Routage dans un réseau arborescent (X 2003)

Durée : 4 heures

Dans ce problème on aborde une question soulevée dans la conception des réseaux : il s'agit d'affecter des adresses distinctes aux nœuds d'un réseau de façon telle que la route entre deux nœuds puisse être déterminée par un calcul utilisant uniquement leurs deux adresses, sans référence à une connaissance globale du réseau. On s'intéresse plus particulièrement aux réseaux ayant une forme d'arbre.

Le problème est composé de trois parties indépendantes.

Dans tout le problème un *n-arbre* est un arbre dont l'ensemble des sommets (on dit aussi parfois les nœuds) est $\{0,1,\ldots,n-1\}$, et dont la racine est 0. Chaque sommet a possède un père noté pere$[a]$, par convention la racine est son propre père, ainsi pere$[0] = 0$. L'ensemble des n-arbres est noté $\mathscr{A}(n)$.

L'ensemble des *ancêtres* du sommet a est constitué de a et des ancêtres de son père pere$[a]$; la racine 0 est ainsi ancêtre de tous les sommets de l'arbre. Le *sous-arbre de racine a*, noté A_a, est formé de tous les sommets de l'arbre A dont a est ancêtre.

Les *fils* d'un sommet a sont ainsi les $b \neq 0$ tels que pere$[b] = a$, (attention la racine n'est pas fils d'elle même). Une *feuille* est un sommet qui n'a pas de fils.

La *hauteur* (on dit aussi parfois profondeur) d'un sommet a, notée $h(a)$ est un entier égal à 0 si a est la racine, elle est égale à la hauteur du père de a augmentée de 1 sinon. Le *plus proche ancêtre commun* à deux sommets a et b, noté ppac(a,b), est le sommet de hauteur maximale qui est à la fois ancêtre de a et de b. Noter que si a est ancêtre de b alors ppac$(a,b) = a$.

Dans tout le problème, on considère que l'entier n est fixé, d'autre part pour un tableau t, $t[i]$ dénote l'élément d'indice i ; une liste contenant les entiers $x_0, x_1, \ldots, x_p$ est représentée par $\langle x_0, x_1, \ldots, x_p \rangle$; la liste vide est notée $\langle\ \rangle$.

On utilisera pour un n-arbre A :

- un tableau de listes d'entiers fils de taille n tel que fils$[i]$ est la liste des fils de i ;
- un tableau d'entiers pere de taille n tel que pere$[i]$ est le père de i.

On a ainsi en Caml :

```
let n = ..... ;;
type vint == int vect ;;
type lint == int list ;;
type vlint == lint vect ;;
```

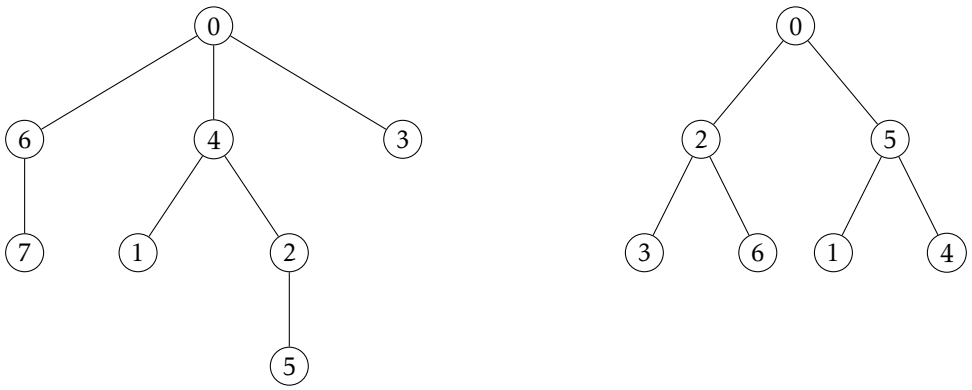

Figure 1 – Un arbre A $\in \mathscr{A}(8)$ et un arbre binaire complet B $\in \mathscr{B}(2)$.

Les tableaux pere et fils de l'arbre A situé à gauche de la figure 1 sont donnés par :

i	0	1	2	3	4	5	6	7
pere	0	4	4	0	0	2	0	6
fils	$\langle 6,4,3 \rangle$	$\langle\ \rangle$	$\langle 5 \rangle$	$\langle\ \rangle$	$\langle 1,2 \rangle$	$\langle\ \rangle$	$\langle 7 \rangle$	$\langle\ \rangle$

Partie I. Fonctions élémentaires

Question 1. Calculer ppac(i, j) pour tous les couples de sommets de l'arbre A donné plus haut. On présentera le résultat sous forme d'un tableau de 8 lignes et 8 colonnes tel que la case sur la ligne i et la colonne j contienne le plus proche ancêtre de i et j.

Question 2. Écrire une fonction

```
hauteur : vint -> int -> int
```

qui étant donné un sommet a d'un arbre, pour lequel on donne son tableau des pères, calcule la hauteur de ce sommet.

Question 3. Écrire une fonction

```
filsEnPere : vlint -> vint
```

qui à partir d'un arbre, donné par ses listes de fils, calcule le tableau des pères.

Question 4. Écrire une fonction

```
ppacMemeH : vint -> int -> int -> int
```

qui calcule le plus proche ancêtre commun de deux sommets a et b supposés de même hauteur dans un arbre pour lequel on donne le tableau des pères.
En déduire une fonction **ppac** qui effectue le même calcul pour deux sommets a et b n'ayant pas nécessairement la même hauteur.

Partie II. Arbres binaires complets

Un *arbre binaire complet* est un arbre dans lequel tout sommet qui n'est pas une feuille a exactement deux fils, et tel que toutes les feuilles sont à la même hauteur. On note $\mathscr{B}(h)$ l'ensemble des n-arbres binaires complets dans lesquels les feuilles sont à hauteur h. Un exemple d'arbre binaire complet B est donné à droite de la Figure 1.

Question 5. Déterminer pour $0 \leqslant k \leqslant h$ le nombre s_k de sommets de hauteur k dans un arbre B de $\mathscr{B}(h)$, justifiez votre réponse. En déduire le nombre n de sommets d'un arbre de $\mathscr{B}(h)$ en fonction de h.

Dans la suite B est un arbre de $\mathscr{B}(h)$ ayant n sommets ; à chaque sommet a de B on associe une étiquette $\ell(a) \in \{1, 2, \ldots, n\}$ satisfaisant la condition suivante :

pour chaque sommet a ayant pour liste de fils $\langle b, c\rangle$: $\max\limits_{u \in \mathrm{B}_b} \ell(u) < \ell(a) < \min\limits_{v \in \mathrm{B}_c} \ell(v)$.

Question 6. Calculer les valeurs de $\ell(a)$ pour les 7 sommets de l'arbre binaire complet représenté sur la droite de la Figure 1 : l'ordre des fils dans les listes correspond à la lecture de gauche à droite de la figure.

Question 7. Écrire une fonction

```
etiquettes : vlint -> vint
```

qui calcule les étiquettes $\ell(a)$ pour tous les sommets d'un arbre de $\mathscr{B}(h)$ donné par ses listes de fils.

Question 8. Un arbre B de $\mathscr{B}(h)$ de racine 0 ayant pour liste de fils $\langle a, b\rangle$ se décompose en deux sous-arbres B_a et B_b. Déterminer les valeurs de $\ell(0)$, de $\ell(a)$ et de $\ell(b)$ pour les fils a et b de la racine.

Question 9. Démontrer que si a et b sont deux sommets quelconques de B alors $r = \ell(\text{ppac}(a, b))$ est déterminé de manière unique à partir de $p = \ell(a)$ et $q = \ell(b)$ par la propriété suivante :

r est parmi les entiers compris (au sens large) entre p et q celui possédant le plus grand diviseur qui soit une puissance de 2.

Question 10. Donner une fonction récursive

```
mu : int -> int -> int
```

qui détermine $r = \ell(\text{ppac}(a, b))$ à partir de $p = \ell(a)$ et $q = \ell(b)$; on supposera que $p \leqslant q$. Votre fonction doit être de complexité $\mathrm{O}(\log_2(q - p))$.

Partie III. Arbres généraux

Dans cette partie on utilise les définitions et notations suivantes :

- dans un arbre A le poids $w[a]$ d'un sommet a est le nombre de sommets du sous-arbre A_a de racine a, ainsi $w[a] = 1$ si et seulement si a est une feuille, noter que $w[0] = n$ (c'est le nombre de sommets de A) ;
- un arbre A est dit *gauche* si pour chaque sommet a (qui n'est pas une feuille) le premier élément de la liste des fils est de poids supérieur ou égal à celui de tous les autres sommets de cette liste. Dans un arbre gauche, le premier élément de la liste des fils d'un sommet est dit *lourd*, les autres sommets sont dits *légers*. La racine est un sommet léger.

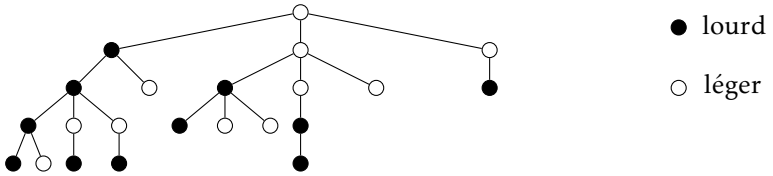

Figure 2 – Sommets lourds et légers.

Question 11. Écrire une fonction

```
poids : vlint -> vint
```

qui calcule les poids de tous les sommets d'un arbre donné par ses listes de fils. Puis une fonction :

```
gauchir : vlint -> vint -> vlint
```

qui calcule un arbre gauche obtenu en réordonnant les fils de chaque sommet de façon telle que le premier fils soit de poids supérieur ou égal à celui des autres. Sont donnés dans cette fonction les listes des fils et le tableau des poids des sommets de l'arbre.

Question 12. Soit a un sommet léger d'un arbre gauche A et b son père, montrer que $w[b] > 2w[a]$. En déduire que le nombre d'ancêtres légers de a est inférieur à $1 + \log_2 n$.

On utilise dans toute la suite la notion de cime d'un sommet.

- Si a est léger cime$[a]$ est égale à pere$[a]$;
- si a est lourd cime$[a]$ est le plus proche ancêtre de a qui est léger.

Ainsi en raison de nos conventions cime$[0] = 0$. On appelle A′ l'arbre gauche obtenu en appliquant la fonction **`gauchir`** à l'arbre A de la figure 1. Les cimes des sommets de l'arbre A′ sont alors données par le tableau suivant :

i	0	1	2	3	4	5	6	7
cime	0	4	0	0	0	0	0	6

Question 13. Écrire une fonction

```
cimes : vlint -> vint
```

qui calcule les sommets cimes de tous les sommets à partir des listes de fils d'un arbre gauche.

On se propose d'attribuer des étiquettes aux sommets d'un arbre gauche A. Pour cela on commence par construire deux tableaux d'entiers t_1 et t_2 associés aux sommets. Le tableau t_1 est tel que $t_1[0] = 0$ et $t_1[a] = i$ si a est le i^{e} élément de la liste des fils de son père.
Le tableau t_2 est tel que $t_2[a] = 0$ si a est léger et $t_2[a] = t_2[\text{pere}[a]] + 1$ si a est lourd.
Les étiquettes $\lambda(a)$ des sommets sont alors des listes d'entiers construites de la façon suivante :

$$\lambda(0) = \langle\,\rangle \qquad \text{et pour } a \neq 0 \quad \lambda(a) = \lambda(\text{cime}[a]) \circ \langle t_1[a], t_2[a] \rangle$$

dans cette formule $\circ$ dénote la concaténation des listes.

Question 14. Donner l'arbre A′ défini à la question 12 ; puis donner les valeurs de $t_2[a]$ pour tous les sommets a ; calculer enfin leurs étiquettes $\lambda(a)$.

Question 15. Écrire une fonction

```
etiquettes : vlint -> vint -> vlint
```

qui calcule les étiquettes des sommets d'un arbre gauche donné par ses listes de fils et pour lequel on donne aussi le tableau des cimes des sommets. On pourra utiliser l'opérateur `@`.

Question 16. Écrire une fonction :

```
trouve : vlint -> lint -> int
```

qui, à partir de l'étiquette d'un sommet d'un arbre gauche A et des listes de fils des sommets de cet arbre, détermine ce sommet.

Question 17.

a) Montrer que, pour tout sommet a d'un arbre, la longueur de l'étiquette $\lambda(a)$ est majorée par 4 fois le nombre de ses ancêtres légers.

b) Indiquer comment on calcule $\lambda(\text{ppac}(a,b))$ à partir des deux listes $u = \lambda(a)$ et $v = \lambda(b)$; on ne demande pas de justifier la réponse à cette question.

On a donné dans ce problème une technique permettant d'étiqueter les nœuds des sommets d'un réseau qui a une forme d'arbre. Dans cet étiquetage la recherche du plus proche ancêtre commun de deux sommets (et donc de la route qui les relie) s'effectue uniquement à l'aide de leurs étiquettes. Des techniques plus complexes, qui généralisent les techniques données ici, utilisent des étiquettes de taille $O(log_2 n)$ *bits qui permettent aussi un calcul du plus proche ancêtre commun en* $O(1)$ *opérations.*

♦ ♦
♦