Détection de cycle dans un graphe dirigé :
conventions de représentations et théorie du problème

LINGI1122 - Méthodes de conception de programmes

Sedda	Mélanie	2246-11-00
Sluysmans	Benoît	6957-11-00
Van Den Eeckhaut	Kim	7561-11-00
Vico	Nicolas	3271-09-00
Volon	Julien	???

1 Introduction

Dans le cadre du cours de méthode de conception de programmes, il nous a été demandé de résoudre le problème suivant : étant donné un graphe dirigé, déterminer si celui-ci contient ou non un cycle. L'algorithme utilisé doit donc ressortir une réponse booléenne.

L'algorithme que nous devons utiliser est le suivant : supprimer du graphe tous les nœuds qui n'ont pas d'arête entrante, ainsi que les arêtes dont ces nœuds sont l'origine. En répétant cette opération, deux cas peuvent survenir : soit il n'y a plus de nœud disponible, et nous pouvons conclure qu'il n'existe pas de cycle dans le graphe de base, soit il ne reste que des nœuds avec au moins une arête entrante, auquel cas il existe au moins un cycle dans le graphe.

2 Conventions de représentations

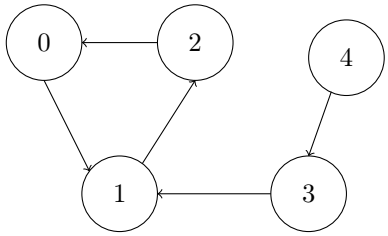
2.1 Présentation de la convention retenue

L'algorithme prend un graphe dirigé en entrée et retourne un booléen, qui indique si le graphe possède un cycle ou non. Il nous faut donc définir une convention de représentation d'un graphe dirigé sur lequel notre algorithme va s'appliquer.

Le graphe sera représenté en utilisant la matrice d'adjacence. Cette matrice est une matrice carrée dont la dimension est égale au nombre de noeuds du graphe. Chaque entrée (i, j) avec $0 \leq i, j \leq n-1$ indique le nombre d'arêtes allant du noeud i au noeud j.

Nous donnons ici un exemple d'une telle représentation. Le graphe donné ci-dessous sera représenté par le tableau suivant :

	0	1	2	3	4
0	0	1	0	0	0
1	0	0	1	0	0
2	1	0	0	0	0
3	0	1	0	0	0
4	0	0	0	1	0



2.2 Motivations

Le fait de représenter un graphe par le tableau à double entrée décrit précédemment possède divers avantages. Ceux-ci sont les suivants :

- **Facilité d'implémentation** : dans l'algorithme à implémenter, les actions principales à effectuer sur le graphe sont de parcourir les noeuds, de déterminer si ils ont des arêtes entrantes et de les supprimer. L'implémentation de ces deux fonctionnalités devient très facile. En effet, pour le parcours des noeuds, il suffit de parcourir le tableau à l'aide de boucles sur ses indices. Pour déterminer si un noeud j n'a aucune arête entrante, il suffit que la somme des éléments de la $j$ème colonne de la matrice d'adjacence soit nulle. Pour la suppression, il suffit de marquer la ligne et la colonne associées au noeud comme ne devant plus être parcourues.
- **Complexité satisfaisante** : on déduit de ce qui a été dit au point précédent que la recherche d'un noeud sans arrêtes entrantes se fait en $O(n^2)$ et que la suppression de celui-ci (ou plus précisément le marquage dans notre cas) se fait en $O(1)$. Ces complexités étant polynomiales, celles-ci sont considérées comme efficaces. De plus, d'autres représentations possibles (listes d'adjacence et liste d'arêtes) ont été considérées et aucune ne donnait une meilleure complexité pour la recherche et la suppression.
- **Marquage** : il a été décidé de marquer la ligne et la colonne associées à un noeud à supprimer au lieu de les supprimer du tableau. En effet, cela évite de devoir recopier le tableau dans un tableau de plus petite taille, ce qui est une opération qui se fait en $O(n^2)$ alors que le marquage se fait en $O(1)$.

3 Théorie du problème

3.1 Définitions

Un **graphe** est un triplet (V, E, ψ) où :
- V est un ensemble dont les éléments sont appelés sommets ou nœuds ;
- E est un ensemble dont les éléments sont appelés arêtes ;
- ψ est une fonction, dite fonction d'incidence, qui associe à chaque arête un sommet ou une paire de sommets.

On peut représenter un graphe par un diagramme. Plusieurs diagrammes peuvent représenter le même graphe.

Un **graphe dirigé**, ou orienté, est un triplet (V, E, ψ) où :
- V est un ensemble dont les éléments sont appelés sommets ou nœuds ;
- E est un ensemble dont les éléments sont appelés arêtes ;
- ψ est une fonction, dite fonction d'incidence, qui associe à chaque arête un couple de sommets. Ici, l'ordre au sein du couple de sommets a de l'importance, il signifie qu'un sommet est le nœud de départ de l'arête, l'autre étant le nœud d'arrivée.

Un **parcours** est une suite $v_0 e_1 v_1 e_2 ... e_n v_n$, où $v_1; v_2; ...$ sont des sommets, et $e_1; e_2; ...$ sont des arêtes. La longueur du parcours est son nombre d'arêtes n. Le sommet d'origine est v_0, le sommet de destination v_n. Les autres sommets sont dits intérieurs. Un parcours est fermé si $v_0 = v_n$.

Un **cycle** est un parcours fermé dont les sommets d'origine et intérieurs sont tous distincts. Un graphe qui ne contient pas de cycle est dit acyclique.

3.2 L'algorithme utilisé est correct

Démonstration : L'algorithme est juste.

Hypothèse : Soit G un graphe possédant n nœuds v et m arêtes e. Tous les nœuds de G possèdent une arête entrante, et G est acyclique.

- On choisit arbitrairement le nœud v_i, par hypothèse le nœud v_i possède au moins une arête entrante.
- On choisit arbitrairement l'une des ces arêtes entrantes et on supprime les éventuelles autres arêtes entrantes de v_i, on arrive alors au nœud v_j, qui possède également au moins une arête entrante(par hypothèse).
  - Soit on arrive à un nœud déjà visité et la démonstration est finie.
  - Soit on arrive à un nœud qu'on n'avait pas encore visité et on réitère l'algorithme.
- Comme le nombre de nœuds de G est fini, on arrive au dernier nœud (puisqu'on n'a pas de cycle jusqu'à présent). Hors, par hypothèse ce dernier nœud possède également une arête entrante qui ne peut pointer vers un autre nœud qu'un de ceux visité ⇒ il y a un cycle et contradiction.

Démonstration : L'algorithme se finit toujours.

Hypothèse : Le graphe G possède un nombre fini de nœuds et d'arêtes.

A chaque itération
- Il ne reste aucun nœud ⇒ Le programme s'arrête.
- Tous les nœuds ont au moins une arête entrante ⇒ Le programme s'arrête.
- Il existe un nœud ne possédant pas d'arête entrante. Ce nœud (et ses arêtes) est retiré par l'algorithme. Comme le graphe G possède, par hypothèse, un nombre fini de nœuds, il finit par se retrouver dans l'une des situations précédentes ⇒ Le programme s'arrête.