

Introduction

Il existe principalement deux algorithmes permettant de parcourir un graphe (sommets et arêtes ou arcs) en temps linéaire, c'est à dire en $O(n + m)$:

- ▶ le **parcours en largeur**, que l'on appellera également **BFS** pour "Breadth-First-Search", basé sur l'utilisation d'un file ;
- ▶ le **parcours en profondeur**, que l'on appellera également **DFS** pour "Depth-First-Search", basé sur l'utilisation d'un pile.

Parcours en profondeur

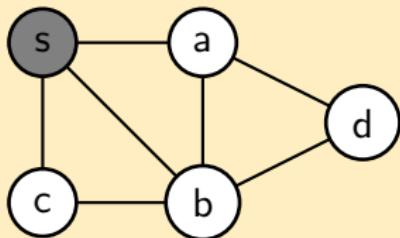
Énoncé

Voir le document *AlgosGraphes.pdf* page 1.

Parcours en profondeur

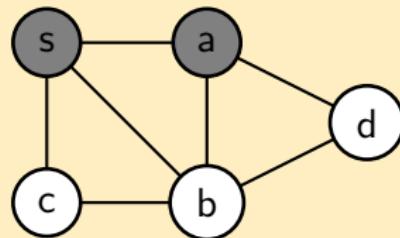
Exemple dans le cas non orienté

La couleur "NOIR" sera affichée avec du bleu afin de laisser l'étiquette lisible.



Pile = [s]

v	s	a	b	c	d
$d(v)$	1				
$f(v)$					

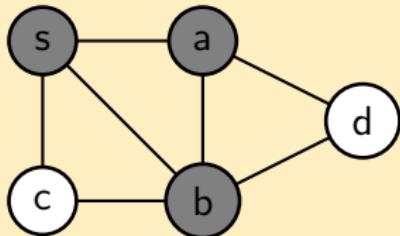


Pile = [a, s]

v	s	a	b	c	d
$d(v)$	1	2			
$f(v)$					

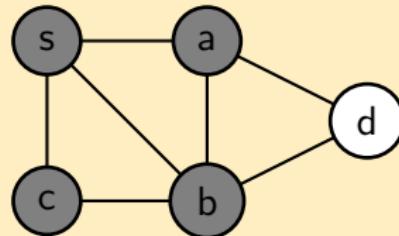
Parcours en profondeur

Exemple dans le cas non orienté



$$Pile = [b, a, s]$$

v	s	a	b	c	d
$d(v)$	1	2	3		
$f(v)$					

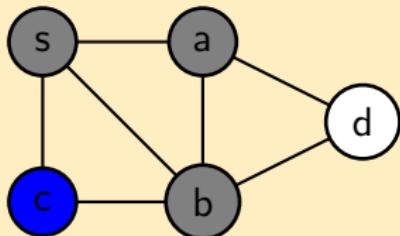


$$Pile = [c, b, a, s]$$

v	s	a	b	c	d
$d(v)$	1	2	3	4	
$f(v)$					

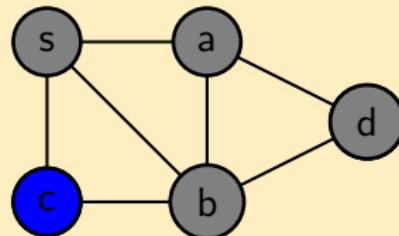
Parcours en profondeur

Exemple dans le cas non orienté



$$Pile = [b, a, s]$$

v	s	a	b	c	d
$d(v)$	1	2	3	4	
$f(v)$				5	

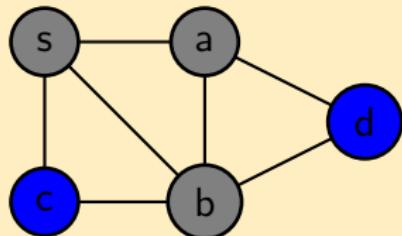


$$Pile = [d, b, a, s]$$

v	s	a	b	c	d
$d(v)$	1	2	3	4	6
$f(v)$				5	

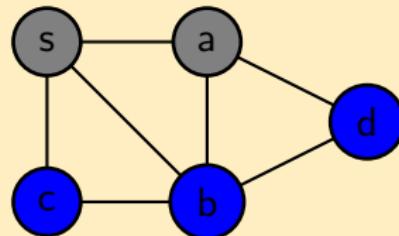
Parcours en profondeur

Exemple dans le cas non orienté



$$Pile = [b, a, s]$$

v	s	a	b	c	d
$d(v)$	1	2	3	4	6
$f(v)$				5	7

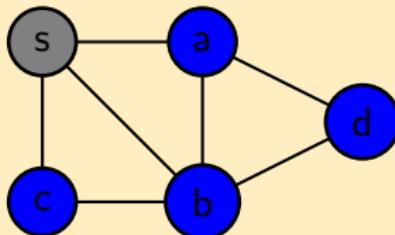


$$Pile = [a, s]$$

v	s	a	b	c	d
$d(v)$	1	2	3	4	6
$f(v)$				8	5

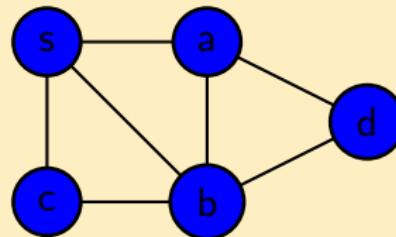
Parcours en profondeur

Exemple dans le cas non orienté



$Pile = [s]$

v	s	a	b	c	d
$d(v)$	1	2	3	4	6
$f(v)$		9	8	5	7

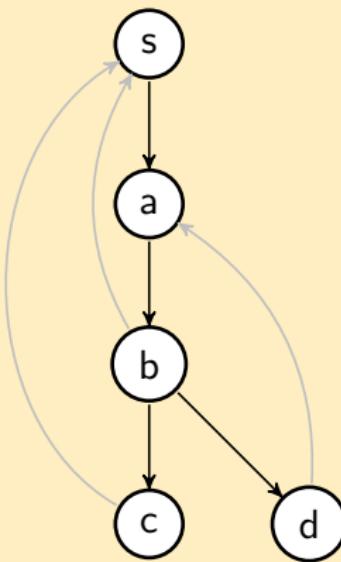


$Pile = []$

v	s	a	b	c	d
$d(v)$	1	2	3	4	6
$f(v)$	10	9	8	5	7

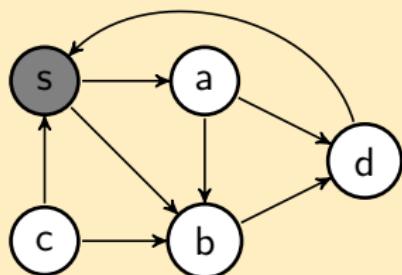
Parcours en profondeur

Arborescence du parcours en profondeur



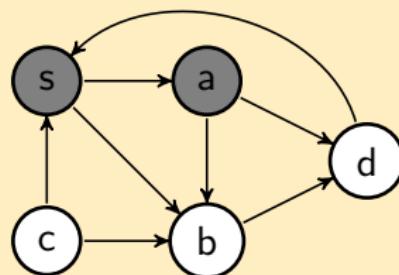
Parcours en profondeur

Exemple dans le cas orienté



$Pile = [s]$

v	s	a	b	c	d
$d(v)$	1				
$f(v)$					

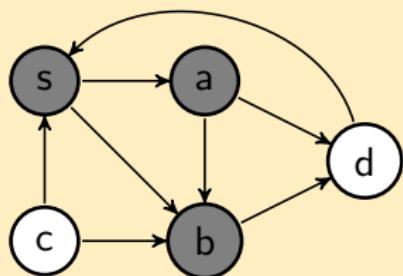


$Pile = [a, s]$

v	s	a	b	c	d
$d(v)$	1	2			
$f(v)$					

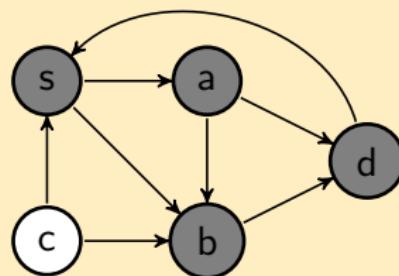
Parcours en profondeur

Exemple dans le cas orienté



$$Pile = [b, a, s]$$

v	s	a	b	c	d
$d(v)$	1	2	3		
$f(v)$					

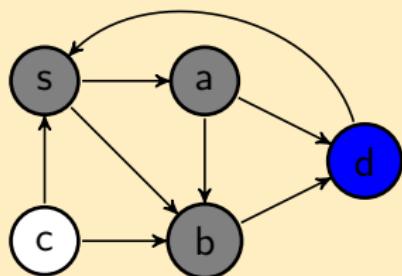


$$Pile = [d, b, a, s]$$

v	s	a	b	c	d
$d(v)$	1	2	3		4
$f(v)$					

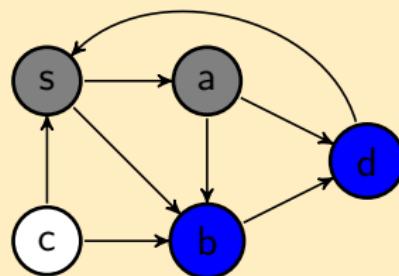
Parcours en profondeur

Exemple dans le cas orienté



$$Pile = [b, a, s]$$

v	s	a	b	c	d
$d(v)$	1	2	3		4
$f(v)$					5

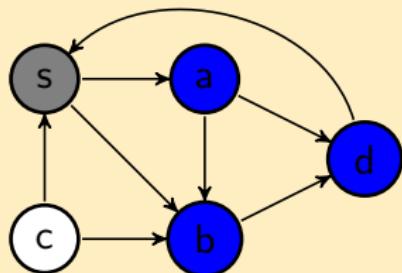


$$Pile = [a, s]$$

v	s	a	b	c	d
$d(v)$	1	2	3		4
$f(v)$			6		5

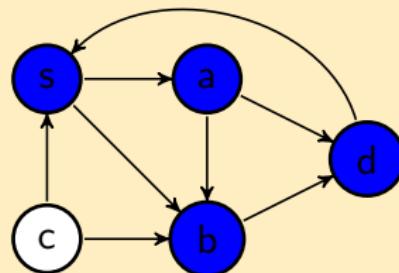
Parcours en profondeur

Exemple dans le cas orienté



Pile = [s]

v	s	a	b	c	d
$d(v)$	1	2	3		4
$f(v)$		7	6		5

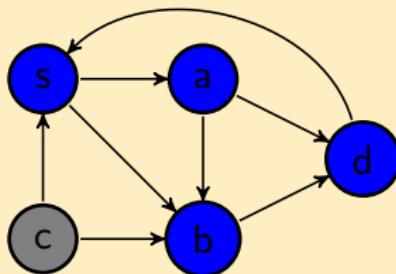


Pile = []

v	s	a	b	c	d
$d(v)$	1	2	3		4
$f(v)$	8	7	6		5

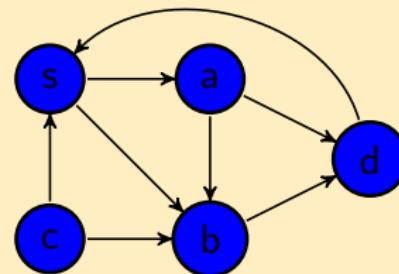
Parcours en profondeur

Exemple dans le cas orienté



$Pile = [c]$

v	s	a	b	c	d
$d(v)$	1	2	3	9	4
$f(v)$	8	7	6		5

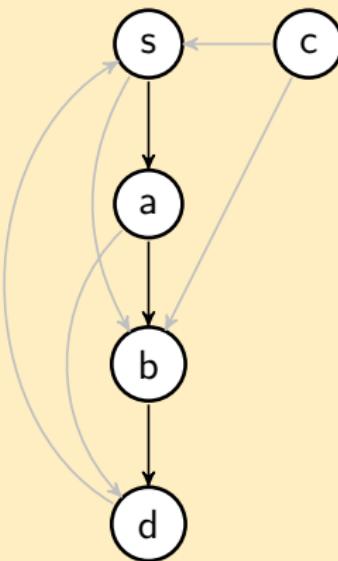


$Pile = []$

v	s	a	b	c	d
$d(v)$	1	2	3	9	4
$f(v)$	8	7	6	10	5

Parcours en profondeur

Arborescence du parcours en profondeur



Type des arcs de l'arborescence

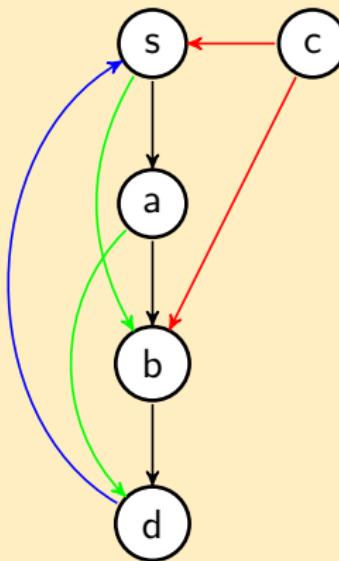
On distingue 4 types d'arcs :

- ▶ de **liaison** : ce sont les arcs reliant un sommet à ses fils ;
- ▶ **avant** : ce sont les arcs reliant un sommet à ses descendants \neq de ses fils ;
- ▶ **arrière ou de retour** : ce sont les arcs reliant un sommet à l'un de ses ancêtres ;
- ▶ **transverse** : ce sont les arcs reliant un sommet aux sommets avec lesquels il n'a aucune relation dans l'arborescence.

Parcours en profondeur

Type des arcs de l'arborescence

Dans le dessin suivant, les arcs de liaison sont en noir, les arcs de retour en bleu, les arcs avant en vert et les arcs transverses en rouge.



Parcours en profondeur

Classe des arcs en fonction des couleurs des sommets

L'idée est de connaître le type d'un arc lors de sa visite en fonction de la couleur du sommet destination (le sommet origine étant toujours GRIS). Soit uv l'arc visité.

- ▶ Si v est BLANC, alors l'arc uv sera un arc de **liaison**.
- ▶ Si v est GRIS, alors l'arc uv sera un arc **retour** (aussi appelé **arrière**).
- ▶ Si v est NOIR, alors l'arc uv sera un arc **avant** ou **transverse**
 - ▶ **avant** si u est un ancêtre de v , c'est à dire que $[d(v), f(v)] \subseteq [d(u), f(u)]$.
 - ▶ **transverse**, si il n'y a pas de relation dans l'arborescence entre u et v et donc $f(u) < d(v)$.

Parcours en profondeur

Classe des arcs dans le cas non orienté

Dans le cas du parcours en profondeur d'un graphe G non orienté, alors les arcs sont soit de liaison, soit retour. Il n'y a pas d'arc avant, ni d'arc transverse.

Justification : Lorsque l'on examine pour la première fois une arête uv , disons à partir d'un sommet u , si v n'a pas encore été visité, alors uv sera un arc de liaison.

Sinon, v est forcément un ancêtre de u car on ne peut finir de visiter v sans regarder l'arête uv . Donc v ne peut être NOIR et donc uv est un arc de retour.

Théorème des parenthèses

Soit u et v deux sommets de G , et $[d(u), f(u)]$, $[d(v), f(v)]$ les intervalles définis par leurs heures de début et fin de visite. On suppose que $d(u) < d(v)$. Deux cas sont possibles :

- ▶ Soit $[d(v), f(v)] \subseteq [d(u), f(u)]$. Dans ce cas v est un descendant de u .
- ▶ Soit $[d(v), f(v)] \cap [d(u), f(u)] = \emptyset$. Dans ce cas, aucun des deux n'est le descendant de l'autre dans l'arborescence.

Justification :

- ▶ dans le cas où $[d(v), f(v)] \subseteq [d(u), f(u)]$, cela signifie que v a été visité quand u était GRIS. Donc v sera un descendant de u et on finira la visite de v avant celle de u .
- ▶ Si $[d(v), f(v)] \cap [d(u), f(u)] = \emptyset$, alors comme $d(v) > d(u)$, on a $f(v) > d(v) > f(u) > d(u)$ et donc, on aura fini de visiter u avant de commencer la visite de v .

Parcours en profondeur

Théorème du chemin blanc

Dans une forêt obtenu par un parcours en profondeur d'un graphe $G = (V, E)$, un sommet v est un descendant d'un sommet u si et seulement si lors du début de visite du sommet u , il existe une chaîne ou un chemin reliant u à v composé exclusivement de sommets blancs.

Théorème du chemin blanc - Justification

si v est un descendant de u , les sommets reliant u à v dans l'arborescence étaient tous BLANC au moment du début de la visite de u .

Inversement, supposons qu'il existe un chemin composé de sommets blancs entre u et v à l'instant $d(u)$, et que v ne soit pas un descendant de u . Choisissons v de telle sorte qu'il soit le sommet de ce chemin blanc le plus proche de u qui ne soit pas un de ses descendants. Et soit w son prédécesseur sur le chemin blanc. w sera bien un descendant de u . Or si v n'est pas un descendant de u et donc de w , quand on finit de visiter w , v sera encore blanc et devrait donc être visité via l'arête wv . Donc v sera un fils de w et donc un descendant de u .

Applications

Il existe de nombreuses applications issues du parcours en profondeur. Parmi elles, citons :

1. le tri topologique ;
2. le calcul des composantes fortement connexes ;
3. la recherche de points d'articulation, un point d'articulation étant un sommet dont la suppression augmente le nombre de composantes connexes ;
4. le test de planarité d'un graphe (Hopcroft et Tarjan).

A noter que dans tous les cas cités ci-dessus, les algorithmes ainsi obtenus sont en temps linéaire, i.e. en $O(n + m)$.

Tri topologique

Définition

Le tri topologique d'un graphe G orienté est un ordonnancement de ses sommets de telle sorte que si v_1, \dots, v_n est l'ordre obtenu, alors :

$$\forall v_i, v_j \in V(G), v_i v_j \in E(G) \Rightarrow i < j$$

Théorème

Un graphe G orienté admet un tri topologique si et seulement si il est sans circuit.

Justification :

\Rightarrow (par contradiction)

soit $v_1 v_2 \dots v_{k-1} v_1$ un circuit de G . Si G admet un tri topologique, alors v_1 doit être rangé avant v_2 car il existe un arc $v_1 v_2$, v_2 avant v_3, \dots, v_{k-1} avant v_1 , ce qui est impossible.

Donc " G graphe orienté admet un tri topologique" \Rightarrow " G est sans circuit".

Tri topologique

Justification (suite)

⇐

A l'inverse, si G est sans circuit, montrons que G admet un tri topologique.

On peut montrer qu'il existe un sommet de G de degré entrant 0. En effet, si l'on considère un chemin P de longueur maximale dans G , disons $P = v_0, v_1, \dots, v_k$, alors v_0 est de degré entrant 0 dans G . Sinon, soit il a un prédécesseur hors de $\{v_1, \dots, v_k\}$ et dans ce cas, P n'est pas un chemin de longueur maximale, soit il a un prédécesseur parmi $\{v_1, \dots, v_k\}$ et dans ce cas, G n'est pas sans circuit.

On prend le sommet v de degré entrant 0 comme le premier sommet dans le tri topologique. $G - v$ est forcément sans circuit, donc possède également un sommet de degré entrant 0, que l'on prendra en deuxième position. On répète cette opération jusqu'à ce que tous les sommets de G aient été placés.

Tri topologique

Algorithme (basé sur le parcours en profondeur)

Voir le document *AlgosGraphes.pdf* page 2.

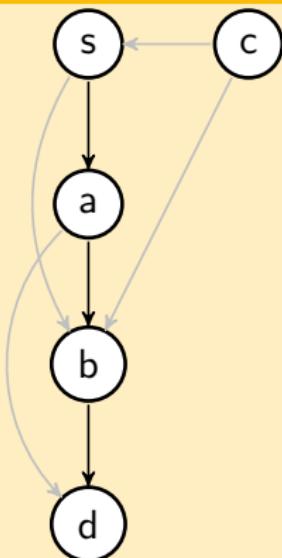
Complexité

Il est facile de constater que la complexité du tri topologique basé sur le parcours en profondeur est identique à celle du parcours en profondeur.

Donc le tri topologique est en $O(n + m)$

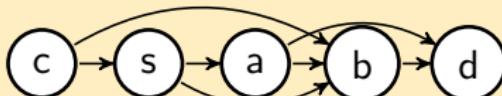
Tri topologique

Exemple



v	s	a	b	c	d
$d(v)$	1	2	3	9	4
$f(v)$	8	7	6	10	5

Tri topologique : c, s, a, b, d



Tri topologique

Justification de l'algorithme

Théorème : un graphe G orienté est sans circuit si et seulement si le parcours en profondeur de G ne produit aucun arc de retour.

Preuve : on va en fait montrer que G possède un circuit si et seulement si son parcours en profondeur produit un arc retour.

Si lors du parcours, on a un arc retour uv , alors le graphe G possède un circuit constitué des arcs de liaisons reliant v à u complété de l'arc uv .

Si G possède un circuit $C = v_1, \dots, v_k, v_1$, alors supposons (sans perte de généralité) que v_1 soit le premier sommet de C visité.

D'après le théorème du chemin blanc, v_k sera un descendant de v_1 et donc $v_k v_1$ un arc retour.

Tri topologique

Justification de l'algorithme

Montrons maintenant que l'ordre inverse de fin de visite de $PP(G)$ est bien un tri topologique de G . Il faut donc montrer que

$$uv \in E(G) \Rightarrow f(u) > f(v)$$

En effet, soit uv un arc de G .

Si uv est un arc de liaison ou avant, alors v est un descendant de u et donc $f(v) < f(u)$ car $[d(v), f(v)] \subset [d(u), f(u)]$.

Si uv est un arc transverse, alors $d(v) < f(v) < d(u) < f(u)$.

Donc dans tous les cas, on aura bien $uv \in E(G) \Rightarrow f(u) > f(v)$.

Composantes fortement connexes

Énoncé de l'algorithme de Kosaraju

Voir le document *AlgosGraphes.pdf* page 2.

Complexité

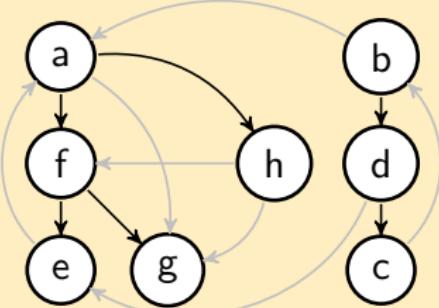
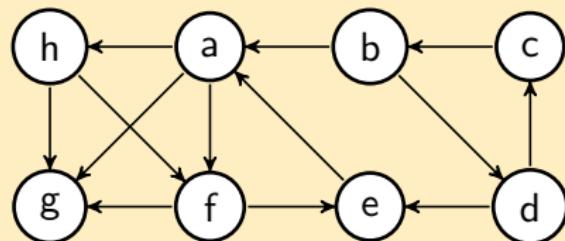
L'algorithme exécute deux parcours en profondeur, soit une complexité de $O(n + m)$. De plus, le calcul du graphe inversé G^{-1} est également en $O(n + m)$ en utilisant par exemple l'algorithme suivant où $Adj_G(v)$ désigne la liste des successeurs du sommet v dans le graphe G .

Inverser(G) :

- 1: $G^{-1} \leftarrow (V(G), \emptyset)$
- 2: **pour tout** $v \in V(G)$ **faire**
- 3: **pour tout** $w \in Adj_G(v)$ **faire**
- 4: INSERER($v, Adj_{G^{-1}}(w)$)
- 5: **fin pour**
- 6: **fin pour**
- 7: **retourner** G^{-1}

Composantes fortement connexes

Exemple



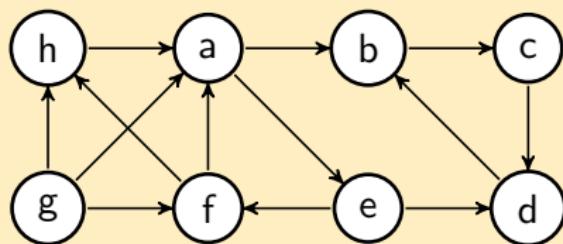
$v :$	a	b	c	d	e	f	g	h
$d(v)$	1	11	13	12	3	2	5	8
$f(v)$	10	16	14	15	4	7	6	9

Ordre : b, d, c, a, h, f, g, e

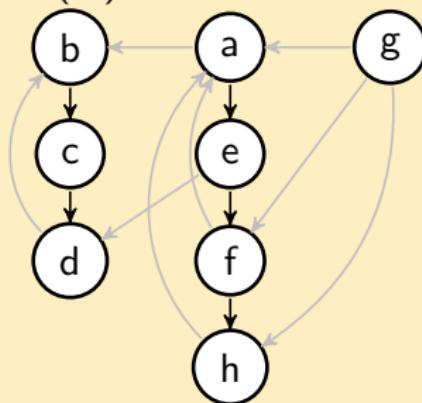
Composantes fortement connexes

Exemple (Suite)

On calcule d'abord le graphe inverse G^{-1} , puis on exécute $PP(G^{-1})$ en traitant dans la boucle 6 de $PP(G^{-1})$ les sommets dans l'ordre inverse de f obtenu lors de $PP(G)$.



Ordre : b, d, c, a, h, f, g, e



CFC : $\{b, c, d\}$, $\{a, e, f, h\}$, $\{g\}$.

Composantes fortement connexes

Justification

Lemme 1 : si deux sommets u et v sont dans une même composante fortement connexe, alors aucun chemin entre eux n'utilise de sommet extérieur à la composante.

Preuve : soit C la composante fortement connexe contenant u et v . Soit w un sommet situé sur un chemin P de u à v .

$P = P_{uw}|P_{wv}$ où P_{uw} est un chemin de u à w et P_{wv} un chemin de w à v . D'autre part, comme $v \in C$, il existe un chemin P_{vu} de v à u . Donc il existe un chemin de u à w : P_{uw} et un chemin de w à u : $P_{wv}|P_{vu}$. Donc $w \in C$.

Composantes fortement connexes

Justification

Lemme 2 : lors d'un parcours en profondeur, tous les sommets d'une même composante fortement connexe se trouveront dans la même arborescence.

Preuve : soit C une composante fortement connexe et soit u le premier sommet de C visité lors d'un parcours en profondeur. Lors de la visite de u , tous les autres sommets de C sont blancs et donc pour tout sommet $v \in C$, il existe un chemin de u à v dont tous les sommets sont blancs. D'après le théorème du chemin blanc, v sera un descendant de u . Et donc tous les sommets de C seront dans la même arborescence que u .

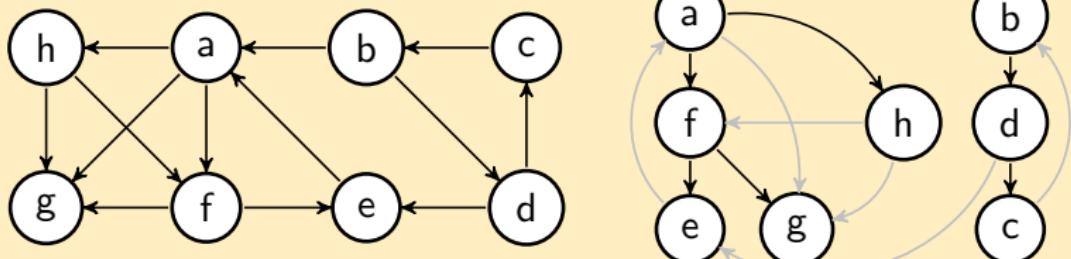
Composantes fortement connexes

Justification

Lors d'un parcours en profondeur d'un graphe G , on appelle **aïeul** d'un sommet u , le sommet noté $\phi(u)$ qui est parmi les sommets v accessibles à partir de u celui qui a la plus grande valeur pour f .
Noter que l'on peut avoir $\phi(u) = u$.

Composantes fortement connexes

Exemple de calcul de ϕ



$v :$	a	b	c	d	e	f	g	h
$d(v)$	1	11	13	12	3	2	5	8
$f(v)$	10	16	14	15	4	7	6	9
$\phi(v)$	a	b	b	b	a	a	g	a

Composantes fortement connexes

Justification

Lemme 3 : si deux sommets u et v sont tels qu'il existe un chemin de u à v , alors pour tout parcours en profondeur, on aura $f(\phi(u)) \geq f(\phi(v))$.

Preuve : comme il existe un chemin de u à v et un chemin de v à $\phi(v)$, alors il existe un chemin de u à $\phi(v)$ et donc $f(\phi(u)) \geq f(\phi(v))$.

Justification

Lemme 4 : pour tout sommet u , $\phi(\phi(u)) = \phi(u)$.

Preuve : il existe un chemin de u à $\phi(u)$. Donc d'après le Lemme 3, $f(\phi(u)) \geq f(\phi(\phi(u)))$. Or par définition, $f(\phi(\phi(u))) \geq f(\phi(u))$. Donc $f(\phi(\phi(u))) = f(\phi(u))$ et donc $\phi(\phi(u)) = \phi(u)$.

Composantes fortement connexes

Justification

Lemme 5 : pour tout sommet u , lors d'un parcours en profondeur, $\phi(u)$ est un ancêtre de u .

Preuve :

Il faut montrer que lorsqu'on visite un sommet u pour la première fois, $\phi(u)$ est gris.

- $\phi(u)$ ne peut être noir, sinon on aurait $f(\phi(u)) < d(u)$ et donc $f(\phi(u)) < f(u)$. Ce qui est impossible par définition de ϕ .

Composantes fortement connexes

Justification

Preuve du Lemme 5 (suite et fin) :

- ▶ Montrons que $\phi(u)$ ne peut être blanc.

Si lors de la visite de u , tous les sommets du chemin entre u et $\phi(u)$ sont blancs, alors $\phi(u)$ sera un descendant de u et on aura $f(\phi(u)) < f(u)$, ce qui est impossible.

S'il existe des sommets non blancs sur le chemin de u à $\phi(u)$, soit v le sommet non blanc sur ce chemin le plus proche de $\phi(u)$. Donc tous les sommets entre v et $\phi(u)$ seront blancs (v non compris). Comme v a un successeur blanc, v ne peut être noir et donc v est gris. Or il existe un chemin blanc de v à $\phi(u)$ et donc $\phi(u)$ sera un descendant de v et donc $f(\phi(u)) < f(v)$. Or v est accessible par un chemin depuis u et $f(v) > f(\phi(u))$, ce qui est contradictoire avec la définition de ϕ .

Composantes fortement connexes

Justification

Lemme 6 : pour tout sommet u , u et $\phi(u)$ appartiennent à la même composante fortement connexe.

Preuve : on a montré dans le Lemme 5 que pour tout sommet u , $\phi(u)$ est un ancêtre de u . Donc il existe un chemin de $\phi(u)$ à u . Par définition de ϕ , il existe également un chemin de u à $\phi(u)$. Donc u et $\phi(u)$ appartiennent à la même composante fortement connexe.

Composantes fortement connexes

Justification

Lemme 7 : deux sommets u et v appartiennent à une même composante fortement connexe si et seulement si ils ont le même aïeul.

Preuve : soient u et v deux sommets appartenant à la même composante fortement connexe C . D'après le Lemme 6, $\phi(u)$ et $\phi(v)$ appartiendront également à C . Donc $\phi(u)$ et $\phi(v)$ sont tous les deux accessibles depuis u et v et donc par définition de ϕ $f(\phi(u)) \geq f(\phi(v))$ car $\phi(u)$ est l'aïeul de u et $f(\phi(v)) \geq f(\phi(u))$ car $\phi(v)$ est l'aïeul de v . Donc $f(\phi(v)) = f(\phi(u))$ et donc $\phi(v) = \phi(u)$.

Inversement, si deux sommets u et v ont le même aïeul, alors ils appartiennent tous les deux à la composante fortement connexe de cet aïeul et donc à la même composante fortement connexe.

Composantes fortement connexes

Justification

Lemme 8 : soit T une arborescence obtenue lors du parcours en profondeur de G^{-1} de racine r . Pour tout sommet u , si $\phi(u)$ désigne l'aïeul de u lors du parcours en profondeur de G (étape 1), alors $\phi(u) = r$ si et seulement si u appartient à T .

Preuve :

r_1 est le sommet ayant la plus grande valeur de f lors de $PP(G)$. Donc il sera l'aïeul de tous les sommets de sa composante fortement connexe, disons C_1 . De plus, lors du début de la visite de r_1 dans de $PP(G^{-1})$, tous les sommets de C_1 seront blancs et donc seront des descendants de r_1 .

De plus, si un sommet v ne possède pas r_1 comme aïeul, cela signifie qu'il n'existe pas de chemin de v à r_1 dans G et donc pas de chemin de r_1 à v dans G^{-1} . Donc v ne sera pas dans l'arborescence de r_1 .

Composantes fortement connexes

Justification

Suite de preuve du Lemme 8 : supposons que l'hypothèse soit vérifiée pour les $k - 1$ premières arborescence. Soit T_k l'arborescence suivante de racine r_k , C_k la composante fortement connexe de r_k dans G et $\phi(C_k)$ l'aïeul des sommets de C_k obtenu lors de l'étape 1.

Comme $\phi(C_k)$ est le sommet ayant la plus grande valeur de f parmi les sommets non encore visités, on aura $r_k = \phi(C_k)$.

Si un sommet $v \in C_k$, v est blanc lors de l'examen de r_k et sera donc ajouté à T_k . De même, soit un sommet $w \notin C_1 \cup \dots \cup C_k$. Comme il n'existe pas de chemin de w à $\phi(C_k)$ dans G , w ne sera pas ajouté à T_k . Donc T_k contiendra exactement les sommets de C_k .

Par induction, le Lemme 8 est donc vrai et justifie l'algorithme CFC.