Université Paris XII
IUT de Fontainebleau
Département informatique
Licence SIL
Structures de données abstraites
2007–2008

PARTIEL 2
Durée : 1h30

Seuls sont autorisés, à titre de documents, les listings comprenant explicitement le nom de l'étudiant imprimé (et celui-ci seulement) et les notes manuscrites (pas de photocopie) comprenant le nom de l'étudiant sur chaque page.

Il est conseillé d'écrire les exercices en langage C++. L'écriture en Java complique les entrées-sorties et l'écriture en langage C allonge le programme (ce qui ne rapportera pas de point supplémentaire dans un cas comme dans l'autre).

Exercice unique.- **Tri par tas**

Le **tri par tas** *(*heapsort *en anglais) est un algorithme de tri proposée en 1964 par J. W.* Williams *et R. W.* Floyd. *Il présente l'intérêt d'être optimal en moyenne et surtout de s'effectuer "en place", c'est-à-dire qu'on n'a pas besoin d'utiliser un tableau auxiliaire.*

- 1^o) (**Représentation d'un tableau par un arbre binaire**)

Un tableau `tab[]` *de taille effective n peut être représenté par un arbre (étiquetté) binaire dont la racine est étiquettée par* `tab[0]`*, dont le fils gauche de* `tab[i]`*, s'il existe, est* `tab[2i+1]` *et le fils droit, s'il existe,* `tab[2i+2]`*.*

Par exemple le tableau [a,e,b] *est représenté par l'arbre de la figure 1.*

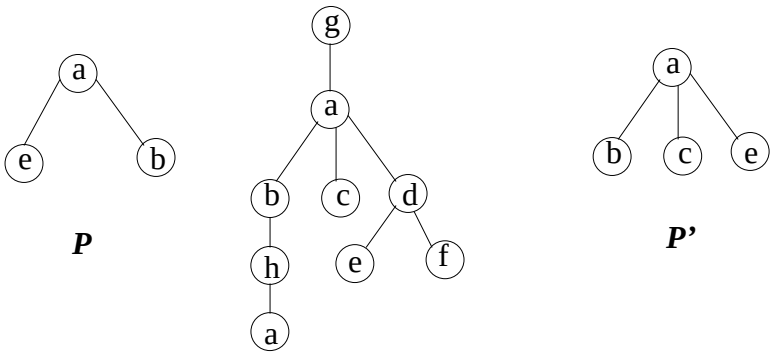

Figure 1:

Tracer l'arbre associé au tableau [44, 55, 12, 42, 94, 18, 6, 67].

- 2^o) (**Arbre ordonné verticalement**)

En supposant qu'il existe une relation d'ordre total sur l'ensemble des étiquettes, un arbre binaire (étiquetté) est dit **ordonné verticalement** *si l'étiquette de tout nœud est inférieure à celle de ses fils. On parle alors de* **tas** *(*heap *en anglais).*

a) L'arbre associé au tableau [2, 4, 3] est-il un tas ?
b) L'atbre associé au tableau [12, 5, 13] est-il un tas ?
c) L'arbre de la question 1^o) est-il un tas ?

On justifiera sa réponse dans chacun des cas.

3^o) (**Pseudo-tas**)

Un **pseudo-tas** *est un arbre dont la racine est quelconque mais dont les deux sous-arbres dont les racines sont le fils gauche et le fils droit de la racine sont des tas.*

L'opération qui consiste à permuter les étiquettes de façon à obtenir un tas s'appelle l'opération de **descente**. *Elle consiste à échanger la racine avec le plus petit de ses deux fils, puis échange de ce fils avec le plus petit de ses deux propres fils, en poursuivant tant que l'élément à descendre a au moins un fils courant qui lui est inférieur et s'effectue toujours sur la branche correspondant au plus petit des deux fils.*

a) Expliquez pourquoi l'algorithme (récursif) suivant est une implémentation de l'opération de descente pour le pseudo-tas de racine j :

```
descente(j,n,tab)
  Si j < n/2 alors
    l = 2j + 1
    Si l < n-1 alors
      Si tab[l] > tab[l+1] alors
        l = l+1
    Si tab[j] > tab[l] alors
      echange(tab[j],tab[l])
      decente(l,n,tab)
```

b) Coder cet algorithme en langage C en supposant que l'arbre est étiqueté par des entiers, y compris les fonctions auxiliaires nécessaires éventuelles.

4^o) (**De l'arbre au tas**)

*Une première étape du tri par tas consiste à transformer un arbre en un tas, en permutant les étiquettes. Remarquons que les feuilles de l'arbre (*tab[j] pour j ¿ n/2 *sont des tas puisqu'elles n'ont pas de fils. Il suffit donc d'appliquer l'opération de descente depuis le premier nœud interne jusqu'à la racine, c'est-à-dire de $n/2$ à 0.*

a) Expliquer pourquoi l'algorithme suivant implémente cet algorithme :

```
arbre2tas(n,tab)
  Pour j de n/2-1 a 0 en decroissant
    descente(j,n,tab)
```

b) Coder cet algorithme en langage C.

5^o) (**Tri d'un tas**)

Tous les descendants de la racine d'un tas sont étiquetés par des éléments plus grands que l'étiquette de la racine. Si on échange l'étiquette de la racine avec tab[n-1] *(qui est une feuille) alors, d'une part, le plus petit élément se trouve en* tab[n-1] *et, d'autre part, le tableau de longueur* $n-1$ *est un pseudo-tas.*

a) Expliquer pourquoi l'algorithme suivant permet de trier (dans l'ordre décroissant) un tas :

```
tri_tas(n,tab)
  Pour i de n-1 a 2 en decroissant
    echange(tab[0],tab[i]);
    descente(0,i,tab)
  echange(tab[0], tab[1])
```

b) Coder cet algorithme en langage C.

6^o) (**Tri par tas**)

a) Expliquer pourquoi l'algorithme suivant permet de trier (dans l'ordre décroissant) un tableau :

```
heapsort(n,tab)
  arbre2tas(n, tab)
  tri_tas(n, tab)
```

Si on veut trier le tableau dans l'ordre croissant, il suffit de renverser le tableau ainsi obtenu.

b) Coder cet algorithme en langage C.

6^o) Écrire un programme C complet utilisant ce tri par tas.