Notes sur le cours 10 : Parcours, parcours en largeur

Le cours 10 sur les graphes ne peut toujours pas avoir lieu. La première partie de ce cours traite des parcours génériques. On y introduit les notions indispensables pour comprendre la notion de parcours. La seconde partie est consacrée au parcours en largeur d'un graphe non orienté. Je vous suggère de bien comprendre la première partie avant d'attaquer la seconde.

En guise d'introduction à la seconde partie, vous pouvez regarder les vidéos suivantes sur la chaîne YouTube de Christian Laforest :

- « Parcours en largeur »qui présente le principe général du parcours en largeur.
- « Distance dans les graphes, parcours en largeur »qui présente la relation entre parcours en largeur et les plus courtes chaînes.
- « Retour sur le parcours en largeur du graphe »qui étudie une implémentation en utilisant une file. Cette vidéo est plus technique que les deux précédentes. Je vous suggére de la regarder en fin de cours quand vous arrivez à l'algorithme du parcours en largeur. De plus, son implémentation est un peu complexe que celle présentée ici.

Transparent 4 Un graphe non connexe ne possède par de parcours générique.

Transparent 7 Un parcours est une liste qui comporte tous les sommets de V, donc $V(L) = L$ et $\mathcal{B}(L) = \emptyset$.

Transparent 8 Il existe des graphes connexes orientés sans racine : par exemple, $V = (\{1,2,3\},\{(1,2),(3,2)\})$ n'a pas de racine.

Transparent 11 Soit $G = (V,E)$ un graphe non orienté.

Theorem 1. *A tout sous-parcours L de G, on peut associer un graphe de liaison $\mathcal{A}(L) = (V(L),H(L))$*

Démonstration. On démontre par récurrence faible la proposition suivante pour $p \in \{1,\dots,n\}$: $\Pi(p)$: A tout sous-parcours $L = (v_1,\dots,v_p)$ de G de longueur p, on peut associer un graphe de liaison $\mathcal{A}(L) = (V(L),H(L))$ de racine v_1.

Base Pour $p = 1$, $L = (v_1)$ et $\mathcal{A}(L) = (\{v_1\},\emptyset)$ est bien un graphe de liaison racine v_1.

Induction Soit maintenant $p \in \{2,\dots,n\}$ tel que $p-1$ soit vérifiée. Soit $L = (v_1,\dots,v_p)$ un sous-parcours de G de longueur p et $L' = (v_1,\dots,v_{p-1})$. Par hypothèse de récurrence, il existe un graphe de liaison $\mathcal{A}(L') = (V(L'),H(L'))$ pour L'.

Par définition des sous-parcours, v_p a un sommet adjacent dans $\{v_1,\dots,v_{p-1}\}$, donc il existe $\alpha \in \{1,\dots,p-1\}$ avec $\{v_\alpha,v_p\} \in E$. On pose alors $H(L) = H(L') \cup \{(v_\alpha,v_p)\}$ et $\mathcal{A}(L) = (V(L),H(L))$ et on vérifie que $\mathcal{A}(L)$ est un graphe de liaison associé à L.

En effet, soit un sommet $v \in V(L)$. Si $v = v_p$, alors son père v_α a bien été visité avant lui par L par construction. Si $v \neq v_p$, cette proriété est vérifiée par récurrence sur $\mathcal{A}(L')$, donc sur $\mathcal{A}(L)$.

Ainsi, $\mathcal{A}(L)$ est bien un graphe de liaison, et la propriété est vérifiée pour p.

□

Transparent 14

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

Démonstration. Si un sommet v est placé dans $\mathcal{B}(L)$ à un moment donné, alors il sera choisi à une itération ultérieure et placé dans la liste. Par la suite, comme il est alors dans $V(L)$, il ne pourra plus être placé une fois de plus dans la bordure.

Comme il y a une itération par sommet, et que il n'y a pas d'itération associée à la racine s, l'algorithme effectue au plus $n-1$ itérations. □

Theorem 2. *Pour tout graphe non orienté $G = (V,E)$ et tout sommet $s \in V$, l'algorithme se termine.*

Démonstration. D'après le lemme précédent, l'algorithme effectue au plus $n-1$ itérations. Le corps de la boucle consiste en trois instructions qui se terminent. Donc l'algorithme se termine. □

Transparent 15

Theorem 3. *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.*

Démonstration. A la fin de l'itération $i \geq 0$, soient L_i et $\mathcal{B}_i$ les valeurs des variables L et $\mathcal{B}$. On démontre par récurrence faible sur i l'invariant de boucle $\Pi(i)$: L_i est un sous-parcours de G de racine s de $i+1$ éléments et $\mathcal{B}_i$ est la bordure de L_i.

Base Pour $i = 0$, $L_0 = (s)$ et $\mathcal{B}_0 = \mathcal{B}(L_0)$. Donc, $\Pi(0)$ est vérifiée.

Induction Supposons que la propriété soit vérifiée pour $i - 1 \geq 0$. Soit alors u_i le sommet de $\mathcal{B}_{i-1}$ choisi à l'itération i. Par hypothèse de récurrence, $\mathcal{B}_{i-1} = \mathcal{B}(L_{i-1})$, donc u_i possède un sommet adjacent v dans $V(L_{i-1})$ et $u_i \notin V(L_{i-1})$. Comme L_{i-1} est un sous-parcours de racine s, on en déduit que $L_i = L_{i-1} + (u_i)$ est également un sous-parcours de racine s. De plus, $\mathcal{B}_i = \mathcal{B}(L_i)$ d'après la dernière instruction de la boucle.

Conclusion La propriété est vérifiée par récurrence.

En sortie de boucle, $\mathcal{B} = \mathcal{B}(L) = \emptyset$ et la liste construite est bien un parcours de G. Pour la seconde partie du théorème, on note $\mathcal{C}(s)$ la composante connexe de s. On montre que $\mathcal{C}(s) = V(L)$:

$\mathcal{C}(s) \subseteq V(L)$ Par l'absurde, soit $u \in \mathcal{C}(s) - V(L)$, alors il existe une chaîne μ entre s et u. Comme $u \notin V(L)$, il existe une arête de $\{x,y\}$ de μ avec $x \in V(L)$ et $y \notin V(L)$. Ainsi $y \in \mathcal{B}(L)$ et donc $\mathcal{B}(L) \neq \emptyset$, ce qui est contraire aux hypothèses.

$V(L) \subseteq \mathcal{C}(s)$ Soit $u \in V(L)$, alors il existe une arborescence $\mathcal{A}(L)$ associée à L. Il existe donc une chaîne dans G de s à u et $u \in \mathcal{C}(s)$.

□

Corollaire 1. *Un graphe non orienté $G = (V,E)$ est connexe si et seulement si l'algorithme construit un parcours en partant d'un sommet quelqconque.*

Démonstration. On démontre la double implication.

$A \Rightarrow B$ Si G est connexe, et que $s \in V$, alors d'après le théorème précédent, l'algorithme construit un sous-parcours composé de tous les sommets de V puisqu'ils sont tous dans la composante connexe de s. On en déduit que $V(L) = V$ et donc L est un parcours.

$B \Rightarrow A$ Réciproquement, si l'algorithme construit un parcours L à partir d'un sommet quelqconque, alors on peut lui associer une arborescence $\mathcal{A}(L) = (V(L),H(L))$ avec $V(L) = V$. On en déduit que G est connexe.

□

Transparent 17 Intuitivement, G possède un parcours d'origine s si s est une racine de G.

Transparent 22 Il n'y a pas unicité dans le cas général. Sur l'exemple, on observe que $L =$ (2,4,1,5,6,3,7) est également un parcours en largeur d'origine 2.

Transparent 23 La définition assure ici l'unicité du graphe de liaison en largeur $\mathcal{A}^\star(L)$ de L.

Transparent 25 Non, si on reprend par exemple le graphe de liaison $\mathcal{A}(L)$ de droite du transparent 23. Ce graphe n'est pas le graphe de liaison **en largeur** $\mathcal{A}^\star(L)$ de L. On observe que la distance du chemin de 2 à 7 est de 4, ce qui ne correspond pas à $dist_2(7)$ qui vaut 3.

Transparent 26 Les files sont des structures linéraires. Les solutions les plus simples pour les implanter sont des tableaux circulaires, ou des listes chainées circulaires.