +∞ pour les autres sommets),
- un sommet *vientDe(U)* : le sommet par lequel on arrive en U en suivant le chemin de S à U de moindre coût connu à ce stade de l'algorithme,

- on gère un ensemble de sommets *traités* qui, à tout instant, contient l'ensemble des sommets pour lequel un chemin optimal issu de S a été construit,
- à chaque étape de l'algorithme (boucle tant que), un chemin optimal va être obtenu reliant S à l'un des sommets du graphe, disons T. Le sommet T est placé dans l'ensemble *traités* et on vérifie alors si les arcs issus de T constituent ou non des « raccourcis » (un arc UV est un raccourci si, en empruntant l'arc UV, on améliore le meilleur chemin connu menant à V ; on a alors intérêt à rejoindre V en passant par U…) :



L'arc UV est un raccourci

Notons que le fait que la fonction coût soit positive ou nulle fait que lorsque un arc UV est traité comme étant un raccourci, il cesse d'être un raccourci. Mais si le chemin optimal conduisant à U est amélioré par la suite, il est alors possible qu'il devienne à nouveau un raccourci…

Lorsque l'algorithme se termine, chaque sommet U possède l'information *meilleurCoût(U)*, donnant le coût du chemin optimal menant de S à U. Ce chemin peut être reconstitué de proche en proche, grâce à l'information *vientDe(U)*.

L'algorithme est alors le suivant :

```
Algorithme Dijkstra
  données       G : graphe orienté, coût : fonction entière, S : sommet,
  résultats     vientDe : tableau[Sommet] de Sommet
                meilleurCoût : tableau[Sommet] d'entiers

# cet algorithme, du à Dijkstra, détermine les chemins de moindre coût
# dans G reliant S à tous les autres sommets

variable    traités : ensemble de sommets

début
       # initialisations
  pour tout sommet U de G
  faire
       meilleurCoût[U] ← infini
  fin_pour_tout

  meilleurCoût[S] ← 0
  vientDe[S] ← S
  traités ← ensemble vide

       # boucle de traitement : tant qu'il reste des sommets non traités
  tant que (traités <> V(G))
  faire
       choisir un sommet X non traité de meilleurCoût minimal
       rajouter X dans traités

            # les arcs issus de X sont-ils des raccourcis ?
       pour tout successeur non traité Y de X
       faire
```

```
            si ( meilleurCoût[X] + coût(XY) < meilleurCoût[Y] )
            alors    meilleurCoût[Y] ← meilleurCoût[X] + coût(XY)
                     vientDe[Y] = X
            fin_si
      fin_pour-tout
fin
```

Illustrons maintenant le fonctionnement de cet algorithme sur notre graphe exemple, en prenant comme sommet initial S = A.

Après la phase d'initialisation, nous avons :

Sommet	A	B	C	D	E	F
meilleurCoût	0	+∞	+∞	+∞	+∞	+∞
vientDe	A	?	?	?	?	?
traités	n	n	n	n	n	n

Étape 1. Le sommet non traité de meilleurCoût minimal est A. Les deux arêtes incidentes à A, AB et AF, sont toutes deux des raccourcis. Le sommet A est ajouté à traités et les deux raccourcis sont pris en compte. On obtient alors :

Sommet	A	B	C	D	E	F
meilleurCoût	0	180	+∞	+∞	+∞	38
vientDe	A	A	?	?	?	A
traités	O	n	n	n	n	n

Étape 2. Le sommet non traité de meilleurCoût minimal est F. Deux des trois arêtes incidentes à F, FB et FE, sont des raccourcis. Le sommet F est ajouté à traités et les deux raccourcis sont pris en compte. On obtient alors :

Sommet	A	B	C	D	E	F
meilleurCoût	0	94	+∞	+∞	103	38
vientDe	A	F	?	?	F	A
traités	O	n	n	n	n	O

Étape 3. Le sommet non traité de meilleurCoût minimal est B. Une seule des trois arêtes incidentes à B, BC, est un raccourci. Le sommet B est ajouté à traités et le raccourci est pris en compte. On obtient alors :

Sommet	A	B	C	D	E	F
meilleurCoût	0	94	114	+∞	103	38
vientDe	A	F	B	?	F	A
traités	O	O	n	n	n	O

Étape 4. Le sommet non traité de meilleurCoût minimal est E. Une seule des quatre arêtes incidentes à E, ED, est un raccourci. Le sommet E est ajouté à traités et le raccourci est pris en compte. On obtient alors :

Sommet	A	B	C	D	E	F
meilleurCoût	0	94	114	181	103	38
vientDe	A	F	B	E	F	A
traités	O	O	n	n	O	O

Étape 5. Le sommet non traité de meilleurCoût minimal est C. Une seule des trois arêtes incidentes à C, CD, est un raccourci. Le sommet C est ajouté à traités et le raccourci est pris en compte. On obtient alors :

Sommet	A	B	C	D	E	F
meilleurCoût	0	94	114	133	103	38
vientDe	A	F	B	C	F	A
traités	O	O	O	n	O	O

Étape 6. Le sommet non traité de meilleurCoût minimal est D. Tous ses voisins sont déjà traités, le sommet D est ajouté à traités et l'algorithme se termine. On obtient finalement :

Sommet	A	B	C	D	E	F
meilleurCoût	0	94	114	133	103	38
vientDe	A	F	B	C	F	A
traités	O	O	O	O	O	O

Grâce aux informations retournées, les chemins optimaux sont les suivants :

- de A à A : « vide », de coût 0,
- de A à B : AFB, de coût 94,
- de A à C: AFBC, de coût 114,
- de A à D : AFBCD, de coût 133,
- de A à E : AFE, de coût 103,
- de A à F: AF, de coût 38.

4.3. Algorithme de Dijkstra original

L'algorithme présenté dans la section précédente et celui proposé dans le cadre du programme de Terminale ES. L'algorithme original de Dijkstra utilise un ensemble supplémentaire, *atteints*, qui contient les sommets du graphe ayant subi au moins une amélioration via un raccourci, et non encore traités. Cet ensemble permet de restreindre la recherche d'un sommet X de *meilleurCoût* minimal aux sommets pour lesquels cette valeur n'est pas infinie…

La boucle de traitement de l'algorithme précédent est alors ainsi modifiée :

```
…
atteints ← { s }
     # boucle de traitement : tant qu'il reste des sommets non traités
tant que (traités <> V(G))
faire
     choisir un sommet X dans atteint de meilleurCoût minimal
     supprimer X de atteints
     rajouter X dans traités
          # les arcs issus de X sont-ils des raccourcis ?
     pour tout successeur non traité Y de X
     faire
          si ( meilleurCoût[X] + coût(XY) < meilleurCoût[Y] )
          alors     meilleurCoût[Y] ← meilleurCoût[X] + coût(XY)
                    vientDe[Y] = X
                    ajouter Y dans atteints
          fin_si
     fin_pour-tout
```

Chapitre 5. Graphes valués et automates

Seule la notion d'automate (désigné sous l'appellation *graphe pondéré permettant de reconnaître un ensemble de mots*) est au programme de la classe de Terminale ES. Nous introduisons ici quelques éléments supplémentaires, notions de base de la *théorie des langages*, permettant de mieux appréhender le contexte lié à l'utilisation d'automates.

5.1. Définitions de base : alphabet, mots et langages

Un *alphabet* est un ensemble A de symboles appelés *lettres*. Dans ce chapitre, nous ne considèrerons que des alphabets *finis*.

Un *mot* m sur un alphabet A est une séquence de lettres prises dans A : $m = a_1 \ldots a_k$ (là encore, nous ne considèrerons que des mots *finis*). On note A* l'ensemble de tous les mots construits sur l'alphabet A.

Le nombre de lettres d'un mot m est sa *longueur*, notée |m|. Ainsi, pour $m = a_1 \ldots a_k$, nous avons $|m| = k$. Il existe un unique mot de longueur nulle, le *mot vide*, noté ε. On note A^n l'ensemble des mots de A* de longueur n. En particulier, $A^0 = \{ \varepsilon \}$ et $A^1 = A$.

Soit a une lettre de A et m un mot de A*. Le *nombre d'occurrences* de la lettre a dans m, noté $|m|_a$, est le nombre de fois où la lettre a apparaît dans m. Ainsi par exemple, $|abcaab|_a = 3$ et, pour toute lettre a, $|\varepsilon|_a = 0$.

Soient u et v deux mots de A*. La *concaténation* de u et v est le mot, noté u.v ou plus simplement uv, obtenu en « collant » le mot v à la suite du mot u. Ainsi, $|uv| = |u| + |v|$ et, pour toute lettre a, $|uv|_a = |u|_a + |v|_a$. Par exemple, si u = aba et v = cabac, alors uv = abacabac. Notons que pour tout mot u, $\varepsilon u = u\varepsilon = u$ (ε est élément neutre pour la concaténation) et, pour tous mots u, v et w, u.vw = uv.w = uvw (la concaténation est associative).

Un *langage* L sur l'alphabet A est un sous-ensemble, fini ou infini, de A*. Par exemple, sur l'alphabet A = {a, b}, on peut définir les langages suivants :

- $L1 = A^2 = \{ aa, ab, ba, bb \}$,
- L2 = { mots d'au plus quatre lettres ayant autant de a que de b } = { ε, ab, ba, aabb, abab, abba, baab, baba, bbaa },
- L3 = { mots ayant deux fois plus de a que de b } (ce langage est infini).

5.2. Opérations sur les langages

Soit A un alphabet. On définit les opérations suivantes sur les langages définis sur A*:

Union.

$$L1 \cup L2 = \{ m \in A^* / (m \in L1) \text{ ou } (m \in L2) \}$$

Intersection.

$$L1 \cap L2 = \{ m \in A^* / (m \in L1) \text{ et } (m \in L2) \}$$

Produit (de concaténation).

$$L1.L2 = L1L2 = \{ m \in A^* / m = m_1 m_2, m_1 \in L1 \text{ et } m_2 \in L2 \}$$

Puissance.

$$L^0 = \{ \varepsilon \}$$

$$L^n = \{ m \in A^* / m = m_1 m_2 ... m_n, m_i \in L \text{ pour tout } i, 1 \leq i \leq n \}, \text{ pour } n \geq 1$$

Étoile et « plus ».

$$L^* = L^0 \cup L^1 \cup ... \cup L^n \cup ...$$

$$L^+ = L^1 \cup L^2 \cup ... \cup L^n \cup ... \qquad \text{(i.e. } L^+ = L.L^* \text{)}$$

On observera que la notation A*, désignant l'ensemble des mots définis sur A, correspond exactement à l'opération étoile appliquée à l'alphabet lui-même, considéré comme un langage. De même, la notation A^n, désignant l'ensemble des mots de longueur n définis sur A, correspond exactement à l'opération puissance.

Exemples. Soient A = { 0, 1 }, et L1, L2, L3 et L4 les langages suivants :

- L1 = { mots finissant par 1 }
- L2 = { mots ayant autant de 0 que de 1 }
- L3 = { palindromes }
- $L4 = A^4$

Nous avons alors :

- $L2 \cap L4 = \{ 0011, 0101, 0110, 1001, 1010, 1100 \}$
- $(L1.L2) \cap L4 = \{ 0101, 0110, 1101, 1110, 0001, 0011, 0111, 1001, 1011, 1111 \}$
- $(L3 \cup L2) \cap L4 = \{ 0011, 0101, 0110, 1001, 1010, 1100, 0000, 1111 \}$
- $L3^* = A^*$ (car 0 et 1 sont des palindromes...)
- $L2^+ = L2$ (voyez-vous pourquoi ?)

5.3. Langages rationnels et langages reconnaissables

5.3.1. Langages rationnels

Un langage sur un alphabet A est *rationnel* (on dit également *régulier*), s'il peut se construire à l'aide des opérations union, étoile et produit à partir des langages élémentaires composés du mot vide ou d'un mot à une seule lettre.

De façon inductive, un langage rationnel se définit donc ainsi :

- $\{ \varepsilon \}$ est un langage rationnel,
- si $a \in \mathbf{A}$, { a } est un langage rationnel,
- si L est un langage rationnel alors L* est un langage rationnel,
- si L1 et L2 sont des langages rationnels, alors $L1 \cup L2$ et L1.L2 sont des langages rationnels.

Notons que l'on peut également utiliser l'opérateur « plus », car $L^+ = L.L^*$, ainsi que l'opérateur « puissance », car $L^n = L.L…L$ (n fois).

Prenons par exemple A = { a, b, c }. Les langages suivants sont rationnels :

- $L1 = (\{ \varepsilon \} \cup \{ a \}).(\{ b \} \cup \{ c \})^*.\{ a \}^3$
  - L1 = { aaa, bcbbcaaa, ..., aaaa, abaaa, ... }
- $L2 = \{ c \}^2.(\{ a \} \cup \{ b \}^* \cup (\{ a \} \cup \{ b \}.\{ c \})^2)^3$
  - L2 = { ccaaa, cc, cca, ccaaaaaa, ccbcabcabcbc, ... }
- $L3 = (\{ a \} \cup \{ b \})^*.\{ c \}$

- L3 = { c, ac, bc, aac, abc, bac, bbc, aaac, ... }

On utilise habituellement quelques conventions d'écriture qui permettent d'alléger les expressions précédentes : un langage singleton { a } est noté simplement a, l'union est notée simplement +. Ainsi, les langages précédents peuvent s'écrire, plus simplement, sous forme de ce que l'on appelle des *expressions rationnelles* :

- $L1 = (\varepsilon + a)(b + c)^* a^3$
- $L2 = c^2(a + b^* + (a + bc)^2)^3$
- $L3 = (a + b)^* c$

Sur l'alphabet A = {0, 1}, nous avons :

- $(11 + 01)^*$ → { ε, 11, 01, 1111, 1101, 0101, 0111, 111111, ... }
- 0(0+1)+11 → { 0011, 0111, 00011, 00111, 01011, 01111, ... }

5.3.2. Langages reconnaissables

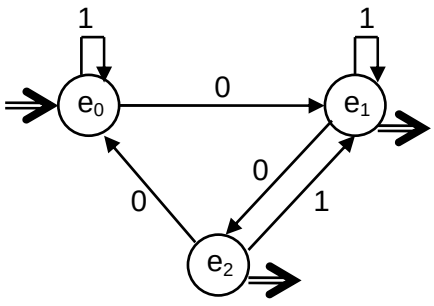
On va maintenant s'intéresser à des « machines » (que l'on appellera *automates*), capables de « lire » des mots, puis de les accepter ou de les refuser. Le langage *reconnu* par un automate est alors l'ensemble des mots que celui-ci accepte. On dira enfin qu'un langage est *reconnaissable* s'il existe un automate qui le reconnaît.

C'est la notion d'automate, vu sous l'angle d'un « graphe orienté valué capable de reconnaître des mots », qui est au programme de la classe de terminale ES.

Plus formellement, un automate (fini, déterministe) est un 5-uplet $< \Sigma, E, e_0, F, \delta >$ où :

- Σ est *l'alphabet d'entrée,*
- E est un ensemble fini d'*états,*
- $e_0 \in E$ est *l'état initial,*
- $F \subseteq E$ est l'ensemble des *états terminaux,*
- $\delta : E \times \Sigma \rightarrow E$ est la *fonction de transition.*

Un automate se représente à l'aide d'un graphe orienté dont les sommets représentent les états et dont les arcs matérialisent la fonction de transition (ils sont valués par une lettre de l'alphabet). L'état initial est repéré par une flèche entrante et les états terminaux par une flèche sortante :



Cet automate se définit formellement ainsi :

- $\Sigma = \{ 0, 1 \}$,
- $E = \{ e_0, e_1, e_2 \}$,
- $F = \{ e_1, e_2 \}$,
- la fonction $\delta : E \times \Sigma \rightarrow E$ est donnée par l'ensemble de triplets suivants :

$$\{ (e_0, 0, e_1), (e_0, 1, e_0), (e_1, 0, e_2), (e_1, 1, e_1), (e_2, 0, e_0), (e_2, 1, e_1) \}$$

Notons que le fait que δ soit une fonction, implique qu'on de doit jamais avoir deux arcs issus d'un même sommet et associés à la même valeur (lettre de l'alphabet).

Intuitivement, l'automate fonctionne de la façon suivante : on se place sur l'état initial et on « lit » le mot à analyser lettre à lettre, en suivant les transitions (arcs) correspondantes à la lettre lue. Lorsque le mot est entièrement lu, il est accepté si et seulement si on se trouve sur l'un des états terminaux.

Ainsi par exemple, pour le mot 110110101000, l'automate effectue le parcours suivant :

$$[\text{état initial } e_0]\ e_0\, e_0\, e_1\, e_1\, e_1\, e_2\, e_1\, e_2\, e_1\, e_2\, e_3\, e_1$$

Le mot est donc accepté car le dernier état (e_1) est un état terminal.

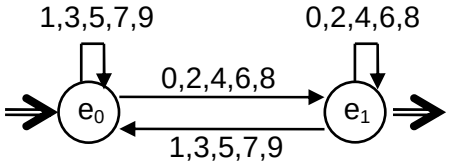
Plus formellement, on étend la fonction δ aux mots en posant $\delta(e,m) = \delta(\delta(e,a),m')$ si $m = am'$. Le langage reconnu par un automate est alors :

$$L = \{ m \in A^* / \delta(e_0,m) \in F \},$$

soit l'ensemble des mots de A^* qui conduisent de l'état initial à l'un des états terminaux.

On pourra « s'amuser » à trouver le langage reconnu par notre automate exemple !...

L'automate suivant reconnaît les entiers positifs, définis sur l'alphabet A = { 0, 1, …, 9 }, multiples de 2. Par convention, un arc étiqueté a, b, etc. correspond à un ensemble d'arcs étiquetés respectivement a, b, etc.



Deux automates finis déterministes sont *équivalents* s'ils reconnaissent le même langage. Il existe un algorithme (de *minimisation*) permettant, à partir d'un automate donné, d'obtenir un automate équivalent *minimal* (en termes de nombre d'états).

5.3.3. Théorème de Kleene

Ce théorème est un résultat majeur de la théorie des langages :

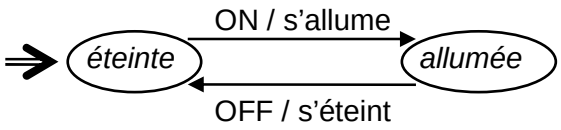
Théorème [Kleene, 1956]. Un langage est rationnel si et seulement si il est reconnaissable.

Ainsi, tout langage reconnu par un automate fini déterministe peut être défini par une expression rationnelle, et réciproquement. Il existe des algorithmes permettant de construire une expression rationnelle à partir d'un automate ou de construire un automate à partir d'une expression rationnelle.

5.4. Automates avec actions

Les automates sont également utiles dans le cadre de la modélisation de *systèmes*. Un système est vu comme une entité pouvant *réagir* à certains événements. Ces réactions seront modélisées par des *actions*, associées aux transitions de l'automate. Ainsi, lorsque l'automate « lit » une lettre (c'est-à-dire reçoit un événement), il change éventuellement d'état et réalise une action (sa réponse à l'événement).

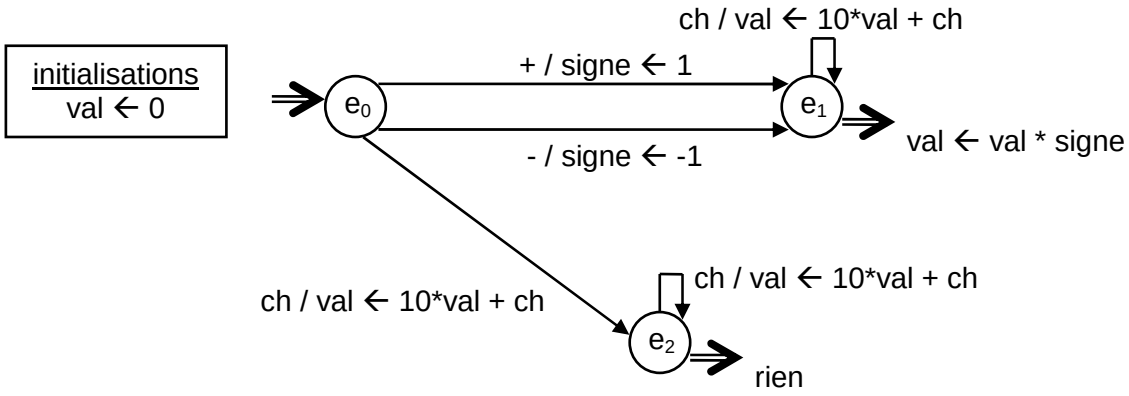
L'exemple suivant modélise le comportement d'une ampoule, susceptible de réagir à deux événements, ON et OFF. Notre ampoule peut alors se trouver dans deux états : *éteinte* ou *allumée*. On obtient alors :



Notons que dans cet exemple, notre automate ne possède pas d'état terminal (le système modélisé a une durée de vie illimitée).

Ceci n'est pas toujours le cas, et il est alors possible d'associer une action à la « sortie » par un état terminal. De même, on associe généralement une action (d'initialisation) à l'entrée dans l'automate.

L'exemple suivant permet de déterminer la valeur d'un entier relatif, signé ou pas, lu chiffre par chiffre (*rien* représente une action « vide »…) :



Chapitre 6. Graphes probabilistes

Dans ce chapitre, nous utiliserons à nouveau des graphes orientés valués (une valeur sera associée à chaque arc) que l'on rencontre également sous l'appellation *graphes pondérés* (on parle alors de *poids* des arcs plutôt que de valeur).

6.1. Graphes probabilistes

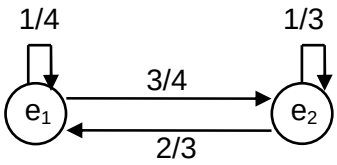
On va s'intéresser à l'évolution de systèmes qui peuvent se trouver dans certains *états*, que l'on notera e_1, e_2, … et changer d'état selon certaines *probabilités de transition*. Ces systèmes se modélisent de façon tout à fait naturelle par des graphes orientés valués spécifiques que l'on appellera *graphes probabilistes* (on trouve également l'appellation *graphes de transition*).

Un graphe probabiliste est un graphe orienté valué G = (E, T, p), où E = { e_1, …, e_k } est un ensemble fini d'états, T un ensemble de transitions (arcs) valués par une fonction p leur associant un nombre réel compris entre 0 et 1, correspondant à une probabilité de transition, et vérifiant la propriété suivante :

pour tout état e_i, la somme des probabilités des transitions issues de e_i vaut 1.

On notera $p_{i,j}$ la probabilité de transition de e_i vers e_j.

Voici un exemple simple de graphe probabiliste à deux états :

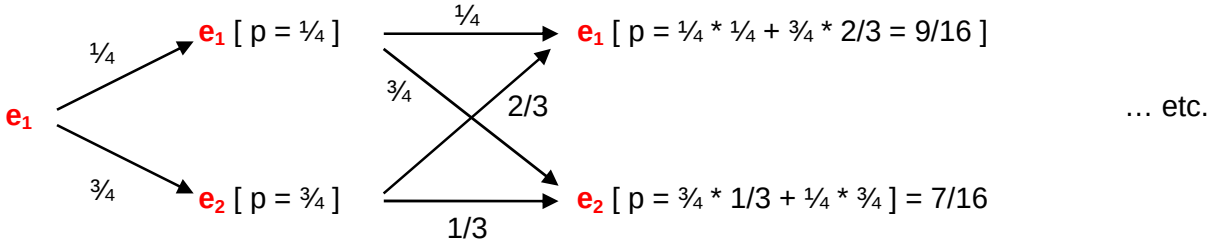


Deux transitions sont issues de e_1, dont la somme des probabilités vaut 0.25 + 0.75 = 1. De même, la somme des probabilités des transitions issues de e_2 vaut 0.33 + 0.67 = 1.

Pour un tel système, on peut définir une *loi de probabilité* sur l'ensemble de ses états, appelée *état probabiliste*, donnant la probabilité de se retrouver dans chacun des états après n transitions (on dit également *à l'instant n*). Cette loi de probabilité peut être représentée par une matrice ligne P_k (indicée par les états). Supposons par exemple que notre système se trouve initialement dans l'état e_1. Nous avons alors :

$$P_0 = [\ 1 \quad 0\ ], \quad P_1 = [\ ¼ \quad ¾\ ], \quad P_2 = [\ 9/16 \quad 7/16\ ], \quad \text{etc.}$$

Ces valeurs se calculent aisément à l'aide de probabilités conditionnelles. Ce calcul est illustré par le schéma suivant :



Remarquons que dans le cas d'un graphe probabiliste à deux états, tous nos états probabilistes sont naturellement de la forme [a 1-a].

6.2. Matrices de transition

À un tel graphe probabiliste, on associe une *matrice de transition* M permettant de retrouver les valeurs des différentes transitions. Sur l'exemple précédent, la matrice est la suivante :

	e1	e2
e1	0.25	0.75
e2	0.67	0.33

$$M = \begin{pmatrix} 0.25 & 0.75 \\ 0.67 & 0.33 \end{pmatrix}$$

Plus formellement, la matrice de transition M associée à un graphe probabiliste G = (E, T, p), avec |E| = k, est la matrice k x k définie par M[i,j] = $p_{i,j}$ (par convention, $p_{i,j}$ = 0 s'il n'existe pas de transition allant de e_i vers e_j).

On remarque aisément le lien entre cette matrice et les états probabilistes vus précédemment. Nous avons en effet $P_1 = P_0 \times M$, $P_2 = P_1 \times M = P_0 \times M^2$, et donc, plus généralement, $P_n = P_0 \times M^n$ pour tout $n \geq 0$ (P_0 représentant l'état initial du système).

6.3. État stable

On dit qu'un état probabiliste P_n est *stable* si $P_{n+1} = P_n \times M = P_n$ (cet état n'évoluera donc plus).

Pour déterminer l'état stable d'un système ayant pour matrice de transition M, il suffit donc de résoudre l'équation P = P x M, avec P = [a, 1-a], $0 \leq a \leq 1$.

L'existence d'un état stable est assurée dans les cas suivants :

Théorème. Si tous les coefficients de M sont strictement positifs, la suite des états probabilistes $(P_n)_{n\geq 0}$ converge vers un état stable π indépendant de l'état initial. Par ailleurs, la suite $(M^n)_{n\geq 0}$ converge vers une matrice M* dont toutes les lignes sont égales à π.

Dans le cas d'un système à deux états, cet état stable se calcule aisément. Nous devons en effet résoudre le système suivant :

$$\begin{pmatrix} a & 1-a \end{pmatrix} = \begin{pmatrix} a & 1-a \end{pmatrix} \times \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}$$

d'où a = q/(p+q) et donc :

$$\pi = \begin{pmatrix} \frac{q}{p+q} & \frac{p}{p+q} \end{pmatrix}$$

6.4. Chaînes de Markov

Tout ceci se fait naturellement en classe de Terminale ES sans évoquer la notion de *chaîne de Markov*. Nous rappelons brièvement ici les notions correspondantes.

Un *vecteur stochastique* de $\mathbb{R}^n$, n > 0, est un vecteur $X = [X_1, \ldots, X_n]$ tel que $X_i \geq 0$ pour tout i, $1 \leq i \leq n$, et $X_1 + \ldots + X_n = 1$. Un matrice n x n à coefficient réels est une *matrice stochastique* si toutes ses lignes sont des vecteurs stochastiques.

Nous avons alors les propriétés élémentaires suivantes :

- si X est un vecteur stochastique de $\mathbb{R}^n$ et M une matrice stochastique n x n, alors XM est un vecteur stochastique de $\mathbb{R}^n$,
- si P et Q sont deux matrices stochastiques n x n, alors PQ est une matrice stochastique n x n.

Ainsi notamment, si M une matrice stochastique n x n, alors M^k est une matrice stochastique n x n pour tout k (même k=0 puisque la matrice identité est stochastique).

Fixons maintenant un ensemble S d'états. Une *chaîne de Markov* à temps discret sur S est une suite de variables aléatoires à valeurs dans S, $X = (X_n)_{n\geq 0}$ satisfaisant les deux conditions suivantes :

- *indépendance* : pour tout $k \geq 0$ et pour tout choix de $i_0, i_1, \ldots, i_{k-1}$ et i dans S pour lesquels $\mathbb{P}[(X_0,X_1,\ldots,X_k) = (i_0, i_1, \ldots,i_{k-1},i)] > 0$, on a :

$$\mathbb{P}[X_{k+1} = j \mid (X_0,X_1,\ldots,X_k) = (i_0, i_1, \ldots,i_{k-1},i)] = \mathbb{P}[X_{k+1} = j \mid X_k = i]$$

- *homogénéité* : pour tous i et j dans S et pour tous les entiers k tels que $\mathbb{P}[X_k = i] > 0$, la probabilité conditionnelle $\mathbb{P}[X_{k+1} = j \mid X_k = i]$ ne dépend pas de k.

Notons que les systèmes précédents, tels que nous les avons définis par l'intermédiaire de graphes probabilistes, vérifient ces deux conditions.

Si $X = (X_n)_{n\geq 0}$ est une chaîne de Markov sur un ensemble S, la matrice $M = (m_{i,j})_{i,j\in S\times S}$ défini par

$$m_{i,j} = \mathbb{P} [X_{k+1} = j \mid X_k = i]$$

est la *matrice de transition* de la chaîne de Markov (on vérifie aisément qu'il s'agit d'une matrice stochastique). La distribution $\pi_0 = (\mathbb{P} [X_0 = i])_{i\in S}$ est la *loi initiale* de la chaîne de Markov.

Notons S = {1, 2, …, s} l'ensemble des états et, pour tout instant k,

$$\pi_k = (\mathbb{P}[X_k = 1], \mathbb{P}[X_k = 2], \ldots, \mathbb{P}[X_k = s].$$

On a alors, pour tout entier k, $\pi_{k+1} = \pi_k M$. De plus, nous avons :

Théorème. Si tous les coefficients de M sont strictement positifs, la suite (π_k) converge vers un vecteur π indépendant de l'état initial, et la suite (M^m) converge vers une matrice M^∞ dont toutes les lignes sont égales à π.

On aura naturellement reconnu en π notre état stable…

Chapitre 7. Corpus d'exercices

7.1. Notions de base

7.1.1. Modélisation

Exercice 1. Construction de graphe.

Construire un graphe orienté dont les sommets sont les entiers compris entre 1 et 12 et dont les arcs représentent la relation « être diviseur de ».

Exercice 2. Traversée du fleuve.

Une chèvre, un chou et un loup se trouvent sur la rive d'un fleuve ; un passeur souhaite les transporter sur l'autre rive mais, sa barque étant trop petite, il ne peut transporter qu'un seul d'entre eux à la fois. Comment doit-il procéder afin de ne jamais laisser ensemble et sans surveillance le loup et la chèvre, ainsi que la chèvre et le chou ?

Exercice 3. Maris jaloux.

Trois maris jaloux et leurs épouses souhaitent traverser une rivière. Ils disposent d'une barque qui ne peut transporter plus de deux personnes à la fois. Comment doivent-ils procéder, sachant qu'aucune femme ne doit rester en compagnie d'un ou deux hommes sans que son mari soit présent ?

Montrez que ce problème n'a pas de solution si les couples sont au nombre de 4.

Exercice 4. Le tonneau.

On souhaite prélever 4 litres de liquide dans un tonneau. Pour cela, nous avons à notre disposition deux récipients (non gradués !), l'un de 5 litres, l'autre de 3 litres... Comment doit-on procéder ?

Exercice 5. Jeu de Fan Tan.

Deux joueurs disposent de 2 ou plusieurs tas d'allumettes. A tour de rôle, chaque joueur peut enlever un certain nombre d'allumettes de l'un des tas (selon la règle choisie). Le joueur qui retire la dernière allumette perd la partie.

- Modéliser ce jeu à l'aide d'un graphe dans le cas où l'on dispose au départ de deux tas contenant chacun trois allumettes, et où un joueur peut enlever une ou deux allumettes à chaque fois.
- Que doit jouer le premier joueur pour gagner la partie à coup sûr ?
- Mêmes questions avec 3 tas de 3 allumettes.

Exercice 6. Modélisation.

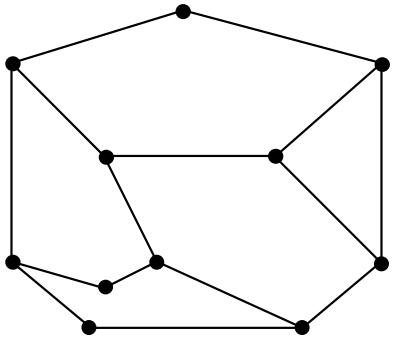
Essayez d'exprimer (et non nécessairement de résoudre…) en termes de graphes les problèmes suivants :

- Peut-on placer huit dames sur un échiquier sans qu'aucune d'elles ne puisse en prendre une autre ?

- Un cavalier peut-il se déplacer sur un échiquier en passant sur chacune des cases une fois et une seule ?
- Combien doit-on placer de dames sur un échiquier 5x5 afin de contrôler toutes les cases ?

Exercice 7. Surveillance d'un musée.

Le graphe ci-dessous représente le plan des couloirs d'un musée. Un gardien placé dans un couloir peut surveiller les deux carrefours placés à ses extrémités. Combien de gardiens sont nécessaires (et comment les placer) afin que tous les carrefours soient surveillés ?



Si l'on place maintenant les gardiens aux carrefours, en supposant qu'un tel gardien peut surveiller tous les couloirs amenant à ce carrefour, combien de gardiens sont nécessaires pour surveiller tous les couloirs ?

Exercice 8. La bibliothèque.

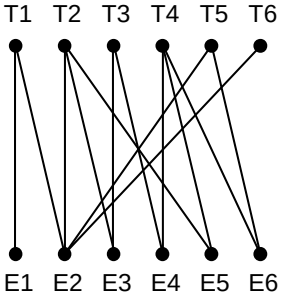
Sept élèves, désignés par A,B,C,D,E,F et G se sont rendus à la bibliothèque aujourd'hui. Le tableau suivant précise « qui à rencontré qui » (la bibliothèque étant petite, deux élèves présents au même moment se rencontrent nécessairement…).

- Quel est l'ordre d'arrivée des élèves à la bibliothèque ?
- leur ordre de départ ?

élève	**A**	**B**	**C**	**D**	**E**	**F**	**G**
a rencontré	D,E	D,E,F,G	E,G	A,B,E	A,B,C,D,F,G	B,E,G	B,C,E

Exercice 9. Affectation d'emplois.

Dans le graphe biparti suivant, les sommets T1, …, T6 représentent des travailleurs et les sommets E1, …, E6 des emplois. Une arête relie un travailleur à un emploi si le travailleur a les qualifications nécessaires pour occuper cet emploi.



Comment affecter les emplois aux travailleurs afin de minimiser le nombre de sans-emploi ?

Exercice 10. Promenade.

Chaque jour, un groupe de 12 enfants fait une promenade, par rang de deux. Combien de jours peuvent-ils se promener si l'on souhaite qu'un enfant n'ait jamais deux fois le même voisin ? Et si maintenant la promenade se fait par rang de trois ?

Exercice 11. Les lapins.

Soit X un ensemble de lapins, et G un graphe orienté ayant X pour ensemble de sommets. On dit que G est un « graphe de parenté » si les arcs de G codent la relation « être l'enfant de »… Quelles conditions doit nécessairement vérifier G pour pouvoir être un graphe de parenté ?

7.1.2. Degré des sommets

Exercice 12. Graphes 3-réguliers.

On s'intéresse aux graphes dont tous les sommets sont de degré trois.

- Construisez de tels graphes ayant 4 sommets, 5 sommets, 6 sommets, 7 sommets.
- Qu'en déduisez-vous ?
- Prouvez-le !

Exercice 13. Graphes 4-réguliers.

La situation est-elle identique pour les graphes dont tous les sommets sont de degré 4 ?

Exercice 14. Suites graphiques (graphes non orientés).

Une suite décroissante (au sens large) d'entiers est graphique s'il existe un graphe dont les degrés des sommets correspondent à cette suite (par exemple, le triangle à trois sommets correspond à la suite 2,2,2). Les suites suivantes sont-elles graphiques ?

- 3, 3, 2, 1, 1
- 3, 3, 1, 1
- 3, 3, 2, 2
- 4, 2, 1, 1, 1, 1
- 5, 3, 2, 1, 1, 1
- 5, 4, 3, 1, 1, 1, 1

Trouvez deux graphes distincts (c'est-à-dire non isomorphes) correspondant à la suite 3, 2, 2, 2, 1.

[Deux graphes G1 et G2 sont isomorphes s'il existe une bijection f entre leurs ensembles de sommets qui préserve les arêtes (f(x)f(y) est une arête de G2 si et seulement si xy est une arête de G1). De façon plus intuitive, cela signifie que l'on peut « renommer » les sommets de G1 de façon à obtenir G2…]

Exercice 15. Suites graphiques (graphes orientés).

Pour les graphes orientés, il faut considérer des suites de couples d'entiers (le premier élément d'un couple correspond au degré entrant, le second au degré sortant). Les suites suivantes sont-elles des suites graphiques ?

- (0,1), (1,1), (1,1), (1,1), (1,0)
- (1,1), (1,1), (1,1), (1,1), (1,1)
- (0,2), (1,1), (1,1), (1,1)
- (0,2), (1,1), (1,1), (2,0)

- (1,2), (1,2), (2,1), (2,1)
- (1,2), (1,2), (2,1), (2,2), (1,1)

Exercice 16. Degrés distincts.

Essayez de construire un graphe non orienté ayant au moins deux sommets et tel que tous les sommets ont des degrés distincts. Qu'en déduisez-vous ?

Exercice 17. Les six personnes.

Montrez que dans un groupe de six personnes, il y en a nécessairement trois qui se connaissent mutuellement ou trois qui ne se connaissent pas (on suppose que si A connaît B, B connaît également A).

Montrez que cela n'est plus nécessairement vrai dans un groupe de cinq personnes.

Exercice 18. Les neuf personnes.

Montrez que dans un groupe de 9 personnes, 4 se connaissent mutuellement ou 3 ne se connaissent pas.

Cela est-il toujours vrai dans un groupe de 8 personnes ?

Exercice 19. Les amis.

Montrez que dans un groupe de personnes, il y a toujours deux personnes ayant le même nombre d'amis présents.

Exercice 20. Le groupe mystère.

Un groupe de personnes est tel que

- (i) chaque personne est membre d'exactement deux associations,
- (ii) chaque association comprend exactement trois membres,
- (iii) deux associations quelconques ont toujours exactement un membre en commun.

Combien y a-t-il de personnes ? d'associations ?

7.2. Coloration de graphes

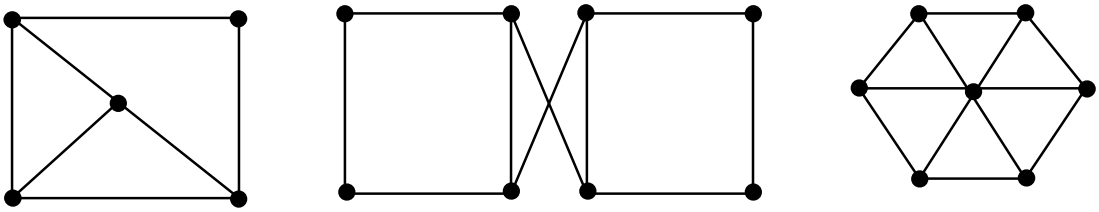
Exercice 21. Triangles.

Tout graphe contenant un triangle (K3) ne peut être colorié en moins de trois couleurs.

- Construire un graphe sans triangle qui nécessite également trois couleurs.
- Comment, à partir du graphe précédent, construire un graphe sans K4 nécessitant 4 couleurs ?
- Un graphe sans K5 nécessitant 5 couleurs ?

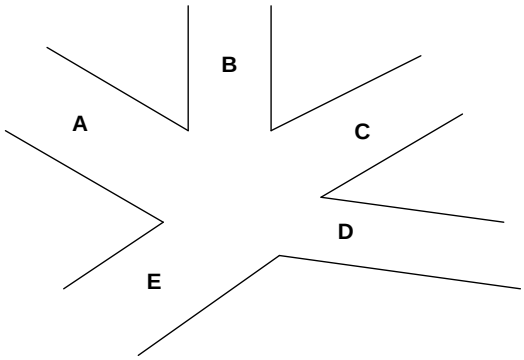
Exercice 22. Nombre chromatique.

Déterminer le nombre chromatique des graphes suivants :



Exercice 23. Carrefour.

Le schéma suivant représente un carrefour.



Le tableau suivant précise les « franchissements » possibles de ce carrefour.

En arrivant par…	A	B	C	D	E
Il est possible d'aller en…	C,E	A,E,D	A,D	C,A	C,D

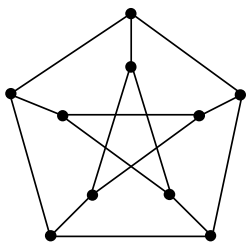
Les franchissements A-C et B-E ne peuvent naturellement pas être autorisés simultanément…

- Modélisez ces incompatibilités à l'aide d'un graphe dont les sommets représentent les franchissements possibles et les arêtes les incompatibilités entre franchissements.
- Proposez une coloration du graphe ainsi obtenu.
- Que peut-on dire d'un ensemble de sommets ayant même couleur ?
- À quoi peut correspondre le nombre chromatique de ce graphe ?

Exercice 24. Graphe de Petersen.

On cherche à colorier le graphe ci-dessous en utilisant des entiers positifs de façon telle que deux sommets voisins ont des couleurs dont la différence, en valeur absolue, est au moins égale à trois.

- Proposez une coloration de ce graphe. Quel est le plus grand entier utilisé ?
- Peut-on faire mieux ?
- Maintenant, on souhaite que, de plus, deux sommets à distance deux aient des couleurs dont la différence, en valeur absolue, est au moins égale à deux. Quelle est la meilleure coloration possible de ce graphe ?



Exercice 25. Places assises.

Sept élèves, désignés par A,B,C,D,E,F et G se sont rendus à la bibliothèque aujourd'hui. Le tableau suivant précise « qui à rencontré qui » (la bibliothèque étant petite, deux élèves présents au même moment se rencontrent nécessairement…).

élève	A	B	C	D	E	F	G
a rencontré	D,E	D,E,F,G	E,G	A,B,E	A,B,C,D,F,G	B,E,G	B,C,E

De combien de places assises doit disposer la bibliothèque pour que chacun ait pu travailler correctement au cours de cette journée ?

Exercice 26. Encadrements du nombre chromatique.

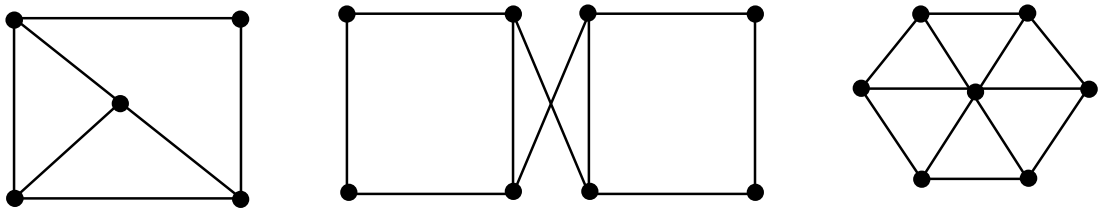
- Trouvez un graphe G pour lequel $\omega(G) > n / \alpha(G)$.
- Trouvez un graphe G pour lequel $\omega(G) < n / \alpha(G)$.
- Trouvez un graphe G pour lequel $\omega(G) = n / \alpha(G)$.
- Trouvez un graphe G pour lequel $\omega(G) = \Delta(G) + 1$.

Exercice 27. Algorithme First-Fit et nombre chromatique.

Montrez que pour tout graphe G, il existe un ordre de ses sommets pour lequel l'algorithme First-Fit produit une coloration optimale, c'est-à-dire utilisant $\chi(G)$ couleurs.

Exercice 28. Algorithme de Welsh et Powell.

Appliquer l'algorithme de Welsh et Powell aux graphes suivants :



Exercice 29. Graphes sans triangles.

Construisez des graphes sans triangles de nombre chromatique respectifs 3, 4, 5, …

Exercice 30. K5 et formule d'Euler.

En utilisant la formule d'Euler, montrez que le graphe K_5 n'est pas planaire.

Exercice 31. $K_{3,3}$ et formule d'Euler.

En utilisant la formule d'Euler, montrez que le graphe $K_{3,3}$ n'est pas planaire.

Exercice 32. Ballon de football.

En utilisant la formule d'Euler, montrez qu'un ballon de football, composé d'hexagones et de pentagones, contient nécessairement 12 pentagones.

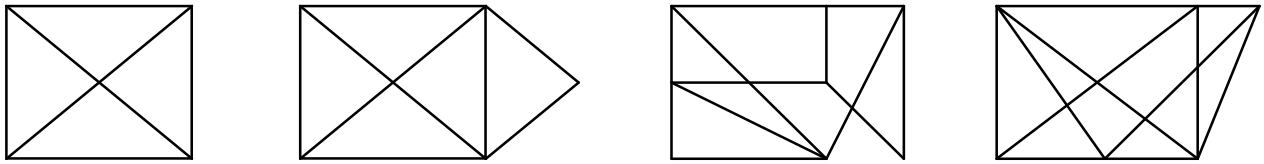
Exercice 33. Graphes planaires sans triangles.

Montrez qu'un graphe planaire à n sommets, $n \geq 3$, ne contenant pas de triangles, possède au maximum 2n-4 arêtes.

7.3. Graphes eulériens

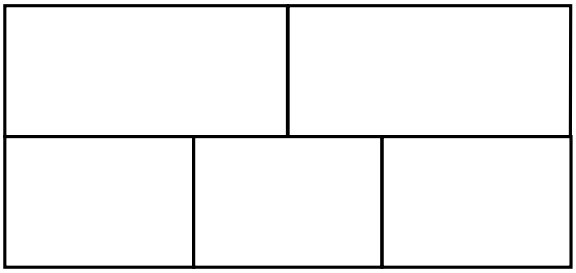
Exercice 34. Dessins.

Est-il possible de tracer les figures suivantes sans lever le crayon (et sans passer deux fois sur le même trait !…) ? Pourquoi ?



Exercice 35. Couper les segments.

Est-il possible de tracer une courbe, sans lever le crayon, qui coupe chacun des 16 segments de la figure suivante ?



Exercice 36. Les ponts de Koenigsberg, variante.

Est-il possible de traverser les sept ponts de la ville de Koenigsberg en empruntant deux fois chaque pont, dans un sens puis dans l'autre ?

Exercice 37. Rendre un graphe eulérien.

Soit G un graphe non eulérien. Est-il toujours possible de rendre G eulérien en lui rajoutant un sommet et quelques arêtes ?

Exercice 38. Dominos.

On considère des dominos dont les faces sont numérotées 1, 2, 3, 4 ou 5.

- En excluant les dominos doubles, de combien de dominos dispose-t-on ?
- Montrez que l'on peut arranger ces dominos de façon à former une boucle fermée (en utilisant la règle habituelle de contact entre les dominos).
- Pourquoi n'est-il pas nécessaire de considérer les dominos doubles ?
- Si l'on prend maintenant des dominos dont les faces sont numérotées de 1 à n, est-il possible de les arranger de façon à former une boucle fermée ?

7.4. Problèmes de chemins

Exercice 39. Tournois.

Un tournoi est un graphe orienté tel que toute paire de sommets est reliée par un arc, dans un sens ou dans l'autre (mais pas dans les deux sens).

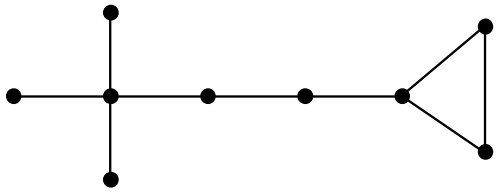
- Pourquoi, selon vous, appelle-t-on de tels graphes des tournois ?

- Montrez que si un tournoi contient un circuit de longueur k, alors il contient également des circuits de longueur k', pour tout k', $3 \leq k' < k$ (une « preuve » à l'aide d'un dessin suffit…).
- Dessinez un tournoi à 6 sommets ne possédant pas de circuit de longueur 4.

Exercice 40. Le robot.

Un robot se promène sur le graphe ci-dessous. Partant d'un sommet quelconque s, appelé sommet de stockage, il doit déposer un cube sur chacun des autres sommets. Il possède suffisamment de cubes sur le sommet de stockage, mais ne peut transporter qu'un cube à la fois (il doit donc repasser par le sommet de stockage avant de livrer un autre cube). Calculer, pour chacun des sommets du graphe, le trajet minimum que doit parcourir le robot si ce sommet est sommet de stockage.

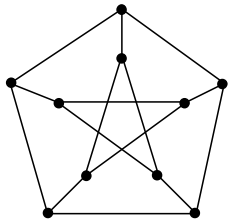
Quel est le « meilleur » sommet de stockage ?



Exercice 41. Petersen (bis).

Considérons le graphe ci-dessous.

- Combien de cycles simples (sans répétition d'arêtes) de longueur 5 ce graphe contient-il ?
- De longueur 6 ?
- De longueur 8 ?
- De longueur 9 ?



Exercice 42. Les diviseurs.

Construire le graphe orienté dont les sommets sont les entiers compris entre 1 et 24 et dont les arcs relient x à y lorsque x divise y. De plus, les arcs sont valués par le quotient y/x (ainsi, l'arc allant de 3 vers 15 a la valeur 5).

Comment reconnaît-on dans ce graphe un nombre premier ?

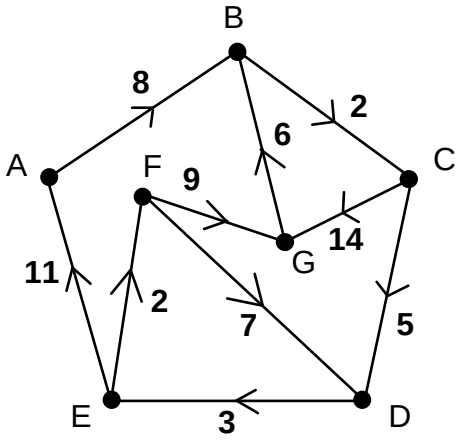
Comment retrouver dans ce graphe la décomposition d'un nombre en facteurs premiers ?

Exercice 43. Plus courts chemins.

(o) Remplir le tableau suivant qui, pour le graphe valué ci-dessous, donne la valeur du plus court chemin d'un sommet à un autre.

	A	B	C	D	E	F	G
A							
B							
C							

D							
E							
F							
G							



Exercice 44. Algorithme de Dijkstra.

Exécutez l'algorithme de Dijkstra sur le graphe précédent, à partir du sommet C, puis à partir du sommet F.

Exercice 45. Compagnie aérienne.

La compagnie Europ'Air dessert différentes villes européennes. Le tableau ci-dessous donne les durées de vol entre ces différentes villes.

Comment déterminer le trajet le plus rapide entre deux villes ?

Comment modifier la méthode précédente afin de prendre en compte la durée des escales dans les différentes villes ?

	A	**B**	**C**	**D**	**E**
A		1h30	2h00		2h15
B	1h40				3h00
C	2h20			2h55	
D			3h20		1h05
E	2h25	3h10	1h10		

7.5. Problèmes d'automates

7.5.1. Automates simples

Exercice 46. Construction d'automate 1.

Proposez un automate déterministe permettant de reconnaître un entier positif inférieur ou égal à 138.

Exercice 47. Construction d'automates 2.

Proposez des automates déterministes permettant de reconnaître :

- les multiples de 3,

- les multiples de 9,
- les multiples de 10,
- les multiples de 100,
- les multiples de 1000,
- les multiples de 50,
- les multiples de 25,
- les multiples de 250,
- les multiples de 6.

Exercice 48. Construction d'automates 3.

Proposez des automates déterministes permettant de reconnaître :

- les mots binaires finissant par 110,
- les mots binaires contenant 110,
- les mots binaires ayant un nombre pair de 0,
- les mots binaires ayant un nombre pair de 0 et impair de 1.

Exercice 49. Les horaires.

Proposez un automate déterministe permettant de reconnaître un horaire donné sous la forme 12:15.

Exercice 50. Les dates.

Proposez un automate déterministe permettant de reconnaître une date donnée sous la forme 03/02 (pour le 3 février), en se limitant à l'année en cours…

7.5.2. Automates avec actions

Exercice 51. Les élèves.

Un élève peut être considéré comme un système très particulier : lorsqu'il est interrogé oralement par un enseignant, il répond s'il est en forme et ne répond pas s'il est fatigué. Après avoir été interrogé, un élève en forme devient fatigué. Lorsqu'une interrogation surprise se présente, l'élève en forme remet une bonne copie, l'élève fatigué une copie plutôt moyenne. Après une interrogation surprise, tous les élèves sont fatigués pour le reste de la journée ! Le soir, tous les élèves se reposent et arrivent en forme le lendemain matin.

Modélisez cette situation à l'aide d'un automate avec actions (on pourra dans un premier temps établir la liste des événements « subis » par un élève et la liste des actions réalisées par celui-ci en réponse à ces événements…)

Exercice 52. Le téléphone portable.

Un téléphone portable est finalement un objet assez simple… Quand vous le sortez de son emballage, il est éteint et toutes les touches sont sans effet… sauf la touche « ON », qui émet un « bip » sympathique et voici votre téléphone allumé… Dorénavant, toutes les touches émettent un « bip » (c'est amusant)… même la touche « OFF » qui, pourtant, éteint votre téléphone…

Modélisez le fonctionnement de ce téléphone portable à l'aide d'un automate avec actions.

Exercice 53. Le téléphone (suite).

On n'arrête pas le progrès… Votre téléphone est également muni d'une touche « MUTE » qui, naturellement, n'a aucun effet si votre téléphone est éteint, mais qui peut faire passer votre téléphone

(allumé en mode normal) en mode silencieux (après avoir émis cependant un « bip » ;-). En mode silencieux, les touches n'émettent plus de « bip », sauf la touche « MUTE » qui, du coup, fait repasser votre téléphone en mode normal… Là où l'on prend vraiment conscience du progrès accompli, c'est que lorsque vous rallumez un téléphone qui a été éteint en mode normal, il se rallume en mode normal, alors que s'il a été éteint en mode silencieux, il se rallume en mode silencieux (en émettant cependant un « bip »…). Étonnant, non ?

Proposez un nouvel automate avec actions pour ce téléphone « moderne »...

Exercice 54. Le Kidor-Dyn.

Le Kidor-Dyn est un animal fabuleux que peu d'entre nous ont eu le loisir de croiser sur leur chemin... À sa naissance (vers 7h00 du matin), le Kidor-Dyn a faim et soif. Lorsqu'il passe près d'une rivière, il boit, et cela lui suffit pour la journée. Lorsqu'il croise une belette, il la croque, et cela lui suffit pour la journée. Il peut cependant, au hasard des rencontres, croquer une seconde belette mais jamais trois dans la même journée. Lorsque le soleil se couche, le Kidor-Dyn fait sa toilette et s'endort, qu'il ait croisé ou pas de rivière ou de belette dans la journée... Le lendemain, il a à nouveau faim et soif, à moins qu'il n'ait rêvé, au cours de la nuit, de pluie (auquel cas il n'a pas soif) ou de belette (auquel cas il n'a pas faim). Par ailleurs, chaque fois qu'un enfant passe à proximité de lui dans la journée, le Kidor-Dyn chante une chanson...

Modéliser la vie trépidante du Kidor-Dyn à l'aide d'un automate avec actions.

7.5.3. Notions de théorie des langages

Exercice 55. Opérations sur les langages.

Les égalités suivantes sont-elles vérifiées pour tous langages L1, L2 et L3 ?

- $L1 (L2 \cup L3) = L1L2 \cup L1L3$
- $L1 (L2 \cap L3) = L1L2 \cap L1L3$
- $(L1 \cup L2)^* = L1^* \cup L2^*$
- $(L1 \cap L2)^* = L1^* \cap L2^*$

Exercice 56. Expressions rationnelles.

Sur l'alphabet A = { 0, 1 }, donnez des expressions rationnelles décrivant les langages suivants (basés sur la représentation binaire des nombres) :

- les nombres pairs
- les multiples de 8
- les nombres congrus à 3 modulo 16
- les mots contenant la séquence 011 (consécutifs)
- les mots sans 0, les mots sans 01, puis les mots sans 011
- les mots ayant un nombre pair de 1
- les mots ayant un nombre impair de 0
- les mots ayant un nombre pair de 1 et impair de 0

7.6. Graphes probabilistes

Exercice 57. L'allumeur de réverbère du Petit Prince (un classique s'il en est !...)

Chaque matin, l'allumeur de réverbère du Petit Prince change l'état de sa planète avec une probabilité 0,75. Au jour 0, le réverbère est éteint.

- Faites un arbre permettant de trouver l'état probabiliste du réverbère au deuxième jour.
- Décrivez cette situation à l'aide d'un graphe probabiliste.
- Soit *M* la matrice de transition associée à ce graphe. Vérifiez que M = N – ½ R, où :

$$N = \begin{pmatrix} ½ & ½ \\ ½ & ½ \end{pmatrix} \quad \text{et } R = \begin{pmatrix} ½ & -½ \\ -½ & ½ \end{pmatrix}$$

- Calculez N^2, R^2, NR et RN puis en déduire M^n pour tout entier naturel n.
- Au jour 0, le réverbère est allumé (respectivement éteint). Calculez la probabilité p_n (respectivement p'_n) que le réverbère soit allumé (respectivement éteint) au *n*-ième jour.

Exercice 58. Fumeurs et non fumeurs (Annales – Centres étrangers – juin 2005)

On a divisé une population en deux catégories : « fumeurs »et « non-fumeurs ». Une étude statistique a permis de constater que, d'une génération à l'autre,

- 60% des descendants de fumeurs sont des fumeurs,
- 10% des descendants de non-fumeurs sont des fumeurs.

On suppose que le taux de fécondité des fumeurs est le même que celui des non-fumeurs.

On désigne par :

- f_n le pourcentage de fumeurs à la génération de rang n,
- $g_n = 1 - f_n$ le pourcentage de non-fumeurs à la génération de rang n, où n est un entier naturel.

On considère qu'à la génération 0, il y a autant de fumeurs que de non-fumeurs. On a donc $f_0 = g_0 = 0,5$.

- Traduisez les données de l'énoncé par un graphe probabiliste.

Justifiez l'égalité matricielle $(f_{n+1} \;\; g_{n+1}) = (f_n \;\; g_n) \times A$, où A désigne la matrice

$$\begin{pmatrix} 0,6 & 0,4 \\ 0,1 & 0,9 \end{pmatrix}$$

- Déterminez le pourcentage de fumeurs à la génération de rang 2.
- Déterminez l'état probabiliste stable et l'interpréter.
- Montrer que, pour tout entier naturel n, $f_{n+1} = 0,5f_n + 0,1$.
- On pose, pour tout entier naturel n, $u_n = f_n – 0,2$.
  - Montrez que la suite (u_n) est une suite géométrique dont on précisera le premier terme et la raison.
  - Donnez l'expression de u_n en fonction de n.
  - Déduisez-en que, pour tout entier naturel n, $f_n = 0,3\times0,5^n+0,2$.

- Déterminez la limite de la suite (f_n) lorsque n tend vers $+\infty$ et interprétez-là.

Exercice 59. Sorties pédagogiques (Annales – Amérique du Sud, novembre 2004)

Au cours de la première semaine de l'année scolaire, un professeur propose aux élèves de sa classe le choix entre deux sorties pédagogiques, une sortie A et une sortie B : 20% des élèves de la classe sont favorables à la sortie A et tous les autres élèves sont favorables à la sortie B. Les arguments des uns et des autres font évoluer cette répartition en cours d'année. Ainsi 30% des élèves favorables à la sortie A et 20% des élèves favorables à la sortie B changent d'avis la semaine suivante.

On note :

- a_n la probabilité qu'un élève soit favorable à la sortie A la semaine n ;
- b_n la probabilité qu'un élève soit favorable à la sortie B la semaine n ;
- P_n la matrice $(a_n ; b_n)$ traduisant l'état probabiliste la semaine n.

- Déterminez l'état initial P_1.
- Représentez la situation par un graphe probabiliste.
- Déduisez-en que $P_{n+1} = P_n \times M$, où M est la matrice

$$\begin{pmatrix} 0,7 & 0,3 \\ 0,2 & 0,8 \end{pmatrix}$$

- Déterminez l'état probabiliste P_3 et en déduire la probabilité qu'un élève soit favorable à la sortie A la troisième semaine.
- Déterminez le réel x tel que $(x \ \ 1-x) \times M = (x \ \ 1-x)$.
- On admet que la suite (a_n) est croissante. La sortie A finira-t-elle par être préférée à la sortie B ?