Écrit Blanc d'informatique
Préparation au CAPES de Mathématiques
12 décembre 2016
Durée 5h

Instructions :

1. Les deux problèmes sont indépendants et *doivent* être traités sur des copies séparées.
2. La clarté et la précision des réponses font partie intégrante de l'évaluation.
3. Des graphes sans dessin ...

Problème 1 : Aidons Superman à s'habiller.

On travaille sur des graphes *orientés*, que l'on suppose représentés par des *listes d'adjacence* : soit S l'ensemble des sommets et A l'ensemble des arcs, on associe à chaque sommet u dans S la liste $Adj[u]$ des éléments v tels qu'il existe un arc de u à v.

Parcours en profondeur :

Le parcours en profondeur d'un graphe G fait usage des constructions suivantes :

- A chaque sommet du graphe est associée une *couleur* : au début de l'exécution de la procédure, tous les sommets sont blancs. Lorsqu'un sommet est rencontré pour la première fois, il devient gris. On noircit enfin un sommet lorsque l'on est en fin de traitement, et que sa liste d'adjacence a été complètement examinée.
- Chaque sommet est également *daté* par l'algorithme, et ceci deux fois : pour un sommet u, $d[u]$ représente le moment où le sommet a été rencontré pour la première fois, et $f[u]$ indique le moment où l'on a fini d'explorer la liste d'adjacence de u. On se servira par conséquent d'une variable entière **globale** "temps" comme compteur événementiel pour la datation.
- La manière dont le graphe est parcouru déterminera aussi une relation de paternité entre les sommets, et l'on notera $\pi[v] = u$ pour dire que u est le père de v (selon l'exploration qui est faite, c'est à dire pendant le parcours v a été découvert directement à partir de u).

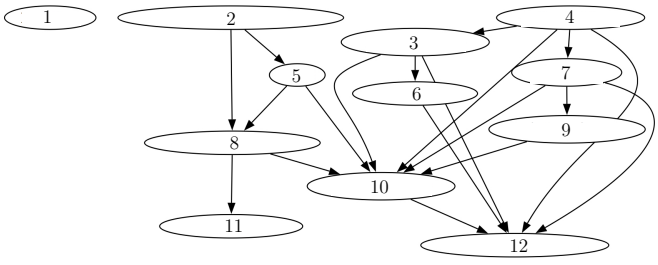
On définit alors le parcours en profondeur (**PP**) à l'aide des Algorithmes 1 et 2.

```
début
    pour tous les sommets u ∈ S/* Initialisation                    */
     faire
        couleur[u] ← BLANC;
        π[u] ← NIL;
    temps ← 0;
    pour tous les sommets u ∈ S/* Exploration                       */
     faire
        si couleur[u] = BLANC alors
            Visiter_PP(u);
        fin
fin
```

Algorithme 1 : PP(S, $Adj[]$, $temps$)

Question #1

Dessiner la liste d'adjacence du graphe suivant :



Question #2

Faire tourner l'algorithme sur ce graphe. Lorsqu'il y a un choix entre deux sommets, on prendra le sommet de plus petit numéro.

```
début
    couleur[u] ← GRIS;
    d[u] ← temps;
    temps ← temps + 1;
    pour tous les v ∈ Adj[u] faire
        si couleur[v] = BLANC alors
            π[v] ← u;
            Visiter_PP(v);
        fin
    couleur[u] ← NOIR;
    f[u] ← temps;
    temps ← temps + 1;
fin
```

Algorithme 2 : Visiter_PP(u)

On donnera la liste des sommets dans l'ordre de parcours, ainsi que $\pi[u]$, $d[u]$ et $f[u]$ pour tout sommet u.

Question #3

Justifiez le fait que l'on n'appelle **Visiter_PP**(u) qu'une et une seule fois par sommet u du graphe, puis une et une seule fois par arc.

Question #4

En déduire la complexité de PP en fonction de |S| et |A|.

Question #5

Soient u et v deux sommets du graphe. Supposons $d[u] < d[v]$ (on rappelle que d est la "date" de première visite). Si v est un descendant de u, que peut-on dire des valeurs relatives de $d[u]$, $d[v]$, $f[u]$, $f[v]$? Et si v n'est pas un descendant de u ? *FAIRE UN DESSIN*

Question #6

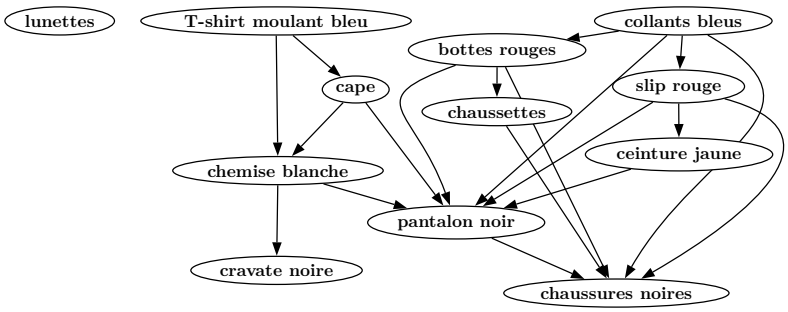
Montrer que si v est un descendant de u, alors à l'instant $d[u]$, il existe un *chemin blanc* de u à v, c'est à dire un chemin qui ne contient que des noeuds coloriés par Blanc. *On s'appuiera sur l'exemple pour raisonner.*

Question #7

Montrer que si à l'instant $d[u]$, il existe un *chemin blanc* de u à v, alors forcément v est un descendant de u. *On raisonnera par l'absurde.*

Tri topologique. n tâches sont données avec des contraintes de précédence $A < B$ signifie que la tâche A doit être effectuée avant la tâche B. L'objectif est de trouver un ordre des tâches qui respecte les contraintes de précédence. Pour cela, on modélise le problème par un graphe orienté :
— les sommets sont les tâches et
— il y a un arc $A \to B$ si et seulement si $A < B$, i.e. si A doit être exécuté avant A.

Voici par exemple le graphe de contraintes pour permettre à Clark Kent de s'habiller le matin avec son habit de Superman sous son costume (par exemple, le slip est mis après les collants et avant le pantalon noir) :



Il s'agit alors de trouver un ordre des sommets qui respecte les dépendances, ie qui respecte l'ordre "père-fils" dans le graphe. On remarque que si le graphe admet un circuit, ce n'est pas possible.

Question #8

Rajouter dans le graphe exemple des premières questions l'arc $12 - 8$ (afin de créer un circuit). Que se passe-t-il lors du parcours en profondeur ?

Question #9

Montrer que si le parcours en profondeur produit un arc arrière (c'est-à-dire un arc uv tel que v est un ancêtre de u dans l'arborescence produite π), alors il y a un circuit dans le graphe.

Question #10

Montrer (par l'absurde) que si un graphe est sans circuit, alors le parcours en profondeur ne produit aucun arc arrière.

On donne l'algorithme suivant pour faire un tri topologique d'un graphe G :

TRI-TOPOLOGIQUE(G)
1 appeler PP(G) pour calculer
les dates de fin de traitement $f[v]$ pour chaque sommet v
2 chaque fois que le traitement d'un sommet se termine,
insérer le sommet début d'une liste chaînée
3 **retourner** la liste chaînée des sommets

Question #11

Appliquer cet algorithme pour habiller Superman grâce à l'exemple de la question 1.

Question #12

Quelle est la complexité de cet algorithme ?

Question #13

(Difficile) Expliquer pourquoi cet algorithme donne le résultat voulu.