Algorithmique
Partiel n° 1 (P1)

Info-spé (s3)
Epita

D.S. 310962.78 BW (18 déc. 2013 - 10 :00)

Consignes (à lire) :

- ☐ Vous devez répondre sur **les feuilles de réponses prévues à cet effet**.
  - Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
  - Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
  - Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
  - Aucune réponse au crayon de papier ne sera corrigée.
- ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- ☐ **Les algorithmes :**
  - Tout algorithme doit être écrit dans le langage Algo (pas de C, Caml ou autre).
  - Tout code Algo non indenté ne sera pas corrigé.
  - Tout ce dont vous avez besoin (types, routines) est indiqué en **annexe** (dernière page) !
- ☐ Durée : 2h00

Des graphes

Exercice 1 (Connexions en vrac... – *4 points*)

1. Dans une réception de n personnes, est-il vrai qu'il y a toujours au moins deux personnes ayant le même nombre de connaissances (on admet que si A connaît B, alors B connaît A) ?

   Justifiez votre réponse.

2. Soit X un ensemble de lapins, et G un graphe orienté ayant X pour ensemble de sommets. On dit que G est un ***graphe de parenté*** si les arcs $x \to y$ de G codent la relation y *est l'enfant de* x et s'il n'y a qu'un *Adam* et qu'une *Eve*. Citez trois conditions que doit nécessairement vérifier G pour pouvoir être un graphe de parenté ?

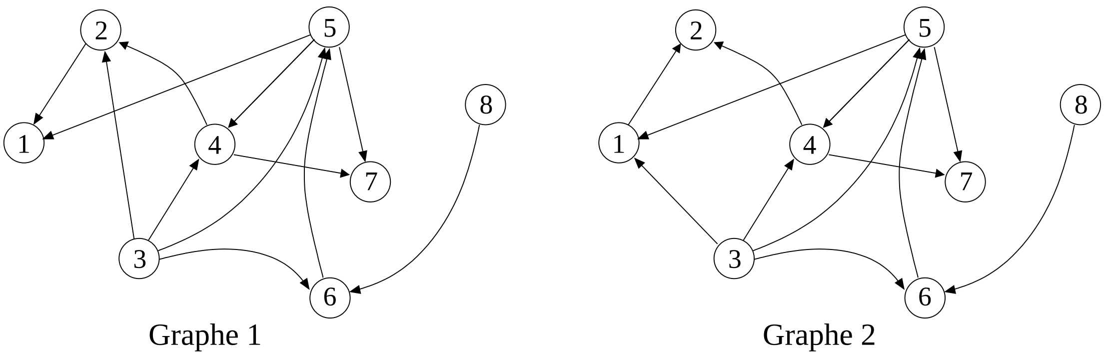

Figure 1 – Deux *graphes de parenté* ?

3. Quel(s) graphe(s) de la figure 1 mérite(nt) l'appellation de ***graphe de parenté*** ?

Exercice 2 (Représentations et question... – *3 points*)

1. Supposons un arbre[1] binaire complet de 7 sommets numérotés hiérarchiquement.
   (a) Donner sa représentation sous la forme d'une matrice d'adjacence.

   (b) Donner sa représentation sous la forme de listes d'adjacence.

2. Supposons un 1-graphe G orienté de n sommets et p arcs. Si ce graphe est implémenté en mémoire sous la forme de listes d'adjacence avec uniquement les listes de successeurs *(pas de listes de prédécesseurs)*.
   (a) Quelle est au pire la complexité pour calculer le demi-degré extérieur d'un sommet quelconque de G ?

   (b) Quelle est au pire la complexité pour calculer le demi-degré intérieur d'un sommet quelconque de G ?

1. Un arbre est un graphe connexe et sans cycle.

Exercice 3 (Plus long cycle – *9 points*)

Rappels :
– Un *cycle* est une chaîne dont les extrémités coïncident.
– Une chaîne est *élémentaire* si elle ne contient pas plusieurs fois le même sommet.

Le but de l'exercice est de déterminer dans un graphe non orienté (en représentation statique) la longueur du plus long *cycle élémentaire* à l'aide d'un parcours profondeur.
Pour calculer la longueur du plus long cycle élémentaire, on construit, pendant le parcours, un vecteur ($prof$) qui contient pour chaque sommet sa profondeur dans la forêt couvrante. Ce vecteur servira de marque.

1. Des arcs pour les cycles :
   (a) Lors du parcours profondeur d'un graphe non orienté, quel type d'arc permet de détecter les cycles ?
   (b) Combien d'arcs de ce type contiendra un *cycle élémentaire* ?
   (c) Comment repérer ces arcs lors du parcours profondeur en utilisant le vecteur $prof$?
2. Écrire les deux algorithmes du parcours profondeur :
   (a) la fonction `plus_long_cycle` (G) qui retourne la longueur du plus long cycle dans le graphe G (0 si pas de cycle). Cet algorithme est la fonction d'appel de la fonction suivante.
   (b) la fonction `plc_rec` (G, s, $prof$) qui lance le parcours profondeur à partir du sommet s et qui remplit le vecteur $prof$ (qui sert aussi de marque) avec les profondeurs des sommets dans l'arbre couvrant. Elle retourne la longueur du plus long cycle dans le sous-graphe parcouru.

Un arbre

Exercice 4 (Arbre rouge-noir – *4 points*)

On s'intéresse à écrire un algorithme qui nous dit si un arbre 2-3-4 sous sa représentation bicolore (rouge-noir) est un arbre 2-3-4 complet, c'est-à-dire que tous ses noeuds ne sont que des 4-noeuds. On considérera que l'arbre bicolore, passé en paramètre, représente forcément un arbre 2-3-4 valide.

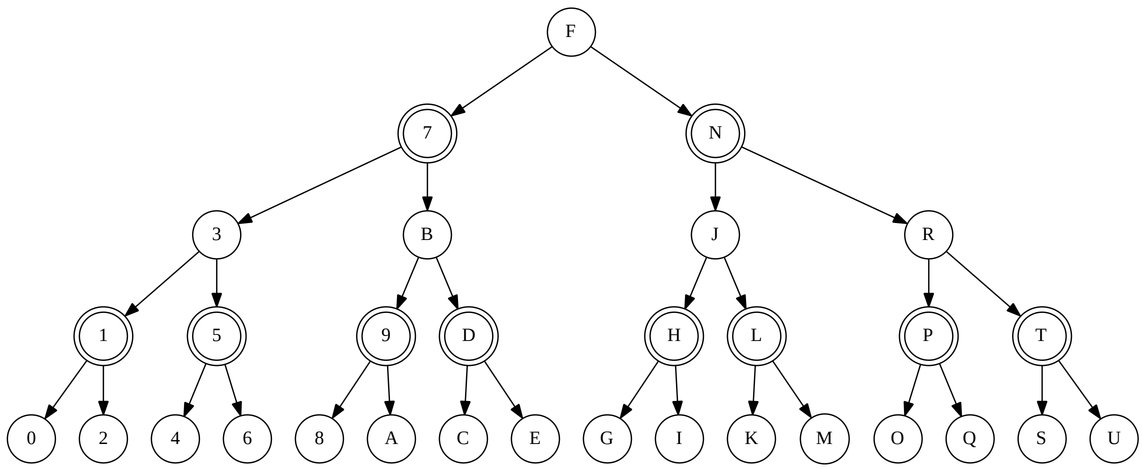

Figure 2 – Arbre rouge-noir

1. L'arbre 2-3-4 sous sa représentation bicolore de la figure 2 est-il complet ? Justifier votre réponse.
2. Écrire l'algorithme permettant de savoir si un arbre 2-3-4, sous sa représentation bicolore, est complet.

Annexes

Représentation statique des graphes

Les graphes utilisés ici sont non valués, les coûts ont donc été enlevés de la représentation.

```
constantes
   Max = 100

types
   t_mat_adj   = Max × Max  entier

   t_graph_stat = enregistrement
       booleen     orient
       entier      ordre
       t_mat_adj   adj
   fin enregistrement t_graph_stat
```

Autres

Le type pour les vecteurs d'entiers :

```
types
     t_vect_entiers = Max entier
```

Fonction utile :

`max (entier` a, b`)` retourne la valeur maximum entre a et b.

Représentation des arbres bicolores

```
types
     /* déclaration du type t_element */
    t_arn = ↑ t_noeud_arn

    t_noeud_arn = enregistrement
        t_element    cle
        booleen      rouge
        t_arn        fg, fd
    fin enregistrement t_noeud_arn
```