Cours Mathématiques Discrètes
IUT Belfort Montbéliard

Pierre-Cyrille HEAM

12 janvier 2016

Chapitre 1

Graphes finis orientés

1.1 Premières définitions

Un **graphe fini orienté** est un couple (V, E) où
— V est un ensemble fini dont les éléments sont appelés **sommets** ou **noeuds** ou **états**.
— $E \subseteq V \times V$ est un ensemble de couples d'éléments de V, appelés **arêtes** ou **transitions**.

Par exemple $(\{1,2,3\}, \{(1,2),(2,2),(3,1)\})$ est un graphe fini orienté. Le choix des lettres V et E viennent de l'anglais pour *vertiex* et *edge*. On rappelle que dans un ensemble (entre $\{\}$) les éléments ne sont pas ordonnés et n'apparaissent qu'au plus une fois. En revanche dans un couple (deux éléments entre parenthèse), les éléments sont ordonnés. Le couple $(1,2)$ est différent du couple $(2,1)$.

Exercice 1
Les couples (V, E) suivants sont-ils des graphes finis orienté ?

1. $V = \{1,2,3\}$ et $E = \{(1,3)\}$?
2. $V = \{1,2,3\}$ et $E = \{(1,6),(2,4)\}$?
3. $V = \{1,2,3\}$ et $\emptyset$?
4. $V = \{1,2,3\}$ et $E = \{\{1,3\}\}$?
5. $V = \mathbb{N}$ et $E = \{(1,3)\}$?

Sauf mention contraire, tous les graphes étudiés dans ce polycopié seront des graphes finis orientés, que nous appellerons simplement *graphes* par la suite.

Un graphe se représente souvent par un dessin où les sommets sont des points (parfois entourés d'un cercle) et les arêtes des flèches : si (x, y) est une arête, on dessine une flèche entre x et y. Par exemple le graphe

$$(\{1,2,3,4\}, \{(1,1),(1,2),(3,1),(2,4),(3,4)\})$$

peut se dessiner comme sur la figure 1.1. Comme on peut le voir, il n'y a pas unicité de la façon de dessiner un graphe.

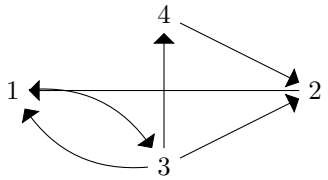
(a) première possibilité (b) deuxième possibilité

FIGURE 1.1 – Représentation d'un graphe

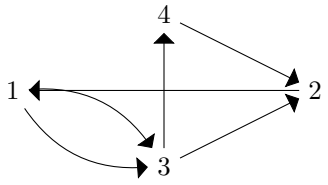
Exercice 2
Dessiner les graphes suivants.

1. $V = \{1,2,3,4,5\}$ et $E = \{(1,2),(2,1),(3,4),(4,4),(4,5),(5,3)\}$.
2. $V = \{1,2,3,4,5\}$ et $E = \{(1,4),(4,1),(3,4),(4,4),(4,5),(5,3)\}$.

Donner sous forme ensembliste (V et E) le graphe suivant :



Le dessin ci-dessous représente-t-il un graphe ? Si oui le donner explicitement, sinon dire pourquoi.



Dans un graphe, les **voisins** d'un sommet x sont tous les sommets y tels que (x, y) soit dans E.

Exercice 3

Pour le graphe de la figure 1.2, quels sont les voisins de 3 ? de 7 ? Quels sommets ont 9 pour voisin ? Donner deux sommets qui ont au moins deux vosins en commun.

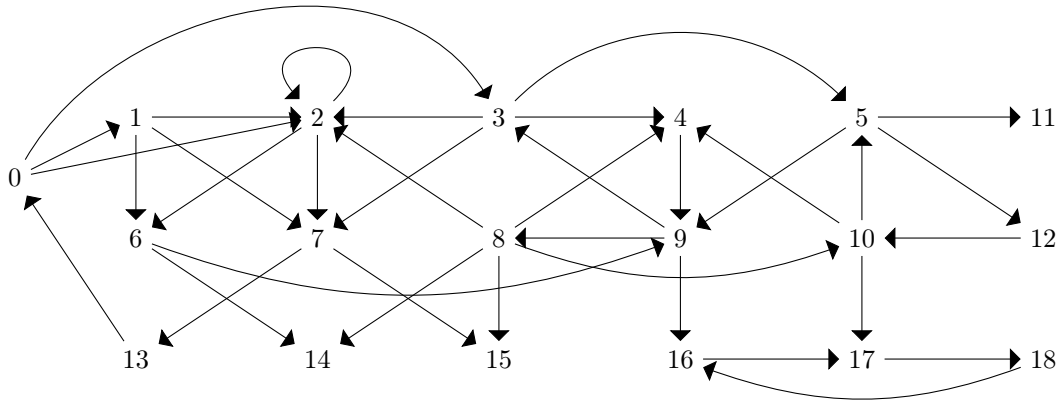


FIGURE 1.2 – Graphe 1

Exercice 4

On considère la fonction f de $\{0, \ldots, 7\}$ dans $\{0, \ldots, 7\}$ qui à x associe son reste modulo 3. Dessiner le graphe où $V = \{0, \ldots, 7\}$ et $(x, y) \in E$ si et seulement si $y = f(x)$. Mêmes question avec $(x, y) \in E$ ssi $x^2 = y \mod 5$ (c'est-à-dire si $x^2 - y$ est un multiple de 5). Même question avec $(x, y) \in E$ ssi $xy = 1 \mod 6$.

Exercice 5

On considère $V = \{maison, voiture, poisson, porte, cartable, grue\}$ et $(x, y) \in E$ si et seulement x est avant y dans l'ordre du dictionnaire. Dessiner le graphe (V, E). Même question mais cette fois ci $(x, y) \in E$ si et seulement x a plus de lettres que y.

Exercice 6

Soit $G = (V, E)$ un graphe. Comment voit-on sur un dessin de G que la relation E est une application bijective ? que la relation E est transitive ? réflexive ?

1.2 Chemins, boucles

Soit $G = (V, E)$ un graphe. Un **chemin** est une suite finie $(x_1, x_2), (x_2, x_3), \ldots, (x_n, x_{n+1})$ d'arêtes consécutives. L'entier n est appelé **taille** ou **longueur** du chemin. Le sommet x_1 est l'**origine** du chemin et x_n sa **destination**. On dit aussi que ce chemin part de x_1 pour arriver en x_n. Dans le graphe de la figure 1.2, $(0, 1), (1, 2), (2, 6)$ est un chemin partant de 0, arrivant en 6 et de longueur 3. Par convention, de tout point part un chemin vers lui même de longueur 0 (qui n'utilise aucune arête).

Une **boucle** est un chemin dont le sommet d'origine est aussi le sommet d'arrivée. Un chemin qui ne passe pas deux fois par le même sommet est appelée **chemin sans boucle**. Une boucle qui ne passe pas deux fois par le même sommet, sauf à l'origine et à l'arrivée, est appelée **boucle simple**. Dans le graphe de la figure 1.2, $(5, 12), (12, 10), (10, 5)$ est une boucle simple. Le chemin

$$(0, 1), (1, 7)(7, 13)(13, 0), (0, 3), (3, 7), (7, 13), (13, 0)$$

est une boucle mais n'est pas une boucle simple.

Exercice 7

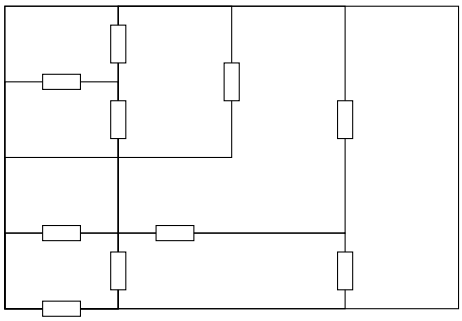
Dans le graphe de la figure 1.2 donner

1. Un circuit de taille 1,
2. Une boucle de longueur 9 passant par 4,
3. Un chemin sans boucle de 0 à 18,
4. Un chemin sans boucle de taille au moins 10.

1.3 Graphes et modélisation

L'objectif de ce chapitre est de voir comment passer d'un problème concret à un problème de graphe.

Exercice 8
On considère le plan ci-dessous d'un musée. La porte d'entrée est en bas à gauche.



1. Représenter sous forme de graphe ce musée, en mettant un sommet par salle.
2. On suppose qu'un gardien peut surveiller la salle dans laquelle il se trouve ainsi que les salles voisines. Combien faut-il de gardiens au minimum ? Formuler ce problème, d'une manière générale, dans le vocabulaire des graphes.
3. Un groupe de visiteurs souhaite admirer toutes les salles du musée. Peut-il le faire sans jamais passer deux fois par la même salle (on ne compte pas le chemin de retour ; le groupe peut partir de n'importe quelle salle aussi) ? Proposer une formulation générale de ce problème en utilisant le vocabulaire des graphes.
4. Le soir, le conservateur souhaite avant de fermer le musée passer dans toutes les salles puis ressortir, tout en minimisant le trajet (il part de l'entrée et ressort par l'entrée/sortie). Quel chemin doit-il prendre ? Formuler ce problème de façon général en utilisant le vocabulaire des graphes.

Exercice 9
On considère un ensemble fini V de joueurs de tennis, et le graphe orienté (V, E) tel que $(x, y) \in E$ si et seulement si x a déjà gagné un match contre y. Donner des formulations dans le vocabulaire de la théorie des graphes des propriétés suivantes.

1. Il y a un joueur qui n'a jamais perdu.
2. Il y a un joueur qui n'a jamais gagné.
3. Un joueur ne joue jamais contre lui-même.
4. Tous les joueurs ont déjà joué au moins un match conte un autre joueur.
5. Il y a un joueur qui a déjà joué contre tout le monde.
6. Il y a un joueur qui a déjà gagné contre tout le monde.
7. Il y a un joueur qui a joué contre tout le monde et perdu tous ses matchs.

Exercice 10
On considère un graphe représentant la carte d'une ville : chaque nœud est un croisement, chaque arête un tronçon de rue, codant le sens de circulation. On considère de plus que les neouds sont coloriés, en rouge (s'il y a un feu de signalisation au croisement) ou en bleu (s'il n'y a pas de feu). Exprimer, dans le vocabulaire de la théorie des graphes les propriétés suivantes.

1. Il n'y a pas de sens unique.
2. Il n'y a pas d'impasse.
3. De n'importe où, je peux aller partout.
4. De n'importe où, je peux aller partout en passant par au plus trois feux de signalisation.
5. Deux croisements consécutifs ne peuvent avoir tout deux des feux.
6. Il n'y a jamais plus de deux rues qui se croisent à la fois.

7. Il n'y a ni pont, ni tunnel.

Exercice 11
On considère le graphe G dont les sommets sont les configurations possible d'un rubics cube, et il y a une arête entre deux configurations si l'on peut passer de l'une à l'autre en jouant un coup (c'est-à-dire une rotation d'un quart de tour d'un coté). Exprimer avec le vocabulaire de la théorie des graphes les propriétés suivantes (on ne cherche pas à savoir si elles sont vraies ou fausse) :

— Il y a toujours une solution.
— A partir de certaines configurations, il n'y a pas de solution.
— De toute position, si je modifie le cube, je peux revenir à cette position.
— Il existe une position dans laquelle je peux résoudre le rubics cube, mais pas en moins de 30 coups.
— Si dans une position je peux résoudre le rubics cube, alors je peux le résoudre en moins de 30 coups.

Exercice 12
On considère un ensemble de n enfants portant chacun un dossard différent, numéroté de 1 à n, et jouant à cache-cache. On considère le graphe dont les sommets sont les entiers de 1 à n et il y a une arête entre i et j si l'enfant avec le dossard i peut voir l'enfant avec le dossard j. Exprimer dans le vocabulaire de la théorie des graphes les propriétés suivantes :

1. Il y a un enfant bien caché qu'aucun autre ne peut voir.
2. Il y a un enfant qui ne voit personne.
3. Un enfant se voit toujours lui-même.
4. Il y a deux enfants qui se voient l'un l'autre.
5. Tout enfant voit au moins un autre enfant.

Exercice 13
On considère un graphe fini orienté $G = (V, E)$ où V est un ensemble d'enfant. Il y a une arête entre x et y si x a déjà invité y à son anniversaire. Pour chacune des propriétés suivantes (indépendantes), donner une traduction dans le vocabulaire des graphes.

1. Tout le monde a déjà invité au moins une personne à son anniveraire.
2. Il y a quelqu'un qui n'a jamais été invité.
3. Il y a quelqu'un qui a déjà été invité par tout le monde.
4. Il y a quelqu'un qui n'a invité que deux personnes à son anniversaire.
5. Personne ne s'invite lui-même à son anniversaire.
6. Il y a deux enfants qui se sont invité mutuellement à leur anniversaire.
7. Il y a quelqu'un qui n'a jamais invité personne à son anniversaire.
8. Tout le monde a toujours retourné une invitation d'anniversaire.

Exercice 14
Un homme se trouve au bord d'une rivière avec un choux, une chèvre et loup. Il possède une barque et veut traverser la rivière. Cependant, il ne peut mettre en même temps, en plus de lui, dans la barque qu'un des trois. Et, malheureusement aussi, il ne peut laisser sur une rive seul le loup avec la chèvre, ni la chèvre avec le choux.

Modéliser le problème avec un graphe et proposer une solution.

Exercice 15
On considère un site Web statique, installé sur un serveur. On se pose les questions suivantes, liées à l'ergonomie du site :

1. Puis-je à l'aide de la souris uniquement accéder à toutes les pages du site à partir de la page d'accueil ?
2. Puis-je toujours en un clic retourner à la page d'accueil ?
3. Puis-je, à partir de n'importe quelle page, accéder à n'importe quelle autre page en moins de 3 clics ?

Proposer une modélisation de cette situation par un graphe, puis décrire des algorithmes pour répondre aux questions posées.

Exercice 16
On considère le musée décrit dans la figure 1.3. Le directeur souhaite, pour une exposition exceptionnelle, placer un gardien dans chaque salle. Il recrute pour cela des gardiens espagnols (qui ne parlent que l'espagnol) et des gardiens italiens (qui ne parlent qu'italien). Afin qu'il ne discutent pas entre eux, il ne veut pas que deux gardiens de la même nationalité soient dans des salles voisines. Est-ce possible ?

Même question avec le musée de la figure 1.4.

Généraliser ce problème sur les graphes. Décrire un algorithme pour le résoudre ?

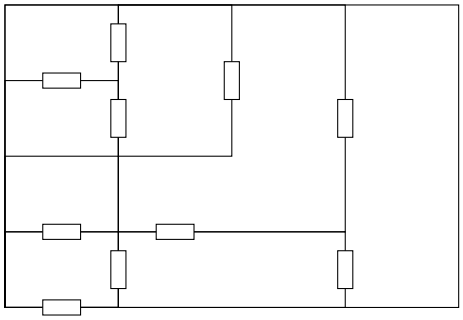


FIGURE 1.3 – Plan du musée 1

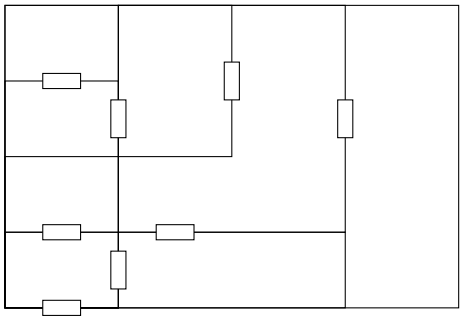


FIGURE 1.4 – Plan du musée 2

1.4 Codage des graphes

On s'intéresse maintenant à coder les graphes (en machine). Il y a deux méthodes principales (qui admettent des variantes), le codage par liste d'adjacence et le codage par matrice d'adjacence.

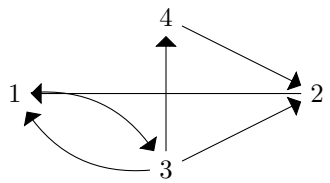
1.4.1 Codage par matrices d'adjacence

Soit $G = (V, E)$ un graphe. On suppose, sans perte de généralité, que $V = \{1, \ldots, n\}$. On appelle **matrice d'adjacence** de V le tableau (matrice) de taille n sur n ne contenant que des 0 ou des 1 et défini par : la case de la i-ème ligne et de la j-ième colonne contient un 1 si et seulement si $(i, j) \in E$.

On considère par exemple le graphe $(\{1, 2, 3, 4\}, \{(1, 1), (1, 2), (3, 1), (2, 4), (3, 4)\})$ dessiné sur la figure 1.1. Sa matrice d'adjacence est

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

Réciproquement, le graphe dont la matrice d'adjacence est $\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$ est le graphe ci-dessous :



Exercice 17

Dessiner le graphe dont la matrice d'adjacence est $\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$.

Quel est la matrice d'adjacence du graphe où $V = \{1, \ldots, 7\}$ et

$$E = \{(1, 2), (4, 2), (3, 5), (7, 1), (6, 4), (1, 5), (5, 0)\} \ ?$$

Exercice 18
Soit (V, E) un graphe. Comment lit-on sur la matrice d'adjacence de ce graphe que la relation E est réflexive ? symétrique ? antisymétrique ?

Exercice 19
Pour tout graphe orienté G, on note G^R le graphe défini par : les sommets de G^R sont les mêmes que ceux de G et (x, y) est une arête de G^R si et seulement si (y, x) est une arête de G. Le graphe G^R s'appelle *miroir* de G.

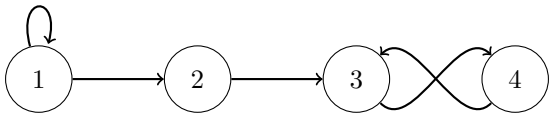


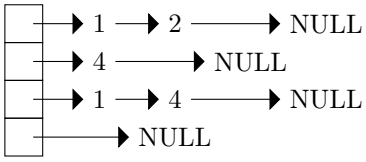
FIGURE 1.5 –

1. Dessiner le graphe miroir du graphe de la figure 1.5.
2. Soit A la matrice d'adjacence d'un graphe G. Quelle est la matrice d'adjacence de G^R, en fonction de A ?
3. Justifier que $(G^R)^R = G$ pour tout graphe G.
4. Justifier que si G contient une boucle de longueur k, alors G^R aussi.

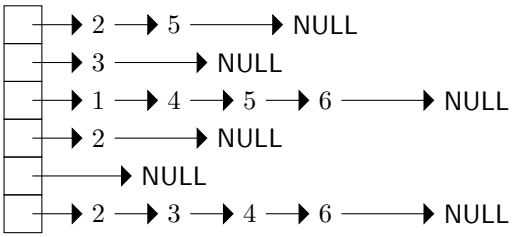
1.4.2 Codage par liste d'adjacence

Le codage d'un graphe par liste d'adjacence consiste à coder, pour chaque sommet du graphe, l'ensemble de ses voisins par une liste *triée*[1]. On considère encore que $V = \{1, \dots, n\}$. On dispose alors d'un tableau de taille n dont la première case contient la liste des voisins de 1, la seconde la liste des voisins de 2, etc.

Par exemple, le graphe de la figure 1.1 se code par le tableau de liste :



Exercice 20
Dessiner le graphe dont la représentation par liste d'adjacence est :



Exercice 21
Dessiner la représentation par liste d'adjacence du graphe de la figure 1.2.

Exercice 22
On considère que l'on code en Python un graphe par liste d'adjacence de la manière suivante :
- Un graphe un un couple de la forme (V, E),
- V est une liste d'éléments tous différents, triés,
- E est un dictionnaire qui à chaque élément de V associe la liste triée de ses voisins.

1. Dessiner le graphe correspondant à
   `([1,2,3],{1:[2],2:[2,3],3:[1,3]})`
2. Écrire une fonction qui compte le nombre d'arêtes d'un graphe.
3. Écrire une fonction qui teste si (i, j) est une arête d'un graphe.

1. Attention, le fait qu'elle soit triée n'est pas toujours imposé dans la littérature, mais c'est le choix que nous prendrons.

4. On appelle $sommet\ bloquant$ un sommet i tel que pour tout sommet j, (i, j) n'est pas une arête. Écrire une fonction qui teste si un sommet est bloquant.
5. On appelle $sommet\ puits$ un sommet i tel que pour tout sommet $j \neq i$, (i, j) n'est pas une arête et tel que (i, i) soit une arête. Écrire une fonction qui teste si un sommet est un puits.
6. Écrire une fonction qui ajoute une arête (i, j) à un graphe.
7. Écrire une fonction qui supprime une arête (i, j) à un graphe (on rappelle que la méthode `remove` de python retire la première occurrence d'une valeur donnée en paramètre).

Chapitre 2

Parcours de Graphes

L'exercice suivant est une introduction aux parcours de graphes.

Exercice 23
On considère le graphe de la figure 2.1. Il y a une flèche entre x et y si x connaît l'adresse de y. On considère aussi que certains individus dans cette liste peuvent être des zombies.

1. On considère que chaque nuit, un zombie va mordre et donc transformer en zombie des individus dont il connaît l'adresse et qui ne sont pas encore zombies. Si c'est possible, chaque zombie est obligé de transformé au moins un zombie par nuit. En supposant qu'au départ seule Karina est un zombie, donner quelques possibilités indiquant, in fine, qui a zombifier qui ? Dessiner les solutions à l'aide d'un graphe où (x, y) est une arête si x a zombifié y.
2. On considère maintenant que chaque nuit, chaque zombie va transformer tous les individus dont il connaît l'adresse et qui ne sont pas encore zombies. En supposant qu'au départ seule Karina est un zombie, donner quelques possibilités indiquant, in fine, qui a zombifier qui ?
3. On suppose maintenant que chaque zombie doit non seulement transformer tous les individus dont il connaît l'adresse et qui ne sont pas encore zombies, mais aussi dans l'ordre croissant de leur numéros. Par exemple, Karina transformera dans l'ordre Raph, Mike puis David. On suppose de plus que chaque nuit les zombies vont mordre à tour de rôle, en commençant par celui qui à été transformé en zombies le plus anciennement. En supposant qu'au départ seule Karina est un zombie, écrire qui a zombifié qui, et dans quel ordre.
4. On suppose maintenant que pour qu'un zombie transforme un individu en zombi, il doit être équipe du Sceptre Maudit. Chaque nuit, le zombie possédant le sceptre va mordre le premier de ses voisins (par ordre de numéro) qui n'est pas encore zombie, le transformant ainsi ; de plus il lui donne le sceptre. Si le zombie qui a le sceptre n'a pas de voisin à mordre, il donne le sceptre au zombie qui l'a transformé. En supposant qu'au départ Karine a le sceptre, écrire qui a zombifié qui, et dans quel ordre.

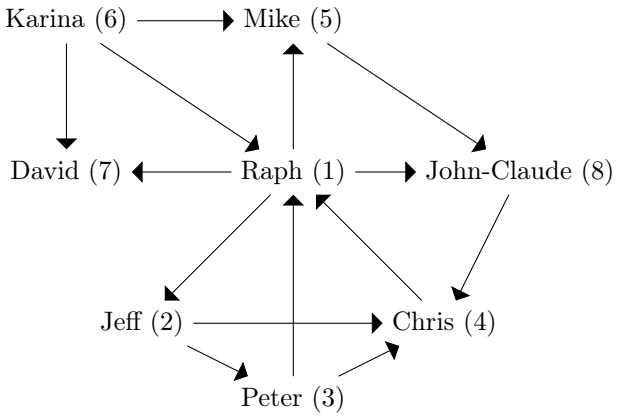


FIGURE 2.1 – Graphe des connaissances

Les graphes obtenus dans les questions de l'exercice précédent s'appellent des *arbres couvrants*. La façon de procédé de la question 3 s'appelle *parcours en largeur* et celle de la question 4 s'appelle *parcours en profondeur*.

2.1 Arbres couvrants

2.1.1 Arbres

Un **arbre** est un graphe $G = (V, E)$ tel que :

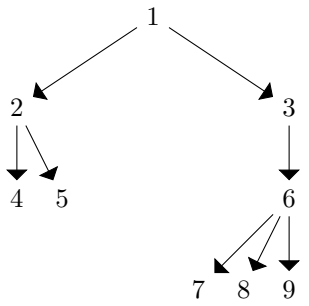


FIGURE 2.2 – Exemple d'arbre

(1) Il n'y a pas de boucle de longueur non nulle dans G (on dit que G est acyclique).

(2) Si (x, y) et (z, y) sont dans E, alors $x = z$.

(3) Il existe un sommet x_0, appelé **racine**, tel que pour tout sommet y, il existe une chemin de x_0 à y.

Un exemple d'arbre dont la racine est 1 est dessiné sur la figure 2.2.

Exercice 24
Dessiner trois arbres différents à 6 sommets.

Exercice 25
Dessiner un graphe qui vérifie (1) et (2) de la définition mais qui n'est pas un arbre. Même question avec (2) et (3). Même question avec (3) et (1).

Dans un arbre on utilise à la fois une terminologie *végétal* et aussi de type *arbre généalogique*. On appelle **feuille** tout sommet d'un arbre qui n'a pas de voisin. Tout chemin de la racine à une feuille est appelée **branche**. Les voisins d'un noeud sont appelés ses **fils**. Tous les sommets atteignables depuis un sommet x sont dits les **descendants** de x et ils constituent le **sous-arbre** enraciné en x. Si y est atteignable depuis x, alors x est un **ancêtre** de y. Si y est un voisin de x, alors x est le **père** de y. Deux sommets qui ont le même père sont de **frère**. Par exemple, dans le dessin de la figure 2.2, 6 est le père de 7, 3 est un frère de 2 et 5 est une descendant de 1.

Exercice 26
Dans le graphe de la figure 2.2, quels sont les descendants de 3 ? les fils de 1, les ancêtres de 6, les feuilles ?

Exercice 27
On considère dans cet exercice des arbres codés par liste d'adjacence comme dans l'exercice en Python de la page 6. On supposera que la racine est le premier sommet de la liste V.

1. Écrire une fonction qui compte le nombre de feuille d'un arbre.
2. Écrire une fonction qui compte le nombre de descendants d'un sommet donné.

2.1.2 Parcours d'arbres

Parcourir un arbre, c'est appliquer un algorithme qui va visiter, une et une seule fois chaque sommet de l'arbre. Les deux parcours les plus connus, sont les parcours préfixes et suffixes. A chaque visite de noeud, une fonction `Traiter` est appliquée, selon ce que l'on souhaite faire.

Le parcours Préfixe est l'algorithme récursif suivant. Initialement, le parcours est lancé à la racine.

```
Parcours-Prefixe(V,E,x):
   Traiter(x)
   Pour chaque fils f de x:
      Parcours-Prefixe(V,E,f)
   FinPour
```

Si l'on applique `Parcours-Prefixe` en 1 sur l'arbre de la figure 2.2, avec pour fonction `Traiter` la fonction qui affiche la valeur du sommet on obtient.

1 `Parcours-Prefixe(2)` `Parcours-Prefixe(3)` (sommets dans l'ordre)

Or `Parcours-Prefixe(2)` retourne

2 `Parcours-Prefixe(4)` `Parcours-Prefixe(5)` et donc retourne

2 4 5

De même `Parcours-Prefixe(3)` retourne

3 6 7 8 9

Et donc `Parcours-Prefixe(1)` retourne

1 2 4 5 3 6 7 8 9

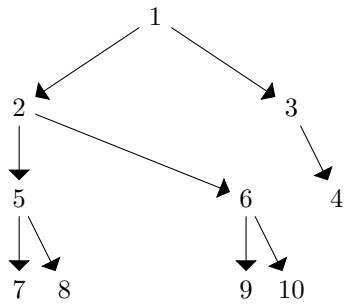


FIGURE 2.3 – Arbre

Exercice 28
Que retourne l'algorithme de parcours préfixe sur l'arbre de la figure 2.3.

La parcours suffixe est identique à la différence près que le traitement se fait après l'appel récursif.

```
Parcours-Suffixe(V,E,x):
   Pour chaque fils f de x:
      Parcours-Suffixe(V,E,f)
   FinPour
   Traiter(x)
```

Exercice 29
Que donne le parcours suffixe sur les arbres des figures 2.2 et 2.3 ?

Exercice 30
Avec le même codage que précédemment, écrire en Python des fonctions de parcours préfixe et suffixe.

2.1.3 Arbres couvrants

Soit $G = (V, E)$. Un arbre couvrant pour G, enraciné en $x \in V$, est un arbre (V', E') tel que

(1) $V' \subseteq V$ et $E' \subseteq E$.

(2) x est la racine de (V', E').

(3) On ne peut pas rajouter à l'arbre un sommet ou une arête tel que (1) et (2) soit toujours vérifier (l'arbre est maximal).

Intuitivement un arbre couvrant est un arbre que l'on peut calquer sur un graphe – conditions (1) et (2) – sans pouvoir le prolonger. Considérons par exemple le graphe de la figure 2.4.

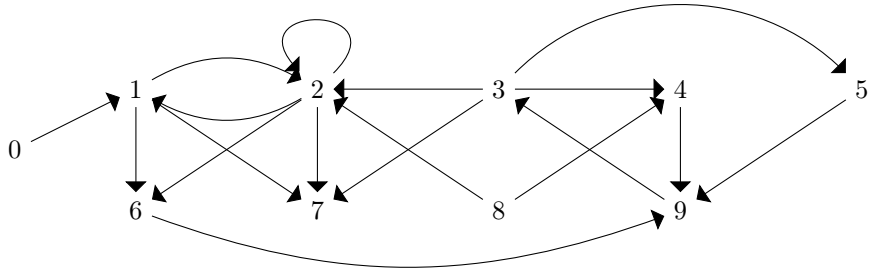


FIGURE 2.4 – Graphe 1

Dans ce graphe, l'arbre suivant de gauche vérifie bien (1) et (2) mais pas (3). En revanche celui de droite est bien un arbre couvrant.

Exercice 31
Écrire une fonction qui étant donné un graphe et un arbre codé en Python comme précédemment, teste si l'arbre est un arbre couvrant du graphe (on ne testera pas si c'est bien un arbre).

L'objectif des algorithmes de parcours de graphes est de construire des arbres couvrants de graphe afin, ensuite, d'appliquer des parcours (type préfixe/suffixe) sur les arbres obtenus. Il existe deux constructions d'arbres couvrant très utilisées, appelées *parcours en largeur* et *parcours en profondeur*.

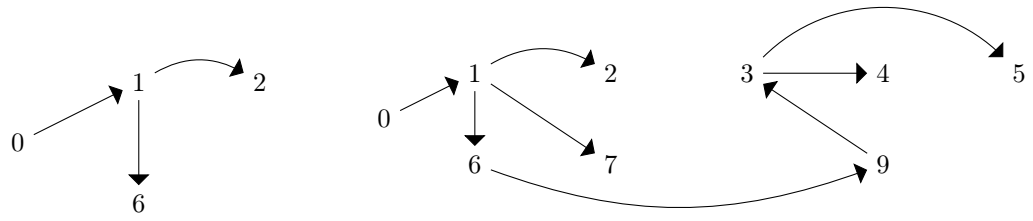


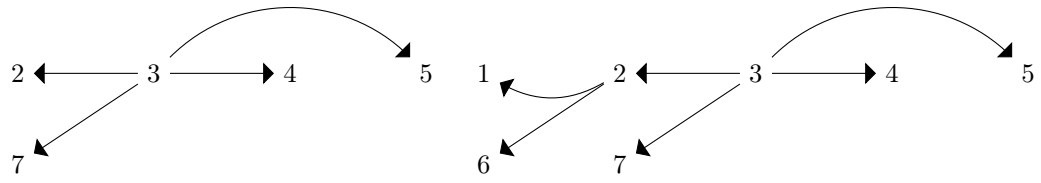
FIGURE 2.5 – Arbres non couvrant et couvrant

2.1.4 Parcours en largeur

On en donne pas explicitement l'algorithme de parcours en largeur à partir de x qui demanderait l'introduction des structures de données. On se contente d'un pseudo-algorithme expliquant la méthode :

1. x est la racine de l'arbre.
2. On insère dans l'arbre tous les voisins de x (sauf éventuellement x), dans l'ordre.
3. On insère (dans l'ordre) dans l'arbre tous les voisins de la feuille la plus anciennement introduite dans l'arbre et pour laquelle c'est possible.
4. On recommence l'étape précédente tant que c'est possible.

Si l'on exécute le parcours sur le graphe de la figure 2.4, en partant du sommet 3. Après les étapes 1 et 2 on a l'arbre A_1. La feuille introduite le plus anciennement (attention à l'ordre qui est l'ordre des numéro des sommets) est 2. On ajoute donc 1 et 6 – et pas 7 qui y est déjà. On obtient alors l'arbre A_2. Voir figure 2.6.



(a) A_1 (b) A_2

FIGURE 2.6 – Parcours en largeur

Ensuite, la feuille la plus anciennement introduite est 4. Son seul voisin est 9 qui n'est pas encore dans l'arbre. On obtient donc l'arbre A_3. Ensuite il s'agit du sommet 5 dont l'unique fils 3 est déjà dans l'arbre. La construction n'évolue donc pas pour 5. Ensuite, on a fini les fils de 3, on passe donc aux fils du premier fils de 3, à savoir 2. Les fils de 1 sont 2, 6 et 7, qui sont déjà dans l'arbre. En continuant ainsi, on voit qu'il n'y a plus rien à ajouter. n obtient donc l'arbre de la figure 2.7

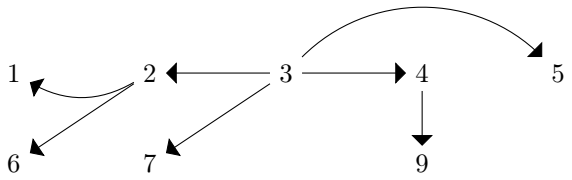
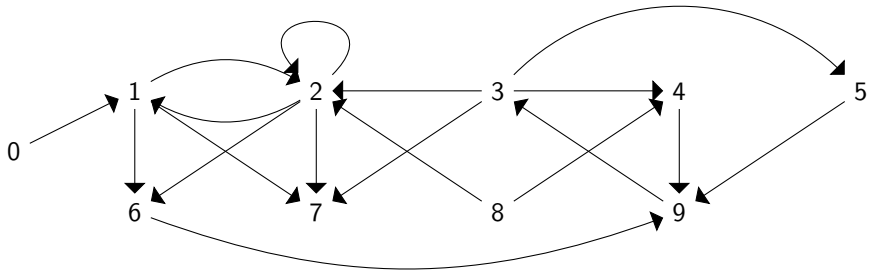


FIGURE 2.7 – Parcours en largeur A_3

Exercice 32
Sur le graphe de la figure ci-dessous, dessiner les arbres obtenus par un parcours en largeur à partir de 2. Même question en partant de 6. Même question en partant de 9.

Même question à partir de 3, puis de 0, puis de 8 sur le graphe de la figure 1.2.



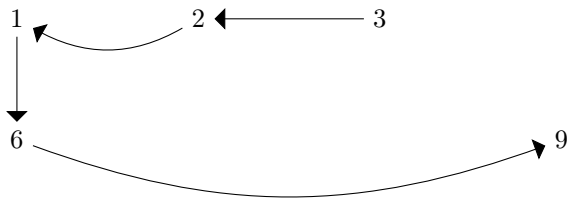


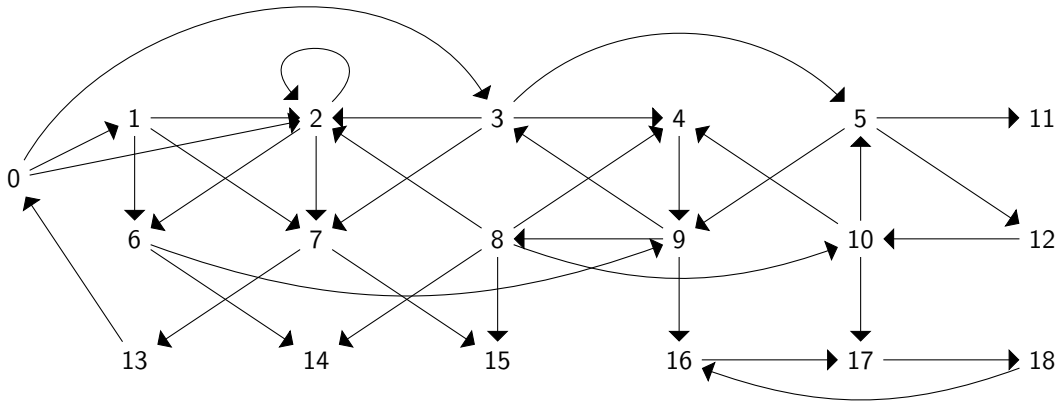
FIGURE 2.8 – Parcours en profondeur (1)

Exercice 33
Écrire en Python l'algorithme de parcours en largeur pour un graphe codé par liste d'adjacence.

Exercice 34
Écrire en Python une fonction qui teste si un graphe donné par liste d'adjacence est un arbre dont la racine est le plus petit sommet.

Exercice 35
Les *composantes fortement connexes* d'un graphe sont des sous-ensembles de sommets telles que : (1) tout sommet appartient à une composante fortement connexe, (2) s'il existe un chemin de p vers q et un chemin de q vers p, alors p et q doivent appartenir à la même composante fortement connexe ; et réciproquement.

1. Deux composantes fortement connexes différentes sont-elles forcément disjointes ? Pourquoi ?
2. Quels sont les composantes fortement connexes du graphe suivant.



3. Décrire une méthode algorithmique pour trouver les composantes fortement connexe d'un graphe.

Exercice 36
Comment utiliser un parcours en largeur pour trouver un chemin de taille la plus courte possible entre deux sommets donnés ?

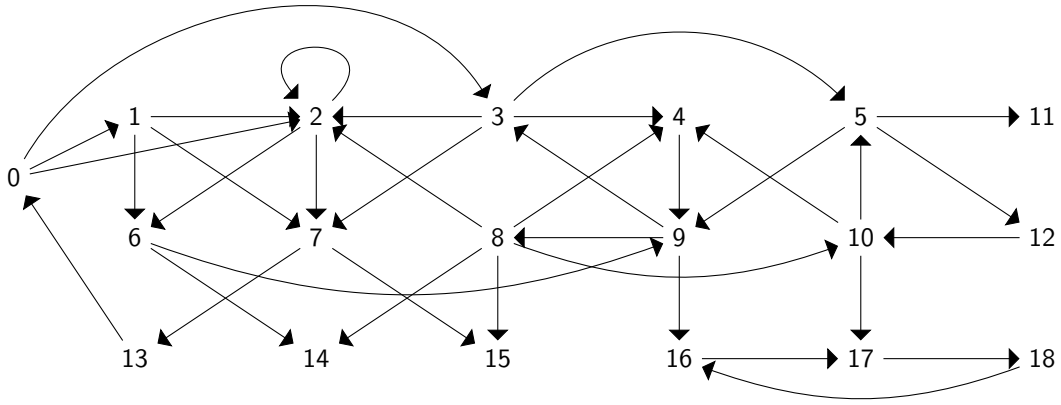
2.1.5 Parcours en profondeur

On en donne pas explicitement l'algorithme de parcours en largeur à partir de x qui demanderait l'introduction des structures de données. On se contente d'un pseudo-algorithme expliquant la méthode :

1. x est la racine de l'arbre.
2. On insère dans l'arbre le premier voisin de x (sauf éventuellement x).
3. On insère dans l'arbre le premier voisins du sommet le plus récemment introduit dans l'arbre et pour lequel c'est possible.
4. On recommence l'étape précédente tant que c'est possible.

On va appliquer la méthode sur l'arbre de la figure 2.4, en partant du sommet 3. L'arbre commence par 3 à la racine. Puis on introduit le plus petit fils de 3, c'est-à-dire 2. La feuille la plus récente de l'arbre est 2, donc on introduit le plus petit fils de 2, à savoir 1. A nouveau, on va introduire 6, puis 9. On obtient alors l'arbre de la figure 2.8.

Exercice 37
Donner les parcours en profondeur obtenus à partir de 0, puis 8, puis 7 sur le graphe ci-dessous.



Exercice 38
Écrire en Python l'algorithme de parcours en profondeur pour un graphe codé par liste d'adjacence.

Chapitre 3

Mots et langages

3.1 Alphabets, mots, langages

Definition 39 *On appelle* alphabet *tout ensemble fini. Les éléments d'un alphabet sont appelés des* lettres.

Par exemple, $A = \{a, b, c\}$ est un alphabet contenant 3 lettres. On peut prendre pour lettre n'importe quels symboles, mais la tradition veut que l'on prennent les lettres minuscules du début de l'alphabet romain.

Definition 40 *Toute suite fini de lettres est appelée* mot fini *ou simplement* mot. *Le cardinal de cette suite est appelée* longueur *du mot. Si u est un mot de longueur n non nulle, on le note $u(1)u(2)\dots u(n)$. Il existe un unique mot de taille nulle notée ε. La taille d'un mot u est notée $|u|$. L'ensemble des mots sur un alphabet A se note A^*.*

Par exemple, si u est le mot sur l'alphabet $\{a, b, c\}$ de taille 3 qui à 1 associe a, à 2 associe c et à 3 associe b, alors u est noté acb. De même $bbab$ représente le mot qui à $1, 2$ et 4 associe b et qui à 3 associe a. On a $|bbab| = 4$.

Exercice 41
On considère l'alphabet $\{a_0, a_1, a_2, a_3\}$. Écrire le mot u de longueur 9 qui à i associe a_j où j est le reste de la division de i^2 par 3. Que vaut $|a_1a_3a_0a_1|$?

Definition 42 *On appelle* langage *tout ensemble de mots.*

Par exemple $\emptyset$, A^* sont des langages sur A. Si a et b sont des lettres l'ensemble des mots de longueur paire, l'ensemble des mots finissant par a, l'ensemble des mots contenants autant de a que de b sont des langages.

Exercice 43
On travaille avec l'alphabet $\{a, b, c\}$. Soit L l'ensemble des mots de longueur paire, K l'ensemble des mots commençant par c et R l'ensemble des mot contenant autant de a que de b. Donner un exemple de mot dans chacun des langages suivants :

1. L, L^c,
2. K, K^c,
3. R, R^c,
4. $K \cap L$,
5. $K \cap L^c$,
6. $R \cap K \cap L^c$.
7. $R^c \cap K^c \cap L^c$.

3.2 Opérations sur les mots et les langages

3.2.1 Produit de mots

Definition 44 *Le produit de deux mots u et v est le mot noté uv de longueur $|u| + |v|$ défini par*

$$\begin{cases} uv(i) = u(i) \text{ si } i \leq |u| \\ uv(i) = v(i - |u|) \text{ si } |u| < i \leq |u| + |v| \end{cases}$$

Par exemple le produit de *abbab* par *cbaa* est le mot *abbabcbaa*. Il suffit de concaténer les mots.

Proposition 45 *Pour tout mot u, $u\varepsilon = \varepsilon u = u$ (le mot vide est un élément neutre pour le produit). De plus le produit est associatif, c'est-à-dire que pour tous mots u, v, w, $u(vw) = (uv)w$.*

Exercice 46
Sur l'alphabet $\{a, b\}$, ecrire tous les mots de longueur inférieure ou égale à 3. Sur un alphabet Σ, combien y a-t-il de mots de longueur k ? Combien y a-t-il de mots de longueur inférieure ou égale à k ?

3.2.2 Opérations sur les langages

Le produit s'étend naturellement aux langages : si L et K sont des langages sur A, alors LK est le langage sur A défini par

$$LK = \{uv \mid u \in L \text{ et } v \in K\}.$$

Par exemple, si L est l'ensemble des mots ne contenant que des a et K est l'ensemble des mots ne contenant que des b, alors LK est l'ensemble des mots de la forme $aa\ldots abb\ldots b$.

Exercice 47
Soit $L = \{ab, bc\}$ et $K = \{aab, bcba, ab\}$. Que vaut LK ? KL ? LL ?

Exercice 48
Soit L l'ensemble des mots de longueur paire et K l'ensemble des mots commençant par a. Décrire par une phrase en francais la plus simple possible LK.

langages.

Exercice 49
On travaille avec l'alphabet $\{a, b, c\}$. Soit L l'ensemble des mots de longueur paire, K l'ensemble des mots commençant par c et R l'ensemble des mot contenant autant de a que de b. Donner un exemple de mot dans chacun des langages suivants :

1. LK,
2. KL,
3. RL,
4. $RL \cap LR$,
5. $(LK)^c$.

Exercice 50
Soient L et K deux langages finis. Justifier que LK est fini et que $|LK| \leq |L||K|$. Donner un exemple où l'inégalité est stricte.

Exercice 51

1. Que vaut LK si L est vide ? si K est vide ?
2. Justifier que si $L_0 = \{\varepsilon\}$, alors pour tout langage K, $L_0K = KL_0 = K$. Justifier que la réciproque est vraie aussi.
3. Justifier que si $K \subseteq R$, alors pour tout langage L, $LK \subseteq LR$ (pour cela, prendre un mot générique $v \in LK$ et prouver qu'il est dans LR).
4. En déduire que si K_1 et K_2 sont des langages, alors $LK_1 \subseteq L(K_1 \cup K_2)$ et $LK_2 \subseteq L(K_1 \cup K_2)$. Pourquoi peut-on dire que $LK_1 \cup LK_2 \subseteq L(K_1 \cup K_2)$?
5. Justifier que $L(K_1 \cup K_2) \subseteq LK_1 \cup LK_2$.
6. Comparer $L(K_1 \cup K_2)$ et $LK_1 \cup LK_2$.
7. Justifier à l'aide des questions précédentes que $L(K_1 \cap K_2) \subseteq LK_1$ et que $L(K_1 \cap K_2) \subseteq LK_2$. En déduire que $L(K_1 \cap K_2) \subseteq LK_1 \cap LK_2$?
8. Donner un exemple qui montre que cette inclusion (question précédente) peut être stricte.

Si u est un mot et L un langage, on note uL le langage $\{u\}L$.

Definition 52 *Pour tout langage L, on appelle* étoile *de L, et l'on note L^* le langage*

$$L^* = \bigcup_{n \geq 0} L^n,$$

où $L^0 = \{\varepsilon\}$ et $L^{n+1} = LL^n$.

Par exemple $\{aa, ab, ba, bb\}^*$ est le langage des mots de longueur paire sur $\{a, b\}$.

Notons que la notation A^* introduite en début de chapitre est cohérente avec la définition 52 si l'on identifie les mots de longueur 1 et les lettres (ce que l'on fait en pratique).

Si u est un mot, on note u^* pour $\{u\}^*$.

Exercice 53
— Quelle est la différence entre $\{ab\}^*$ et ab^* ?
— Décrire $\{ab, bc\}^* \cap a\{a, b, c\}^*$.

Exercice 54

1. Justifier que $L^* \cup K^* \subseteq (L \cup K)^*$.

2. Donner un exemple de langages L et K tels que $(L \cup K)^* \neq L^* \cup K^*$. Comparer de même L^* avec $(L^*)^*$.
3. Justifier que $(L \cap K)^* \subseteq L^* \cap K^*$.
4. Montrer par un exemple que cette inclusion est stricte.
5. Peut-on avoir $L^* = \emptyset$?

Exercice 55
On considère un alphabet A contenant les lettres a et b. En utilisant uniquement les symboles a, b, A, le produit, l'union et l'étoile, définir les langages suivants :

1. l'ensemble des mots commençant par a.
2. l'ensemble des mots finissant par b.
3. l'ensemble des mots commençant par a et finissant par b.
4. l'ensemble des mots contenant au moins un a.
5. l'ensemble des mots commençant et finissant par a.
6. l'ensemble des mots contenant au moins une fois ab.
7. l'ensemble des mots contenant au moins une fois ba.
8. l'ensemble des mots contenant au moins une fois ba ou au moins une fois ab.
9. l'ensemble des mots contenant au moins une fois ba et au moins une fois ab.
10. l'ensemble des mots ne contenant pas aa.
11. l'ensemble des mots ne contenant pas ab.

Exercice 56
On considère un alphabet $A = \{a, b\}$. En utilisant uniquement les symboles $a, b, \emptyset$, le produit, l'union, l'intersection et le complément, exprimer le langage $ab(ab)^*$.

Definition 57 *Pour un alphabet A, on appelle langage rationnel la plus petite famille de langages contenant les langages finis et stable par produit, union et étoile.*

Les langages rationnels jouent un rôle fondamental en informatique. Ils possèdent de nombreuses caractérisations intéressantes, en alliant expressivité et propriétés algorithmiques. Comme nous le verrons plus tard, tous les langages ne sont pas rationnels.

Chapitre 4

Automates finis

4.1 Automates finis

Definition 58 *Un* automate fini *est un tuple* (Q, A, E, I, F) *où*
- Q *est un ensemble fini dont les éléments sont appelés* états*,*
- A *est un alphabet,*
- $E \subseteq Q \times A \times Q$ *est l'ensemble des* transitions*,*
- $I \subseteq Q$ *est l'ensemble des états* initiaux*, et*
- $F \subseteq Q$ *est l'ensemble des états finaux.*

On représente souvent un automate par un graphe dont les sommets sont les états de l'automate, les arêtes (étiquetées) sont les transitions de l'automate. Par ailleurs les états initiaux sont marqués avec une petite flèche entrant dans l'état et les états finaux soit par une petite flèche sortante, soit par un double rond autour de l'état (dans ce cours nous choisirons cette dernière convention). Par exemple, l'automate

$$(\{1,2,3\}, \{a,b\}, \{(1,a,1),(1,a,2),(2,b,3),(3,a,3),(3,b,3),(3,b,1)\}, \{1,2\}, \{2,3\})$$

sera représenté par le dessin de la figure 4.1.

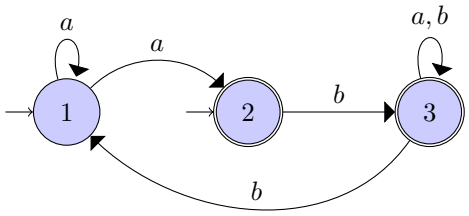


FIGURE 4.1 – Exemple de dessin d'automate

Exercice 59
Dessiner l'automate (Q, A, E, I, F) où $Q = \{1,2,3\}$, $A = \{a,b\}$, $I = \{2\}$, $F = \{1,3\}$ et

$$E = \{(1,a,2),(2,a,3),(3,a,1),(1,b,1),(1,b,3),(3,b,1)\}.$$

Notez que certains livres imposent que I soit un singleton, restriction qui n'est pas nécessaire.

Definition 60 *Un chemin de dans un automate fini est une suite finie de transitions* $(p_1, a_1, q_1) \dots (p_n, a_n, q_n)$ *telle que pour tout* i *entre* 1 *et* $n-1$ $q_i = p_{i+1}$*. De plus si* p_1 *est initial et* q_n *est final, alors le chemin est dit* acceptant *ou* réussi*. La suite vide de transition est un chemin de tout état vers lui-même. L'entier* n *est appelé* longueur *du chemin et le mot* $a_1 \dots a_n$ *l'*étiquette *du chemin. L'étiquette du chemin vide est* ε*.*

Par exemple le $(3,b,3)(3,b,3)(3,a,3)(3,b,1)(1,a,2)$ est un chemin de l'automate de la figure 4.1, entre 3 et 2, de longueur 5 et d'étiquette *bbaba*. Il n'est pas acceptant car 3 n'est pas un état initial. En revanche le chemin $(2,b,3)(3,a,3)(3,b,1)(1,a,2)$ est acceptant.

Exercice 61
Donner un chemin acceptant sur l'automate de la figure 4.1 d'étiquette *aaaabb*. Même question avec *baba*.

Definition 62 *Un mot est* accepté *ou* reconnu *par un automate fini s'il est l'étiquette d'un chemin acceptant. L'ensemble des mots reconnus par un automate* $\mathcal{A}$ *est appelé* langage reconnu *par* $\mathcal{A}$*. On le note* $L(\mathcal{A})$*.*

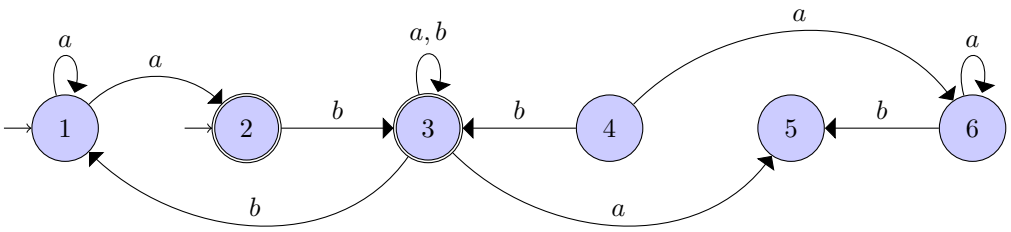


FIGURE 4.2 – États accessibles/co-accessibles

Exercice 63

Dessiner des automates reconnaissant exactement les langages suivants :

- l'ensemble des mots de longueur paire,
- l'ensemble des mots finissant par b,
- l'ensemble des mots commençant pas a,
- l'ensemble des mots contenant aa,
- l'ensemble des mots ayant un nombre paire de a,
- l'ensemble des mots ayant un nombre de b multiple de 3,
- l'intersection des deux langages ci-dessus,
- l'ensemble des mots ne contenant pas aba.
- l'ensemble des mots où tout b est précédé d'un a.
- l'ensemble des mots tels que s'il commence par a, alors il doit finir par b.
- l'ensemble des mots tels que s'il finit par b, alors il doit commencer par a.
- l'ensemble des mots où le nombre de a plus deux fois le nombre de b est un multiple de 3.

Exercice 64

Montrer (en donnant une construction) que si L_1 et L_2 sont reconnaissables par des automates finis, alors $L_1 \cup L_2$ aussi.

4.2 États accessibles, co-accessibles

Definition 65 *Dans un automate fini, un état est dit* accessible *s'il existe un chemin d'un état initial vers cet état. Un état est dit* co-accessible *s'il existe un chemin depuis cet état vers un état final. Un automate est dit* émondé *si tous ses états sont à la fois accessibles et co-accessibles.*

Par exemple, dans l'automate de la figure 4.2, les états 1, 2 et 3 sont à la fois accessibles et co-accessibles. L'état 4 est co-accessible mais pas accessible. L'état 5 est accessible mais pas co-accessible. L'état 6 n'est ni accessible, ni co-accessible.

Exercice 66

Décrire un algorithme, utilisant un parcours de graphe, pour tester si un état est accessible. Même question avec la co-accessibilité.

Exercice 67

Justifier que dans un automate fini, il n'existe pas de chemin réussi passant par un état non accessible. Même chose avec *co-accessible*.

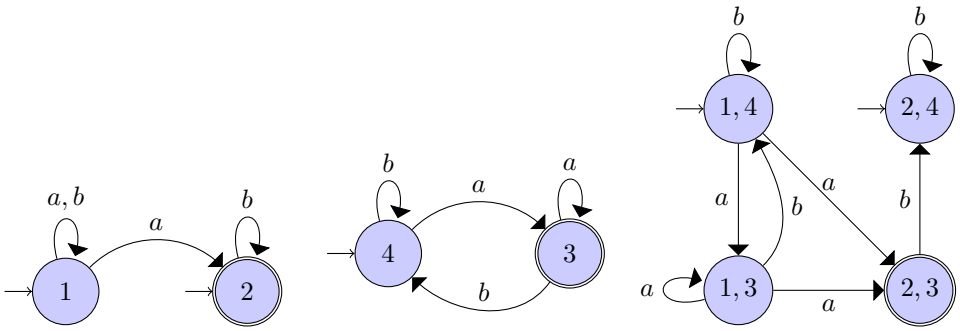
Exercice 68

Pour tout automate fini, montrer qu'il existe un automate émondé reconnaissant le même langage. On pourra s'aider de l'exercice précédent.

Exercice 69

On considère le langage L sur $\{a, b\}$ des mots de la forme $a^k b^k$ où $k \geq 0$. Le but de l'exercice est de montrer qu'il n'existe pas d'automate reconnaissant L. Pour cela, répondre aux questions suivantes :

1. On suppose qu'il existe $\mathcal{A} = (Q, A, E, I, F)$ reconnaissant L. Quel supposition peut-on faire sur $\mathcal{A}$ d'après l'exercice précédent ?
2. On pose Q_i l'ensemble des états atteignables depuis un état initial en lisant a^i. Si $q \in Q_i$, pourquoi les mots que l'on peut lire à partir de q pour arriver en un état final de $\mathcal{A}$ sont de la forme $a^\ell b^{i+\ell}$?
3. En déduire que si $i \neq j$, alors $Q_i \cap Q_j = \emptyset$ (supposer que l'intersection est non vide, prendre un état quelconque dans cette intersection et obtenir un contradiction grace à la question précédente).



(a) $\mathcal{A}_1$ (b) $\mathcal{A}_2$ (c) $\mathcal{A}_1 \times \mathcal{A}_2$

FIGURE 4.3 – Exemple de produit d'automates

4. Justifier que pour tout $i \in \mathbb{N}$, Q_i est non-vide.
5. En utilisant le fait que Q est un ensemble fini, en déduire une contradiction.

Exercice 70

Justifier qu'un automate $\mathcal{A}$ reconnaît le langage vide si et seulement s'il existe un état final accessible, si et seulement s'il existe un état initial co-accessible. En déduire un algorithme utilisant un parcours de graphe pour tester si un automate reconnaît le langage vide.

4.2.1 Produit d'automates

Definition 71 *Soient $\mathcal{A}_1 = (Q_1, A, E_1, I_1, F_1)$ et $\mathcal{A}_2 = (Q_2, A, E_2, I_2, F_2)$ deux automates finis sur le même alphabet. On appelle produit de $\mathcal{A}_1$ par $\mathcal{A}_2$, que l'on note $\mathcal{A}_1 \times \mathcal{A}_2$ l'automate $(Q_1 \times Q_2, A, E, I_1 \times I_2, F_1 \times F_2$ où*

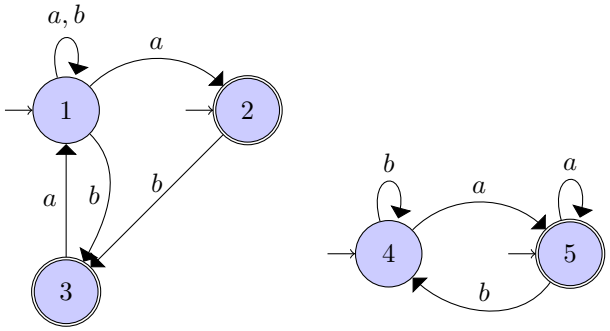
$$E = \{((p_1, p_2), a, (q_1, q_2)) \mid (p_1, a, q_1) \in E_1 \text{ et } (p_2, a, q_2) \in E_2\}.$$

Un exemple est décrit dans la figure 4.3.

Proposition 72 *Pour tout automate $\mathcal{A}_1$ et $\mathcal{A}_2$, $L(\mathcal{A}_1 \times \mathcal{A}_2) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.*

Exercice 73

Calculer le produit des automates de la figure 4.4



(a) $\mathcal{A}_1$ (b) $\mathcal{A}_2$

FIGURE 4.4 – Exercice sur le produit

Exercice 74

Montrer que si p_1 est accessible dans $\mathcal{A}_1$ et p_2 accessible dans $\mathcal{A}_2$, alors (p_1, p_2) est accessible dans $\mathcal{A}_1 \times \mathcal{A}_2$ (pour tout automate $\mathcal{A}_1$ et tout automate $\mathcal{A}_2$). Que penser de la réciproque ? Prouver la proposition 72.

Exercice 75

Proposer un algorithme qui étant donné deux automates teste s'ils reconnaissent au moins un mot en commun.

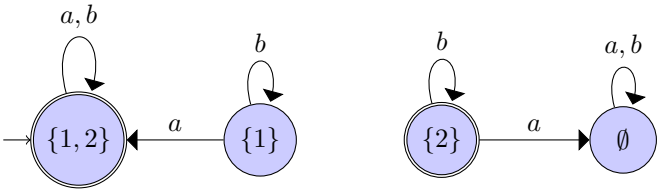


FIGURE 4.5 – Determinisation

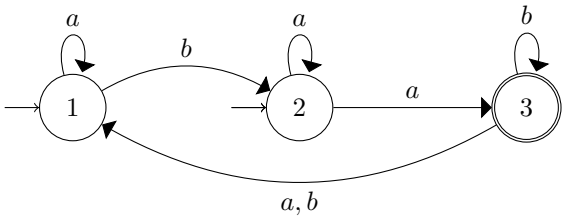


FIGURE 4.6 –

4.2.2 Automates complets

Definition 76 *Un automate est dit* complet *si pour tout état q et toute lettre a, il existe au moins une transition de la forme (p, a, q).*

L'automate $\mathcal{A}_1$ de la figure 4.3 n'est pas complet car aucune transition étiquetée par a ne sort de 2. En revanche, sur la même figure, $\mathcal{A}_2$ est complet.

Exercice 77

Montrer que le produit de deux automates complets est complet.

Exercice 78

Montrer que pour tout automate il existe un automate complet reconnaissant le même langage. Montrer par un exemple que certains langages sont reconnaissable par un automate, mais par aucun automate à la fois complet et émondé.

4.2.3 Automates déterministes

Definition 79 *Un automate est* déterministe *si pour tout état p et toute lettre a il existe au plus une transition de la forme (p, a, q)* **et** *s'il n'a qu'un seul état initial.*

L'automate $\mathcal{A}_1$ de la figure 4.3 n'est pas déterministe car deux transitions étiquetées par a partent de l'état 1. Par ailleurs il dispose aussi de deux états initiaux. En revanche l'automate $\mathcal{A}_2$ sur cette même figure est déterministe.

Definition 80 *Soit $\mathcal{A} = (Q, A, E, I, F)$ un automate fini. Le* déterminisé *de $\mathcal{A}$ est l'automate $(2^Q, A, E', \{I\}, F')$ où $F' = \{P \subseteq Q \mid P \cap F \neq \emptyset\}$ et $(P, a, R) \in E$ si et seulement si*

$$R = \{q \mid \exists p \in P \text{ tel que } (p, a, q) \in E\}.$$

Considérons par exemple l'automate $\mathcal{A}_1$ de la figure 4.3. Son ensemble d'état est $\{1, 2\}$ donc l'ensemble des états de son déterminisé est $\{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$. L'automate $\mathcal{A}_1$ a deux états initiaux, donc son déterminisé a pour état initial $\{1, 2\}$. L'automate $\mathcal{A}_1$ a un seul état final $\{2\}$ donc les états finaux de son déterminisé sont $\{2\}$ et $\{1, 2\}$. Il faut ensuite construire les transition. En partant de $\{1, 2\}$ et en lisant a, on obtient $\{1, 2\}$. En partant de $\{2\}$ et en lisant b on obtient le vide. En faisant ainsi pour chaque sous-ensemble, on obtient au final l'automate de la figure 4.5.

Exercice 81

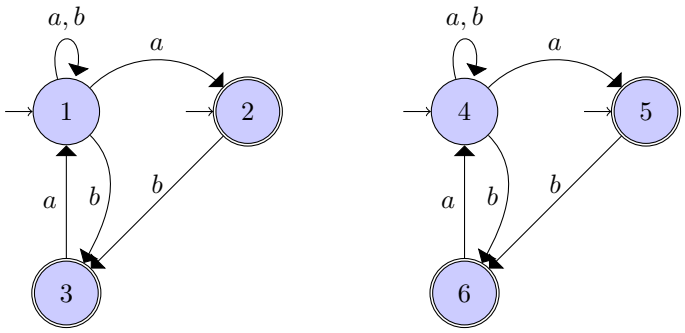
Déterminiser l'automate $\mathcal{A}_1$ de la figure 4.4.

Exercice 82

Déterminiser l'automate de la figure 4.6.

Proposition 83 *Un automate et son déterminisé reconnaissent le même langage.*

Exercice 84



(a) $\mathcal{A}_1$ (b) $\mathcal{A}_2$

FIGURE 4.7 – Isomorphisme

Montrer la proposition 83.

Exercice 85

Montrer que si un langage est reconnaissable par un automate fini, alors son complémentaire aussi.

Théorème 86 *Il y a équivalence entre*

1. *L est reconnaissable par un automate fini,*
2. *L est reconnaissable par un automate fini déterministe,*
3. *L est rationnel.*

L'équivalence entre les points 1. et 2. a été vu dans ce chapitre. Montrer que tout langage reconnaissable est rationnel n'est pas très difficile. La réciproque est un peu plus technique.

Exercice 87

On travaille sur l'alphabet $\{a, b\}$.

1. Dessiner un automate reconnaissant les mots dont l'avant-avant dernière lettre est un a.
2. Le déterminiser.

Exercice 88

Pour chacune des combinaisons des propriétés (*déterministe, complet, accessible, co-accessible*), dessiner un automate ayant ou non ces propriété - ce qui fait 16 automates.

4.2.4 Isomorphismes d'automates

Definition 89 *Deux automates $\mathcal{A}_1 = (Q_1, A, E_1, I_1, F_1)$ et $\mathcal{A}_2 = (Q_2, A, E_2, I_2, F_2)$ sur une même alphabet sont* isomorphes *s'il existe une application bijective φ de Q_1 dans Q_2 telle que $\varphi(I_1) = I_2$, $\varphi(F_1) = F_2$ et $(p, a, q) \in E_1$ ssi $(\varphi(p), a, \varphi(q)) \in E_2$.*

Intuitivement, deux automates sont isomorphes s'ils sont égaux au nom des états près.
Par exemple les deux automates de la figure 4.7 sont isomorphes avec $\varphi(1) = 4$, $\varphi(2) = 5$ et $\varphi(3) = 6$.

Proposition 90 *Deux automates isomorphes reconnaissent le même langage.*

Exercice 91

Montrer que la réciproque de la proposition 89 est fausse.

Proposition 92 *La relation est isomorphe à est une relation d'équivalence sur l'ensemble des automates de même alphabet.*

Exercice 93

Montrer la proposition 91.

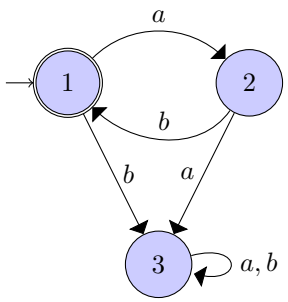


FIGURE 4.8 – Automate minimal de $(ab)^*$.

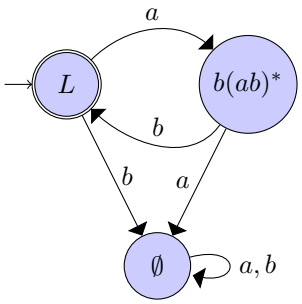


FIGURE 4.9 – Automate minimal de $(ab)^*$.

4.2.5 Automate minimal

Théorème 94 *Pour tout langage reconnaissable, il existe un unique automate déterministe complet (à isomorphisme près) reconnaissant ce langage et ayant un nombre minimal d'états. On l'appelle* automate minimal *du langage.*

Par exemple, on peut montrer que l'automate minimal de $(ab)^*$ est celui de la figure 4.8.

Proposition 95 *Dans tout automate minimal, pour tout couple d'états p, q distincts, il existe un mot u tel que l'on puisse atteindre un état final depuis p en lisant u, mais pas depuis q, ou le contraire.*

Si on reprend l'exemple de la figure 4.8, on peut lire ab depuis 1 pour arriver dans un état final, mais en le lisant à partir de 2 ou 3, on n'arrive pas à un état final.

Proposition 96 *Soit L un langage régulier. Le langage L a un nombre fini de quotients gauche. De plus l'automate*

$$(\{u^{-1}L \mid u \in A^*\}, A, \{(u^{-1}L, a, (ua)^{-1}L) \mid u \in A^*\}, \{L\}, \{u^{-1}L \mid u \in L\})$$

est l'automate minimal de L.

Par exemple, pour $L = (ab)^*$. On a $a^{-1}L = b(ab)^*$, $b^{-1}L = \emptyset$, $(aa)^{-1}L = \emptyset$, $(ab)^{-1}L = L$, et pour tout u, $u^{-1}\emptyset = \emptyset$. L'automate minimal de L est donc l'automate de la figure 4.9.

Il existe plusieurs algorithmes vus en TP pour minimiser un automate.

Exercice 97
Soit L_n (avec $n \geq 1$) le langage des mots sur $\{a, b\}$ dont la nième lettre en partant de la fin est un a.

1. Quel est le langage L_1 ?
2. Donner un automate non déterministe $\mathcal{A}_n$ avec $n+1$ états reconnaissant L_n.
3. Déterminiser $\mathcal{A}_2$.
4. Déterminiser $\mathcal{A}_3$.
5. Montrer que l'automate minimal de L_n a un nombre exponentiel, par rapport à n, d'états.

4.2.6 Automates et langages rationnels

Le theorème suivant est connu sous le nom de *théorèeme de Kleene*

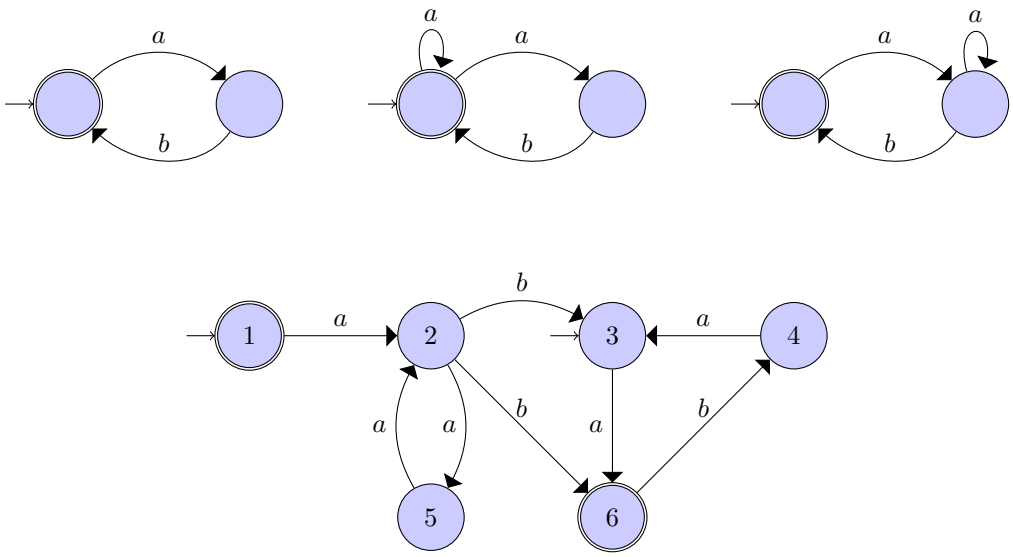
Théorème 98 *Un langage est reconnaissable par un automate fini si et seulement s'il est rationnel.*

Un corrolaire important de ce théorème est que la classe des langages rationnelle est close par intersection est complément, ce qui n'a rien d'évident.

Vous avez vu en TP un algorithme pour transformer un automate fini en une expression régulière (une expression utilisant les langages finis, le produit, l'union et l'étoile) reconnaissant le même langage.

Exercice 99

Donner des expressions régulières reconnaissant les mêmes langages que les automates suivants :



Exercice 100

Soit L le langage des mots sur $\{a, b\}$ contenant le facteur abb. Donner une expression régulière reconnaissant L. Donner une expression régulière reconnaissant le complémentaire de L.

4.3 Exercices (plus difficiles)

Exercice 101

Pour tout langage L on pose $\sqrt{L} = \{u \mid uu \in L\}$. Le but de l'exercice est de montrer que si L est rationnel, alors $\sqrt{L}$ l'est aussi.

1. Que vaut la racine des langages suivants : $(ab)^*$, $\{abaaba\}$, $\{ab, ba\}$, A^*b ?
2. A-t-on $\sqrt{L}^2 = L$ pour tout L ?
3. On fixe à présent un langage n. On appelle $\mathcal{A} = (Q, A, E, \{i\}, F\}$ son automate minimal. Pour toute lettre a, on appelle φ_a l'application de Q dans Q défini par : pour tout état p et toute lettre a, $\varphi(p)$ est l'unique élément de Q tel que $(p, a, \varphi(p)) \in E$. En reprenant l'exemple de la figure 4.8, que sont φ_a et φ_b ?
4. Pour tout mot u on défini φ_u par : φ_ε est l'identité et sinon, si $u = va$ avec $a \in A$, $\varphi_{va} = \varphi_a \circ \varphi_v$. En reprenant l'exemple de la figure 4.8, quelle application est φ_{abab} ? φ_{aba} ?
5. Montrer que dans le cas général $\{\varphi_u \mid u \in A^*\}$ est fini.
6. On considère l'automate fini suivant $\mathcal{B}$:

$$(\{\varphi_u \mid u \in A^*\}, A, \{(\varphi_u, a, \varphi_{ua}) \mid u \in A^*,\ a \in A\}, \{\varphi_\varepsilon\}, \{\varphi_u \mid u \in L\}).$$

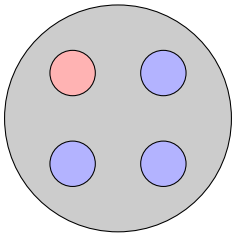
   Montrer que $L(\mathcal{B}) = L$.
7. Montrer que l'automate

$$(\{\varphi_u \mid u \in A^*\}, A, \{(\varphi_u, a, \varphi_{ua}) \mid u \in A^*,\ a \in A\}, \{\varphi_\varepsilon\}, \{\varphi_u \mid u^2 \in L\}),$$

   reconnaît $\sqrt{L}$.

Exercice 102

Le jeu du barman aveugle (historiquement c'est ainsi que s'appelle le jeu) est le suivant : sur un plateau rond tournant 4 jetons identiques sont placés en carré, comme sur la figure ci-dessous. Chaque jeton possède une face bleue et une face rouge.



Le joueur A a les yeux bandés (le barman). Le joueur B place les 4 jetons sur les faces qu'il désire, avec la seule contrainte que tous les jetons ne soient pas sur la même face (couleur). A chaque coup, le joueur B tourne le plateau de 0, 90, 180 ou 270 degrés, selon son choix. Le joueur A ne sait pas de combien le plateau a tourné. Puis le joueur A, retourne autant de jetons qu'il veut (en gardant les yeux bandés). Lorsqu'il a fini, si tous les jetons ont la même couleur, il a gagné.

Montrer que A a une stratégie gagnante, c'est-à-dire que A peut gagner à tous les coups s'il joue bien.

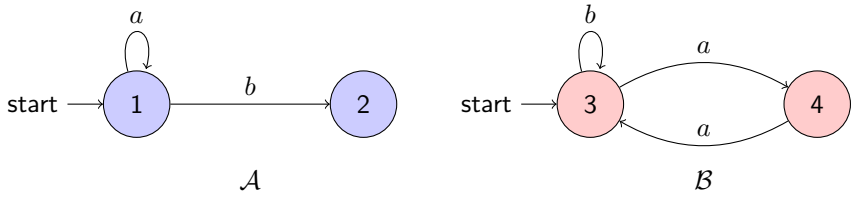
Exercice 103

On considère deux automates $\mathcal{A}$ et $\mathcal{B}$ et deux joueurs A et B.

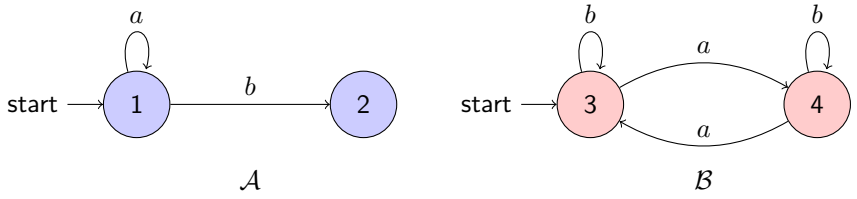
Au premier coup, A place un jeton sur un état initial de l'automate $\mathcal{A}$. Puis B fait de même sur l'automate $\mathcal{B}$.

A chaque coup, A bouge son jeton sur l'automate $\mathcal{A}$ en suivant une transition de son choix. B doit faire de même sur l'automate $\mathcal{B}$ en utilisant une transition ayant la même lettre. Si B ne peut pas, A gagne. Si l'on revient dans une configuration déjà jouée, B gagne.

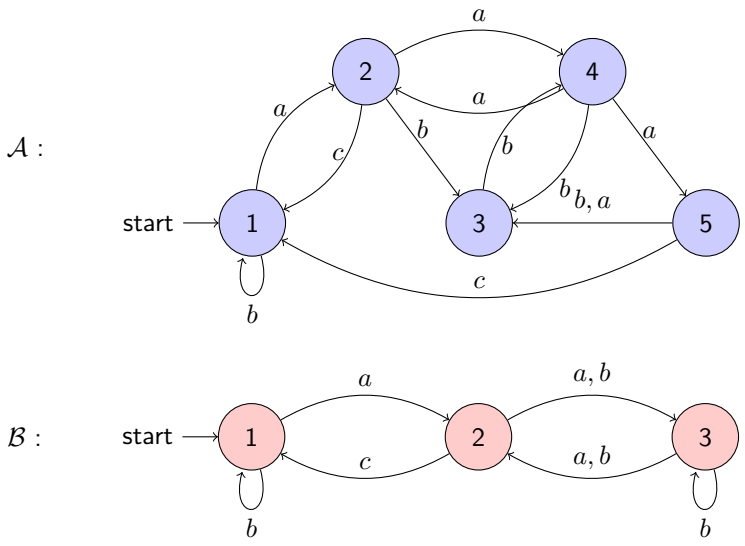
1. Montrer que toutes les parties terminent (on ne peut pas jouer indéfiniment).
2. Montrer que sur l'exemple ci-dessous, A a une stratégie gagnante (si A joue bien, il est sûr de gagner).



3. Montrer que sur l'exemple ci-dessous, B a une stratégie gagnante.



4. Qui a une stratégie gagnante dans le cas suivant ?



5. Donner un algorithme pour décider si A a une stratégie gagnante.
6. Montrer/Justifier que si B a une stratégie gagnante, alors $L(\mathcal{A}) \subseteq L(\mathcal{B})$. Montrer que la réciproque n'est pas vraie.