Nom : prénom : numéro de carte :

Institut Galilée — Algorithmique, arbres et graphes
Année 2006-2007 — L2

Examen du jeudi 31 mai 2007

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.

Comme dans le cours, on considère des éléments qui sont faits 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.

Première partie

11 pt

1 Notation asymptotique, complexité

Exercice 1. *Pour chacune des assertions suivantes, dire si elle est vraie ou fausse et* ***justifier*** *votre réponse en rappelant la définition.*

1. $\frac{n \log n}{2} = \Omega(n)$ (1 pt)
2. $\log(n!) = O(n \log n)$ (1 pt)
3. $n^3 + n^2 + n + 1 = \Theta(n^3)$ (1 pt)

Exercice 2. *Soit la fonction* MaFonction *donnée ci-dessous en pseudo-code et en langage C (au choix).*

1. *En supposant que le tableau passé en entrée est trié par ordre croissant, que renvoie exactement cette fonction (sous quel nom connaissez-vous cet algorithme) ?* (0,5 pt)
2. *Donner une majoration asymptotique, en pire cas, du temps d'exécution de* MaFonction *en fonction de la taille n du tableau en entrée. Démontrer ce résultat dans les grandes lignes (on pourra se contenter de raisonner sur les cas $n = 2^k - 1$ pour $k \geq 1$ entier).* (1.5 pt)

Fonction MaFonction(T, c)

Entrées *: Un tableau croissant d'entiers $T[1..n]$ de taille* Taille(T) $= n$ *et un entier c.*
Sorties *: Une case du tableau T ou bien une case vide.*

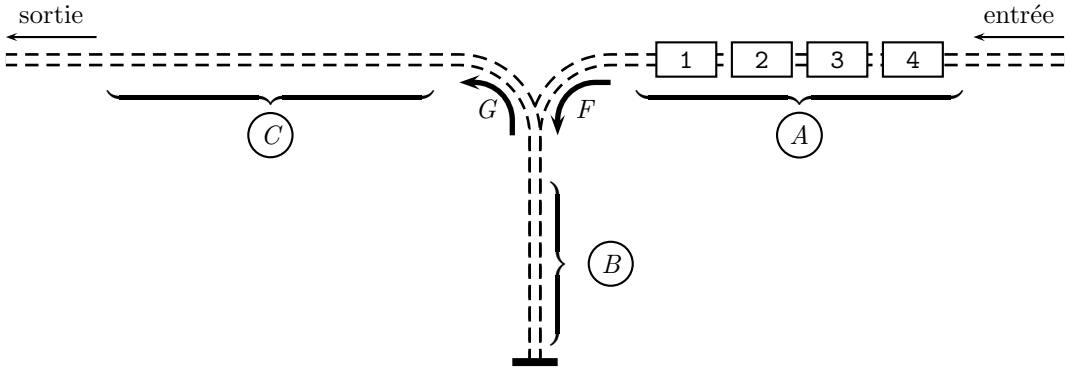
```
si Taille(T) > 0 alors
    m = ⌊Taille(T)/2⌋ (partie entière inférieure);
    si c = T[m] alors
        retourner T[m];
    sinon
        si c < T[m] alors
            T' = le sous tableau T[1..m − 1];
            retourner MaFonction(T', c);
        sinon
            T'' = le sous tableau T[m + 1..Taille(T)];
            retourner MaFonction(T'', c);
sinon
    retourner case vide;
```

```
int * mafonction(int T[], int taille, int c) {
  int milieu;
  if (taille > 0) {
    milieu = taille / 2;
    if (c == T[milieu]) {
      return &(T[milieu]);
    }
    else {
      if (c < T[milieu]) {
        return mafonction(T, milieu, c);
      }
      else { /* On a c > T[milieu] */
        return mafonction(T + milieu + 1, taille - milieu - 1, c);
      }
    }
  }
  return NULL;
}
```

Fig. 1: `mafonction` en langage C

2 Piles, files, tas

Exercice 3. *On suppose que 4 wagons numérotés de* 1 *à* 4 *sont placés en entrée sur le réseau ferroviaire suivant.*



Les actions possibles sont : Ajouter(*wagon*) *qui fait entrer un nouveau wagon sur le réseau (le wagon arrive par l'entrée dans la zone A),* Retirer() *qui fait sortir un wagon (le wagon sort de la zone C par la sortie) ainsi que* F() *qui fait passer un wagon de la zone A à la zone B et* G() *qui fait passer un wagon de la zone B à la zone C. Par exemple, la séquence d'actions :* Ajouter(1), Ajouter(2), Ajouter(3), Ajouter(4), F(), F(), F(), F(), G(), G(), G(), G(), Retirer(), Retirer(), Retirer(), Retirer() *donnera, par ordre de sorties :* 4, 3, 2, 1.

1. *Si la séquence de wagons ajoutés est* 1, 2, 3, 4 *dans cet ordre, peut-on obtenir en sortie les wagons dans l'ordre* 2, 4, 3, 1 *(expliquer) ?* (0.5 pt)
2. *Même question avec six wagons en entrée* 1, 2, 3, 4, 5, 6 *et la sortie* 3, 2, 5, 6, 4, 1 *puis la sortie* 1, 5, 4, 6, 2, 3. (1 pt)
3. *On peut modéliser le réseau comme l'assemblage de trois structures de données élémentaires, une par zone (zone A, zone B, zone C). Quelles sont ces structures de données ? Comment coder les quatre actions possibles à partir des opérations élémentaires des structures de données choisies ?* (1 pt)

4. *On suppose que l'on a une structure de données réalisant le réseau et ses quatre actions. Comment l'utiliser pour implanter une pile ? Une file ? (Décrire les opérations d'ajout et de retrait de ces deux structures de données élémentaires à partir des quatre actions).* (1 pt)
5. *Donner une séquence de wagons en entrée, la plus courte possible, et un ordre de sortie que l'on ne peut pas réaliser (expliquer).* (0.5 pt)

Exercice 4 (Insertion / suppression)**.** *On se donne le tas max (ou maximier) de la figure 2. En utilisant les algorithmes vus en cours :*

1. *Insérer un élément de clé* 17 *dans ce tas. Répondre en représentant le nouveau tas.* (0.5 pt)
2. *En repartant du tas initial, retirer l'élément de clé maximale. Répondre en représentant le nouveau tas.* (0.5 pt)

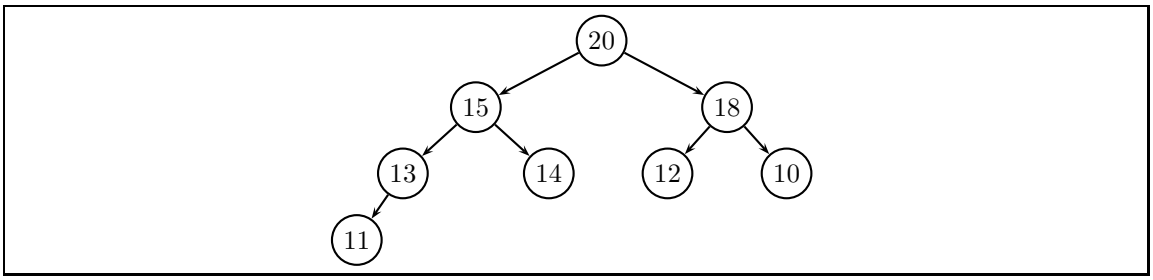


Fig. 2: Tas

3 Arbres rouge noir

Exercice 5. *Est-il possible de colorier tous les nœuds de l'arbre binaire de recherche de la figure 3 pour en faire un arbre rouge noir ?* (1 pt)

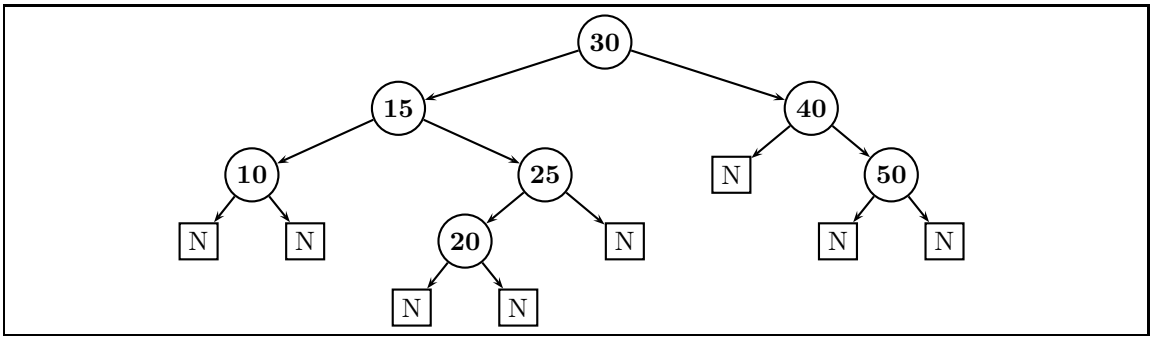


Fig. 3: Coloriage

Seconde partie

9 pt

Dans cette partie on s'intéresse au problème suivant : étant donné un ensemble A de n éléments deux à deux comparables, déterminer quel est l'élément de rang k (le k-ième plus petit élément), c'est à dire l'élément $x \in A$ tel que exactement $k-1$ éléments de A sont plus petits que x. Bien entendu, on peut supposer que k est choisi tel que $1 \leq k \leq n$. On pourra considérer que les éléments de A sont distincts (la comparaison ne les déclare jamais égaux). Si on a par exemple les éléments 23, 62, 67, 56, 34, 90, 17 (n vaut 7) alors l'élément de rang 3 est 34.

Les trois exercices suivants portent sur l'écriture d'un algorithme réalisant la fonction de recherche de l'élément de rang k dans un ensemble A, SÉLECTION(S, k), où S est la structure de donnée contenant les éléments de A.

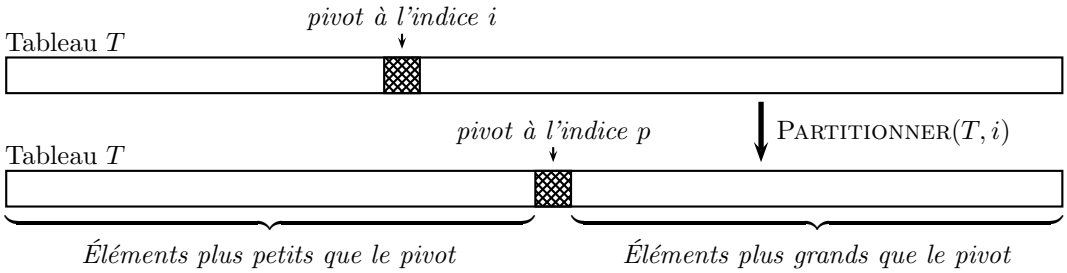
Rang dans un tableau

On suppose que les éléments de A sont donnés en entrée dans un tableau T non trié de n éléments.

Exercice 6. *Quel est le moyen* le plus simple *(3-4 lignes de pseudo-code ou de C) de réaliser la fonction* SÉLECTION(T, k) *à partir d'algorithmes du cours ? Quelle borne asymptotique minimale obtiendra t-on pour le temps d'exécution de cette fonction en fonction de n ? (Préciser le ou les algorithmes du cours que vous utilisez).* (1 pt)

Rappel sur le tri rapide. On rappelle le fonctionnement du tri rapide (*quicksort*), algorithme permettant de trier un tableau d'éléments deux à deux comparables (le résultat, c'est à dire le tableau contenant les éléments dans l'ordre est rendu en place). Le principe de fonctionnement est le suivant. Si le tableau contient un ou zéro élément, il n'y a rien à faire.

Partition Si le tableau contient plus d'un élément, on choisit (au hasard ou bien en prenant le premier élément du tableau) un élément du tableau, d'indice i, appelé *pivot* et on partitionne le tableau autour de cet élément. Plus précisément, la partition fait en sorte que, après partition, le pivot est à l'indice p, les éléments plus petits que le pivot sont (dans le désordre) aux indices inférieurs à p et les éléments plus grands que le pivot sont (dans le désordre) aux indices supérieurs à p. Le pivot est donc à sa place définitive.



Appels récursifs On relance le tri sur chacun des deux sous-tableaux obtenus par partition : le tableau des éléments d'indices inférieurs à p et le tableau des éléments d'indices supérieurs à p.

Exercice 7. *Dans cet exercice, on va mettre en œuvre un algorithme inspiré du tri rapide pour réaliser la fonction* SÉLECTION(T, k). *L'idée est qu'il n'est pas besoin de trier tout le tableau pour trouver l'élément de rang k.*

On suppose que les indices du tableau commencent à 1. *La remarque importante est que le pivot est l'élément de rang p ($p + 1$ si les indices commencent à zéro). Si après partition on trouve un p égal à k, alors l'élément de rang k est le pivot.*

On suppose que l'on a une fonction PARTITIONNER(T, i) *qui effectue la partition autour de l'élément d'indice i (pivot) et renvoie l'indice p du pivot après partition.*

1. *Après partition autour d'un pivot où chercher l'élément de rang k lorsque $k < p$? Et lorsque $k > p$? Décrire simplement le principe d'un algorithme récursif, basé sur la partition, réalisant* SÉLECTION(T, k). *(A t-on besoin de faire deux appels récursifs comme dans le tri rapide ?)* (1.5 pt)
2. *Écrire en pseudo-code ou en C, l'algorithme* SÉLECTION(T, k) *sous forme itérative.* (2 pt)
3. *Écrire l'algorithme de partionnement* PARTITIONNER(T, i). (1.5 pt)

Rang dans un arbre binaire de recherche avec comptage des descendants

On enrichit la structure de données *arbre binaire de recherche* (ABR) en ajoutant à chaque nœud x un entier $\textsc{Total}(x)$ donnant le nombre d'éléments stockés dans le sous-arbre dont le nœud x est racine, l'élément stocké dans x inclus. Exemple figure 4.

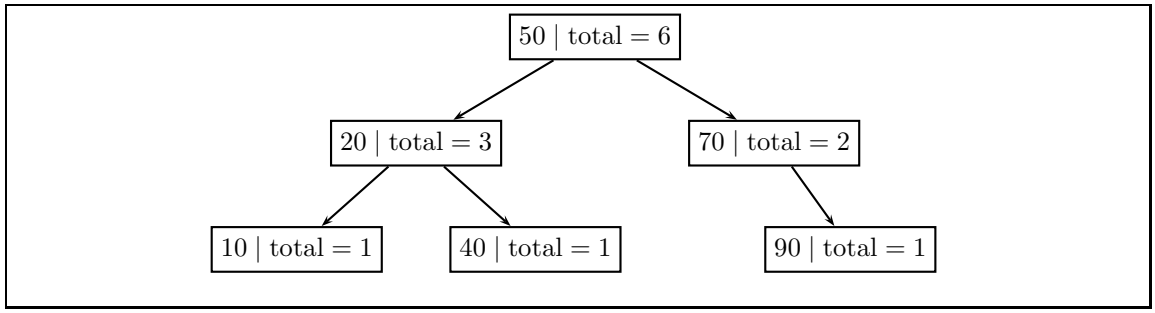


Fig. 4: Un exemple d'arbre binaire de recherche avec comptage des descendants

Dans la suite, on suppose que les éléments de A sont stockés dans un ABR enrichie par ce comptage, de racine r.

Exercice 8. *Écrire un algorithme (C ou pseudo-code) réalisant la fonction* $\textsc{Sélection}(r, k)$. *Donner en fonction de la hauteur h de l'arbre, un majorant asymptotique de son temps d'exécution.* (1.5 pt)

En guise d'aide, on rappelle, en pseudo-code, l'algorithme de recherche dans un ABR.

Fonction $\textsc{Rechercher}(r,\ c)$

Entrées : Le nœud racine r d'un arbre binaire de recherche et une clé c.
Sorties : Le nœud de l'arbre contenant l'élément de clé c s'il en existe un, ou le nœud vide N sinon.

```
si EstVide(r) alors
    retourner N ;
sinon
    si c = Clé(r) alors
        retourner r ;
    sinon
        si c < Clé(r) alors
            retourner Rechercher(Gauche(r), c) ;
        sinon
            retourner Rechercher(Droite(r), c) ;
```

Exercice 9. *Pour réaliser les ABRs avec comptage des descendants, il faut adapter les fonctions d'ajout et de suppression d'un élément. Expliquer les modifications à apporter. Et pour les rotations ?* (1.5 pt)