2009–2010

UNIVERSITÉ DE RENNES 1

Algorithmique (ALBA)

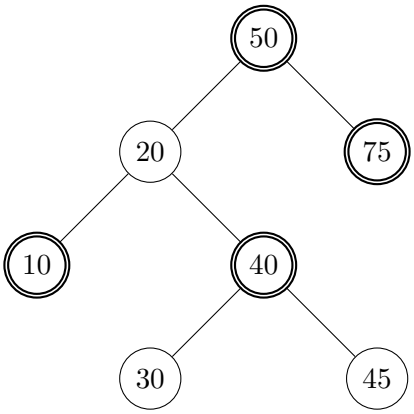
Examen (décembre 2009)

Les notes de cours, TD et TP sont autorisées. Tous les autres documents, les calculatrices et les téléphones portables sont interdits.

Durée : 2 heures

Exercice 1

On considère l'arbre rouge-noir suivant. (Les nœuds noirs sont représentés avec un trait double, les nœuds rouges avec un trait simple).



1. Vérifier que cet arbre respecte bien les propriétés définissant les arbres rouge-noir, et que c'est bien un arbre binaire de recherche pour les entiers contenus dans les nœuds (avec l'ordre usuel sur les entiers).
2. Insérer 35 dans l'arbre ci-dessus. On détaillera les opérations effectuées pour équilibrer l'arbre.

Exercice 2

Soit T un tableau indicé de 0 à $n-1$, contenant n entiers compris entre 0 et k (pour un certain entier $k > 0$). On considère l'algorithme de tri suivant.

```
R ← nouveau tableau indicé de 0 à n-1
X ← nouveau tableau indicé de 0 à k
pour i de 0 à k
    X[i] ← 0
pour i de 0 à n-1
    X[T[i]] ← X[T[i]] + 1
pour i de 1 à k
    X[i] ← X[i-1] + X[i]
i ← n
tant que i > 0
    i ← i - 1
    R[X[T[i]]] ← T[i]
    X[T[i]] ← X[T[i]] - 1
renvoyer R
```

1. Expliquer le fonctionnement de l'algorithme et montrer qu'il renvoie bien un tableau trié contenant les mêmes entiers que le tableau de départ.
2. Déterminer le complexité asymptotique de l'algorithme de tri précédent. (Les comparaisons d'entiers, additions, soustractions, multiplications et les accès en lecture ou en écriture à une case de tableau compteront comme une opération élémentaire chacun).
3. Comparer cette complexité à celle des algorithmes vus en cours, en particulier à la borne inférieure sur le complexité des tris par comparaisons.

Exercice 3

Trouver tous les entiers $x \in \mathbb{Z}$ tels que

$$\begin{cases} 9x \equiv 6 \pmod{91} \\ 7x \equiv 17 \pmod{33}. \end{cases}$$

(La démarche employée devra être clairement expliquée).

Exercice 4

Étant donnés un graphe non orienté connexe et deux nœuds de ce graphe, donner un algorithme qui détermine la longueur (en nombre d'arêtes) du plus court chemin entre ces deux nœuds. (On pourra s'inspirer de l'un des algorithmes de parcours de graphe vus en cours). L'algorithme proposé devra être écrit sous forme de pseudocode et son fonctionnement devra être clairement expliqué.