Théorie des Graphes
Connexité et Chemins

TPS 1A INOC

theoleyre@unistra.fr

Objectif : Propriétés fondamentales de connexité et de recherche de chemins

Durée : 3 heures

Partie 1

Graphes non orientés

Exercice 1: Connexité et transitivité

Quelle propriété satisfait la relation d'un graphe non orienté ?
Montrer qu'un graphe $G = (S, A)$ est connexe si et seulement si sa fermeture transitive est le graphe complet d'ordre $|S|$.

Réponse :
La relation d'un graphe non orienté est symétrique.
Soit A_i la suite définie par $A_0 = A$ et $A_{i+1} = A_i \cup \{(x, z) \mid (\exists y \mid (x, y) \in A_i \wedge (x, z) \in A_i)\}$. La partie droite de la proposition (fermeture transitive) correspond donc à cette suite A_i. Il est nécessaire de montrer les deux sens de la proposition :

1. Soit $G = (S, A)$ un graphe connexe, et $(u, v) \in S$. Il existe donc une chaine de sommets de u à v, que nous noterons $c = \{u, x_0, .., x_i, ..x_{,k-1}, v\}$ (la chaine est donc de longueur k+1).
   Démontrons que la fermeture transitive possède bien une arête quelle que soit la longueur de la chaine de u à v. Supposons que la propriété est vrai pour une chaine de longueur k : c'est donc vrai pour la chaine $c' = \{u...x_{k-1}\}$. Ainsi, A_{k-1} possède une arête (u, x_{k-1}). Ainsi, la fermeture transitive ajoutera une arête entre u et v dans A_k car il existe un sommet x_{k-1} tel qu'il existe une arête (u, x_{k-1}) et (x_{k-1}, v) dans A_{k-1}. La propriété est donc vraie quelle que soit la valeur de k.
   Or, la propriété est trivialement vraie pour k=0 (A_0 comprenant toutes les arêtes du graphe).
2. Supposons maintenant que la fermeture transitive forme un graphe complet. Alors si u et v

sont voisins dans la fermeture transitive, il existe une chaine de u et v. Si u et v sont voisins dans A_k mais ne le sont pas dans A_{k-1}, alors il existe un sommet intermédiaire x_k tels qu'ils sont tous deux voisins de ce sommet dans A_{k-1}. Soit $\{u, x_k, v\}$ cette sous-chaine. Nous pouvons donc ensuite appliquer un raisonnement récursif (chaine construite de façon récursive, en la découpant ensuite en deux sous-chaines, de u vers x_k et de x_k vers v, ces sous-chaines étant toutes deux de longueur strictement inférieure à celle de u vers v).

Exercice 2: Cycles et cardinalités

Soit $G = (S, A)$ un graphe non orienté à n sommets.

1. Soit $n \geq 1$. Démontrer que si G est connexe, alors $Card(A) \geq n - 1$.
2. Soit G complet. Combien possède-t-il d'arêtes ? Démontrer ce résultat.
3. Démontrer que si G est sans cycle, alors $Card(A) \leq n - 1$.

1. Soit $n \geq 1$, supposons G connexe et $|A| < n - 1$. On essaye de construire un tel graphe en ajoutant un à un les sommets en partant de deux sommets reliés par une arête. Pour ajouter un troisième sommet, il faut une arête entre ce sommet et un des autres sommets déjà construits. Pour ajouter le i-ème sommet, il faut une arête. On en est à i sommets et $i - 1$ arêtes. D'où la contradiction.
2. Soit G complet à n sommet. Alors chaque sommet est relié à tous les autres, donc a un degré de $n-1$. On utilise la formule $\sum_{s \in S} deg(s) = 2|A|$ pour écrire $(n-1)n = 2|A|$, d'où $|A| = \frac{n(n-1)}{2}$.
3. Supposons G sans cycle, et $|A| > |S| - 1$ avec $|S| = n$.
   - Soit G est connexe. Montrons qu'alors le retrait de $n - 1$ arêtes entraîne la création de n composantes connexes.

     En effet, si après le retrait de l'arête (a, b), $G = (S, A \setminus \{(a, b)\})$ est encore connexe, il existe un chemin $a = s_0, ..., s_m = b$ dans G', et $a = s_0, ..., s_m, a = a$ est un cycle dans G, d'où la contradiction. Le retrait d'une arête entraîne donc la création de deux composantes connexes G_1 et G_2. Le retrait d'une nouvelle arête, que nous supposons sans perte de généralités être dans G_2, divise G_2 en deux composantes connexes, par le même argument. Finalement, par récurrence, le retrait de $n - 1$ arêtes entraîne la création de n composantes connexes.

     Soit $G' = (S, A')$ le graphe G auquel on a retiré $n - 1$ arêtes. G' a n sommets et n composantes connexes, donc G' n'a pas d'arête et $|A'| = 0$. Or $|A'| = |A| - n - 1 > n - 1 - (n - 1) > 0$, et $0 > 0$, d'où la contradiction.
   - Soit G n'est pas connexe. Alors parmi les composantes connexes de G, il en existe au moins une, disons $G' = (S', A')$, qui vérifie $|A'| > |S'| - 1$.

     En effet, si toutes les composantes connexes $G_i = (S_i, A_i)$ sont telles que $|A_i| \leq |S_i| - 1$, en notant m le nombre de composantes connexes de G, on a $\sum_{1 \leq i \leq m} |A_i| \leq \sum_{1 \leq i \leq m} |S_i| - m$ et $|A| \leq |S| - m$, d'où la contradiction avec l'hypothèse.

     On applique alors le raisonnement du premier cas à G' pour montrer la contradiction.

Exercice 3: Graphe biparti

Soit n,m des entiers naturels non nuls. Le graphe complet biparti $K_{n,m}$ est le graphe dont l'ensemble des sommets est partitionné en deux ensembles V_1 et V_2 de cardinalité respective n et m, et dont les arêtes sont (u, v), avec $u \in V_1$, $v \in V_2$.

1. Quel est le nombre d'arêtes de $K_{n,m}$?

$$|A| = \frac{1}{2}\sum_{u \in S} deg(u) = \frac{1}{2}\left(\sum_{u \in V_1} deg(u) + \sum_{u \in V_2} deg(u)\right) \quad (1.1)$$

$$= \frac{1}{2}(n * m + m * n) = n * m \quad (1.2)$$

On peut également de façon alternative compter les arêtes en dénombrant les arêtes partant des sommets de V_1 (respectivement V_2) puisque par définition, elles comportent toutes un sommet dans V_1 et un autre dans V_2.

2. $K_{n,m}$ est-il connexe ? Quel est son diamètre ? Quels sont les centres ?

Cas 1 : u et v ne sont pas dans le même ensemble (sans perte de généralité, u est dans V_1)

$$\forall u \in V_1, \forall v \in V_2, (u, v) \in A \quad (1.3)$$

Cas 2 : u et v sont dans le même ensemble V_1

$$\forall u \in V_1, \forall v \in V_1, \exists w \in V_2 | (u, w) \in A \wedge (w, v) \in A \quad (1.4)$$

Cas 3 : u et v sont dans le même ensemble V_2, le cas est similaire au précédent.
En conclusion, $K_{n,m}$ est connexe, et son diamètre est égal à son rayon (2). Tout sommet est son centre.

3. Sous quelles conditions $K_{n,m}$ est-il Eulérien ?

Un graphe comprend un circuit d'Euler ssi le nombre d'arêtes incidentes à chaque sommet est pair. Ainsi, n et m doivent être pairs.

4. Donnez une condition nécessaire (non triviale) pour que $K_{n,m}$ soit Hamiltonien

Hamiltonien : je passe par chaque sommet, et je reviens au premier (sans passer forcément par toutes les arêtes). Dans $K_{n,m}$, j'alterne entre V_1 et V_2. Donc il faut que n=m (démonstration par l'absurde de non existence pour $n \neq m$).

Exercice 4: La chèvre, le loup et le chou

Un passeur souhaite transporter sur la rivière une chèvre, un loup, un chou. La barque ne comprend que 2 places (lui compris) et il ne peut pas laisser sans surveillance sur la même berge le loup et la chèvre, ainsi que le chou et la chèvre.

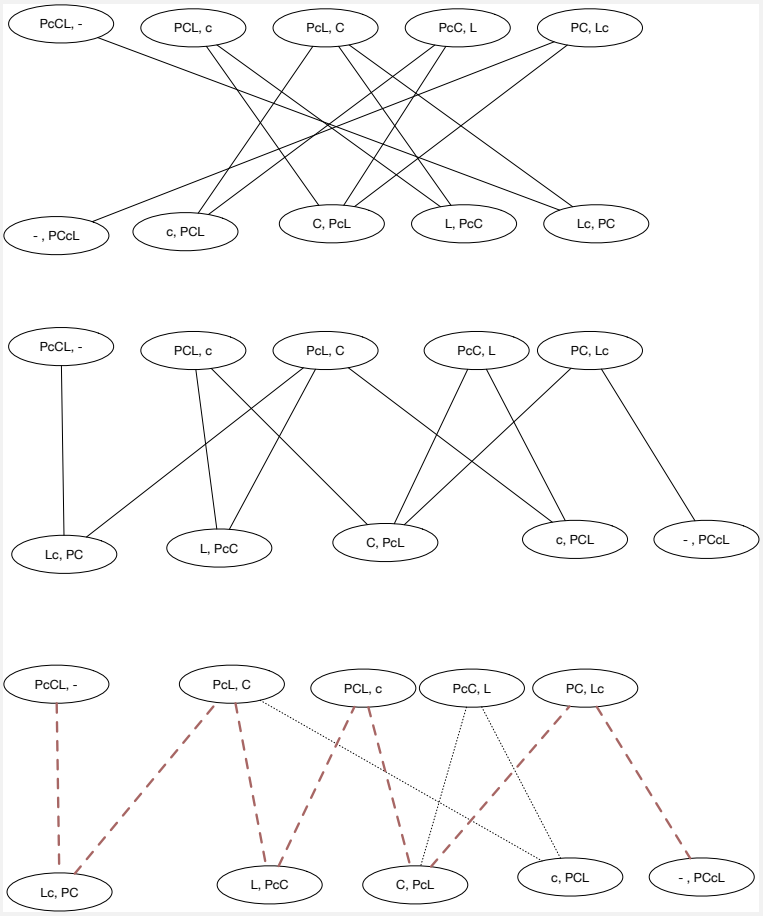
Pouvez vous l'aider ?

On peut modéliser ce problème en écrivant un graphe dont les sommets sont tous les états possibles du problème. On détaille ces états possibles en énumérant, pour chaque protagoniste, la rive sur laquelle il se trouve. On nomme rive gauche la rive sur laquelle se trouvent le passeur (P), le loup (L), la chèvre (c) et le chou (C) au début, et rive droite l'autre rive. On note A l'ensemble $\{P, L, c, C\}$.
On peut écrire un état du problème sous la forme d'un couple d'ensembles (Rg, Rd) où $Rg \in \mathcal{P}(A)$ contient les éléments présents sur la rive gauche, et $Rd \in \mathcal{P}(A)$ les éléments présents sur la rive droite du fleuve.
Les sommets du graphe sont tous les couples $(Rg, Rd) \in \mathcal{P}(A) \times \mathcal{P}(A)$ tels que tout élément de A appartienne à un et un seul des ensembles Rg et Rd (et aucun élément de A n'appartient ni à Rg ni à Rd).
On appelle sommet interdit un sommet où $P \in Rg$ (resp. Rd) et $L, c \in Rd$ (resp. Rg) ou $c, C \in Rd$ (resp. Rg).
Une arête existe entre deux sommets (Rg_1, Rd_1) et (Rg_2, Rd_2) si et seulement si

— $p \in Rg_1$ (resp. Rd_1) et $p \in Rd_2$ (resp. Rg_2),

— un au plus des éléments x de A vérifie $x \in Rg_1$ (resp. Rd_1) et $x \in Rd_2$ (resp. Rg_2).

La résolution du problème passe alors par la recherche d'un chemin passant par les arêtes autorisées entre les sommets $(A, \emptyset)$ et $(\emptyset, A)$.
Le graphe n'est pas orienté : le passeur peut parcourir la rivière dans les deux sens.



Partie 2

Graphes orientés

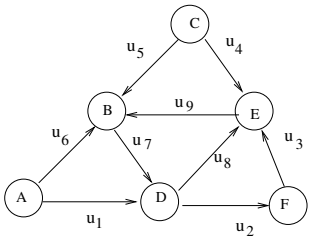
Exercice 1: Chemins et connexité

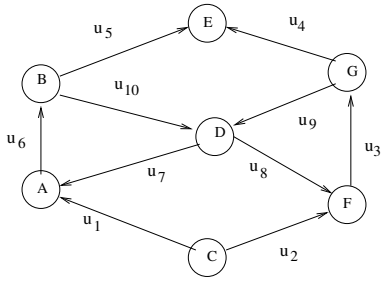
1. On considère le graphe orienté ci-après (gauche).
   (a) Donner un chemin simple de A à F passant par E.
   (b) Donner un chemin élémentaire de A à F.

simple : pas deux fois par le meme arc (u1, u8, u9, u7, u2)
élémentaire : pas deux fois par le meme sommet (u1, u2)
NB : un chemin élémentaire est également simple (pas possible de reprendre la même arête si on ne passe pas par le meme sommet).

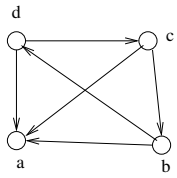
2. On considère le graphe orienté ci-après (droite).
   (a) Donner un circuit partant de A et passant par G.
   (b) Donner un circuit élémentaire à partir de A.

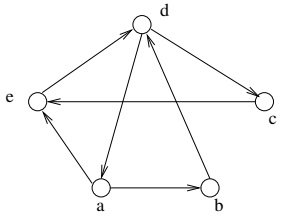
circuit : u6,u10,u8,u3,u9,u7
circuit élémentaire (pas par le meme sommet) : u6,u10,u7

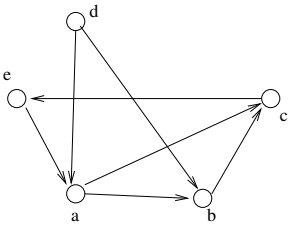




3. Donner les composantes fortement connexes des graphes orientés représentés ci-après.







Composante fortement connexe : tout sommet possède un chemin vers tout autre sommet de la composante.
graphe 1 : (b,c,d) (a)
graphe 2 : (a,b,c,d,e)
graphe 3 : (d) (a,b,c,e)

Exercice 2: Démonstration

Démontrer la proposition suivante. Soit $G = (S, A)$ un graphe orienté. La relation $\sim_G$ définie sur S par $x \sim_G y \Leftrightarrow$ « il existe un chemin de x à y dans G et un chemin de y à x dans G » est une relation d'équivalence sur S.

On montre que $\sim_G$ vérifie les trois propriétés suivantes :

- réflexivité : $x \sim_G x$ car il existe un chemin de x à x : c'est le chemin de longueur 0.
- symétrie : par définition de la relation
- transitivité : Soient x, y, z trois sommets de S tels que $x \sim_G y$ et $y \sim_G z$.
  Alors il existe un chemin p1 de x à z et un chemin p2 de z à x. Le chemin formé par la concaténation de p1 et p2 est toujours valide puisque z appartient aux 2 chemins. $\sim_G$ est donc transitive.

Partie 3

Connexité, Chemin, Coupe

Exercice 1: Labyrinthe

Un labyrinthe est composé de salles reliées entre elles par des portes. L'une des salles est l'entrée du labyrinthe, une autre est la sortie.

1. Modéliser un tel labyrinthe sous forme de graphe.
2. Donner un algorithme permettant :
   (a) de savoir s'il est possible de sortir du labyrinthe ;
   (b) si oui, de calculer un chemin de la salle d'entrée à la salle de sortie.
3. Montrer que l'algorithme proposé termine et est correct.

On modélise le problème sous forme de graphe en associant chaque salle à un sommet et chaque porte à une arête. Sortir du labyrinthe revient alors à trouver un chemin de l'entrée du labyrinthe à sa sortie. On se donne également la possibilité de marquer les sommets du graphe pour éviter de visiter plusieurs fois le même sommet.
On propose une procédure récursive pour calculer un chemin de l'entrée à la sortie.
préconditions : S liste des sommets, A liste d'arêtes, x un sommet, C pile de sommets
retour : Vrai si il existe un chemin de entrée à sortie, C contient alors ce chemin, faux sinon.
fonction :

```
Marquer x.
Si x==sortie alors
    Empiler x dans C,
    Retourner vrai.
FinSi.
Pour s non marqué appartenant à S
    Si Arete(A,x,s) alors
        Si SortieRec(S,A,s,C) alors
            Empiler x dans C,
            Retourner vrai.
        FinSi
    FinSi
FinPour
Retourner faux.
```

La fonction `Arete`(Liste A, Sommet x, Sommet s) retourne vrai si A contient une arête de x vers s, faux sinon. On appelle initialement la fonction `SortieRec` avec S la liste des sommets du graphe, A la liste des arêtes, x l'entrée du graphe, C une pile vide. Cet appel termine, car :

- à chaque appel de récursif de `SortieRec`, un sommet est marqué
- dans la boucle Pour, `SortieRec` n'est appelée que pour les sommets non encore marqués.
- le nombre de sommets est fini.

Pour montrer la correction de cet appel, il faut montrer :

1. Si `SortieRec`(S,A,entrée,C) retourne vrai avec C initialement vide, alors les sommets contenus dans C forment une chaîne de entrée à sortie.
2. Sinon, il n'existe pas de chemins de entrée à sortie dans le graphe initial.

Pour montrer 2, montrons sa contraposée. Supposons qu'il existe un chemin entrée= $s_0, s_1..., s_{n-1}, s_n$ =sortie. Comme il existe une arête de s_0 à s_1, SortieRec(A,S,s_0,C) retourne vrai si SortieRec(A,S,s_1,C) retourne vrai. De même, si $i \leq n$ SortieRec(A,S,s_{i-1},C) retourne vrai si SortieRec(A,S,s_i,C) retourne vrai. Or SortieRec(A,S,s_n,C) retourne vrai. Par récurrence, on a bien SortieRec(A,S,entrée,C) retourne vrai s'il existe un chemin de entrée à sortie.
Le point 1 se montre aussi par récurrence : soit $s_0, s_1..., s_{n-1}, s_n$ le chemin contenu dans C si SortieRec(A,S,entrée,C) renvoie vrai. Il s'agit de montrer que pour $i \leq n$, s_{i-1} est adjacent à s_i, ce qui est direct.

Exercice 2: Coûts au pire

1. Considérer l'algorithme ci-contre, calculant la fermeture transitive d'une relation d'un graphe.
   (a) Dans le cas d'un graphe non orienté $G = (S, A)$, avec $|S| = n$ et $|A| = m$, exprimer le coût au pire de cet algorithme en fonction de n et m.

La multiplication matricielle coûte $O(n^3)$ (R de dimension N, R*R+R) [a]
Le tant que est fait n fois au plus (diamètre max du graphe), et sa vérification coute O(m), un test pour chaque arête
La complexité au final est en $O(n^3)$.

a. En réalité, le meilleur algorithme connu de calcul de produit matriciel est en $O(n^\alpha logn)$. Par exemple, l'algortihme de Strassen fournit une solution avec $\alpha = 2,807$ (https://fr.wikipedia.org/wiki/Algorithme_de_Strassen). De même, l'algorithme de Coppersmith-Winograd fournit une version avec $\alpha = 2,376$ (https://fr.wikipedia.org/wiki/Algorithme_de_Coppersmith-Winograd, https://perso.univ-rennes1.fr/christophe.ritzenthaler/agregation/cours-complexite2.pdf.

   (b) Adapter cet algorithme pour résoudre la question 2(a) du problème du labyrinthe.

Vérification de connexité entre entrée et sortie : il suffit de créer un retour à l'intérieur du tant que. S'il existe une arête dans R-rond entre entrée et sortie, retourner vrai.
A la fin du programme retourner "il existe une arête entre entrée et sortie dans R-rond".

2. Calculer le coût au pire de l'algorithme proposé précédemment pour le labyrinthe.

Une arête n'est visitée qu'une seule fois (un sommet non marqué n'est pas retraité, et on teste toutes les arêtes d'un sommet).
La complexité est donc en $O(m)$.

3. Quelle différence entre ces deux procédures justifie une telle différence en coût au pire ?

La procédure de fermeture transitive ne part pas d'un sommet particulier : elle teste la transitivité en partant de **tous** les sommets. Au contraire, le labyrinthe ne cherche que les sommets joignables en partant de l'entrée : il est donc moins couteux.

Tester si une relation est **réflexive** :
fonction : estréflexive
paramètres : R matrice d'adjacence de $\mathcal{R}$
résultat : Vrai ou Faux.
entier $i = 1$;
retour$\leftarrow \neg(R(i,i) = 0)$;
tant que retour et $i \leq n-1$
$i \leftarrow i+1$;
retour$\leftarrow \neg(R(i,i) = 0)$;
fin tant que
sortir avec la valeur retour ;

Tester si une relation est **antisymétrique** :
fonction : estantisymétrique
paramètres : R matrice d'adjacence de $\mathcal{R}$
résultat : Vrai ou Faux.
entiers $i = 1$, $j = 2$;
retour$\leftarrow \neg((R(i,j) \neq 0) \wedge (R(j,i) \neq 0))$;
tant que retour et $i \leq n-1$
tant que retour et $j \leq n-1$
$j \leftarrow j+1$;
retour$\leftarrow \neg((R(i,j) \neq 0) \wedge (R(j,i) \neq 0))$;
fin tant que
$i \leftarrow i+1$;
$j \leftarrow i+1$;
fin tant que
sortir avec la valeur de retour ;

Tester si une relation est **symétrique** :
fonction : estsymétrique
paramètres : R matrice d'adjacence de $\mathcal{R}$
résultat : Vrai ou Faux.
matrice $R' =$transposée(R) ;
sortir avec la valeur de $R - R' = 0_{\mathcal{M}_{n,n}(\mathbb{N})}$;

Tester si une relation est une **relation d'ordre** :
fonction : estordre
paramètres : R matrice d'adjacence de $\mathcal{R}$
résultat : Vrai ou Faux.
sortir avec la valeur estréflexive$(R) \wedge$estantisymétrique$(R) \wedge$esttransitive(R) ;

Calculer la **fermeture transitive** d'une relation :
fonction : fermeture
paramètres : R matrice d'adjacence de $\mathcal{R}$
résultat : $\mathring{R}$ matrice d'adjacence de $\mathring{\mathcal{R}}$
matrices $R_{temp} \leftarrow R$, $\mathring{R} \leftarrow R * R + R$;
tant que $\neg(\mathring{R} \subseteq R_{temp})$
$R_{temp} \leftarrow \mathring{R}$;
$\mathring{R} \leftarrow R_{temp} * R_{temp} + R_{temp}$;
fin tant que
sortir avec la valeur de $\mathring{R}$;

Tester si une relation est **transitive** :
fonction : esttransitive
paramètres : R matrice d'adjacence de $\mathcal{R}$
résultat : Vrai ou Faux.
sortir avec la valeur $(R \subseteq$fermeture$(R))$;

Remarque : Dans la fonction fermeture, l'instruction $(\mathring{R} \subseteq R_{temp})$ signifie $\mathring{R}(i,j) \neq 0 \Rightarrow R_{temp}(i,j) \neq 0$. Mais grâce à l'instruction $\mathring{R} \leftarrow R_{temp} * R_{temp} + R_{temp}$, c'est ici équivalent à : $\forall (i,j) \in E^2, (R_{temp}(i,j) = 0) \Leftrightarrow (\mathring{R}(i,j) = 0)$. Il en va de même dans la fonction esttransitive.