THEORIE DES GRAPHES - NOTIONS FONDAMENTALES

Amina ELJABRI

FSTM
Département Informatique

PLAN

Définitions

Définition 1

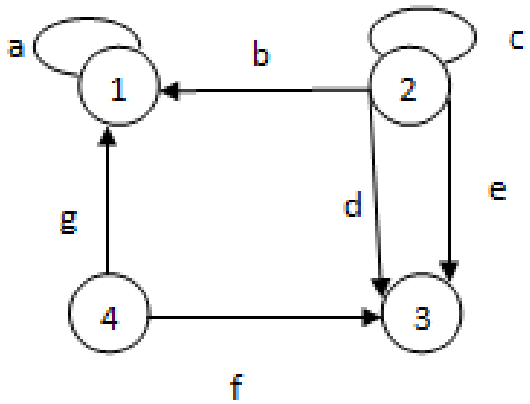
Un graphe est défini par :

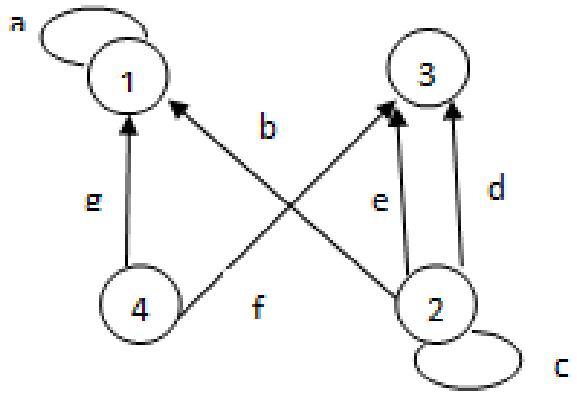
- Deux ensembles X et U qui sont respectivement : ensemble de sommets (ou nœuds) et ensemble d'arcs.
- Deux applications $I : U \to X$ et $T : U \to X$ telles que :$I(u)$ et $T(u)$) sont respectivement l'extrémité initiale et terminale d'un arc u de U.

Exemple : $X = \{1, 2, 3, 4\}$; $U = \{a, b, c, d, e, f, g\}$
I(a)=1; I(b)=2; I(c)=2; I(d)=2; I(e)=2; I(f)=4; I(g)=4;
T(a)=1; T(b)=1; T(c)=2; T(d)=3; T(e)=3; T(f)=3; T(g)=1;

Remarques

- On associe à un graphe une représentation graphique telle que les sommets sont des points (ou cercles) et les arcs des morceaux de courbes munis d'une orientation pour différencier l'extrémité initiale de l'extrémité terminale.
- Un graphe peut avoir plusieurs représentations graphiques. Une représentation graphique d'un graphe G le définit parfaitement.





Définitions

Définition 2

Un graphe est dit simple s'il ne possède pas deux fois le même arc. Il peut être défini par : Un ensemble de sommets X et une partie U de X^2.

- Un graphe général peut être considéré comme un graphe simple évalué : (X, U, m) où m est une application qui associe à chaque arc sa multiplicité (ie le nombre d'occurrences de l'arc dans le graphe).
- Il existe une bijection entre l'ensemble des relations binaires sur X et l'ensemble des graphes simples sur X. Si R est la relation binaire associée à G = (X, U) alors on a :
  $x\ R\ y \Leftrightarrow (x\ y) \in U$

Définitions

Nous désignerons par la suite un graphe G par G = (X, U) et un arc u par (x y) où x = I(u) et y = T(u).

Définition 3

Soit u = (x y) un arc de G. On dit que :

- x et y sont deux sommets adjacents.
- u est adjacent à x et y.
- x et y sont adjacents à u (on dit aussi incidents).
- u est une boucle si x et y sont confondus.
- Deux arcs sont adjacents s'ils ont une extrémité commune.

Définitions

Définition 4

A tout sommet x de G on associe :

- $\Gamma^+(x) = \{y \in X/(x\ y) \in U\}$ désigne l'ensemble des successeurs de x.
- $\Gamma^-(x) = \{y \in X/(y\ x) \in U\}$ désigne l'ensemble des prédécesseurs de x.
- $d^+(x) = |\{u \in U/I(u) = x\}|$=degré extérieur de x.
- $d^-(x) = |\{u \in U/T(u) = x\}|$=degré intérieur de x.
- dg(x) = $d^+(x) + d^-(x)$ = degré de x.

théorèmes

Theorem

Soit $G = (X , U)$ un graphe. On a :

$$\sum_{x \in X} d^+(x) = \sum_{x \in X} d^-(x)$$

corollaire 1

La somme des degrés des sommets d'un graphe G est égale à 2 fois le nombre d'arcs de G.

corollaire 2

Le nombre de sommets de degré impair est pair

Définitions

Définition 5

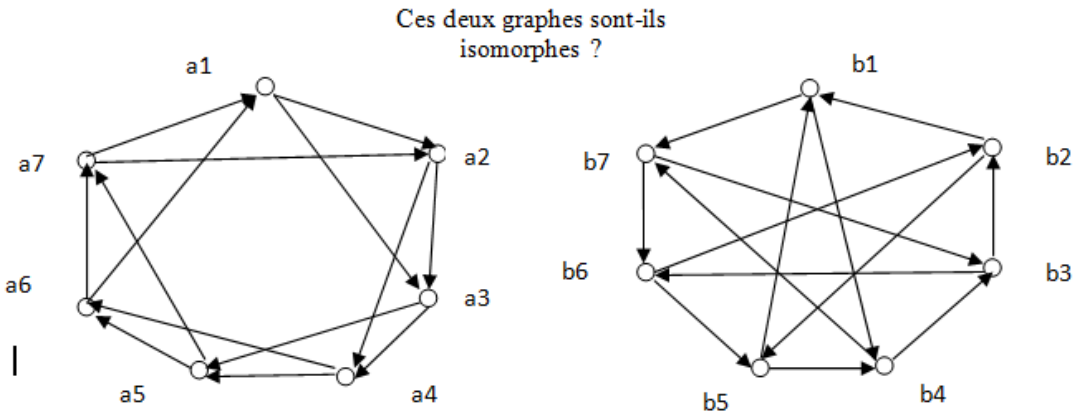
Un graphe simple G = (X, U) est dit :

- Symétrique si : $(x\ y) \in U \Longrightarrow (y\ x) \in U$
- Antisymétrique si : $(x\ y) \in U \Longrightarrow (y\ x) \notin U$
- Complet si : $(x\ y) \notin U \Longrightarrow (y\ x) \in U$
- Transitif si : $(x\ y) \in U\ et\ (y\ z) \in U \Longrightarrow (x\ z) \in U$

Définitions

Définition 6

Deux graphes G1 = (X1, U1) et G2 = (X2, U2) sont dits isomorphes s'il existe deux bijections :
$Bx : X1 \to X2$ et $Bu : U1 \to U2$ telles que :
$\forall u = (x_1\ y_1) \in U1, \forall v = (x_2\ y_2) \in U2$ *on a* :
$v = Bu(u) \Leftrightarrow x_2 = Bx(x_1)$ *et* $y_2 = Bx(y_1)$



Chemin

Soient G = (X, U) un graphe et x, y deux sommets de G. un chemin de x à y est une séquence d'arcs telle que :

- L'extrémité initiale du premier arc de la séquence est x
- L'extrémité initiale de chaque arc de la séquence coïncide avec l'extrémité terminale de l'arc qui le précède.
- L'extrémité terminale du dernier arc de la séquence est y.
- Un chemin est dit simple si la séquence d'arcs qui le compose ne comporte pas plusieurs fois le même arc.
- Un chemin est dit élémentaire si chaque sommet du graphe est adjacent à au plus deux arcs du chemin.
- Remarque : un chemin élémentaire est simple mais l'inverse n'est pas nécessairement vrai.

Chemin et circuit

On appelle circuit une séquence circulaire d'arcs tous distincts tels que:

- chaque arc de la séquence est adjacent à l'arc qui le précède par son extrémité initiale et à l'arc qui le suit par son extrémité terminale.
- Un circuit est élémentaire si tout sommet du graphe est adjacent à au plus deux arcs de la séquence.

Theorem

Un chemin de x à y dans un graphe G contient un chemin élémentaire de x à y.
Tout circuit est union disjointe de circuits élémentaires.

Chaine

Soit G = (X, U) un graphe et x, y deux sommets de G. une chaine joignant x et y dans G est une séquence d'arcs telle que:

- le premier arc de la séquence est adjacent à x par l'une de ses extrémités et au deuxième arc par son autre extrémité.
- Le dernier arc de la séquence est adjacent à y par l'une de ses extrémités et à l'avant dernier arc par son autre extrémité.
- Chaque arc intermédiaire de la séquence est adjacent au précédent par l'une de ses extrémités et au suivant par son autre extrémité.
- Une chaine est simple si elle ne comporte pas plusieurs fois le même arc.
- Une chaine est élémentaire si tout sommet du graphe est adjacent à au plus deux arcs de la chaine.

Chaine et Cycle

- Un cycle est une séquence circulaire d'arcs tous distincts tel que chaque arc est adjacent au précédent par l'une de ses extrémités et au suivant par l'autre extrémité.
- Un cycle est dit élémentaire si tout sommet du graphe est adjacent à au plus deux arcs du cycle.

Theorem

Toute chaine joignant deux sommets a et b dans un graphe G contient une chaine élémentaire.
Tout cycle est union disjointe de cycles élémentaires.

Graphe partiel - sous graphe

Soient G = (X, U) un graphe, $X' \subset X$ et $U' \subset U$.
et soit $U1 = \{u \in U / I(u) \in X' et T(u) \in X'\}$ et $U2 = U1 \cap U'$.

- GX' = (X', U1) est appelé sous graphe construit sur X' .
- G'=(X,U') est appelé graphe partiel de G construit sur U'.
- G'' = (X', U2) est appelé sous graphe partiel construit sur X' et U'.

Définition

On note K_n le graphe complet à n sommets et S_n le graphe sans arêtes à n sommets.
Un sous graphe de G isomorphe à K_p est appelé clique
Un sous graphe isomorphe à S_p est appelé stable.

soit un graphe G (X , U), et F un sous-ensemble de sommets. On dit que F est un sous ensemble stable de X s'il n'existe aucun arc du graphe reliant deux sommets de F.

Exercice

Considérons le puzzle mathématique suivant : on dispose de trois récipients qui peuvent contenir 8, 5, 3 litres. On suppose qu'au début le récipient de 8l est plein et que les deux autres sont vides. On suppose que les récipients ne sont pas gradués et on ne peut pas évaluer une fraction du contenu d'un récipient.
Une opération de transvasement aura pour effet de vider complètement un des récipients et/ou d'en remplir un autre à ras bord.
? On voudrait isoler dans chacun des deux premiers récipients 4l par une série de transvasements. Utiliser les graphes pour trouver la solution.

Représentation des graphes

Soit le graphe G = (X , U). On suppose que les sommets de X sont numérotés de 1 à n, avec n = $|X|$.
Il existe essentiellement deux façons classiques de représentation d'un graphe en machine :

- par une matrice d'adjacence
- ou par un ensemble de listes d'adjacence.

Représentation par matrice d'adjacence

La représentation par matrice d'adjacence de G consiste à définir une matrice A carrée d'ordre n booléenne telle que :

$$a_{ij} = \begin{cases} 1 \ si \ (i \ j) \in U \\ 0 \ sinon \end{cases}$$

Si le graphe est pondéré c. à. d. si tout arc $u \in U$ est doté d'un poids c(u), on peut définir la matrice A par :

$$a_{ij} = \begin{cases} c((i \ j)) \ si \ (i \ j) \in U \\ 0 \ ou \ \infty \ sinon \end{cases}$$

Représentation par matrice d'adjacence

- La matrice d'adjacence d'un graphe à n sommets nécessite de l'ordre de $O(n^2)$ espaces mémoires. Si le nombre d'arcs est très inférieur à n^2, une telle représentation n'est pas optimale.
- Le test d'existence d'un arc ou d'une arête avec une telle représentation est efficace.
- Le calcul du degré d'un sommet nécessite le parcours de toute la ligne correspondant au sommet dans la matrice, quelle que soit la valeur de ce degré.
- Le parcours de l'ensemble des arcs du graphe prendra un temps de l'ordre de n^2.
- Cette représentation n'est pas optimale dans la majorité des cas.

Représentation par matrice d'incidence sommet-arc

Soit le graphe G = (X , U). On suppose que les sommets de X sont numérotés de 1 à n, avec $n =\mid X \mid$ et que les arcs sont numérotés de 1 à m, avec $m =\mid U \mid$. La représentation par matrice d'incidence sommets-arcs consiste à définir une matrice A à n lignes et m colonnes de coefficient général.

$$a_{ik} = \begin{cases} 1 \ si \ I(u_k) = x_i \\ -1 \ siT(u_k) = x_i \\ 0 \ sinon \end{cases}$$

Une telle représentation n'est pas optimale pour les mêmes raisons que pour la représentation par matrice d'adjacence. (l'espace mémoire utilisé est de l'ordre de mxn). La somme de chaque colonne est égale à 0 (un arc a une origine et une destination) ; la matrice est totalement unimodulaire, i.e. Toutes les sous-matrices carrées extraites de A ont pour déterminant 1, -1 ou 0.

Représentation par listes d'adjacence

La représentation par listes d'adjacence de G consiste à définir un tableau T de n listes, une pour chaque sommet de X.
Pour chaque sommet $x_i \in X$, la liste T[i] est une liste chainée de tous les sommets x_j successeur de x_i.
Si le graphe est pondéré, un élément de la liste T[i] contiendra, en plus du numéro du sommet x_j, le poids de l'arc $(x_i\ x_j)$.
Dans le cas des graphes non orientés, pour chaque arête $(x_i\ x_j)$, x_j figurera dans la liste chainée T[i] et x_i figurera dans la liste chainée T[j].

Représentation par listes d'adjacence

- Les listes d'adjacence d'un graphe à n sommets et m arcs ou arêtes nécessite de l'ordre de $O(n+m)$ cases mémoires.
- Le test de l'existence d'un arc ou d'une arête $(x_i\ x_j)$ est moins efficace(ce test nécessite de parcourir la liste chainée T[i]).
- Le calcul du degré d'un sommet est plus efficace: il suffit de parcourir la liste d'adjacence associée au sommet.
- Le parcours de l'ensemble des arcs nécessite le parcours de toutes les listes d'adjacence et prendra un temps de l'ordre de m (au lieu de n^2).
- Le calcul des prédécesseurs nécessite le parcours de toutes les listes d'adjacence de T.?Solution: ?

Exploration d'un graphe

Quels sont les points que l'on peut atteindre depuis un sommet de départ a ?
Données : G = (X,U), a un sommet particulier ;
Sorties : les sommets atteignables à partir de a marqués.

Algorithme Général

Liste $= \{a\}$
etat[a] = marqué ;
Répéter
x =choix(Liste) ;
$Liste = Liste \setminus \{x\}$;
$\forall\ y \in \Gamma^{+}(x)$
si (etat[y] $\neq$ marqué)
etat[y] = marqué ;
$Liste = Liste \cup \{y\}$;
Tant que $Liste \neq \emptyset$
L'ensemble Liste peut être géré en pile ou en queue ; dans le premier cas, on a une exploration en profondeur (DFS : Depth first search); dans le deuxième cas, on a une exploration en largeur (BFS : Breath first search).

Application

Liste $= \{a\}$
etat[a] = marqué ;
num = 0; Répéter
x =choix(Liste) ;
numero[x] = num; num = num + 1; $Liste = Liste \setminus \{x\}$;
$\forall\ y \in \Gamma^+(x)$
si (etat[y] $\neq$ marqué)
etat[y] = marqué ;
$Liste = Liste \cup \{y\}$;
Tant que $Liste \neq \emptyset$

Parcours en profondeur à partir d'un sommet : DFS récursif

DFSrecursif($a \in X$, etat[] booleen) {
etat[a] = 1;
Ecrire(a); //ordre préfixe
$\forall\ y \in \Gamma^+(a)$
si (etat[y] $\neq$ 1)
DFSrecursif(y,etat);
Ecrire(a); //ordre suffixe
}
DFS(G = (X,U),$a \in X$,etat : tableau booléen) {
$\forall\ x \in X$ etat[x] = 0;
DFSrecursif(a,etat);
}

Parcours en profondeur à partir d'un sommet : DFS récursif

DFSrecursif($a \in X$, etat[] booleen) {
etat[a] = 1;
Ecrire(a); //ordre préfixe
$\forall\ y \in \Gamma(a)$/*$\Gamma(a) = \Gamma^{+}(a) \cup \Gamma^{-}(a)$*/
si (etat[y] $\neq$ 1)
DFSrecursif(y,etat);
Ecrire(a); //ordre suffixe
}
DFS(G = (X,U),$a \in X$,etat : tableau booléen) {
$\forall\ x \in X$ etat[x] = 0;
DFSrecursif(a,etat);
}

Recherche d'un chemin de s à t

Idée :
Marquer les sommets du nom de leur prédécesseur (pour retrouver le chemin). S'arrêter dès que t est marqué.
Données : G = (X,U) ; deux sommets s et t de X.
Sortie : un chemin de s à t

Algorithme de recherche d'un chemin

Pour tout $x \in X$ etat[x] = nonatteint;
Liste = $\{s\}$; etat[s] = atteint ;
Tant que ($Liste \neq \emptyset$ et etat[t]=nonatteint) faire
x =choix(Liste) ; $Liste = Liste \setminus \{x\}$;
$\forall\ y \in \Gamma^+(x)$
si (etat[y] $\neq$ atteint)
etat[y]=atteint; pere(y) = x; $Liste = Liste \cup \{y\}$;
fintantque
si (etat[t]=atteint)
Ecrire"Chemin de s à t : dans l'ordre inverse"
y = t ; Tant que ($y \neq s$) Ecrire(y); y = pere(y);
Ecrire(s);
sinon Ecrire"pas de chemin de s à t";

Recherche de chemins dans un graphe avec circuits

Le problème d'énumérer des chemins élémentaires issus d'un sommet s donné est un problème complexe puisque ces chemins sont les branches de l'arborescence de racine s. Le choix d'une exploration en profondeur s'explique par la propriété suivante :

Lors d'une exploration en profondeur, l'état de la pile est un chemin élémentaire d'origine s et d'extrémité le sommet de la pile.

Les statuts possibles d'un sommet sont cette fois-ci dehors, en attente, terminé (au lieu de marqué et non marqué). Le sommet s est préalablement en état attente.
Le statut terminé d'un sommet peut être remis en cause lorsque ce sommet est à nouveau visité à partir d'un nouveau prédécesseur, toute l'exploration issue de ce sommet doit être refaite.

Algorithme de recherche de tous les chemins

Données : G = (X,U) ; $s \in X$.
Sortie : afficher tous les chemins issus de s.
Enumeration-recursive(G :graphe, $s \in X$)
Debut
Chemin : une pile ;
Pour tout $x \in X$ faire
    etat[x] = dehors ;
etat[s] = en attente ;
Empiler(s , chemin) ;
Enumerer(s) ;
Fin

Algorithme de recherche de tous les chemins

Enumerer (s : sommet) //version récursive
Pour tout $y \in \Gamma^+(s)\{$
Si (etat [y] $\neq$ en attente) {
  Empiler(y, chemin) ;
  Ecrire (chemin) ;
  etat[y] = en attente ;
  Enumerer(y) ;
}
Si (etat[y] en attente)
  Ecrirecircuit(y,chemin) ; }
Depiler(chemin) ;
etat[s] = terminé ;
Fin—Enumerer

Enumeration(G :graphe, s : sommet) //version itérative
aexplore(s) = $\Gamma^+(s)$; Empiler(s,Q) ;
Tant que (Q non vide)
y = tete(pile) ;
si (aexplore(y) non vide) alors
  z = element(aexplore(y)) ;
  $aexplore(y) = aexplore(y) \setminus \{z\}$;
  si ($z \notin Q$) alors
    empiler(z,Q) ; $aexplore(z) = \Gamma^+(z)$;
    ecrirechemin ;(qui n'est autre que le contenu de la pile)
  sinon
    ecrirecircuit(z) ;
sinon
  $T = T \cup \{y\}$;
  Depiler(Q) ;
Finsi
Fintantque

Détecter un circuit à partir d'un sommet

visiter($a \in X$, $\forall\, x \in X$ visite[x] et critic[x]: booléens) : booléen {

visite[a] = 1;

si (critic[a] = 1)

retourner 1;

sinon {

/*Marquer le sommet courant dans la liste critique*/

critic[a] = 1;

/*Répéter pour tout sommet adjacent à ce sommet*/

$\forall\, y \in \Gamma^+(a)$

si (visiter(y, visite, critic)) retourner 1;

critic[a] = 0; retourner 0;

}

Rq. visite et critic sont initialisés à 0.

Détecter un circuit dans tout le graphe

DetecterCircuit(G = (X, U)) : booléen
{ visite et critic deux tableaux de booléens;
/*Marquer tous les sommets avec non visités et $\notin$ liste critique*/
  $\forall\ i \in X$ {
    visite[i] = false;
    critic[i] = false;
  }
/*Appeler la fonction récursive qui détecte l'existence d'un circuit dans les différentes arborescences*/
  $\forall\ i \in X$
    si (visiter(i, visite, critic) = true)
      retourner true;
  retourner false;
}

Problème

Un ingénieur du service technique d'une ville propose, pour la zone centrale, un plan de circulation. Avant d'adopter ce plan, la direction doit examiner s'il autorise la circulation des automobiles, c'est-à-dire s'il ne rend pas certains points inaccessibles.
Nous verrons comment la notion de forte connexité peut aider la direction à prendre la décision convenable.

composantes fortement connexes

Définition

Etant donné un graphe G = (X, U), on définit la relation binaire CF de connexité forte sur l'ensemble des sommets de G par :
x CF y $\Leftrightarrow$ il existe dans G un chemin de x à y et un chemin de y à x ou bien x=y.

- La relation CF est une relation d'équivalence.
- Les classes d'équivalence sont les composantes fortement connexes du graphe G.
- Un graphe est dit fortement connexe s'il ne possède qu'une composante fortement connexe.

Détermination de la composante fortement connexe contenant un sommet a

Algorithme CFC(Données : G = (X, U, I, T) et a∈ X, Sorties : X' la composante fortement connexe)
DEBUT $X^+ = X^- = \{a\}$
Pour tout x∈ X^+ faire
  Pour tout u∈ U tel que : I(u) = x et $T(u) \notin X^+$ faire
    $X^+ = X^+ \cup \{T(u)\}$
Pour tout x ∈ X^- faire
  Pour tout u ∈ U tel que : T(u) = x et $I(u) \notin X^-$ faire
    $X^- = X^- \cup \{I(u)\}$
$X' = X^+ \cap X^-$
FIN

Graphe réduit

Définition

On appelle graphe réduit d'un graphe G le graphe Gr = (CF,Ur) dont l'ensemble des sommets CF est en bijection avec les composantes fortement connexes de G et :
(I J) $\in$ Ur si $\exists$ u=(a b) $\in$ U tq a$\in$ I et b $\in$ J.

Theorem

Le graphe réduit d'un graphe G n'admet pas de circuit.

Remarque : si x et y (dans G)appartiennent à un même circuit alors ils appartiennent à la même composante fortement connexe.

détermination de toutes les composantes fortement connexes

plusieurs chercheurs se sont penchés sur la question de la recherche des composantes fortement connexes d'un graphe orienté.
Les algorithmes qu'ils ont conçus et qui portent leurs noms ont prouvé leur efficacité. On cite :
Algorithme de R.E.Tarjan
Algorithme de Kosaraju
Algorithme de Gabow(Path-based strong component algorithm)

Algorithme de R.E.Tarjan

Idée : on marque les noeuds avec un indice que l'on augmente.

Debut

$num \leftarrow 0$;

$P \leftarrow \emptyset$;

$partition \leftarrow \emptyset$;

Pour tout $x \in X$

  $x.num = -1$;

Pour tout $x \in X$

  $si(x.num = -1)$

    parcours(x);

Finsi

Finpour

Fin.

function parcours(x)
$x.num \leftarrow num$; $x.numAccessible \leftarrow num$;
$num \leftarrow num + 1$; empiler(x,P); $x.dansP \leftarrow oui$;
Pourtout $y \in \Gamma^+(x)$
si (y.num = -1)
parcours(y);
$x.numAccessible \leftarrow min(x.numAccessible, y.numAccessible)$;
sinon si (y.dansP = oui)
$x.numAccessible \leftarrow min(x.numAccessible, y.num)$;
Finsi Finpour
si (x.numAccessible = x.num)
/*C'est une racine, calcule la CFC associée*/ $X' \leftarrow \emptyset$;
répéter
$w \leftarrow tete(P); depiler(P); w.dansP \leftarrow non; X' \leftarrow X' \cup \{w\}$;
Tant que $(w \neq x)$;
$partition \leftarrow partition \cup X'$;

Path-based strong component algorithm

L'algorithme applique une DFS et maintient 2 piles S et P.
La pile S contient tous les sommets qui n'ont pas encore été affecté à une CFC dans l'ordre dans lequel ils ont été atteint par la DFS.
La pile P contient tous les sommets dont on ne sait pas s'ils appartiennent à une CFC différente de celles trouvées
On met num(x)=0 pour tous les sommets et c=1

DFSC($a \in X$)

$num(a) \leftarrow c$; incrémenter c;
Empiler a sur S; Empiler a sur P;
Pour chaque voisin y de a
  si (num(y)=0) alors
    DFSC(y);
  sinon si y n'a pas été placé dans une CFC alors
    Tant que ($num(sommet(P)) > num(y)$)
    dépiler P;
Finpour
si (a est le sommet de P)
  Dépiler S jusqu'à dépiler a.
  Mettre tous les sommets dépilés dans une CFC
  Dépiler a de P

Path-based strong component algorithm

Procedure Strong(G)
Début
S et P deux piles vides;
Pourtoutx $\in X$ faire num[x] $= 0$;
c $= 1$;
Pourtoutx $\in X$ faire
  si (num[x] $= 0$)
    DFSC(x);
Fin.

Problème

- A quel moment un réseau informatique satisfait-il à la propriété que tous les ordinateurs, pris deux à deux, puissent partager l'information? Des messages peuvent-ils être envoyés au moyen d'un ou de plusieurs ordinateurs intermédiaires? En modélisant ce problème à l'aide de graphe, la question devient : Existe-t-il toujours une chaine entre deux sommets de ce graphe.
- Soit la situation suivante : deux pays A et B sont en guerre. L'état major de A a décidé de porter un coup décisif à B. Le soutien logistique de B est effectué par l'intermédiaire du réseau ferroviaire. On peut donc soit détruire les gares (nœuds) soit couper les voies(arcs).

Nous verrons comment la notion de connexité peut aider à résoudre ce genre de problèmes.

Connexité

Définition

Soit un graphe G = (X, U), on définit la relation C de connexité sur X par : x C y $\Leftrightarrow$ il existe dans G une chaine joignant x et y ou bien x=y

- C est une relation d'équivalence.
- Les classes l'équivalence s'appellent les composantes connexes.

Détermination de la composante connexe contenant un sommet a

Algorithme CC(Données : G = (X, U, I, T) et a∈ X,
Sorties : X' : la composante connexe)
DEBUT
$X' = \{a\}$
Pour tout $x \in X'$ faire
Pour tout $u \in U$ tq I(u) = x et T(u) $\notin$ X' faire

$$X' = X' \cup \{T(u)\}$$

Pour tout $u \in U$ tq T(u) = x et I(u) $\notin$ X' faire

$$X' = X' \cup \{I(u)\}$$

FIN

Parcours en profondeur à partir d'un sommet : DFS récursif

DFSrecursif($a \in X$, etat[] booleen) {
etat[a] = 1;
Ecrire(a); //ordre préfixe//ajouter à la composante de a
$\forall\ y \in \Gamma(a)//\Gamma(a) = \Gamma^{+}(a) \cup \Gamma^{-}(a)$
si (etat[y] $\neq$ 1)
DFSrecursif(y,etat);
Ecrire(a); //ordre suffixe
}
DFS(G = (X,U),$a \in X$,etat : tableau booléen) {
$\forall\ x \in X$ etat[x] = 0;
DFSrecursif(a,etat);
}

Graphe non orienté

Définition

Un graphe non orienté G = (X, E) est défini par la donnée de :

- Deux ensembles X et E dits ensemble de sommets et d'arêtes
- Une application IT qui associe à chaque arête ses deux extrémités

$$IT : E \rightarrow P2(X) \cup X$$

(P2(X):l'ensemble des parties de X à deux éléments)

Remarques

Les notions de demi-degré, chemin, forte connexité et de circuit sont orientées. Elles n'ont pas d'équivalents dans les graphes non orientés. Par contre, celle d'adjacence, de degré, d'isomorphisme, de chaine, de cycle et de connexité y ont un équivalent.

- Deux sommets sont adjacents s'ils sont extrémités d'une même arête.
- Deux arêtes sont adjacentes si elles ont une extrémité commune
- Une arête a est adjacente à un sommet x si x=IT(a)
- dg(x) = nombre d'arêtes adjacentes à x.
- La somme des degrés des sommets d'un graphe G (sans boucle) est égale à 2 fois le nombre d'arêtes.
- le nombre de sommets de degré impair est pair

h-connexité

Définition

Un sommet dont la suppression déconnecte le graphe est appelé point d'articulation. Un ensemble d'articulation est un ensemble de sommets dont la suppression déconnecte le graphe (ou le réduit à un sommet).
Une arête dont la suppression déconnecte le graphe est appelée arête de coupure (ou encore isthme). Un ensemble F inclus dans E d'un graphe G = (X,E) est un ensemble de coupure ou plus simplement une coupe si F est un ensemble minimal (pour l'inclusion) tel que E – F n'est pas connexe.

h-connexité

Définition

Un graphe non orienté est dit h-connexe (respectivement h-arêtes-connexes) si la suppression de tout ensemble de h-1 sommets (respectivement arêtes) ne déconnecte pas le graphe. La connectivité (respectivement l'arête connectivité) est la valeur maximum de h pour laquelle le graphe est h-connexe (respectivement h-arêtes-connexe).
La connectivité (respectivement l'arête-connectivité) est égale au nombre minimum de sommets (respectivement arêtes) qu'il faut supprimer pour déconnecter le graphe.

theorem (Dirac 1960)

Un graphe G = (X, E) non orienté est h-connexe (*avec* $h \geq 1$) si et seulement si $\mid X \mid \geq h+1$ et $Y \subseteq X$ tel que $\mid Y \mid = h$, $\forall x \in X \setminus Y$, il existe un ensemble de chaines 2 à 2 sommets disjointes (sauf en x) reliant x à tous les sommets de Y.

La résolution du problème militaire consiste alors à déterminer la connectivité ou l'arête connectivité du réseau ferroviaire.

Graphe eulérien

Définition

Dans un graphe non orienté, une chaine eulérienne est une chaine qui emprunte une et une seule fois chaque arête du graphe. De même un cycle eulérien est un cycle qui emprunte une et une seule fois chaque arête du graphe. Un graphe comportant une chaine ou un cycle eulérien est appelé graphe eulérien.

Conditions nécessaires et suffisantes

Theorem

Un graphe (simple ou multiple) connexe admet un cycle eulérien si et seulement s'il n'a pas de sommet de degré impair.

Theorem

Un graphe (simple ou multiple) connexe admet une chaine eulérienne entre deux sommets a et b si et seulement si le degré de a et le degré de b sont impairs, et les degrés de tous les autres sommets du graphe sont pairs.

Conditions nécessaires et suffisantes

On retrouve ces différentes notions dans les graphes orientés et en plus les notions de chemin eulérien et de circuit eulérien.

Theorem

Un multigraphe orienté fortement connexe admet un circuit eulérien si et seulement si $d^+(x) = d^-(x)\ \forall x \in X$.

Theorem

Un multigraphe orienté connexe admet un chemin eulérien de a à b si et seulement si

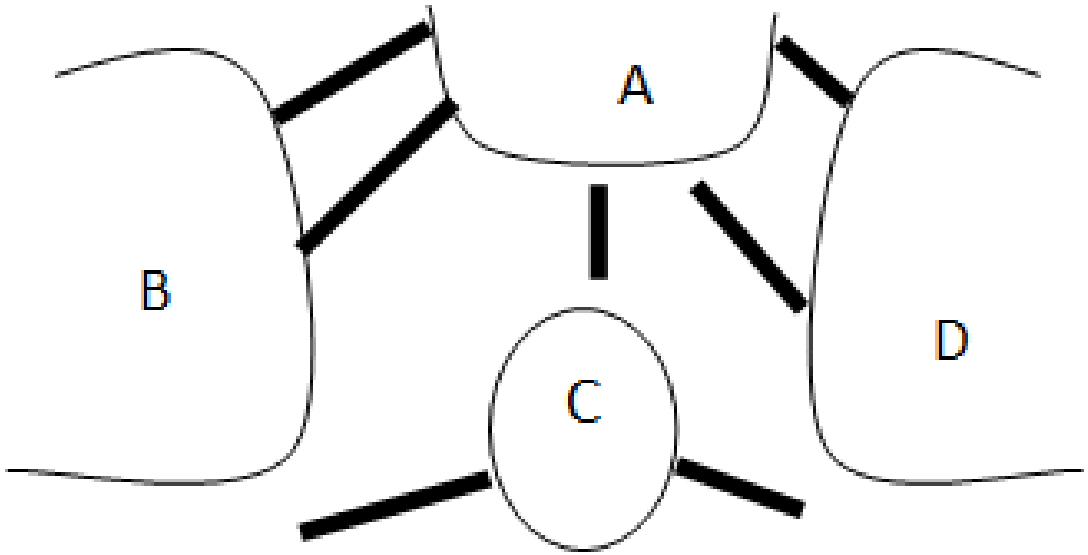
$d^+(a) = d^-(a) + 1$

$d^+(b) = d^-(b) - 1$

$d^+(x) = d^-(x)\ \forall x \in X \setminus \{a, b\}$

Exercice

On considère la disposition des ponts de la ville de Koenigsberg (Kaliningrad aujourd'hui)(ville que Euler visita lors de son voyage en Russie), où A et C sont deux îles et B et D sont des berges :



Un piéton peut-il en se promenant traverser chacun des sept ponts de la ville une et une seule fois, et revenir au point de départ ?

Notion de graphe hamiltonien

Dans un graphe simple non orienté comportant n sommets, une chaine hamiltonienne est une chaine élémentaire de longueur n-1. Autrement dit, une chaine hamiltonienne passe une et une seule fois par chacun des sommets du graphe.
On appelle cycle hamiltonien un cycle élémentaire de longueur n. un graphe possédant un cycle hamiltonien est appelé graphe hamiltonien.
Si le graphe ne possède pas de cycle hamiltonien mais possède une chaine hamiltonienne est dit semi-hamiltonien.

Conditions suffisantes

On ne connait aucune condition nécessaire et suffisante d'existence de cycles (chaines, circuits ou chemins) hamiltoniens, valable pour tous les graphes. On peut juste donner des conditions suffisantes, portant en particulier sur les degrés des sommets d'un graphe simple.
Théorème de Ore dû à Øystein Ore, énoncé en 1960 :

Theorem

Un graphe simple à n sommets ($n \geq 3$) tel que la somme des degrés de toute paire de sommets non adjacents vaut au moins n est hamiltonien.

Théorème de Bondy et Chvátal

S'inspirant du théorème de Ore, John Adrian Bondy et Václav Chvátal ont trouvé en 1976 une méthode pour déterminer si un graphe est hamiltonien : tant qu'il reste des sommets i et j tels que $deg(i) + deg(j) \lneq n$, on ajoute l'arête i,j au graphe. Quand on ne peut plus en ajouter, on a obtenu ce qu'on appelle la fermeture du graphe. On applique alors le théorème suivant :

Theorem

Un graphe est hamiltonien si et seulement si sa fermeture est hamiltonienne.

Remarque :
la fermeture d'un graphe G est le graphe construit à partir de G tel que l'ensemble des sommets est le même que G et les arcs sont les couples de sommets entre lesquels il existe un chemin dans G.

Applications

De nombreux problèmes de recherche opérationnelle consistent à chercher un chemin ou un cycle hamiltonien dans un graphe.
Le plus connu est probablement le problème du voyageur de commerce, qui doit trouver un itinéraire "optimal" passant par chaque ville de son réseau commercial.
La carte routière est représentée par un graphe non orienté valué : les sommets correspondent aux villes, les arêtes aux routes, et les poids aux distances.
Il s'agit de trouver un cycle hamiltonien de pois minimal (le poids d'un cycle est la somme des poids des arêtes le composant).