Graphes #2 : Détection de cycle

Pierre Karpman

Lycée Champollion MP2I

https://membres-ljk.imag.fr/Pierre.Karpman/CPGE/2025/

Table des matières

Détection de cycle dans un graphe non-orienté

Proposition

Soit $G = (S, A)$ un graphe non-orienté, C ses composantes connexes, alors il est sans cycle ssi. $\#A = \#S - \#C$

Esquisse de preuve

On construit une suite de sous-graphes de G avec $G_0 = (S, \varnothing)$ et G_{i+1} obtenu en ajoutant une (nouvelle) arête quelconque de A à G_i. On s'arrête quand on atteint G.
On montre l'invariant sur les $G_i = (S, A_i)$: $\#A_i = \#S - \#C_i$ ou (exclusif)
($\#A_i > \#S - \#C_i$ et G_i possède un cycle).

- Cas de base G_0 : $\#S$ composantes connexes, 0 arêtes et pas de cycles : $0 = \#S - \#S$ est vérifié
- Conservation : l'arête ajoutée peut :
  - relier deux sommets de composantes connexes différentes, ce qui ne crée pas de nouveau cycle et diminue de 1 le nombre de composantes connexes dans G_{i+1}
  - relier deux sommets de la même composante connexe, ce qui crée un cycle et n'augmente pas le nombre de composantes connexes dans G_{i+1}

Proposition *bis*

- L'algorithme utilisé dans la preuve de la proposition donne une façon alternative de calculer les composantes connexes d'un graphe non orienté
  - Et de détecter efficacement la présence d'un cycle
- Ce n'est **pas** un algorithme de parcours
- Vous verrez l'année prochaine une implémentation efficace de la structure de données *d'ensembles disjoints* (implicitement) utilisée par cet algorithme

Détection de cycle dans un graphe non-orienté *bis*

(Maintenant) (relativement) immédiat par parcours

- On compte le nombre de composantes connexes
  - Par ex. *via* un parcours
- On conclut en comparant au nombre d'arêtes et de sommets
- Coût (pour une représentation par listes d'adjacence) : linéaire en la taille de la représentation (**si** l'on se débrouille bien, *cf.* TD)

Optimisation

- Montrez que l'on peut détecter la présence d'un cycle d'un graphe non orienté représenté par listes d'adjacence en coût $O(\#S)$ (indépendant de $\#A$)

Table des matières

Détection de cycle dans un graphe orienté

Idée brute

On peut détecter la présence d'un cycle dans la composante connexe de v en effectuant un parcours d'origine v et tester si l'on revoit un sommet après l'avoir visité

- Mais attention aux détails ! Il faut s'assurer que la seconde rencontre se fait depuis un sommet accessible…
- Comment faire plus efficace sans itérer sur tous les sommets ?

Approche ordonnée : utiliser la classification des arcs d'un parcours en profondeur

On va montrer qu'un graphe orienté possède un cycle contenant v ssi. tout parcours en profondeur d'origine v possède un arc arrière et plus généralement qu'un graphe orienté possède un cycle ssi. il possède un arc arrière dans tout parcours en profondeur

Parcours en profondeur : temps de visite d'un sous arbre

Proposition

Soit v, w deux sommets visités lors d'un même parcours en profondeur depuis un même sommet, alors w est un descendant de v dans l'arborescence du parcours ssi. $v_a < w_a < w_p < v_p$

Preuve (esquisse)

w descendant $\Rightarrow$ [...] :

Par récurrence sur la profondeur des sommets dans le sous-arbre de racine v
Cas de base (profondeur 1) : w est adjacent à v et visité depuis v, et le résultat suit du fait que le parcours est en profondeur
Conservation : on suppose la propriété vraie pour les sommets à profondeur n. Soit w' un sommet à profondeur $n+1$, il est visité depuis un sommet w à profondeur n. Par le fait que le parcours est en profondeur, $w_a < w'_a < w'_p < w_p$; par H.R. $v_a < w_a < w_p < v_p$, et la propriété est conservée

Parcours en profondeur : temps de visite d'un sous arbre (suite)

Preuve (esquisse, suite)

$v_a < w_a < w_p < v_p \Rightarrow [\dots]$

Par récurrence forte sur $w_a - v_a$

Cas de base ($w_a - v_a = 1$) : v ne peut pas être une feuille de l'arborescence, et est nécessairement parent de w dans l'arborescence

Conservation : on supp. la propriété vraie pour tous les w satisfaisant l'hypothèse et t.q. $w_a - v_a \leq n$, et l'on montre la conservation pour un éventuel w' (satisfaisant l'hypothèse et) t.q. $w'_a - v_a = n + 1$

w' n'est pas racine du parcours puisque $w'_a > 1$, il a donc un parent w dans l'arborescence, qui est t.q. $w_a < w'_a$

Si $w = v$ il n'y a plus rien à montrer ; sinon par le fait que le parcours est en profondeur et $v_a < w'_a < w'_p < v_p$, l'on a deux possibilités :

- $w_a < v_a < w'_a < w'_p < v_p < w_p$
- $v_a < w_a < w'_a < w'_p < w_p < v_p$

Parcours en profondeur : temps de visite d'un sous arbre (fin)

Preuve (esquisse, fin)

Premier cas : $w_a < v_a < w'_a < w'_p < v_p < w_p$
Par le sens précédent, w n'est pas descendant de v dans l'arborescence
Comme $w'_a > w_a + 1$, w possède un autre enfant v' dans l'arborescence t.q. $w'_a \geq v'_p + 1$
Par le fait que le parcours est en profondeur, on a $w_a < v_a < v'_a < v'_p < w'_a < \cdots < v_p$
Alors $v'_a - v_p \leq n$ et par H.R. v' est descendant de v dans l'arborescence, ce qui est une contradiction
Ce cas ne peut donc pas se produire

Second cas : $v_a < w_a < w'_a < w'_p < w_p < v_p$
On a $v_a - w_a \leq n$ et par H.R. w est descendant de v dans l'arborescence, ce qui est donc aussi le cas de son enfant w'

Parcours en profondeur : existence de chemin

Proposition

Soit v, w deux sommets t.q. pour un certain parcours en profondeur on a $v_a < w_a < w_p < v_p$, alors il existe un chemin simple élémentaire $v \rightsquigarrow w$

Preuve

D'après la proposition précédente, w est descendant de v dans l'arborescence, qui fournit donc un tel chemin

Parcours en profondeur : existence de chemin *bis*

Proposition

Soit v, w t.q.'il existe un chemin $v \rightsquigarrow w$, alors pour tout parcours en profondeur d'origine v on a $v_a < w_a < w_p < v_p$

Preuve

Un parcours d'origine v visite tous les sommets w t.qu'il existe un chemin $v \rightsquigarrow w$
L'arborescence de ce parcours a racine v et il suffit donc d'appliquer la proposition précédente (précédente)

Parcours en profondeur : détection de cycle ancré

Proposition

Un graphe contient un cycle contenant le sommet v ssi. il existe un arc arrière $w \to v$ dans tout parcours en profondeur d'origine v.

Preuve

- soit $w \to v$ un tel arc arrière, on a $v_a < w_a < w_p < v_p$ et il existe donc un chemin simple élémentaire $v \rightsquigarrow w$ et donc un cycle $v \rightsquigarrow w \to v$ contenant v
- soit $v \rightsquigarrow w \to v$ un tel cycle, alors pour tout parcours en profondeur d'origine v on a $v_a < w_a < w_p < v_p$ et l'arc $w \to v$ est donc un arc arrière

Corollaire

On peut détecter la présence d'un cycle contenant un sommet v et reconstruire un tel cycle en temps $O(A_v)$ et espace $O(S_v)$ (si optimisé)

Parcours en profondeur : détection de cycle quelconque

On peut donc utiliser un parcours en profondeur pour décider si un graphe (orienté ou non) est acyclique

Option brutale

On teste pour chaque sommet s'il appartient à un cycle : coût en temps $O(SA)$

Option moins brutale

On fait un parcours depuis un sommet quelconque non visité (à répéter **correctement** en cas de graphe non connexe) en marquant les temps de visite, et l'on teste à la fin s'il y a un arc arrière : coût en temps $O(A + S)$ pour le parcours, $O(A)$ pour le test

- Correct ?

Parcours en profondeur : détection de cycle quelconque *bis*

Proposition

Un graphe orienté possède un cycle ssi. tout parcours en profondeur possède un arc arrière

Preuve (esquisse)

Présence d'un arc arrière $\Rightarrow$ [...] :
Soit $w \to v$ un arc arrière pour un parcours, alors $v_a < w_a < w_p < v_p$ et [identique preuve précédente]

Présence d'un cycle $\Rightarrow$ [...] :
Soit v le sommet du cycle de temps de pré-visite minimum et w son prédécesseur dans le cycle, alors $v_a < w_a$ et l'on considère deux cas :

- $w_p < v_p$: dans ce cas $w \to v$ est un arc arrière
- $v_p < w_a$: $w \to v$ serait un arc transverse, mais absurde car il existe un chemin $v \rightsquigarrow w$ et v est le premier sommet du cycle à être pré-visité

Parcours en profondeur : détection de cycle quelconque *ter*

Option encore moins brutale peut-être

On fait un parcours depuis un sommet quelconque non visité (à répéter **correctement** en cas de graphe non connexe) en marquant les temps de visite, et l'on détecte un arc arrière dès que possible : coût en temps $O(A + S)$ dans le pire cas, mais temps moyen possiblement plus faible si beaucoup de cycles

Graphes particuliers

On peut disposer d'algorithmes spécifiques pour des graphes possédant une structure particulière. C'est par exemple le cas pour les graphes orientés *fonctionnels* dont tous les sommets ont un degré sortant égal à un (*cf.* TP)

Table des matières

Graphes orientés sans cycles

DAG

Un graphe orienté sans cycle (en anglais : *directed acyclic graph* ou *DAG*) est un graphe orienté sans cycle

Quelques propriétés notables

Un DAG possède au moins :

- une *source* : un sommet sans arc entrant
- un *puit* : un sommet sans arc sortant

Modélisation de dépendance

Les DAGs modélisent naturellement des relations de dépendance où un arc $v \to w$ traduit le fait que w dépend de v
On peut satisfaire ces dépendances en considérant un *ordre topologique* des sommets

Graphes orientés sans cycles : tri topologique

Proposition

Les sommets d'un DAG peuvent être *triés topologiquement* : il existe (au moins) un ordre total $<$ sur les sommets t.q. pour toute paire de sommets (v, w) la présence d'un arc $v \to w$ implique $v < w$

Preuve

Le graphe étant sans cycle il ne possède pas d'arc arrière dans un parcours en profondeur ; ainsi pour tout arc $v \to w$ on a $w_p < v_p$ car :

- si $w_a < v_a < v_p < w_p$: pas d'arc $v \to w$ car il serait arrière
- si $v_a < v_p < w_a < w_p$: pas d'arc $v \to w$ car il serait pris lors d'un parcours depuis v

Donc l'ordre des sommets par temps de post-visite **décroissant** dans un tel parcours est un ordre total qui satisfait la contrainte