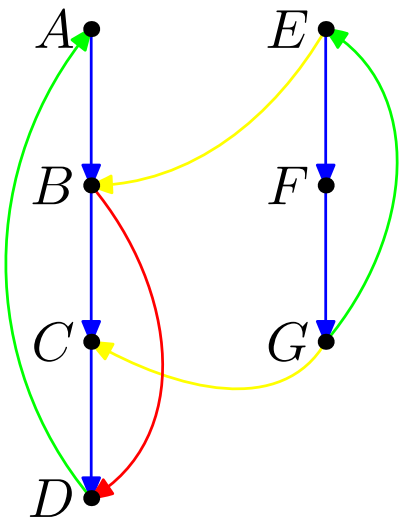
Algorithmique des graphes

Cours 5 – Applications de PP (DFS)

František Kardoš

frantisek.kardos@u-bordeaux.fr

Parcours en profondeur – graphes orientés



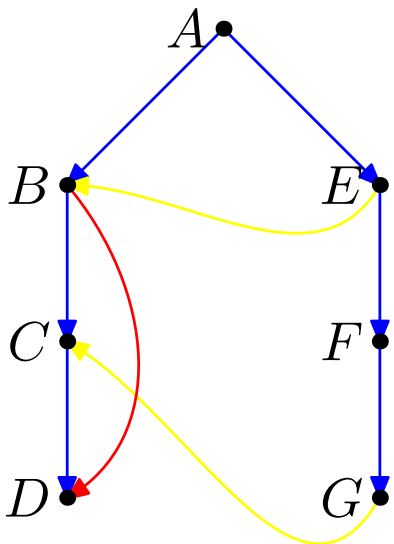
arcs de liaison
arcs de retour
arcs d'avant
arcs transverses

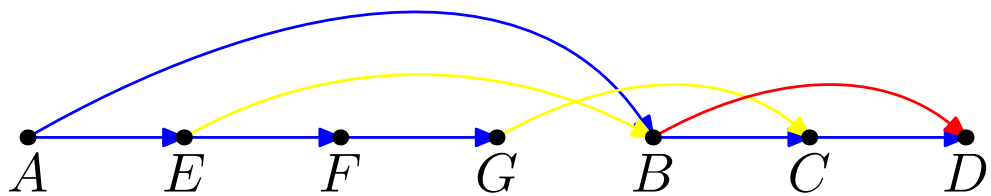
Parcours en profondeur : Applications

Dans un graphe orienté sans circuit, un parcours en profondeur permet d'établir un *tri topologique* – trouver un ordre linéaire des sommets tel que tout arc suive la même direction (de gauche à droite) :

Parcours en profondeur : Applications

Dans un graphe orienté sans circuit, un parcours en profondeur permet d'établir un *tri topologique* – trouver un ordre linéaire des sommets tel que tout arc suive la même direction (de gauche à droite) : il suffit d'ordonner les sommets par l'ordre décroissant de la fin de visite.





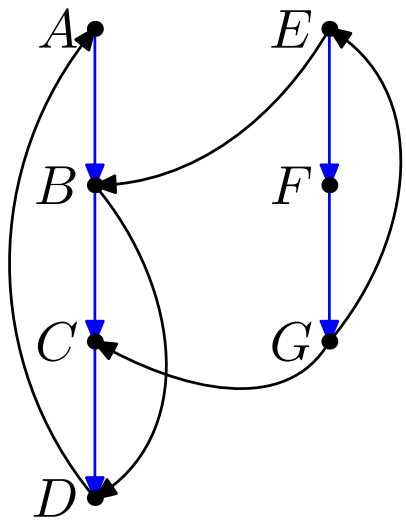
Encore quelques définitions

Un graphe orienté est dit *fortement connexe* si ses sommets sont tous accessibles entre eux, i.e., pour toute paire de sommets u et v il existe un chemin de u à v et aussi un chemin de v à u.

Encore quelques définitions

Un graphe orienté est dit *fortement connexe* si ses sommets sont tous accessibles entre eux, i.e., pour toute paire de sommets u et v il existe un chemin de u à v et aussi un chemin de v à u.

Le graphe ci-dessous est-il fortement connexe ?



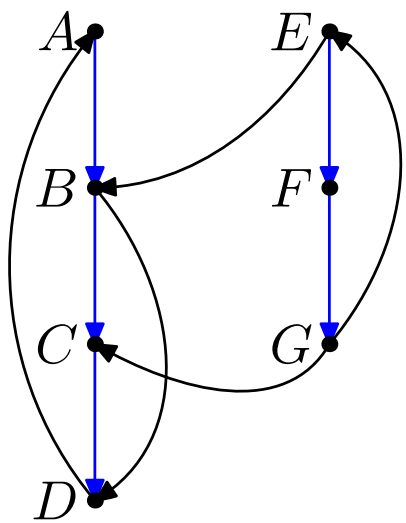
Encore quelques définitions

Dans un graphe orienté, une *composante fortement connexe* est un sous-ensemble X de sommets de G induisant un graphe fortement connexe, X étant maximal par rapport à la connexité forte.

Encore quelques définitions

Dans un graphe orienté, une *composante fortement connexe* est un sous-ensemble X de sommets de G induisant un graphe fortement connexe, X étant maximal par rapport à la connexité forte.

Quelles sont les composantes fortement connexes ?



Quelques observations

Vrai ou faux ?

- Soit uv un arc d'une composante fortement connexe. Alors il existe un chemin de v à u.

Quelques observations

Vrai ou faux ?

- Soit uv un arc d'une composante fortement connexe. Alors il existe un chemin de v à u.
- Soit uv un arc d'une composante fortement connexe. Alors uv est contenu dans un circuit.

Quelques observations

Vrai ou faux ?

- Soit uv un arc d'une composante fortement connexe. Alors il existe un chemin de v à u.
- Soit uv un arc d'une composante fortement connexe. Alors uv est contenu dans un circuit.
- Soit u et v deux sommets d'un graphe orienté. Si u et v appartiennent au même circuit, alors u et v sont dans la même composante fortement connexe.

Quelques observations

Vrai ou faux ?

- Soit uv un arc d'une composante fortement connexe. Alors il existe un chemin de v à u.
- Soit uv un arc d'une composante fortement connexe. Alors uv est contenu dans un circuit.
- Soit u et v deux sommets d'un graphe orienté. Si u et v appartiennent au même circuit, alors u et v sont dans la même composante fortement connexe.
- Soit u et v deux sommets d'un graphe orienté. Si u et v sont dans la même composante fortement connexe, alors u et v appartiennent au même circuit.

Quelques observations

Vrai ou faux ?

- Soit uv un arc d'une composante fortement connexe. Alors il existe un chemin de v à u.
- Soit uv un arc d'une composante fortement connexe. Alors uv est contenu dans un circuit.
- Soit u et v deux sommets d'un graphe orienté. Si u et v appartiennent au même circuit, alors u et v sont dans la même composante fortement connexe.
- Soit u et v deux sommets d'un graphe orienté. Si u et v sont dans la même composante fortement connexe, alors u et v appartiennent au même circuit. FAUX !

Calcul de composantes fortement connexes

Lemme

Soit G un graphe orienté, soit C un circuit dans G. Soit G/C le graphe obtenu en réduisant C en un seul sommet, disons c. Alors G/C a autant de composantes connexes que G, en plus,

- *Si une composante X ne couvre pas C, alors X est une composante de G/C ;*
- *Si X est une composante couvrant C, alors $X \setminus V(C) \cup \{c\}$ est une composante de G/C.*

Note. En réduisant un circuit C, on peut supprimer les boucles et pour tout ensemble d'arcs parallèles il suffit d'en garder un.

Calcul de composantes fortement connexes

On applique le lemme tant que le graphe contient encore des circuits.

Calcul de composantes fortement connexes

On applique le lemme tant que le graphe contient encore des circuits.

Le graphe final est un graphe sans circuit, dont tout sommet représente une composante fortement connexe du graphe de départ.

Calcul de composantes fortement connexes

On applique le lemme tant que le graphe contient encore des circuits.

Le graphe final est un graphe sans circuit, dont tout sommet représente une composante fortement connexe du graphe de départ.

On l'appelle le *graphe des composantes fortement connexes* $CFC(G)$.

Tester la connexité forte avec Parcours en profondeur

Calculer la composante fortement connexe d'un sommet u :

Idée : Pour un sommet u, déterminer

- l'ensemble $Desc(u)$ des sommets accessibles depuis u
- l'ensemble $Anc(u)$ des sommets u est accessible depuis

avec deux parcours en profondeur, un sur G, et l'autre sur G^{-1}.

La CFC(u) est alors $Desc(u) \cap Anc(u)$.

Parcours en profondeur et CFC

Soit G un graphe orienté, soit X une CFC de G. Soit x le premier sommet de X découvert par un parcours de G en profondeur. Alors x est l'ancêtre commun de tous les sommets de X. Autrement dit, tous les sommets de X sont découverts et visités avant la fin de la visite de x. Par ailleurs, si

$$d(x) = \min\{d(u) : u \in X\},$$

alors

$$f(x) = \max\{f(u) : u \in X\}.$$

Parcours en profondeur et CFC

Soit X_1 et X_2 deux CFC d'un graphe orienté G. Notons x_1 (resp. x_2) le premier sommet de X_1 (X_2) découvert par un parcours de G en profondeur.

- Si $X_1 \to X_2$ dans $CFC(G)$, alors x_1 est un ancêtre de x_2, $I(x_1) \supset I(x_2)$.
- Si $X_1 \leftarrow X_2$ ou X_1 et X_2 ne sont pas du tout reliés par un arc dans $CFC(G)$, alors $I(x_1) \cap I(x_2) = \emptyset$.

Algorithme de Kosaraju

Soit G un graphe orienté.

1. Exécuter un parcours en profondeur de G.
2. Exécuter un parcours en profondeur sur le graphe transposé G^{-1} en explorant les sommets dans l'ordre décroissant de la fin de visite du premier parcours.

Les arborescences produites par le second parcours sont les CFC de G.

Algorithme de Kosaraju – exemple

$A : B, G$
$B : E$
$C :$
$D : F$
$E : D, G$
$F : E$
$G : B, C$

Algorithme de Kosaraju – complexité

Soit G un graphe orienté.

1. Exécuter un parcours en profondeur de G – $O(m+n)$.
(*) Calculer le graphe transposé G^{-1} – $O(m+n)$ si graphe représenté par listes de successeurs.
2. Exécuter un parcours en profondeur de G^{-1} – $O(m+n)$.

La complexité de l'algorithme de Kosaraju est donc $O(m+n)$, la même que celle d'un parcours en profondeur.

Algorithme de Tarjan

permet de calculer les CFC en un seul parcours en profondeur.

Pendant le parcours en profondeur de G, on utilise un tableau supplémentaire $t[\,]$ pour calculer, pour chaque sommet de G, le sommet le plus haut (l'ancêtre le plus ancien) vers lequel il est possible de remonter grâce à des arcs de retour.

Algorithme de Tarjan

permet de calculer les CFC en un seul parcours en profondeur.

Pendant le parcours en profondeur de G, on utilise un tableau supplémentaire $t[\,]$ pour calculer, pour chaque sommet de G, le sommet le plus haut (l'ancêtre le plus ancien) vers lequel il est possible de remonter grâce à des arcs de retour.

Si jamais pour un sommet u à la fin de sa visite on a $t[u] = u$, on sais que u est la racine d'une composante fortement connexe.

Algorithme de Tarjan – implémentation

Dans $t[\,]$, au lieu de stocker
le sommet le plus haut (l'ancêtre le plus ancien) vers lequel il est possible de remonter grâce à des arcs de retour,
on va stocker
le début de visite du sommet le plus haut (l'ancêtre le plus ancien) vers lequel il est possible de remonter grâce à des arcs de retour.

Algorithme de Tarjan – implémentation

Dans $t[\,]$, au lieu de stocker
le sommet le plus haut (l'ancêtre le plus ancien) vers lequel il est possible de remonter grâce à des arcs de retour,
on va stocker
le début de visite du sommet le plus haut (l'ancêtre le plus ancien) vers lequel il est possible de remonter grâce à des arcs de retour.

Pour calculer $t[u]$, au début de visite de u, on l'initialise à $d[u]$.
On le met à jour à la fin de visite de chacun de ses fils, et aussi lors d'une découverte d'un arc de retour.

Algorithme de Tarjan – implémentation

Algorithme 1 : DFS(G)

Données : graphe G
début
  pour chaque *s sommet de* G **faire**
    couleur[s] $\leftarrow$ BLANC ;
    $\pi[s] \leftarrow$ NULL ;
  fin
  temps $\leftarrow 0$;
  pour chaque *s sommet de* G **faire**
    si *couleur*[s] $=$ *BLANC*
    alors
      Visiter(s) ;
    fin
  fin
fin

Algorithme 2 : Visiter(u)

début
  couleur[u] $\leftarrow$ GRIS ;
  $d[u] \leftarrow$ temps $\leftarrow$ temps $+ 1$;
  pour chaque *v voisin de u*
  faire
    si *couleur*[v] $=$ *BLANC*
    alors
      $\pi[v] \leftarrow u$;
      Visiter(v) ;
    fin
  fin
  couleur[u] $\leftarrow$ NOIR ;
  $f[u] \leftarrow$ temps $\leftarrow$ temps $+ 1$;
fin

Algorithme de Tarjan – implémentation

Algorithme 1 : DFS(G)

Données : graphe G
début
  pour chaque s *sommet de* G **faire**
    couleur[s] $\leftarrow$ BLANC ;
    $\pi[s] \leftarrow$ NULL ;
  fin
  temps $\leftarrow 0$;
  pour chaque s *sommet de* G **faire**
    si *couleur*[s] $=$ *BLANC* **alors**
      Visiter(s) ;
    fin
  fin
fin

Algorithme 2 : Visiter(u)

début
  couleur[u] $\leftarrow$ GRIS ;
  $d[u] \leftarrow$ temps $\leftarrow$ temps $+ 1$;
  $t[u] \leftarrow d[u]$;
  pour chaque v *voisin de* u **faire**
    si *couleur*[v] $=$ *BLANC* **alors**
      $\pi[v] \leftarrow u$;
      Visiter(v) ;
      $t[u] \leftarrow min(t[u], t[v])$;
    fin
    si *couleur*[v] $=$ *GRIS* **alors**
      $t[u] \leftarrow$
      $min(t[u], d[v])$;
    fin
  fin
  couleur[u] $\leftarrow$ NOIR ;
  $f[u] \leftarrow$ temps $\leftarrow$ temps $+ 1$;
fin

Algorithme de Tarjan – implémentation

Algorithme 1 : DFS(G)

Données : graphe G
début
    pour chaque s *sommet de* G **faire**
        couleur[s] $\leftarrow$ BLANC ;
        $\pi[s] \leftarrow$ NULL ;
    fin
    temps $\leftarrow 0$;
    CFC $\leftarrow [\,]$;
    $L \leftarrow [\,]$;
    pour chaque s *sommet de* G **faire**
        si *couleur*[s] $=$ *BLANC* **alors**
            Visiter(s) ;
        fin
    fin
fin

Algorithme 2 : Visiter(u)

début
    couleur[u] $\leftarrow$ GRIS ;
    $d[u] \leftarrow$ temps $\leftarrow$ temps $+ 1$;
    $t[u] \leftarrow d[u]$;
    $L.append(u)$;
    pour chaque v *voisin de* u **faire**
        si *couleur*[v] $=$ *BLANC* **alors**
            $\pi[v] \leftarrow u$;
            Visiter(v) ;
            $t[u] \leftarrow min(t[u], t[v])$;
        fin
        sinon
            si v *in* L **alors**
                $t[u] \leftarrow min(t[u], d[v])$;
            fin
        fin
    fin
    couleur[u] $\leftarrow$ NOIR ;
    $f[u] \leftarrow$ temps $\leftarrow$ temps $+ 1$;
    si $t[u] = d[u]$ **alors**
        CFC.$append(L[u :])$;
        $L \leftarrow L[: u]$;
    fin
fin

Algorithme de Tarjan – exemple

u	: $Succ(u)$	$d[u]$	$f[u]$	$t[u]$
A	: B			
B	: D			
C	: A			
D	: C, E, F, H			
E	: G			
F	: E			
G	: F			
H	: I			
I	: J			
J	: H, K			
K	:			

Algorithme de Tarjan – exemple

u	: $Succ(u)$	$d[u]$	$f[u]$	$t[u]$
A	: B	1	21	1
B	: D	2	20	1
C	: A	4	5	1
D	: C, E, F, H	3	20	1
E	: G	6	11	6
F	: E	8	9	6
G	: F	7	10	6
H	: I	12	19	12
I	: J	13	18	12
J	: H, K	14	17	12
K	:	15	16	15

CFC :
$[E, G, F]$
$[K]$
$[H, I, J]$
$[A, B, D, C]$