TD d'Éléments d'Algorithmique n° 6
(Correction)

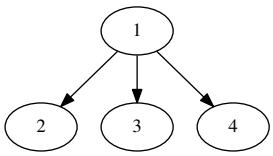
I) Graphes

Exercice 1. *Petits exemples de graphes.*

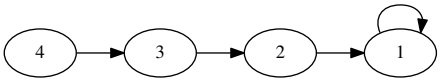
1. Dessiner les graphes orientés G_1, G_2, G_3 correspondant aux matrices d'adjacence suivantes :

$$M_1 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad M_3 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

2. Donner les matrices d'adjacences M_4 et M_5 des graphes orientés suivants :

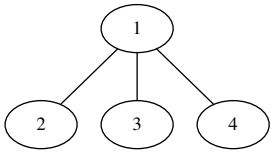


(a) G_4

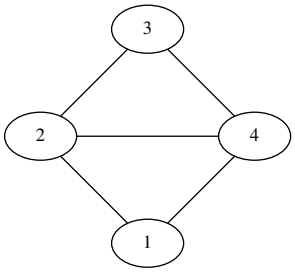


(b) G_5

3. Donner la matrice d'adjacence M_6 et M_7 des graphes non orientés suivants :



(c) G_6

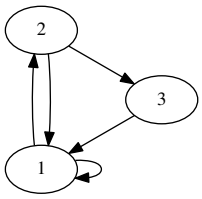


(d) G_7

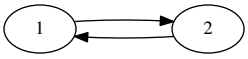
4. Parmi ces graphes :
   - lesquels ont un chemin du sommet 2 au sommet 1 ?
   - lesquels sont *fortement connexes* (c'est-à-dire qu'entre deux sommets, il existe toujours un chemin) ?

5. Calculer les puissances successives des matrices M_1, ..., M_5 dans le semi-anneau de Boole $(\{0,1\}, \vee, \wedge, 0, 1)$ (où l'addition est le "ou" $\vee$, et la multiplication est le "et" $\wedge$).

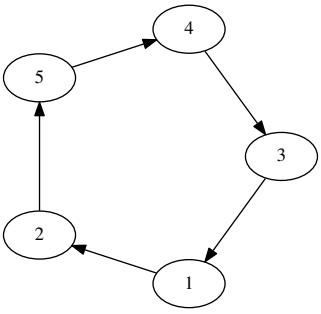
Correction :



(e) G_1



(f) G_2



(g) G_3

1.
2. TODO
3. TODO
4. chemin de 2 à 1 : tous sauf G_4 ; fortement connexe : tous sauf G_4, G_5.
5. TODO

Exercice 2. *Dénombrement et énumération de graphes.*

1. Combien y a-t-il de graphes orientés (avec ou sans boucles — une boucle est une arête d'un sommet vers lui-même) à n sommets ? Écrire un programme qui les énumère.
2. Combien y a-t-il de graphes orientés sans boucles à n sommets ? Écrire un programme qui les énumère.
3. Combien y a-t-il de graphes non orientés sans boucles à n sommets ? Écrire un programme qui les énumère.
4. Combien y a-t-il de graphes orientés à n sommets et k arêtes ?

Correction :

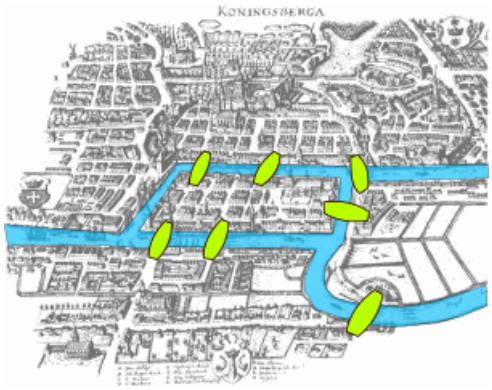
1. 2^{n^2}
2. $2^{n(n-1)}$
3. $2^{n(n-1)/2}$
4. $\binom{n^2}{k}$

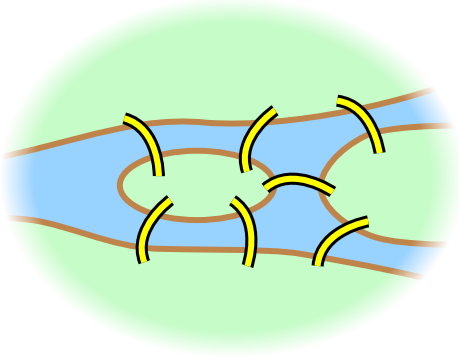
Exercice 3. *Problème des sept ponts de Königsberg.*

La ville de Königsberg (aujourd'hui Kaliningrad, Russie) est construite autour de deux îles situées sur le Pregel et reliées entre elles par un pont. Six autres ponts relient les rives de la rivière à l'une ou l'autre des deux îles, comme représentés sur le plan ci-dessouss.

Le problème consiste à déterminer s'il existe ou non une promenade dans les rues de Königsberg permettant, à partir d'un point de départ au choix, de passer une et une seule fois par chaque pont, et de revenir à son point de départ, étant entendu qu'on ne peut traverser le Pregel qu'en passant sur les ponts.

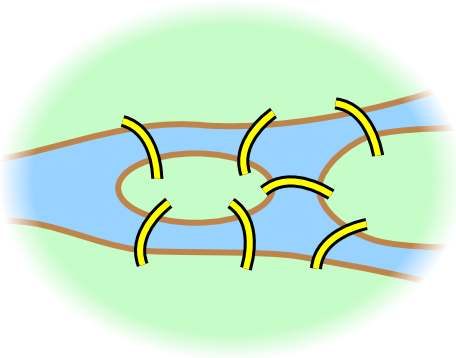
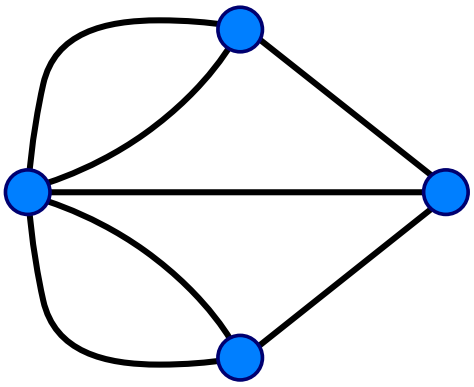
Extrait de Wikipedia (`http://fr.wikipedia.org/wiki/Probl%C3%A8me_des_sept_ponts_de_K%C3%B6nigsberg`)





1. Dessiner un graphe non orienté représentant le problème (où chaque arête représentera un pont). Exceptionnellement, dans cet exercice, le graphe pourra avoir plusieurs arêtes entre deux sommets.
2. Un graphe est dit *eulérien* s'il existe un chemin d'un sommet s (arbitraire) à lui-même passant par toutes les arêtes du graphe une et une seule fois. Vérifier que le problème consiste à déterminer si le graphe est eulérien ou non.
3. Prouver que si un graphe est eulérien alors chaque sommet est de degré pair (i.e., chaque sommet est incident à un nombre pair d'arêtes ; ou encore pour chaque sommet, il existe un nombre pair d'arêtes passant par ce sommet).
4. Conclure.
5. Combien de ponts faut-il construire au minimum pour permettre une telle promenade ? Ou plus radicalement, quels ponts faut-il détruire pour rendre ce parcours possible tout en causant un minimum de dégâts, et sans déconnecter la ville ?

Correction :

1.

Exercice 4. *Lemme d'Ogden.*

Un arbre binaire complet est un arbre avec deux types de sommets :

- les *nœuds* internes qui ont chacun un fils gauche et un fils droit
- les *feuilles* qui n'ont pas de fils.

On considère des arbres binaires complets, dans lesquels les feuilles peuvent être marquées. On dit alors qu'un nœud (interne) est une *jonction* si et seulement si chacun de ses deux sous-arbres contient au moins une feuille marquée.

Le *lemme d'Odgen* dit que si un arbre binaire complet contient au moins $k = 2^h$ feuilles marquées alors il contient une feuille f telle que la branche joignant f à la racine passe par au moins h jonctions.

La *profondeur d'une feuille* est le nombre d'arêtes qui la sépare de la racine. La *profondeur d'un arbre binaire complet* est la profondeur de sa feuille la plus profonde.

1. Montrer qu'un arbre binaire complet de profondeur h contient au plus 2^h feuilles.
2. Démontrer le lemme d'Odgen.
3. Donner un algorithme qui prend en entrée un arbre binaire complet avec n sommets dont au moins 2^h feuilles sont marquées et qui renvoit une feuille telle que le chemin de la racine à f passe par h jonctions. Quelle est la complexité de votre algorithme ?

Correction :

1. Si un arbre de profondeur h contient une feuille à profondeur inférieure à h alors on obtient un arbre de profondeur h avec plus de feuilles en remplaçant cette feuille par un nœud avec 2 fils. On peut donc supposer que toutes les feuilles sont à profondeur h. Or il y a 2^h chemins possibles entre la racine et une telle feuille, et on ne peut pas arriver à deux feuilles distinctes par le même chemin.
2. Considérons un arbre binaire complet A avec 2^h feuilles marquées. L'ensemble des chemins entre la racine et les 2^h feuilles marquées de A forme un arbre binaire complet B dont les nœuds sont les jonctions de A et les arêtes des branches de A. Vu le nombre de feuilles de B cet arbre a profondeur au moins h (question précédente) : il contient donc une feuille à profondeur au moins h, c'est la feuille recherchée pour le lemme.
3. - Solution récursive : on va écrire une fonction qui renvoit prend un arbre A et renvoit un couple (f, p) tel que : f est une feuille ; il y a p jonctions entre la racine de A et f ; et f est la feuille de A avec le plus de jonctions au dessus d'elle. Pour calculer (f, p) pour l'arbre $A = (A_1, A_2)$ on applique récursivement la fonction à A_1 pour avoir (f_1, p_1) et à A_2 pour avoir (f_2, p_2) et on determine $i \in \{1, 2\}$ tel que $p_i = \max(p_1, p_2)$ et on renvoit $(f_i, p_i + 1)$ si p_1 et p_2 sont tous les 2 non nuls, (f_i, p_i) sinon.
   - Solution iterative : Un parcours en profondeur à gauche permet de calculer lors des remontées le couple (f, p) associé à chaque nœud comme précédement : lorsqu'on quitte un nœud après avoir visité les 2 sous arbres on a récupéré (f_1, p_1) et (f_2, p_2) et on peut faire le calcul comme dans le cas récursif.