Cheminement dans les Graphes

1. Chaînes / Chemins

1.1 Chaîne

On appelle chaîne dans un graphe non orienté (resp. orienté) $G=(X, E)$ (resp. $G=(X, U)$), une suite alternée de sommets et d'arêtes (resp. d'arcs) :

$\mu = x_0\, e_1\, x_1 \ldots x_{k-1}\, e_k\, x_k$ (resp. $\mu = x_0\, u_1\, x_1 \ldots x_{k-1}\, u_k\, x_k$)

Tel que pour i de 1 à k, x_{i-1} et x_i sont extrémités de l'arête e_i (resp. de l'arc u_i).

On dit que μ est une chaîne joignant les sommets x_0 et x_k de longueur k.

1.2 Chemin

On appelle chemin dans un graphe orienté $G=(X, U)$, une suite alternée de sommets et d'arcs :

$\gamma = x_0\, u_1\, x_1 \ldots x_{k-1}\, u_k\, x_k$

Tel que pour i de 1 à k,

le sommet x_{i-1} est extrémité initiale de l'arc u_i et le sommet x_i est son extrémité terminale.

On dit que γ est un chemin de x_0 vers x_k de longueur k.

1.3 Chaîne / Chemin simple

On dit qu'un chemin ou une chaîne est simple si tous les arcs ou les arêtes les composant sont distincts.

1.4 Chaîne / Chemin élémentaire

On dit qu'un chemin ou une chaîne est élémentaire si tous les sommets les composant sont distincts.

1.5 Remarques

- La longueur d'une chaîne (chemin) est égale au nombre d'arêtes (arcs) formant cette chaîne (chemin).
- S'il existe dans un graphe un chemin d'un sommet x vers un sommet y, on notera : $x\ \alpha\ y$.
- Toute chaîne (ou chemin) élémentaire est aussi simple. L'inverse n'est pas toujours vrai.
- Dans un graphe simple, une chaîne ou un chemin peuvent être déterminés juste en énumérant la suite des sommets qui les composent.

1.6 Proposition

Soit $G=(X, E)$ un graphe non orienté. De toute chaîne joignant deux sommets x et $y \in X$, on peut extraire une chaîne élémentaire joignant x et y.

Soit $G=(X, U)$ un graphe orienté. De tout chemin allant du sommet $x \in X$ vers le sommet $y \in X$, on peut extraire un chemin élémentaire allant de x à y.

2. Cycles / Circuits

2.1 Chaîne fermée / Chemin fermé

Un chemin (resp. chaîne) dont les extrémités sont confondues est dit chemin fermé (resp. chaîne fermée).

2.2 Cycle

On appelle cycle dans un graphe non orienté (resp. orienté) $G=(X, E)$ (resp. $G=(X, U)$), toute chaîne fermée simple :

$\mu = x_0\, e_1\, x_1 \ldots x_{k-1}\, e_k\, x_k$ (resp. $\mu = x_0\, u_1\, x_1 \ldots x_{k-1}\, u_k\, x_k$) Tel que $k > 0$, et $x_0 = x_k$.

On dit que μ est un cycle de longueur k.

2.3 Circuit

On appelle circuit dans un graphe orienté $G=(X, U)$, tout chemin fermé simple :

$\gamma = x_0\, u_1\, x_1 \ldots x_{k-1}\, u_k\, x_k$ Tel que $k > 0$, et $x_0 = x_k$.

On dit que μ est un circuit de longueur k.

2.4 Cycle / Circuit élémentaire

On dit qu'un cycle ou circuit est élémentaire si tous les sommets qui les composent sont distincts.

2.5 Remarques

- La longueur d'un cycle ou d'un circuit élémentaire est aussi égale au nombre de sommets formant ce cycle ou circuit.
- Dans un graphe simple, un cycle ou un circuit peuvent être déterminés juste en énumérant la suite des sommets qui les composent.
- Une boucle est un cycle élémentaire de longueur 1.
- Une boucle dans un graphe orienté est un circuit élémentaire de longueur 1.
- Tout cycle est aussi chaîne. Tout circuit est aussi chemin. Tout circuit est aussi cycle. Tout chemin est aussi chaîne.
- Dans le cas d'une k-coloration, tout graphe contenant un cycle de longueur impaire nécessite au minimum 3 couleurs.

2.6 Proposition

Soit $G=(X, E)$ un graphe non orienté (resp. $G=(X, U)$ un graphe orienté). De tout cycle (resp. tout circuit) passant par une arête $e \in E$ (resp. un arc $u \in U$), on peut extraire un cycle (resp. un circuit) élémentaire passant par e (resp. u).

2.7 Existence d'un cycle

Si G est un graphe vérifiant $\delta(G) \geq k \geq 2$ Alors G contient un cycle.

Si de plus G est simple alors G admet un cycle élémentaire de longueur $\geq k+1$ et une chaine élémentaire de longueur $\geq k$.

Conséquence : Si $m \geq n$ (m étant la taille de G et n le nombre de sommets dans G) alors G admet un cycle.

2.8 Existence d'un circuit

Si G est un graphe vérifiant $\delta^+(G) \geq k \geq 1$ (resp. $\delta^-(G) \geq k \geq 1$) Alors G contient un circuit.

Si de plus G est simple alors G admet un circuit élémentaire de longueur $\geq k+1$ et un chemin élémentaire de longueur $\geq k$.

3. Matrice de fermeture transitive

Soit $G=(X, U)$ un 1-graphe orienté d'ordre n. A partir de sa matrice d'adjacence M, on peut calculer la matrice de fermeture transitive de G qu'on note $\hat{M}$ où chaque élément $\hat{m}_{ij} = \begin{cases} 1 \ si \ \exists i \alpha j \\ 0 \ sinon \end{cases}$

3.1 Calcul Matriciel Direct

On peut avoir la matrice de fermeture transitive par calcul matriciel comme suit : $\hat{M} = \bigvee_{l=1}^{n} M^{[l]}$

Où chaque matrice $M^{[l]}$ se calcule par récurrence (sur l) à travers le produit matriciel booléen comme suit :

$$\begin{cases} M^{[1]} = M \\ M^{[l+1]} = M^{[l]} * M \end{cases}$$

Chaque élément de $M^{[l]}$: $m_{ij}^{[l]} = \vee (\underset{k=1}{\overset{n}{m}}_{ik}^{[l-1]} \wedge m_{kj})$ où l varie de 2 à n.

La matrice $M^{[l]}$ représente tous les chemins dans G de longueur l.

3.2 Algorithme de Warshall

Le calcul direct de la matrice transitive $\hat{M}$ nécessite trop d'opérations matricielles. L'algorithme de Warshall, donné ci-dessous, permet de la calculer avec un gain considérable en nombre d'opérations : n^2 tests et au plus n^3 opérations $\vee$, c'est donc un algorithme en $O(n^3)$

```
Algorithme Warshall (Entrée : M : « Matrice »; Sortie : M : « Matrice »)
   Début
     Pour j de 1 à n Faire
         Pour i de 1 à n Faire
             Si M[i,j] = 1 Alors
                Pour k de 1 à n Faire M[i,k]  = M[i,k] ∨ M[j,k]  Fait;
             Fsi;
         Fait;
     Fait;
   Fin.
```

4. Exploration (Parcours) d'un graphe

L'exploration d'un graphe est un parcours (via les arcs ou les arêtes) permettant d'examiner de façon exhaustive (visiter) les sommets. L'exploration d'un graphe permet d'étudier une ou plusieurs propriétés du graphe tel que la connexité, la forte connexité, biparti, …

4.1 Algorithme

Principe :

- Il consiste à déterminer l'ordre dans lequel seront visités les sommets. Ainsi, le parcours commence d'un sommet de départ *r* qu'on appelle *racine* et donne comme résultat une liste ordonnée de sommets où *r* apparaît en premier et les autres sommets apparaissent une seule fois.

Données :

- En entrée :
  - `X` : ensemble de *n* éléments représentant les identificateurs des sommets.
  - `U` : ensemble de *m* éléments sous forme (`i`, `j`) représentant les arcs où $i \in X$ et $j \in X$.
  - `r` : identificateur du sommet de départ (la racine).
- En sortie :
  - `P` : Liste ordonnée de sommets.
- Autres :
  - `L` : Liste ordonnée de sommets qui peut être implémentée en tant que pile ou file (liste FIFO).

L'algorithme :

```
P ← ∅ ;     L ← {r} ;
Tant que ((L ≠ ∅) et (P ≠ X))
      Choisir_extraire (i∈L) ;
      Pour (tout (i, j)∈U)   Si (j∉P) Alors Ajouter j à L ;
      Pour (tout (j, i)∈U)   Si (j∉P) Alors Ajouter j à L ;
      Ajouter i à la fin de P ;
```

Remarque :

- Nous supposons que la fonction `choisir_extraire` existe et consiste à choisir un sommet de *L* de façon déterministe puis le supprime de *L*. Nous expliquons ci-dessous ses différentes implémentations.
- Il est possible d'appliquer cet algorithme sur un graphe non-orienté.

4.2 Exploration en largeur

Cette exploration consiste à parcourir le graphe à partir du sommet de départ la racine (*r*) puis ses voisins puis les voisins des voisins non explorés et ainsi de suite jusqu'à la fin.

Nous pouvons utiliser l'algorithme d'exploration donné ci-dessus en déclarant `L` comme une liste FIFO (premier arrivé, premier sorti) et la fonction `choisir_extraire` devient `defiler` et la fonction `Ajouter` devient `enfiler`.

4.3 Exploration en profondeur

Cette exploration consiste à parcourir le graphe à partir du sommet de départ la racine (*r*) puis tracer une chaîne à partir de ce sommet puis choisir un autre sommet (parmi les voisins des sommets de cette chaîne dans l'ordre) et faire de même jusqu'à la fin.

Nous pouvons utiliser l'algorithme d'exploration donné ci-dessus en déclarant `L` comme une pile (dernier arrivé, premier sorti) et la fonction `choisir_extraire` devient `depiler` et la fonction `Ajouter` devient `empiler`.

5. Connexité

5.1 Définition

Un graphe est dit connexe s'il existe une chaîne joignant chaque paire de sommets *x* et *y* ($x \neq y$).

5.2 Composante connexe

Soit un graphe $G = (X, E)$ (resp. $G = (X, U)$):

- Le sous graphe engendré par un sommet isolé est considéré comme une composante connexe de *G*.
- Si tout sous graphe engendré par un ensemble de sommets $S \subseteq X$ (G_S) est connexe et le sous graphe engendré par $S \cup \{x\}$ et $x \notin S$ n'est pas connexe Alors G_S est une composante connexe de *G*.

Remarque : Un graphe connexe contient une seule composante connexe.

5.3 Algorithme de Connexité

Principe :

- A partir d'un sommet *r*, on tentera d'atteindre tous les autres sommets du graphe en faisant un prolongement d'une chaîne.
- Pour chaque sommet nous allons vérifier les arcs sortants et les arcs entrants.

Données :

- En entrée :
  - `X` : ensemble de *n* éléments représentant les identificateurs des sommets.
  - `U` : ensemble de *m* éléments sous forme (`i`, `j`) représentant les arcs où $i \in X$ et $j \in X$.
  - `r` : identificateur d'un sommet de *X* qu'on appelle racine.
- En sortie :
  - `Connexe` : Booléen, sera à `Vrai` si le graphe est connexe.
- Autres :
  - `CC` : sous-ensemble de *X*. Le sous graphe engendré par cet ensemble représente une Composante Connexe.
  - `Marque` : vecteur de *n* éléments booléens pour marquer les sommets traités.

L'algorithme :

```
C ← {r} ;
Pour (tout i ∈ X)        Marque[i] ← faux ;
Tant que (∃ i∈C tel que Non(Marque[i]))
      Pour (tout (i, j)∈U)   C ← C ∪ {j} ;
      Pour (tout (j, i)∈U)   C ← C ∪ {j} ;
      Marque[i] ← vrai ;
Si C=X Alors Connexe ← Vrai ;
       Sinon Connexe ← Faux ;
```

Remarque :

Nous pouvons aussi utiliser l'algorithme d'exploration afin de vérifier la connexité. Il s'agit juste de vérifier si la sortie `P = X`.

5.4 Graphe *k*-connexe

Un graphe *G* est dit *k*-connexe si et seulement si *G* est connexe d'ordre $n \geq k+1$ et il n'existe pas d'ensemble $S \subset X$ de cardinal *k*-1 tel que le sous graphe engendré par $X-S$ n'est pas connexe.

En d'autres termes, en supprimant moins de *k* sommets, le graphe sera toujours connexe.

5.5 Point d'articulation

Dans un graphe *G* qui est 1-connexe, un point d'articulation est tout sommet $x \in X$, tel que le sous graphe de *G* engendré par le sous-ensemble de sommets $X-\{x\}$ n'est non connexe.

En d'autres termes, en supprimant moins de *k* sommets, le graphe sera toujours connexe.

6. Forte Connexité

6.1 Définition

Un graphe orienté $G=(X, U)$ est fortement connexe (f.c.) s'il existe entre chaque paire de sommets *x* et $y \in X$ ($x \neq y$) un chemin de *x* à *y* ($x \alpha y$) et un chemin de *y* à *x* ($y \alpha x$).

6.2 Composantes fortement connexes (C.F.C.)

Soit $G=(X, U)$ un graphe orienté :

- Le sous graphe engendré par un sommet $x \in X$ tel que $d_G^+(x) = 0$ ou $d_G^-(x) = 0$ forme une composante fortement connexe de *G*.
- Si le sous graphe engendré par un ensemble de sommets $S \subseteq X$ (G_S) est fortement connexe et le sous graphe engendré par $S \cup \{x\}$ et $x \notin S$ n'est pas fortement connexe Alors G_S est une composante fortement connexe de *G*.

6.3 Ascendants / Descendants

Soit $G=(X, U)$ un graphe orienté, On définit pour chaque sommet $x \in X$ deux ensembles :

- L'ensemble des descendants de x : $D(x) = \{ y \in X / x \alpha y \}$
- L'ensemble des ascendants de x : $A(x) = \{ y \in X / y \alpha x \}$

6.4 Algorithme de Calcul de C.F.C.

Principe :

- A partir d'un sommet *r*, on calcule la c.f.c. à laquelle appartient *r*.
- On calcule l'ensemble des descendants de *r* (noté *D*) puis l'ensemble des ascendants de *r* (noté *A*).
- La c.f.c. est $A \cap D \cup \{r\}$

Données :

- En entrée :
  - `X` : ensemble de *n* éléments représentant les identificateurs des sommets.
  - `U` : ensemble de *m* éléments sous forme (`i`, `j`) représentant les arcs où $i \in X$ et $j \in X$.
  - `r` : identificateur d'un sommet de *X* qu'on appelle racine.
- En sortie :
  - `CFC` : sous-ensemble de *X*. Le sous graphe engendré par cet ensemble représente la C.F.C.
- Autres :
  - `Marque` : vecteur de *n* éléments booléens.
  - `A, D`: sous-ensembles de *X*. Ils représentant l'ensemble des ascendants et l'ensemble des descendants de la racine.

L'algorithme :

```
(* Initialisation *)
   D ← {r} ;   Pour (tout i ∈ X)    Marque[i] ← faux ;
(* Recherche des descendants de r *)
   Tant que ((∃ i ∈ D) et (Marque[i]=faux) )
         Marque[i] ← vrai ;
         Pour (tout (i, j) ∈ U )  D ← D ∪ {j} ;
(* Réinitialisation *)
   A ← {r} ;          Pour (tout i ∈ X)      Marque[i] ← faux ;
(* Recherche des ascendants de r *)
   Tant que ((∃ i ∈ A) et (Marque[i]=faux))
         Marque[i] ← vrai ;
         Pour (tout (j, i) ∈ U)   A ← A ∪ {j}
(* Calcul de la C.F.C. *)
   CFC  ← D ∩ A
```

Remarque : Il est possible d'utiliser cet algorithme comme procédure et l'appeler à répétition jusqu'à obtention de toutes les CFC d'un graphe. Il faut aussi savoir qu'il existe d'autres algorithmes plus simplifiés.

6.5 Graphe réduit

A tout graphe orienté $G=(X, U)$ on associe le graphe simple $G_R=(X_R, U_R)$ appelé graphe réduit de *G* défini comme suit :

X_R = { A chaque c.f.c. de *G* correspond un sommet C_i }

$U_R = \{ (C_i, C_j) / i \neq j$ et $\exists x \in C_i$ et $\exists y \in C_j$ et $(x, y) \in U \}$

6.6 Remarques

- Un graphe fortement connexe possède une seule C.F.C.
- Le graphe réduit d'un graphe ne possède pas de circuits.

7. Parcours Eulériens

7.1 Définition

Un parcours Eulerien passe une fois et une seule fois par chaque arête (resp. arc) du graphe. Le parcours peut être une chaîne, un chemin, un cycle ou un circuit. Soit *G* un graphe contenant *m* arêtes (resp. *m* arcs) :

Une chaîne simple, un chemin simple, un cycle simple ou un circuit de longueur *m* est appelé Eulérien.

7.2 Théorème 1 – Euler 1766

Un multigraphe G admet une chaîne Eulérienne si et seulement si il est connexe (à des points isolés près) et le nombre de sommets de degré impair est 0 ou 2.

7.3 Conséquences

Un graphe G admet une chaîne Eulérienne d'un sommet x à un sommet y ($x \neq y$) si et seulement si $d_G(x)$ et $d_G(y)$ sont impairs et $\forall z$ sommet de G ($z \neq x$ et $z \neq y$), on a $d_G(z)$ pair.

Un graphe G admet un cycle Eulérien si et seulement $\forall x$ sommet de G, on a $d_G(x)$ pair.

7.4 Détermination d'une chaîne Eulérienne

- On part d'un sommet a de degré impair (S'il n'y pas de sommets de degrés impairs, on démarre de n'importe quel sommet). On construit une chaîne à partir de a comme suit :
- A chaque étape k on obtient une chaîne de longueur k et le graphe sera réduit à G_k qui correspond au graphe partiel de G engendré par l'ensemble des arêtes (resp. d'arcs) initial auquel on supprime ceux faisant partie de la chaîne.
- A chaque étape k, en arrivant à un sommet x, il faut éviter de prendre toute arête (resp. arc) qui est isthme dans G_k. Sauf s'il s'agit de la seule et unique possibilité, on la prend et on termine.
- On termine lorsqu'il n'y a plus d'arêtes (resp. arcs) dans le graphe G_k.

7.5 Proposition

Un graphe $G=(X, U)$ admet un circuit Eulérien si et seulement si pour tout sommet x, on a $d^+_G(x) = d^-_G(x)$. (On dit que G est pseudo-symétrique)

8. Parcours Hamiltoniens

8.1 Définition

Un parcours Hamiltonien passe une fois et une seule fois par chaque sommet du graphe. Le parcours peut être une chaîne, un chemin, un cycle ou un circuit. Soit G un graphe d'ordre n (contenant n sommets) :

Une chaîne (resp. un chemin) élémentaire de longueur n-1 est appelé chaîne Hamiltonienne (resp. chemin Hamiltonien).

Un cycle (resp. circuit) élémentaire de longueur n est appelé cycle (resp. circuit) Hamiltonien.

8.2 Remarque

- Un graphe qui contient un cycle Hamiltonien est appelé graphe Hamiltonien.
- Un graphe semi-Hamiltonien est un graphe qui contient une chaîne Hamiltonienne, mais pas de cycle Hamiltonien.
- Le plus petit graphe Hamiltonien d'ordre n est le graphe cycle (Graphe connexe non-orienté à n arêtes. Il est 2-régulier)

8.3 Propositions

- Un graphe complet d'ordre $n \geq 3$ est Hamiltonien.
- Tout graphe tournoi (un graphe orienté simple et complet) d'ordre n, noté T_n contient un chemin Hamiltonien.
- Tout tournoi d'ordre n (T_n) fortement connexe contient un circuit Hamiltonien.

8.4 Théorème

Le théorème ci-dessous est utilisé pour démontrer qu'un graphe n'est pas Hamiltonien (ne contient pas de cycle Hamiltonien).

Si $G=(X,E)$ est un graphe Hamiltonien, alors pour tout ensemble de sommets $S \subset X$, on a : $p(G_{X-S}) \leq |S|$

où $p(G_{X-S})$ est le nombre de composantes connexes du sous graphe de G induit par l'ensemble $X-S$