Université Claude Bernard – Lyon1 | Licence Sciences et Technologies – L3
Algorithmique, Programmation et Complexité – LIF9 | Printemps 2016

Interrogation 3 : structures arborescentes et Master Theorem

Durée : 40 minutes.

Nom :

Prénom :

Attention : le sujet est recto-verso, n'oubliez pas de tourner la page.

1 Arbres

Soit un tas binaire stocké via le tableau suivant :

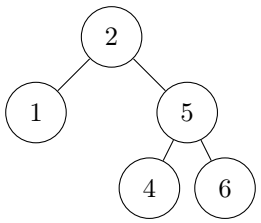
15	8	12	7	3	6	11	5	1	2	0	4

a) Dessinez ci-dessous l'arbre correspondant à ce tableau.

b) Remplissez ci-dessous le nouvel état du tableau après l'insertion de 13 dans le tas.

c) Remplissez ci-dessous le nouvel état du tableau après suppression de la racine du tas.

d) Soit l'arbre binaire de recherche suivant, indiquez sur chaque nœud son déséquilibre au sens des AVL, puis dessinez à côté l'arbre après insertion de la valeur 3 et rééquilibrage. Vous indiquerez également les opérations réalisées pour rééquilibrer l'arbre.



2 Master Theorem

On considère l'algorithme suivant (quickselect) ayant pour but de calculer la k$^{\text{ième}}$ plus petite valeur d'un tableau (numérotée à partir de 0). Par exemple, en fournissant en entrée $k = 3$ le tableau

8	22	2	4	16	14	10	0	6	20	12	18	24

l'algorithme renverra 6. Cet algorithme est très proche du tri rapide, mais ne trie que partiellement le tableau pour parvenir au résultat.

```
Fonction quickselect → nombre
    données : t un tableau, debut et fin des indices de début et fin (exclue), k l'indice recherché
    résultat : la k^ième plus petite valeur du tableau (numérotée à partir de 0)
    Algorithme
        si fin − debut = 1 alors
            retourner t[debut] /* Cas d'arrêt */
        sinon
            pivot ← t[debut] /* Pivot comme le tri rapide */
            pos ← debut
            pour i allant de debut à fin − 2 faire
                si t[pos + 1] < pivot alors /* l'élément est plus petit on le place à gauche */
                    t[pos] ← t[pos + 1]
                    pos ← pos + 1
                sinon /* l'élément est plus grand on le place à droite */
                    tmp ← t[pos + 1]
                    t[pos + 1] ← t[fin − 1 − i + pos]
                    t[fin − 1 − i + pos] ← tmp
            t[pos] ← pivot /* placement du pivot entre les plus petits et les plus grands */
            si pos < k alors
                quickselect(t, pos + 1, fin, k) /* le pivot est avant k, recherche à droite */
            sinon
                quickselect(t, debut, pos + 1, k) /* le pivot est après k, recherche à gauche */
```

Soit T une fonction de la forme

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

le master theorem affirme que :

1. si $f(n) = \mathrm{O}(n^c)$ avec $c < \log_b a$, alors la complexité de f est trop faible devant la complexité liée à la récursion. Dans ce cas
$$T(n) = \Theta(n^{\log_b a}).$$
2. si $f(n) = \Omega(n^c)$ avec $c > \log_b a$, alors la complexité de f est prépondérante devant la complexité liée à la récursion. Dans ce cas il n'est pas toujours possible de déterminer la complexité finale à cause des appels récursifs à f. Si f respecte le critère de régularité : $\exists k < 1, \exists n_0 \geq 0, \forall n \geq n_0, af(n/b) \leq kf(n)$, alors
$$T(n) = \Theta(f(n)).$$
3. si $f(n) = \Theta(n^{\log_b a})$, alors les deux éléments interagissent et la complexité finale est donnée par
$$T(n) = \Theta(n^{log_b a} \log n).$$

e) En supposant qu'à chaque étape, il y a autant d'éléments plus petits que le pivot que d'éléments plus grands, formulez la complexité de l'algorithme quickselect sous la forme $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

f) D'après le Master Theorem, quelle est la complexité de cet algorithme ? Justifiez.

g) Dans le pire cas, le tableau est trié et le pivot est systématiquement le plus grand ou le plus petit élément de la portion de tableau étudiée. Quelle est dans ce cas la complexité de l'algorithme ? (Plus difficile)

h) Dans le cas du tableau trié, avec quelle complexité pouvez vous fournir le résultat attendu ?