Institut Galilée — Algorithmique, arbres et graphes
Année 2005-2006 — L2

Examen du jeudi 8 juin 2006

Aucun document autorisé – pas de calculatrice

Le barème (uniquement indicatif) n'est pas directement proportionnel à la difficulté des questions ou au temps nécessaire pour y répondre.

Première partie : questions de cours

10 pt

Comme dans le cours, on considère des éléments qui sont fait d'une clé et de données satellites. On considère ici que les clés sont des entiers. Dans les arbres et dans les tableaux que l'on représente on se contente de donner les clés des éléments sans faire apparaître les données satellites associées.

1 Notation asymptotique, tris

Exercice 1.

1. *Si* $f(n) = 10n + 100$*, est-ce que* $f(n) = O(n)$ *?* $f(n) = O(n \log n)$ *?* $f(n) = \Theta(n^2)$ *?* $f(n) = \Omega(\log n)$ *? (Répondre par oui ou par non, sans justification.)* (0.5 pt)
2. *En notation asymptotique quelle est la borne minimale en temps des tris par comparaison, en pire cas et en moyenne ?* (0.5 pt)

2 Piles, files, tas

Exercice 2.

1. *Si, partant d'une pile p vide, on ajoute (en empilant), les entiers* 1*, puis* 2*, puis* 3*, puis* 4*, puis* 5 *et que, ensuite, on supprime (par dépilement) deux éléments, quels entiers contient la pile ?* (0.5 pt)
2. *Même question avec une file (utiliser les fonctions d'ajout et de suppression des files à la place de l'empilement et du dépilement).* (0.5 pt)

Exercice 3 (Insertion / suppression)**.** *On se donne le tas max (ou maximier) de la figure 1. En utilisant les algorithmes vus en cours :*

1. *Insérer un élément de clé* 44 *dans ce tas. Répondre en représentant le nouveau tas.* (0.5 pt)
2. *En repartant du tas initial, supprimer l'élément de clé maximale. Répondre en représentant le nouveau tas.* (0.5 pt)

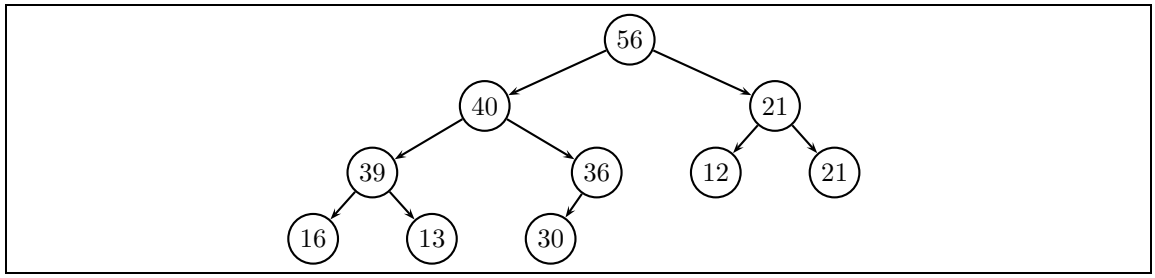

Fig. 1: Tas

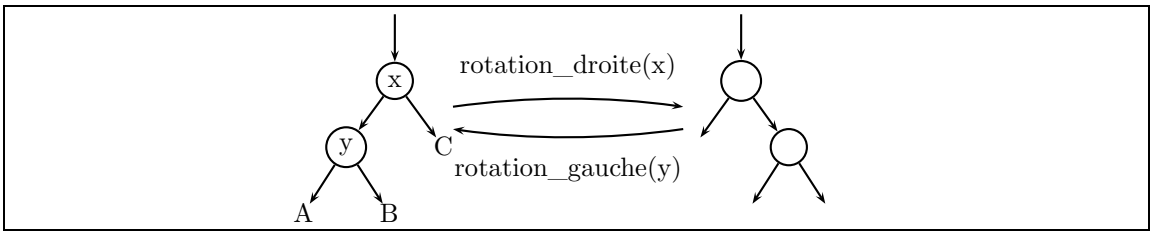

Fig. 2: Rotations gauche et droite

3 Arbres binaires de recherche (ABR)

Exercice 4 (Insertion)**.** *Décrire en quelques lignes le principe de l'algorithme d'insertion d'un élément de clé c dans un arbre binaire de recherche.* Attention : lorsque la clé c est déjà présente dans l'arbre, il y a le choix entre deux possibilités. Dans cet exercice, fixer votre choix (toujours à gauche ou bien toujours à droite) et conserver cette convention pour tout l'examen. (1 pt)

Exercice 5 (Commutativité de l'insertion)**.** *L'opération d'insertion d'éléments dans un arbre binaire de recherche est-elle une opération « commutative » au sens où l'insertion de x puis de y dans un arbre binaire de recherche produit le même arbre que l'insertion de y puis de x ? Si oui, dire pourquoi, si non donner un contre exemple.* (1 pt)

Exercice 6 (Insertion)**.** *Dessiner l'arbre binaire de recherche obtenu par insertions successives des éléments de clés* $24, 16, 32, 45, 11, 12, 28, 26, 32$ *en partant de l'arbre vide.* (1 pt)

Exercice 7. *Les opérations de recherche d'insertion et de suppression dans les ABR se font en temps $O(h)$ où h est la hauteur de l'arbre c'est à dire le nombre maximum de nœuds sur une branche. En pire cas, que vaut cette hauteur en fonction du nombre n d'éléments de l'arbre ? Donner, pour tout n, une suite d'insertions (juste les clés) telle que l'arbre binaire de recherche obtenu réalise ce pire cas.* (1 pt)

4 Rotations, arbres rouge noir

Exercice 8. *Rappel : une rotation doit préserver la propriété des arbres binaires de recherche.*

1. *Compléter la partie droite de la figure 2, avec les noms des différents éléments (x, y) et sous-arbres (A, B, C).* Vous pouvez répondre sur le sujet. (0.5 pt)
2. *Dans la figure 3, quelles série de rotations (centre et sens) peut-on utiliser pour faire remonter à la racine le nœud 32, fils droit de 24 ?* (0.5 pt)

Exercice 9. *Colorier tous les nœuds de l'arbre binaire de recherche de la figure 3 pour en faire un arbre rouge noir.* Vous pouvez répondre sur le sujet. (1 pt)

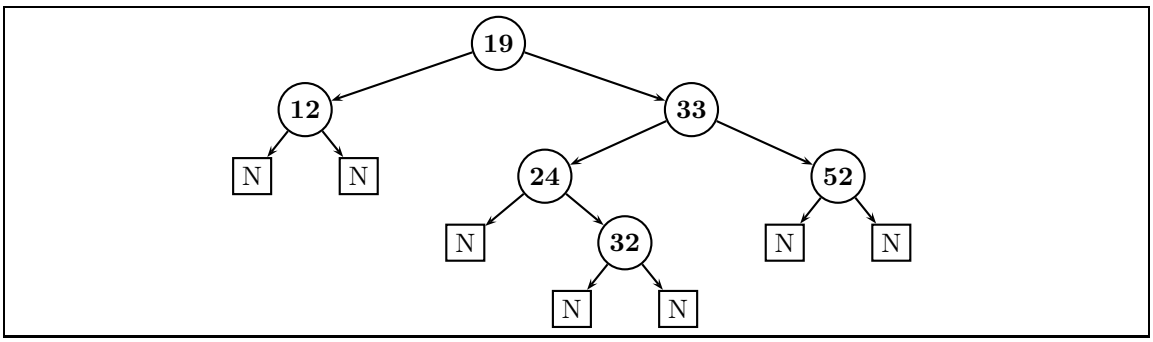

Fig. 3: Coloriage

Exercice 10. *Pour chaque arbre de la figure 4, dire s'il s'agit d'un arbre rouge noir. Si non, pourquoi ?* (1 pt)

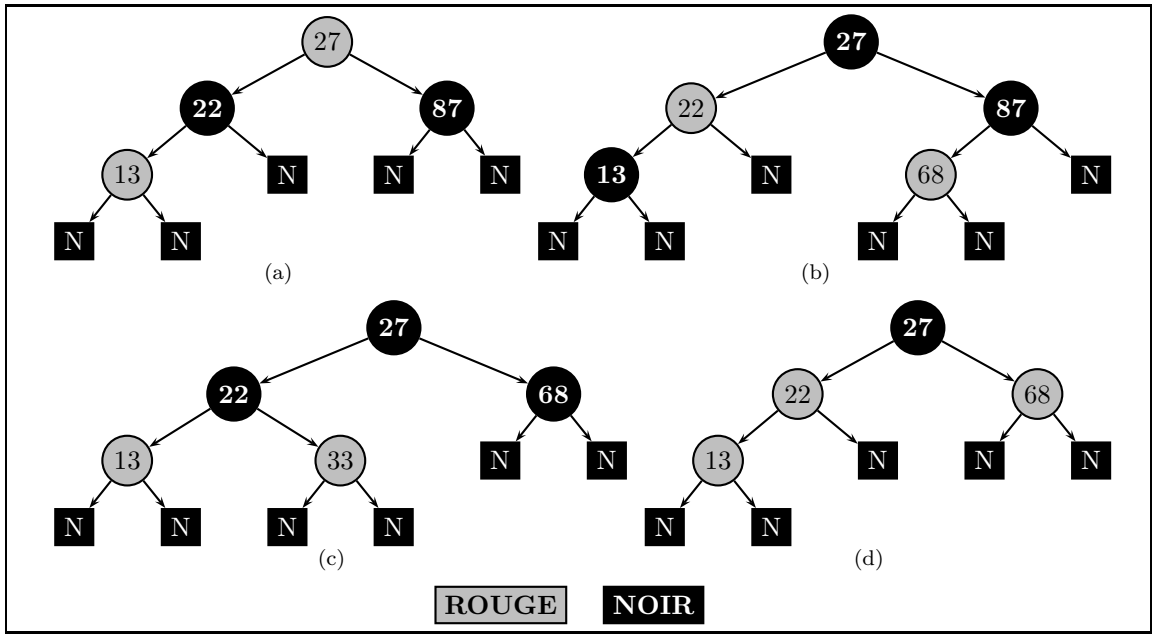

Fig. 4: Rouge noir ?

Seconde partie

10 pt

On pourra considérer que le type des éléments est donnée par le code :

```
typedef struct {   /* Un objet de type element_t est donné par :    */
  int cle;         /* - une clé (= un entier)                       */
  data_t donnee;   /* - et une donnée satellite (type non détaillé) */
} element_t;
```

Pour représenter les arbres binaires de recherche ont peut utiliser le type :

```
typedef struct noeud_s {  /* Un noeud est la donnée :                        */
  element_t e;            /*  - d'un élément                                 */
  struct noeud_s *parent; /*  - d'un pointeur vers un noeud parent           */
  struct noeud_s *gauche; /*  - d'un pointeur vers un noeud fils gauche      */
  struct noeud_s *droite; /*  - d'un pointeur vers un noeud fils droit       */
} noeud_t, * abr_t;       /* Un ABR est un pointeur sur un noeud (la racine) */
```

Exercice 11. *Rappel : la fonction de recherche des arbre binaire de recherche prend en argument une clé c et un pointeur vers la racine d'un arbre binaire de recherche A et rend un des éléments stockés dans A dont la clé est c.*

1. *Écrire le code de la fonction de recherche en C (algorithme vu en cours).* (1 pt)

Lorsque qu'il y a plusieurs éléments de clé c dans l'arbre, la fonction renvoie un seul de ces éléments, e.

2. *En supposant qu'on n'a effectué que des insertions et des suppressions, est-ce que l'élément e renvoyé est le dernier élément de clé c a avoir été inséré dans A, le premier élément de clé c a avoir été inséré dans A, ou ni l'un ni l'autre ? Justifier votre réponse (par un raisonnement ou un contre-exemple).* (2 pt)

3. *Écrire une fonction de recherche qui prend en entrée un arbre binaire de recherche A et une clé c et affiche tous les éléments de A de clé c. Pour l'affichage d'un élément, on se donne une fonction* `void elt_affiche(element_t e)`. (1 pt)

Exercice 12. *Même question qu'à l'exercice 5 pour les commutations entre la suppression et l'insertion. L'insertion et la suppression dans les ABR sont elles des opérations qui « commutent » au sens où la suppression de x puis l'insertion de y dans un arbre binaire de recherche produit le même arbre que l'insertion de y puis la suppression de x ? Si oui, dire pourquoi, si non donner un contre exemple.* (2 pt)

Exercice 13 (À la fois ABR et tas ?)**.** *Un tas est nécessairement un arbre binaire quasi-complet. Est-il toujours possible d'organiser un ensemble de n clés en tas max de manière à ce que cet arbre binaire soit aussi un arbre binaire de recherche ? (Justifier par un raisonnement ou un contre-exemple).* (1 pt)

Exercice 14 (Partition d'un ABR – difficile ! –)**.** *On veut écrire une fonction de partition d'un arbre binaire de recherche autour d'un élément. Plus précisément, étant donné*

- *un arbre binaire de recherche A*
- *et un élément x de clé c,*

on veut obtenir deux ABR, G et D, tels que

- *G contient tous les éléments de A de clé inférieure à c*
- *et D tous les éléments de A de clé supérieure à c.*

Un élément de A de clé c sera placé indifféremment soit dans G soit dans D (dans un premier temps on peut considérer qu'il n'y a pas deux clés semblables).

On veut que la fonction de partition fasse le moins d'opérations possibles. On évitera en particulier l'algorithme consistant à parcourir la liste de tous les éléments de A.

Pour simplifier l'écriture en C, on peut considérer que la fonction reçoit A ainsi qu'un arbre binaire B à un seul nœud contenant l'élément x et rend son résultat en faisant de G le sous arbre gauche de B et de D le sous-arbre droit de B (remarque : l'arbre B obtenu est un arbre binaire de recherche dont les éléments sont ceux de A plus l'élément x, placé à la racine).

1. *Décrire un algorithme de partition (on pourra l'expliquer sur un exemple), et si possible écrire la fonction C (on pourra utiliser des fonctions intermédiaires).* (2.5 pt)
2. *Donner une majoration asymptotique de son temps d'exécution en fonction de la hauteur de l'arbre A.* (0.5 pt)