TD13: Graphes

MP2I Lycée Pierre de Fermat

Dans tout le TD, on considère des graphes **finis**. Certaines questions sont marquées d'un **(*)**, ce qui signifie que vous trouverez à la fin du TD une indication pour cette question si vous bloquez.

Définition 1. Soit $G = (S, A)$ un graphe et $S' \subseteq S$. Le sous-graphe de G induit par S' est $G' = (S', A \cap S'^2)$. C'est donc le graphe obtenu en ne gardant que les sommets de S', et que les arêtes dont les deux bouts sont dans S'.

Exercice 1.

Exercices divers

Q1. Représenter les graphes correspondant aux matrices d'adjacence suivantes. Sont-ils non-orientés ?

(i) $\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$ (ii) $\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$ (iii) $\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$

Q2. Donner les représentations par listes d'adjacence des graphes précédents.

Q3. Montrer qu'un graphe non-orienté sans boucles possède un nombre pair de sommets de degré impair.

On considère l'hypercube de dimension n, noté H_n qui est un graphe dont l'ensemble des sommets est $\{0,1\}^n$, et dans lequel deux sommets $(a_1, \ldots, a_n)$ et $(b_1, \ldots, b_n)$ sont reliés si et seulement si ils diffèrent d'exactement une coordonnée.

Q4. Dessiner H_2, H_3, H_4.

Q5. Donner le nombre de sommets de H_n, ainsi que le degré de ses sommets. En déduire le nombre d'arêtes de H_n.

Exercice 2.

Graphes connexes, graphes acycliques

On considère des graphes non-orientés, sans boucles.

Soit $G = (S, A)$ un graphe non orienté, et notons $n = |S|, m = |A|$. On dit que G est acyclique s'il ne contient aucun cycle.

Q1. Dessiner un graphe à 10 sommets, acyclique, avec 3 composantes connexes, dont chacune a au moins 2 sommets. Combien d'arêtes a t-il ?

Nous allons montrer les deux propriétés suivantes :

1. Si G est connexe, alors $m \geq n - 1$
2. Si G est acyclique non vide, alors $m \leq n - 1$

Graphes connexes Nous allons montrer la propriété 1 par récurrence forte sur le nombre de sommets n de G.

Q2. Dessinez quelques graphes connexes pour vous convaincre du résultat.

Q3. Que dire pour $n = 1$?

Q4. Soit $G = (S, A)$ connexe possédant au moins 2 sommets, et soit u un sommet quelconque de G. On considère le sous-graphe G' de G induit par $S \setminus \{u\}$. Montrez que le nombre de composantes connexes de G' est au plus $\mathbf{deg}(u)$.

Q5. On reprend le graphe G' de la question précédente. En notant $C_1, \ldots, C_p$ les composantes connexes de G, en notant $G_1, \ldots, G_p$ les sous-graphes induits de G correspondants, et en notant n_i le nombre de sommets de G_i, m_i le nombre d'arêtes de G_i pour $i \in [\![1, p]\!]$:

a) Donnez un lien entre m, $\mathbf{deg}(u)$ et les m_i.

b) Donnez un lien entre n et les n_i.

Q6. Montrez par récurrence forte la propriété 1.

Graphes acycliques La preuve de la propriété 2 se fait aussi par récurrence, faible cette fois, sur le nombre de sommets du graphe.

Q7. Dessinez quelques graphes acycliques pour vous convaincre du résultat.

Q8. (*) Montrez qu'un graphe acyclique non vide possède un sommet de degré au plus 1.

Q9. Montrez par récurrence la propriété 2.

Arbres Un graphe est un **arbre** s'il est connexe ET acyclique.

Soit $G = (S, A)$ un graphe, avec $n = |S|$ et $m = |A|$. Nous allons montrer que les trois propriétés suivantes sont équivalentes :

(i) G est un arbre

(ii) G est connexe et $m = n - 1$.

(iii) G est acyclique et $m = n - 1$.

On pourra donc voir les arbres comme des **graphes connexes minimaux**, ou bien comme des **graphes acycliques maximaux**.

Q10. Montrer que (i) implique (ii) et (iii)

Q11. (*) Montrer par l'absurde que (ii) implique (iii).

Q12. On considère un graphe $G = (S, A)$ non orienté acyclique, avec $n = |S|, m = |A|$ et p composantes connexes. Donner une relation entre n, m et p.

Q13. En déduire que (iii) implique (ii).

Exercice 3. *Line graph*

On considère un graphe $G = (S, A)$ non-orienté. Le **line graph** de G, noté $L(G)$ est définit comme suit.

— L'ensemble des sommets de $L(G)$ est l'ensemble des arêtes de G, A.

— Deux sommets de $L(G)$ sont reliés par une arête si les arêtes correspondantes dans G ont une extrémité en commun.

Q1. Dessiner le graphe dont la matrice d'adjacence est la suivante, puis dessiner son line graph.

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

Q2. Dessiner le line graphe de K_4 (le graphe complet à 4 sommets, c'est à dire le tétraèdre).

Q3. Montrer que si un graphe est régulier, de degré d, alors son line-graph est régulier, et donner son degré.

Q4. (*) Si $G = (S, A)$ est un graphe non-orienté, avec n sommets et m arêtes, déterminer le nombre de sommets et d'arêtes de $L(G)$.

Exercice 4. *Démo du cours : parcours en largeur*

Soit $G = (S, A)$ un graphe orienté. On définit la distance entre deux sommets $u \in S$ et $v \in S$ comme la longueur du plus court chemin les reliant, ou $+\infty$ si aucun chemin ne relie u à v.
On considère le graphe $G_0 = (S_0, A_0)$ ci-dessous :

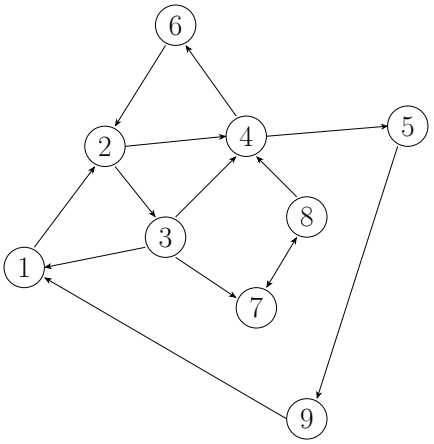

FIGURE 1 – Graphe G_0

Sur ce graphe par exemple, la distance entre les sommets 5 et 2 est de 3 car il existe un chemin de taille 3 ($5 \to 9 \to 1 \to 2$) mais aucun chemin strictement plus court.

Q1. Donner les distances entre le sommet 1 et chaque sommet du graphe, puis entre 4 et chaque sommet du graphe.

Nous avons vu empiriquement que le parcours en largeur d'un graphe énumère ses sommets par ordre croissant de distance à la source. Le but de cet exercice est de démontrer cette propriété, et d'en déduire que le parcours en largeur construit bien des plus courts chemins. On propose l'algorithme suivant :

Algorithme 1 : Parcours en largeur : calcul des distances

Entrée(s) : $G = (S, A)$ un graphe, $s \in S$
Sortie(s) : d tableau des distances de s à chaque sommet de G

```
1  d ← dictionnaire vide // d[u] contient la distance entre s et u
2  F ← file vide;
3  d[s] = 0;
4  F.enfiler(s);
5  tant que F non vide faire
6      u ← F.defiler();
7      pour v voisin de u faire
8          si d[v] = +∞ alors
9              d[v] = d[u] + 1;
10             F.enfiler(v);
11 retourner d
```

Q2. Appliquer l'algorithme sur G_0 en partant du sommet 1, puis en partant du sommet 4. Noter les tableaux d obtenus.

Q3. On note $|d|$ le nombre de clés du dictionnaire d. Donner un variant de la boucle while et en déduire la terminaison de l'algorithme.

Q4. Selon l'implémentation utilisée pour les graphes, quelle sera la complexité de cet algorithme ?

Montrons maintenant que le tableau d renvoyé correspond bien au tableau des distances.
On fixe dans la suite le sommet s depuis lequel on a lancé le parcours, et on note $\delta(u)$ la distance entre s et u. Il faut donc montrer qu'à la fin de l'algorithme $d[u] = \delta(u)$ pour tout $u \in S$.

Q5. (*) Montrez qu'à la fin de l'algorithme, pour toute clé u de d, $\delta(u) \leq d[u]$.

Nous allons maintenant montrer que $d[u]$ atteint exactement $\delta(u)$. Nous allons formaliser le fait que le parcours en largeur traite les sommets par ordre de distance à la source, en fractionnant l'algorithme selon les instants où tous les sommets d'une distance donnée sont dans la file en même temps.

Pour $n \in \mathbb{N}$, on pose $\mathcal{P}(n)$ la propriété suivante : "Il existe un numéro de tour de la boucle while t_n dans l'exécution de l'algorithme tel que :

(i) Pour tout sommet $u \in S$ tel que $\delta(u) \leq n$, $d[u] = \delta(u)$;
(ii) Pour tout sommet $u \in S$ tel que $\delta(u) > n$, u n'est pas une clé de d ;
(iii) La file contient exactement les sommets $u \in S$ tels que $\delta(u) = n$."

Q6. Vérifiez que $\mathcal{P}(0)$ est vraie, et que l'instant t_0 correspondant est l'entrée dans la boucle.

On suppose maintenant $\mathcal{P}(n)$ vraie pour un certain $n \in \mathbb{N}$. En particulier, à l'instant t_n correspondant, la file contient les sommets $u_1, \ldots, u_{k_n}$ qui sont exactement les sommets à distance n de s. On considère l'instant t_{n+1} se situant k_n passages de boucle après t_n, autrement dit lorsqu'on a défilé tous les u_i et enfilé tous leurs voisins encore non vus.

Q7. Montrer que pour $u \in S$, si $\delta(u) = n + 1$, alors à l'instant t_{n+1}, $d[u] = n + 1$. On pourra raisonner sur le sommet précédant u dans un plus court chemin depuis s.

Q8. En déduire que la partie (i) de la propriété est vérifiée à l'instant t_{n+1}.

Q9. Montrer que la partie (ii) de la propriété est vérifiée à l'instant t_{n+1}.

Q10. Montrer que la partie (iii) de la propriété est vérifiée à l'instant t_{n+1}

Q11. Conclure sur la correction de l'algorithme.

Exercice 5.

Démo du cours : Tri topologique

Q1. Rappeler ce qu'est un tri topologique d'un graphe. A quelle condition nécessaire et suffisante un graphe admet-il un tri topologique ?

On rappelle l'algorithme plus efficace proposé en cours pour calculer un tri topologique :

Algorithme 2 : Tri topologique

Entrée(s) : $G = (S, A)$ graphe orienté sans cycle
Sortie(s) : L tri topologique de G

```
1  n ← |S|;
2  L ← liste vide;
3  P ← pile vide;
4  d⁻ ← dictionnaire vide ;
   // Initialiser d⁻
5  pour u sommet de G faire
6      d⁻[u] ← 0;
7  pour u sommet de G faire
8      pour v successeur de u faire
9          d⁻[v] = d⁻[v] + 1;
   // Empiler les sources
10 pour u sommet de G faire
11     si d⁻[u] = 0 alors
12         P.empiler(u);
   // Parcours
13 tant que P non vide faire
14     u ← P.depiler();
15     Ajouter u à la fin de L;
16     pour v voisin de u faire
17         d⁻[v] = d⁻[v] − 1;
18         si d⁻[v] = 0 alors
19             P.empiler(v);
20 retourner L
```

Q2. Appliquer l'algorithme précédent pour calculer un tri topologique du graphe sur les sommets $\{1, \ldots, 7\}$ dont les listes d'adjacence sont :

```
[
    [2, 5],
    [3],
    [4],
    [],
    [3],
    [2, 5, 7],
    [4]
]
```

On considère les deux propriétés suivantes :

$$\forall u \in S, d^-[u] = |\{v \in S \mid (v,u) \in A \text{ et } v \notin L\}| \quad (1)$$

$$\forall u \in S, d^-[u] = 0 \iff (u \in P \text{ ou } u \in L) \quad (2)$$

Q3. Montrer que ces deux propriétés sont bien des invariants de la boucle while.

Nous allons utiliser ces deux propriétés pour montrer la correction de l'algorithme. Pour cela, nous voulons montrer que L énumère bien tous les sommets de S, et que l'ordre de chaque arête est bien bien respecté.
On se place en sortie de la boucle while.

Q4. Montrer que si $(x,y) \in A$ et x, y apparaissent tous les deux dans L, alors x apparaît avant y dans L. On pourra considérer l'instant où y a été ajouté dans L.

Q5. (*) Montrer que s'il existe un sommet $u \in S$ tel que u n'apparaît pas dans L, alors il existe un sommet $u' \in S$ (éventuellement égal à u) n'apparaissant pas dans L **et** dont tous les prédécesseurs sont dans L.

Q6. En déduire qu'il n'existe pas de sommet $u \in S$ n'apparaissant pas dans L, et conclure sur la correction de l'algorithme.

Exercice 6.

Matrice d'incidence

Soit $G = (S, A)$ un graphe orienté sans boucles, avec $n = |S|$ et $m = |A|$. On note $S = \{s_1, \ldots, s_n\}$ et $A = \{a_1, \ldots, a_m\}$ La matrice d'incidence de G est $M = (m_{ij})_{1 \leq i \leq n, 1 \leq j \leq m}$ définie par :

$$\forall i \in [\![1,n]\!], \forall j \in [\![1,m]\!], m_{ij} = \begin{cases} 1 \text{ si l'arête } a_j \text{ arrive sur } s_i \\ -1 \text{ si l'arête } a_j \text{ part de } s_i \\ 0 \text{ sinon} \end{cases}$$

La matrice d'incidence indique ainsi les relations entre les différentes arêtes et sommets d'un graphe.

Q1. Dessiner le graphe dont la matrice d'incidence est :

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

Q2. Quelle est la complexité spatiale nécessaire pour cette représentation ?

Q3. Donner un algorithme permettant de déterminer si deux sommets (u, v) d'un graphe sont voisins, à partir de cette représentation. Quelle est sa complexité ?

Q4. Donner un algorithme permettant de récupérer la liste des successeurs d'un sommet u dans cette représentation. Quelle est sa complexité ?

Q5. Soit $G = (S, A)$ un graphe, et M sa matrice d'incidence. Décrire la matrice d'incidence du graphe obtenu en inversant le sens de toutes les arêtes de G.

Q6. Soit $G = (S, A)$ un graphe, et M sa matrice d'incidence. On note ${}^tM$ sa transposée. Que contient la matrice $M\,{}^tM$?

Exercice 7. *Chemins eulériens et hamiltoniens*

On considère des graphes non orientés, sans boucles.

Un chemin dans un graphe G est **hamiltonien** s'il passe une fois et une seule par chaque sommet de G. On dit qu'un cycle est hamiltonien s'il passe une fois et une seule par chaque sommet, sauf le sommet qui sert de départ et d'arrivée, qui est visité deux fois.

Un chemin est **eulérien** s'il passe une fois et une seule par chaque arête.

Q1. Dans les graphes suivants, dire s'il existe des chemins eulériens / cycles eulériens / chemins hamiltoniens / cycles hamiltoniens, et les donner lorsqu'ils existent :

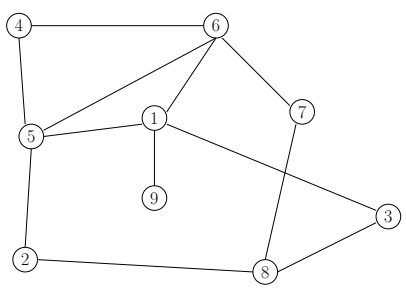

FIGURE 2 – G_1

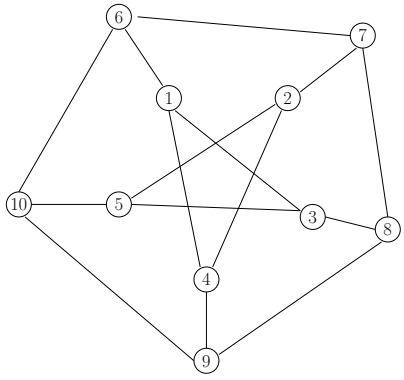

FIGURE 3 – G_2

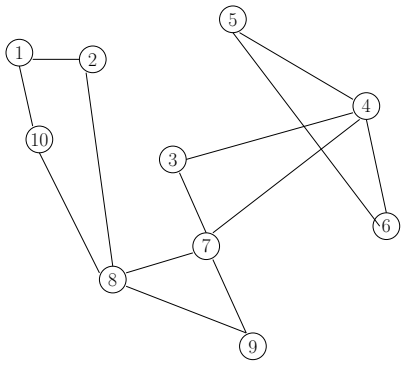

FIGURE 4 – G_3

Q2. Montrer que si un graphe admet un cycle eulérien, alors il est connexe, et tous ses sommets sont de degré pair.

Montrons maintenant que c'est une condition nécessaire et suffisant : un graphe admet un cycle eulérien si et seulement si il est connexe et que tous ses sommets sont de degré pair (on dira que G est de degrés pairs).

Q3. (*) Montrer qu'un graphe connexe de degrés pairs admet un cycle d'au moins 3 sommets. Déduire de votre preuve un algorithme permettant de trouver un tel cycle.

Q4. Soit $G = (S, A)$ un graphe connexe de degrés pairs, et C un cycle. Montrer que le graphe G' obtenu en supprimant de G les arêtes de C est toujours de degrés pairs.

Q5. (*) Déduire des deux questions précédentes un algorithme permettant de construire un cycle eulérien dans un graphe de degrés pairs.

Q6. On considère maintenant des **chemins** eulériens. Donner un critère nécessaire et suffisant pour qu'un graphe admette un chemin eulérien n'étant pas un cycle. Démontrer ce critère en réutilisant le résultat montré sur les cycles eulériens.

Rappel : un DAG (directed acyclic graph) est un graphe orienté sans cycle.

Q7. Montrer que si un DAG admet un chemin hamiltonien, alors il admet un unique tri topologique.

Q8. Proposer un algorithme polynomial pour déterminer s'il existe un chemin hamiltonien dans un DAG.

Pour les graphes quelconques, savoir s'il existe un chemin ou un cycle hamiltonien est très dur : c'est un problème **NP**-complet, et on ne connaît donc pas à l'heure actuelle d'algorithme polynomial pour y répondre.

Q9. Montrer que si un graphe G admet un cycle hamiltonien, alors il est 2-connexe, c'est à dire que G est connexe et qu'en enlevant n'importe quel sommet s de G, le sous-graphe induit obtenu est aussi connexe.

Exercice 8. *Chaînes maximales*

L'objectif de cet exercice est de démontrer le résultat suivant :

Proposition 1. Soit G un graphe non-orienté connexe. Soient C et D deux chaînes sans cycles de longueurs maximales dans G. Alors, C et D ont un sommet en commun.

Soit $G = (S, A)$ un graphe non-orienté connexe. Supposons, par l'absurde, qu'il existe dans G deux chaînes $C = u_0u_1 \dots u_k$ et $D = v_0v_1 \dots v_k$ de longueurs maximales, sans aucun sommet en commun.

Q1. Faire un schéma de la situation.

Q2. Montrer qu'il existe deux indices $i, j \in [\![0, k]\!]$ tels qu'il existe un chaîne entre u_i et v_j ne passant par aucun autre sommet de C ou D.

Q3. Conclure.

Indications

— Exercice 2, Question 8 : on pourra raisonner par l'absurde, et arriver à construire une suite infinie de sommets distincts.

— Exercice 2, Question 11 : essayez de construire un graphe connexe n'ayant pas assez d'arêtes.

— Exercice 3, Question 4 : A quoi correspond un sommet de G dans $L(G)$?

— Exercice 4, Question 5 : on pourra montrer que $d[u]$ contient toujours la longueur d'un chemin valide de s vers u.

— Exercice 5, Question 5 : on pourra raisonner par l'absurde et construire un cycle dans G.

— Exercice 7, Question 3 : on pourra raisonner par l'absurde, et arriver à construire une suite infinie de sommets distincts.

— Exercice 7, Question 5 : on pourra raisonner récursivement, et utiliser les deux questions précédentes pour casser un graphe connexe en lui enlevant un cycle.