Parcours, parcours en largeur

Alix Munier-Kordon et Maryse Pelletier

LIP6
Sorbonne Université
Paris

LU2IN003 Initiation à l'algorithmique

Parcours d'un graphe

1. Le parcours d'un graphe consiste à visiter un à un tous ses sommets dans un certain ordre en passant par les arêtes (ou les arcs).
2. La notion de parcours peut s'appliquer à un graphe orienté ou non.
3. Les algorithmes de parcours sont nombreux, et sont utilisés pour étudier les graphes. Ils permettent par exemple de répondre efficacement aux questions suivantes :
   - Un graphe non orienté est-il connexe ?
   - Un graphe orienté possède t'il un circuit ?
   - Quelles sont les distances minimales (en nombre d'arêtes) de tout sommet à une origine s?

Plan du cours

1. Parcours générique d'un graphe non orienté
2. Parcours générique d'un graphe orienté
3. Parcours en largeur d'un graphe non orienté connexe

Parcours générique d'un graphe non orienté

Soit $G = (V, E)$ un graphe non orienté et un sommet $s \in V$.

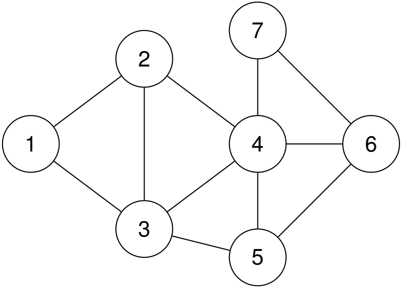
Definition (Parcours d'un graphe non orienté)

Un parcours des sommets de G d'origine s est une liste L des sommets de G telle que :

1. s est le premier sommet de L,
2. chaque sommet apparait exactement une fois dans L,
3. tout sommet $u \in V$ sauf l'origine s est adjacent à au moins un sommet placé avant lui dans la liste.

Est-ce que tout graphe non orienté possède au moins un parcours ?

Parcours d'un graphe non orienté



1. $L = (3, 5, 4, 7, 1, 2, 6)$ est un parcours d'origine 3;
2. $L = (3, 5, 7, 6, 4, 2, 1)$ n'est pas un parcours car 7 n'est pas adjacent à un sommet de $\{3, 5\}$.

Sous-parcours

Definition (sous-parcours)

Un sous-parcours d'origine v_1 d'un graphe orienté $G = (V, E)$ est une sous-liste $L = (v_1, \ldots, v_k)$ pour $k \geq 1$ telle que

1. v_1 est le premier sommet de L,
2. chaque sommet apparait au plus une fois dans L,
3. tout sommet v_α de L sauf l'origine v_1 est adjacent à au moins un sommet placé avant lui dans la liste L.

Definition (sommet visité)

Un sommet $v \in V$ est visité par le sous-parcours L si v apparait dans L. On note $V(L)$ l'ensemble des sommets visités par L.

$L = (2, 4, 5)$ est un sous-parcours d'origine 2. L'ensemble des sommets visités par L est $V(L) = \{2, 4, 5\}$.

Bordure d'un sous-parcours

Soit $G = (V, E)$ un graphe non orienté et L un sous-parcours de G.

Definition (Bordure)

La bordure d'un sous-parcours L de $G = (V, E)$ est l'ensemble des sommets de V qui ne sont pas visités par L et adjacents à un sommet visité par L.

$$\mathcal{B}(L) = \{v \in V - V(L), \exists e = \{u, v\} \in E, \text{ avec } u \in V(L)\}$$

- Pour le sous-parcours $L = (2, 4, 5)$ d'origine 2, $\mathcal{B}(L) = \{1, 3, 6, 7\}$;
- Pour le sous-parcours $L = (7, 6)$ d'origine 7, $\mathcal{B}(L) = \{4, 5\}$.

Que vaut la bordure d'un parcours ?

Racine et Arborescence

Definition (Racine d'un graphe orienté)

Soit $G = (V, A)$ un graphe orienté. Une racine de G est un sommet $s \in V$ tel que, pour tout $u \in V - \{s\}$, il existe un chemin de s à u.

Definition (Arborescence)

Une arborescence $\mathcal{A} = (V, A)$ est un graphe orienté tel que :

- le graphe non orienté associé obtenu en enlevant l'orientation des arcs de $\mathcal{A}$ est un arbre;
- $\mathcal{A}$ possède une unique racine r.

Est-ce que tout graphe orienté connexe contient une racine ?

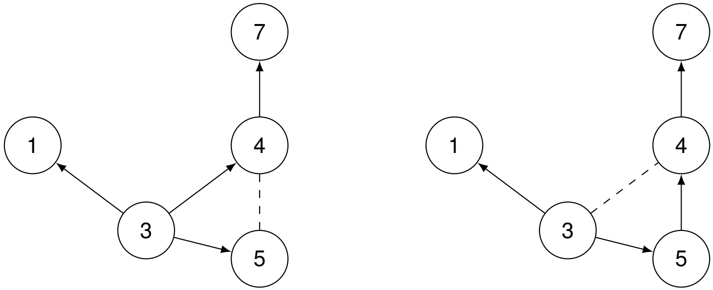
Graphe de liaison associé à un sous-parcours

Soit $G = (V, E)$ un graphe non orienté et L un sous-parcours d'origine $s \in V$.

Definition (Graphe de liaison associé à un sous-parcours)

$\mathcal{A}(L) = (V(L), H(L))$ est *un graphe de liaison associé au sous-parcours* L si tout sommet $v \in V(L) - \{s\}$ a pour unique prédécesseur un sommet $u \in V(L)$ tel que u est visité avant v dans L et $\{u, v\} \in E$.

Graphe de liaison associé à un sous-parcours (Example)



Deux graphes de liaison associés au sous-parcours
$L = (3, 5, 4, 7, 1)$

Graphe de liaison associé à un sous-parcours

Soit $G = (V, E)$ un graphe non orienté.

Theorem

A tout sous-parcours L de G, on peut associer un graphe de liaison $\mathcal{A}(L) = (V(L), H(L))$.

Se démontre par récurrence faible sur $|V(L)|$.

Theorem

Soit $G = (V, E)$ un graphe non orienté et L un parcours d'origine $s \in V$. Alors le graphe de liaison associée au parcours L est une arborescence de racine s.

A faire en TD.

Pour un sous-parcours L fixé, $\mathcal{A}(L)$ n'est pas unique (voir le transparent précédent).

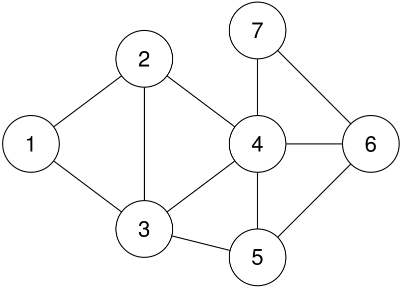
Algorithme de construction d'un parcours

Algorithm 1 Calcul d'un parcours associé à un graphe non orienté $G = (V, E)$

Require: Un graphe non orienté $G = (V, E)$, un sommet s
Ensure: Un parcours L d'origine s

$L := (s)$, $\mathcal{B} := \mathcal{B}(L)$
while $\mathcal{B} \neq \emptyset$ **do**
  Choisir un sommet $u \in \mathcal{B}$
  $L := L + (u)$
  $\mathcal{B} := \mathcal{B}(L)$
end while

Exemple d'exécution de l'algorithme



u	L	$\mathcal{B}$
$\star$	(3)	$\{1,2,4,5\}$
5	$(3,5)$	$\{1,2,4,6\}$
4	$(3,5,4)$	$\{1,2,6,7\}$
7	$(3,5,4,7)$	$\{1,2,6\}$
1	$(3,5,4,7,1)$	$\{2,6\}$
2	$(3,5,4,7,1,2)$	$\{6\}$
6	$(3,5,4,7,1,2,6)$	$\emptyset$

Terminaison de l'algorithme de construction d'un parcours

Lemma

Pour tout graphe non orienté $G = (V, E)$ et tout sommet $s \in V$, l'algorithme effectue au plus $n - 1$ itérations.

Theorem

Pour tout graphe non orienté $G = (V, E)$ et tout sommet $s \in V$, l'algorithme se termine.

Validité de l'algorithme de construction d'un parcours

Theorem

Pour tout graphe non orienté $G = (V, E)$ et tout sommet $s \in V$, l'algorithme construit un sous-parcours L de G de racine s et composé de tous les sommets de la composante connexe de s.

Corollaire

Un graphe non orienté $G = (V, E)$ est connexe si et seulement si l'algorithme construit un parcours en partant d'un sommet quelqconque.

Parcours d'un graphe orienté

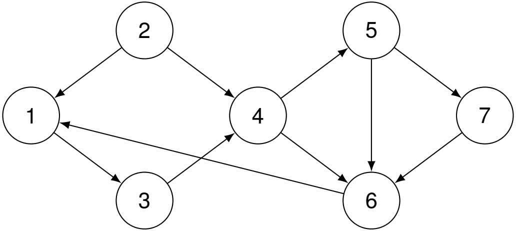
Soit $G = (V, A)$ un graphe orienté et un sommet $s \in V$.

Definition (Parcours d'un graphe orienté)

Un parcours des sommets de G d'origine s est une liste L des sommets de G telle que :

1. s est le premier sommet de L,
2. chaque sommet apparait exactement une fois dans L,
3. tout sommet $u \in V$ sauf l'origine s est successeur d'un sommet v placé avant dans la liste (*i.e* $(v, u) \in A$).

Parcours d'un graphe orienté



1. 2 est une racine de G;
2. $L = (2, 4, 5, 1, 6, 3, 7)$ est un parcours d'origine 2;
3. $L = (2, 1, 3, 6, 4, 5, 7)$ n'est pas un parcours car 6 n'est pas le successeur d'un sommet de $\{1, 2, 3\}$.

A votre avis, à quelle condition G possède un parcours d'origine s ?

Problème de la plus courte chaîne

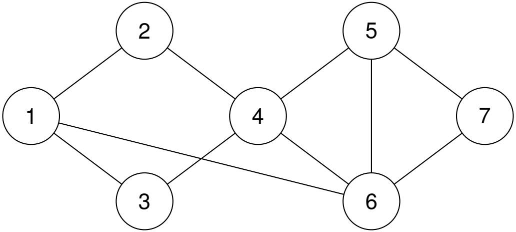
Definition (distance d'une chaîne)

La distance $dist(\mu)$ d'une chaîne μ est le nombre d'arêtes qui la composent.

Definition (Problème de la plus courte chaîne)

Soit $G = (V, E)$ un graphe non orienté connexe et un sommet $s \in V$. Le problème de la plus courte chaîne consiste à calculer, pour tout sommet $u \in V$, le nombre minimum d'arêtes d'une chaîne de s à u et noté $dist_s(u)$.

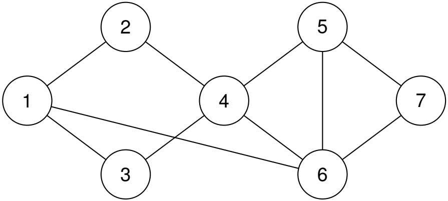
Problème de la plus courte chaîne

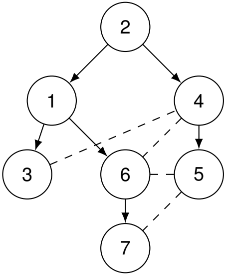


$u \in V$	1	2	3	4	5	6	7
$dist_2(u)$	1	0	2	1	2	2	3

Principe général du parcours en largeur

Le parcours en largeur d'origine s est un parcours qui visite les sommets niveau par niveau: d'abord tous les sommets à distance 1 de s, puis à distance 2, puis 3..etc..
(ce n'est pas une définition)





$L = (2, 1, 4, 3, 6, 5, 7)$ est un parcours en largeur d'origine 2.

Sommets ouverts, fermés

Definition (sommet ouvert, fermé)

Soit L un sous-parcours. Un sommet visité $u \in V(L)$ est ouvert si il possède au moins un sommet adjacent qui n'est pas dans L. Un sommet visité $u \in V(L)$ est fermé si tous ses adjacents sont dans $V(L)$.

Pour le sous-parcours $L = (2, 1, 3, 6)$ du graphe précédent,

- 2 est ouvert car $4 \notin V(L)$;
- 1 est fermé car tous ses sommets adjacents sont dans $V(L)$.

Parcours en largeur

Definition (Parcours en largeur)

Soit $G = (V, E)$ un graphe non orienté connexe et $L = (v_1, \dots, v_n)$ un parcours de G d'origine v_1.
L est un parcours en largeur si pour tout sous-parcours $L_k = (v_1, \dots, v_k)$ avec $k < n$, v_{k+1} est un sommet adjacent du premier sommet ouvert de L_k.

Pour le sous-parcours en largeur $L = (2, 1, 4, 3, 6, 5)$ du graphe précédent,

- Pour $L_4 = (2, 1, 4, 3)$, 2 est fermé, donc le premier sommet ouvert est 1 et 6 est un adjacent de 1.
- Pour $L_5 = (2, 1, 4, 3, 6)$, 2 et 1 sont fermés, donc le premier sommet ouvert est 4 et 5 est un adjacent de 4.

Pour une origine s fixé, est-ce que un parcours en largeur est unique ?

Graphe de liaison en largeur

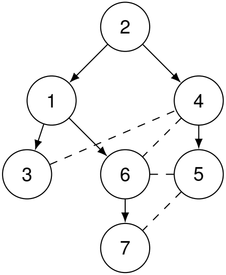
Definition (Graphe de liaison en largeur)

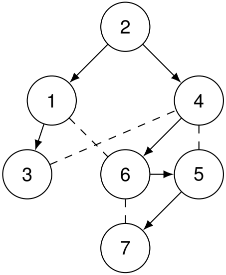
Soit $G = (V, E)$ un graphe non orienté connexe et $L = (v_1, \ldots, v_n)$ un parcours en largeur de G d'origine v_1.
Le graphe orienté $\mathcal{A}^\star(L) = (V(L), H(L))$ est le graphe de liaison en largeur de L si :

- $\mathcal{A}^\star(L) = (V(L), H(L))$ est un graphe de liaison de L;
- Pour tout sous-parcours $L_k = (v_1, \ldots, v_k)$ avec $k < n$, v_{k+1} a pour prédécesseur le plus petit sommet ouvert de L_k.

A votre avis, est-ce que le graphe de liaison en largeur $\mathcal{A}^\star(L)$ associé au parcours en largeur L est unique ?

Graphe de liaison en largeur





$L = (2, 1, 4, 3, 6, 5, 7)$ est un parcours en largeur d'origine 2.

- A gauche, le graphe de liaison en largeur $\mathcal{A}^\star(L)$ de L;
- A droite un graphe de liaison $\mathcal{A}(L)$ associé à L qui n'est pas le graphe de liaison en largeur de L.

Propriété sur les plus courtes chaînes

Soit $G = (V, E)$ un graphe non orienté connexe, L un parcours en largeur de G d'origine s et $\mathcal{A}^\star(L)$ le graphe de liaison en largeur de L.

Theorem

Pour tout sommet $u \in V$, le chemin de s à u de $\mathcal{A}^\star(L)$ est associé à une plus courte chaîne de G entre s et u.

Non démontré dans ce cours

Corollaire

Pour tout sommet $u \in V$, $dist_s(u)$ est égale à la distance du chemin de s à u de $\mathcal{A}^\star(L)$.

Est-ce que ce théorème est vérifié pour tout graphe de liaison $\mathcal{A}(L)$ d'un parcours en largeur L ?

Algorithme de construction d'un parcours en largeur

Definition (File)

Une file est une structure de données telle que les premiers éléments ajoutés à la file seront les premiers à en être retirés (first in, first out).

Les primitives minimales généralement associées sont :

- Enfiler (F,x) qui stocke un elément dans la file F;
- Défiler(F) qui retourne l'élément en tête de la file F
- FileVide(F) qui est vraie si la file F est vide.

Pouvez-vous proposer des structures de données simples pour implanter une file ?

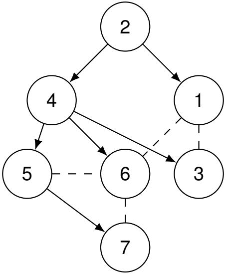
Algorithme de construction d'un parcours en largeur

Require: Un graphe non orienté $G = (V, E)$, un sommet s
Ensure: Un parcours en largeur L d'origine s, les valeurs
  $dist_s(u), u \in V$
  for all $u \in V$ **do**
    $dist_s(u) := +\infty$
  end for
  $L := ()$, Enfiler (F,s), $dist_s(s) := 0$
  while not FileVide(F) **do**
    $u :=$Défiler(F), $L := L + (u)$
    for all $\{u, v\} \in E$ **do**
      if $dist_s(v) = +\infty$ **then**
        Enfiler(F,v), $dist_s(v) = dist_s(u) + 1$
      end if
    end for
  end while

Algorithme de construction d'un parcours en largeur

- Au démarrage de l'algorithme, les distances sont initialisées à l'infini, sauf celle de l'origine s.
- La file permet de gérer la bordure du sous-parcours en largeur. La gestion de la priorité assure que l'on choisit tout le temps un adjacent du premier sommet ouvert du sous-parcours.
- Le test sur la distance permet aussi de s'assurer que tout sommet placé dans la file n'est ni dans $V(L)$ ni dans F.

Exemple d'exécution de l'algorithme de construction d'un parcours en largeur



$L = (2, 4, 1, 5, 6, 3, 7)$

u	L	F
$\star$	$()$	(2)
2	(2)	$(4, 1)$
4	$(2, 4)$	$(1, 5, 6, 3)$
1	$(2, 4, 1)$	$(5, 6, 3)$
5	$(2, 4, 1, 5)$	$(6, 3, 7)$
6	$(2, 4, 1, 5, 6)$	$(3, 7)$
3	$(2, 4, 1, 5, 6, 3)$	(7)
7	$(2, 4, 1, 5, 6, 3, 7)$	$()$

$u \in V$	1	2	3	4	5	6	7
$dist_2(u)$	1	0	2	1	2	2	3

Conclusion

- Les parcours constituent une classe d'algorithmes importante sur les graphes qui consiste à visiter un à un tous ses sommets dans un certain ordre en passant par les arêtes (ou les arcs) à partir d'une origine fixé s.
- Tout parcours peut être associé à un graphe de liaison qui est une arborescence.
- Le parcours en largeur est une classe de parcours particulière qui permet d'obtenir les valeurs $dist_s(u), u \in V$.