Parcours en profondeur et tri topologique

Dans ce TD on étudie un algorithme d'exploration d'un graphe $G = (X, A)$.
En partant d'un sommet donné (appelé racine) on a généralement deux façon d'explorer un graphe :
– une exploration en largeur (qu'on verra en cours)
– une exploration en profondeur (qu'on voit ici).

Exercice 1. *Parcours en profondeur*

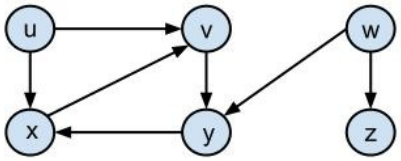
La stratégie d'une recherche en profondeur est de descendre plus "profondément" dans le graphe chaque fois que cela est possible. Dans cet algorithme les arcs sont explorés à partir du sommet v découvert le plus récemment et dont on n'a pas encore exploré tous les arcs incidents. Lorsque tous les arcs de v ont été explorés, l'algorithme revient en arrière pour explorer les arcs qui partent du sommet à partir duquel v à été découvert.

Data : Un graphe $G = (X, A)$
for $u \in X$ **do**
    couleur[u] = blanc;
    $\pi[u]$ =nil;
end
temps = 0;
for $u \in X$ **do**
    if *couleur*[u] =*blanc* **then**
        Visiter-DFS(u);
    end
end

Algorithme 1 : Algorithme DFS

Data : Un sommet u, le graphe G
couleur[u] = gris;
$d[u]$= temps;
temps = temps +1;
for $v \in Adj[u]$ **do**
    if *couleur*[v] =*blanc* **then**
        $\pi[v] = u$;
        Visiter-DFS(v);
    end
end
couleur[u] = noir;
$f[u]$= temps;
temps = temps +1;

Algorithme 2 : Algorithme Visiter-DFS



1. Donner le déroulement du parcours en profondeur sur le graphe de la figure.

2. Dans un graphe $G = (X, A)$ (orienté ou non) pour deux sommets quelconque u et v, une et une seule des conditions est vérifiée :
- Les intervalles $[d[u], f[u]]$ et $[d[v], f[v]]$ sont complètement disjoints, ou
- L'intervalle $[d[u], f[u]]$ est entièrement inclus dans l'intervalle $[d[v], f[v]]$ et u est descendent de v dans l'arborescence en profondeur correspondante, ou
- L'intervalle $[d[v], f[v]]$ est entièrement inclus dans $[d[u], f[u]]$ et v est un descendant de u dans l'arborescence en profondeur correspondente.

3. Montrez que le sommet v est un descendant propre du sommet u dans la forêt en profondeur d'un graphe G si et seulement si $d[u] < d[v] < f[v] < f[u]$.

4. Donner la complexité de l'algorithme.

Exercice 2. *Tri topologique*

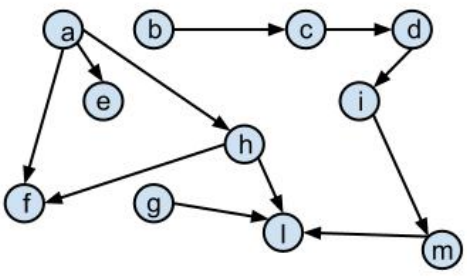
On va utiliser l'algorithme du parcours en profondeur pour effectuer un tri topologique sur des graphes orintés. Le tri topologique d'un graphe orienté acyclique $G = (X, A)$ consiste à ordonner linéairement tous ses sommets de sorte que l'arc (u, v), alors u apparaisse avant v. Soit $G = (X, A)$ un graphe orienté acyclique et la relation binaire $\Longrightarrow_G$ sur X définie par :

$$<_G v \iff \exists \text{ un chemin de } u \text{ à } v \text{ dans } G$$

1. Montrez que la relation $<_G$ est une relation d'ordre partiel sur S. C'est à dire :
- $<_G$ est réflexive : $\forall u \in S, u <_G u$
- $<_G$ est anti-symétrique : $u \neq v$ et $u <_G v$ alors $v \not<_G u$
- $<_G$ est transitive : $u <_G v$ et $v <_G w$ implique que $u <_G w$
- $<_G$ est partielle : $\exists u, v \in S$, avec $u \not<_G v$ et $v \not<_G u$

2. Montrer comment utiliser le parcours en profondeur pour effectuer un tri topologique sur des graphes acycliques orientés. Appelez cet algorithme Tri Top.

3. Donner l'ordre dans lequel Tri Top trie les sommets quand on l'exécute sur le graphe suivant :



4. Montrez que pour une paire quelconque de sommets distinct $u, v, \in X$, s'il existe un arc dans le graphe G de u à v alors $f[v] < f[u]$ (avec la notation de Visiter-DFS).

5. Donnez la complexité de l'algorithme Tri Top.