

Algorithmique des graphes

Cours 4 – Parcours en profondeur

František Kardoš

`frantisek.kardos@u-bordeaux.fr`

Exercices du cours précédent

Comment définir l'isomorphisme pour des graphes pas forcément simples ?

Exercices du cours précédent

Comment définir l'isomorphisme pour des graphes pas forcément simples ?

Comment trouver dans une liste d'entiers positifs une tranche d'une somme donnée en temps linéaire ?

Parcours en profondeur : le principe

Exploration d'un graphe donné par un agent mobile. Il peut se déplacer d'un sommet au voisin en suivant une arête les reliant.

Parcours en profondeur : le principe

Exploration d'un graphe donné par un agent mobile. Il peut se déplacer d'un sommet au voisin en suivant une arête les reliant.

L'agent marque des sommets pour savoir s'il les a déjà découverts. Un sommet est marqué dès que l'agent le découvre (pour la première fois).

Initialement, aucun sommet n'est marqué.

Parcours en profondeur : le principe

Pour décider par quelle arête il va partir d'un sommet, il privilégie toute autre arête à celle qui lui a servi à découvrir le sommet actuel.

Parcours en profondeur : le principe

Pour décider par quelle arête il va partir d'un sommet, il privilégie toute autre arête à celle qui lui a servi à découvrir le sommet actuel.

S'il retrouve un sommet déjà découvert en franchissant une arête, il fait un demi-tour immédiatement.

Parcours en profondeur : le principe

Pour décider par quelle arête il va partir d'un sommet, il privilégie toute autre arête à celle qui lui a servi à découvrir le sommet actuel.

S'il retrouve un sommet déjà découvert en franchissant une arête, il fait un demi-tour immédiatement.

Il déclare un sommet u *visité* s'il a déjà exploré tout ce qui était accessible et non-découvert à partir de u au moment où celui-ci a été découvert. Il termine la visite de u par le retour par l'arête de découverte de u .

Parcours en profondeur : l'algorithme brut

Algorithme 1 : Parcours en profondeur DFS(G)

Données : graphe G , marque des sommets (initialisé à Faux),

père π des sommets (initialisée à null)

début

pour chaque s *sommet de G* **faire**

si s *non marqué* **alors**

 Visiter(s) ;

fin

fin

fin

Algorithme 2 : fonction récursive Visiter(u)

début

 marque[u] \leftarrow Vrai ;

pour chaque v *voisin de u* **faire**

si v *non marqué* **alors**

$\pi[v] \leftarrow u$;

 Visiter(v) ;

fin

fin

fin

Parcours en profondeur

- ▶ Tout sommet du graphe est visité (et marqué) une et une seule fois
- ▶ À la fin, tous les sommets sont marqués
- ▶ Chaque sommet a au maximum un père
- ▶ Certaines arêtes sont des *arêtes de liaison* – celles reliant un sommet à son père
- ▶ Le graphe composé des arêtes de liaison est sans cycle

Parcours en profondeur

- ▶ Une fois le sommet de départ s choisi, l'algorithme visite tous les sommets accessibles depuis s
- ▶ Il est impossible qu'il reste une arête (un arc) reliant un sommet marqué à un autre sommet non marqué
- ▶ L'algorithme utilise implicitement une pile (LIFO) pour stocker des appels récursifs à la fonction $\text{Visiter}(u)$
- ▶ À n'importe quel moment du déroulement de l'algorithme, en tête de la pile est le sommet actuel, précédé par la suite de ses ancêtres

Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

C : G

D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E

Pile :

Parcours en profondeur – graphes non-orientés

A•

A : B, G

B : A, E, G

C : G

D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E

Pile : A

Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

C : G

D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E



Pile : A, B

Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

C : G

D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E



Pile : A, B

Parcours en profondeur – graphes non-orientés

A : B , G

B : A , E , G

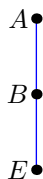
C : G

D : E , F

E : B , D , F , G

F : D , E

G : A , B , C , E



Pile : A, B, E

Parcours en profondeur – graphes non-orientés

A : B , G

B : A , E , G

C : G

D : E , F

E : B , D , F , G

F : D , E

G : A , B , C , E



Pile : A, B, E

Parcours en profondeur – graphes non-orientés

A : B , G

B : A , E , G

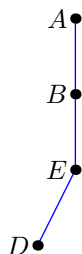
C : G

D : E , F

E : B , D , F , G

F : D , E

G : A , B , C , E



Pile : A, B, E, D

Parcours en profondeur – graphes non-orientés

A : B , G

B : A , E , G

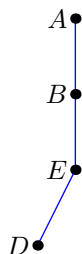
C : G

D : E , F

E : B , D , F , G

F : D , E

G : A , B , C , E



Pile : A, B, E, D

Parcours en profondeur – graphes non-orientés

A : B , G

B : A , E , G

C : G

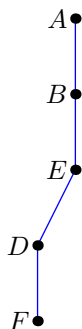
D : E , F

E : B , D , F , G

F : D , E

G : A , B , C , E

Pile : A, B, E, D, F



Parcours en profondeur – graphes non-orientés

A : B , G

B : A , E , G

C : G

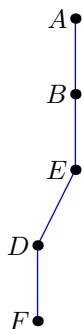
D : E , F

E : B , D , F , G

F : D , E

G : A , B , C , E

Pile : A, B, E, D, F



Parcours en profondeur – graphes non-orientés

A : B , G

B : A , E , G

C : G

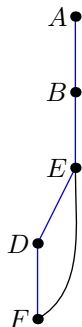
D : E , F

E : B , D , F , G

F : D , E

G : A , B , C , E

Pile : A , B , E , D , F



Parcours en profondeur – graphes non-orientés

A : B , G

B : A , E , G

C : G

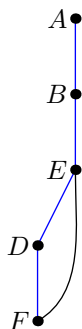
D : E , F

E : B , D , F , G

F : D , E

G : A , B , C , E

Pile : A , B , E , D



Parcours en profondeur – graphes non-orientés

A : B , G

B : A , E , G

C : G

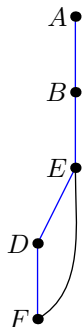
D : E , F

E : B , D , F , G

F : D , E

G : A , B , C , E

Pile : A , B , E



Parcours en profondeur – graphes non-orientés

A : B , G

B : A , E , G

C : G

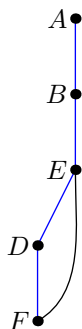
D : E , F

E : B , D , F , G

F : D , E

G : A , B , C , E

Pile : A, B, E



Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

C : G

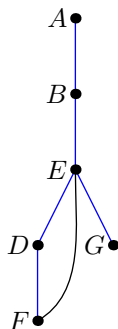
D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E

Pile : A, B, E, G



Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

C : G

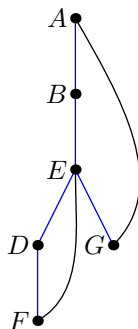
D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E

Pile : A, B, E, G



Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

C : G

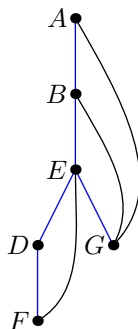
D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E

Pile : A, B, E, G



Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

C : G

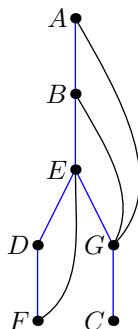
D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E

Pile : A, B, E, G, C



Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

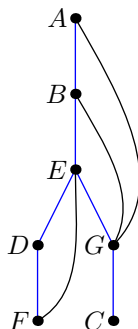
C : G

D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E



Pile : A, B, E, G, C

Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

C : G

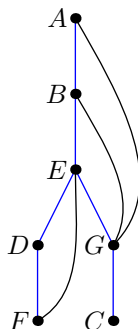
D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E

Pile : A, B, E, G



Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

C : G

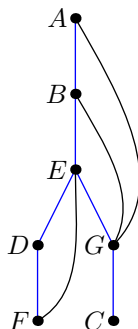
D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E

Pile : A, B, E, G



Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

C : G

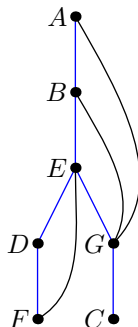
D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E

Pile : A, B, E



Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

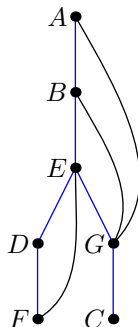
C : G

D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E



Pile : A, B

Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

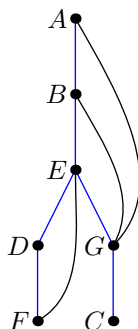
C : G

D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E



Pile : A, B

Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

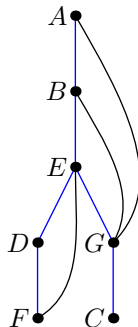
C : G

D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E



Pile : A

Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

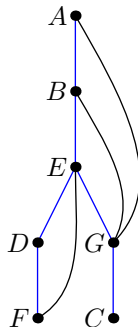
C : G

D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E



Pile : A

Parcours en profondeur – graphes non-orientés

A : B, G

B : A, E, G

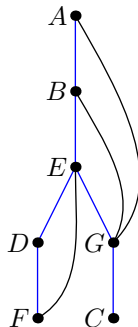
C : G

D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E



Pile :

Parcours en profondeur – graphes non-orientés

- ▶ Toute arête qui n'est pas une arête de liaison relie un sommet à un de ses ancêtres – c'est une *arête de retour*
- ▶ Chaque arête de retour est découverte deux fois, mais elle ne nous donne pas de nouvelle information sur les sommets du graphe

Parcours en profondeur – graphes non-orientés

- ▶ Toute arête qui n'est pas une arête de liaison relie un sommet à un de ses ancêtres – c'est une *arête de retour*
- ▶ Chaque arête de retour est découverte deux fois, mais elle ne nous donne pas de nouvelle information sur les sommets du graphe

Exercice : Transformer l'algorithme DFS(G) en une version non-réursive, en explicitant la pile.

Parcours en profondeur : début et fin de visite

Algorithme 3 : DFS(G)

Données : graphe G

début

pour chaque s sommet de

G faire

 couleur[s] \leftarrow BLANC ;

π [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 4 : 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

π [v] \leftarrow u ;

 Visiter(v) ;

fin

fin

couleur[u] \leftarrow NOIR ;

f [u] \leftarrow temps \leftarrow temps + 1 ;

fin

Parcours en profondeur : début et fin de visite

A : B, G

B : A, E, G

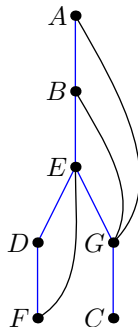
C : G

D : E, F

E : B, D, F, G

F : D, E

G : A, B, C, E



Pile :

Parcours en profondeur – graphes orientés

Parcours en profondeur – graphes orientés

Il existe plusieurs types d'arcs :

- ▶ *arc de liaison* relie un sommet et son père (dans le sens père \rightarrow fils)

Parcours en profondeur – graphes orientés

Il existe plusieurs types d'arcs :

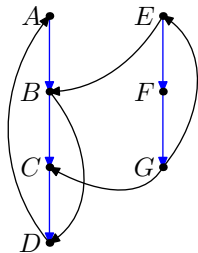
- ▶ *arc de liaison* relie un sommet et son père (dans le sens père \rightarrow fils)
- ▶ *arc de retour* relie un sommet à son ancêtre (dans le sens descendant \rightarrow ancêtre) – c'est un arc qui permet de remonter dans l'arbre de liaison
- ▶ *arc d'avant* relie un sommet à son descendant autre que fils (dans le sens ancêtre \rightarrow descendant) – c'est un arc qui permet de redescendre dans l'arbre de liaison

Parcours en profondeur – graphes orientés

Il existe plusieurs types d'arcs :

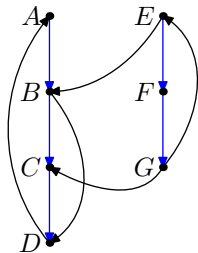
- ▶ *arc de liaison* relie un sommet et son père (dans le sens père \rightarrow fils)
- ▶ *arc de retour* relie un sommet à son ancêtre (dans le sens descendant \rightarrow ancêtre) – c'est un arc qui permet de remonter dans l'arbre de liaison
- ▶ *arc d'avant* relie un sommet à son descendant autre que fils (dans le sens ancêtre \rightarrow descendant) – c'est un arc qui permet de redescendre dans l'arbre de liaison
- ▶ *arc transverse* relie un sommet à tout autre sommet dont la visite a bien été déjà terminée

Parcours en profondeur – graphes orientés



arcs de liaison

Parcours en profondeur – graphes orientés



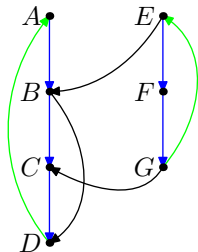
arcs de liaison

arcs de retour

arcs d'avant

arcs transverses

Parcours en profondeur – graphes orientés



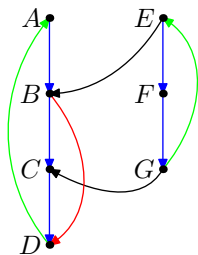
arcs de liaison

arcs de retour

arcs d'avant

arcs transverses

Parcours en profondeur – graphes orientés



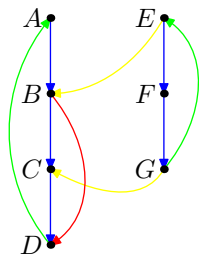
arcs de liaison

arcs de retour

arcs d'avant

arcs transverses

Parcours en profondeur – graphes orientés



arcs de liaison

arcs de retour

arcs d'avant

arcs transverses

Parcours en profondeur : Applications

On peut utiliser un parcours en profondeur dans un graphe orienté pour détecter la présence de circuits :

- ▶ Dans un graphe orienté, chaque arc de retour témoigne d'un circuit
- ▶ Pour chaque circuit, il y a au moins un arc de retour

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.

