Algorithmes et Structures de données | 2021

Journée 3 – Vendredi 26 février

Exercices

Exercice 1 – Implémentation et test du tri par insertion en Java

Le but de cet exercice est d'implémenter et de tester l'algorithme de tri par insertion en utilisant le langage Java. Pour ce faire, il s'agit de traduire en Java l'algorithme donné en pseudo-code dans les slides, afin de trier un tableau de nombres entiers.

Exercice optionnel. Généraliser l'implémentation de cet algorithme à n'importe quel type d'objets implémentant l'interface `Comparable<T>`.

Exercice 2 – Implémentation et test du tri par fusion en Python

Le but de cet exercice est d'implémenter et de tester l'algorithme de tri par fusion en utilisant le langage Python. Pour ce faire, il s'agit de traduire en Python l'algorithme donné en pseudo-code dans les slides, afin de trier un tableau de nombres entiers.

Exercice 3 – Écriture en pseudo-code et analyse du tri par sélection

Le but de cet exercice est d'écrire en pseudo-code l'algorithme de tri par sélection (décrit ci-dessous) et de l'analyser en terme de sa correction (correctness) et de sa complexité computationnelle.

Tri par sélection. Cet algorithme consiste à trier n nombres stockés dans le tableau A en trouvant d'abord le plus petit élément de A et en l'échangeant avec l'élément en $A[1]$. Ensuite, on trouve le deuxième plus petit élément de A et on l'échange avec l'élément en $A[2]$, et ainsi de suite pour les $n - 1$ premiers éléments de A.

a. Écrire le pseudo-code de cet algorithme.

b. Quel est l'invariant de boucle que cet algorithme maintient ?

c. Pourquoi cet algorithme doit-il s'exécuter uniquement pour les $n - 1$ premiers éléments, plutôt que pour l'ensemble des n éléments ?

d. Donner les ordres de croissance du tri de sélection dans le meilleur et le pire des cas.

Exercice 3 – Écriture en pseudo-code et analyse du tri par sélection

Le but de cet exercice est d'écrire en pseudo-code l'algorithme de tri par sélection (décrit ci-dessous) et de l'analyser en terme de sa correction (correctness) et de sa complexité computationnelle.

Tri par sélection. Cet algorithme consiste à trier n nombres stockés dans le tableau A en trouvant d'abord le plus petit élément de A et en l'échangeant avec l'élément en $A[1]$. Ensuite, on trouve le deuxième plus petit élément de A et on l'échange avec l'élément en $A[2]$, et ainsi de suite pour les $n-1$ premiers éléments de A.

```
SELECTION-SORT(A)
n = A.length
for j = 1 to n − 1
    smallest = j
    for i = j + 1 to n
        if A[i] < A[smallest]
            smallest = i
    exchange A[j] with A[smallest]
```

a. Écrire le pseudo-code de cet algorithme.

b. Quel est l'invariant de boucle que cet algorithme maintient ?

L'algorithme maintient l'invariant de boucle suivant : au début de chaque itération de la boucle extérieure, le sous-tableau $A[1 \,..\, j-1]$ est constitué des $j-1$ plus petits éléments du tableau $A[1 \,..\, n]$ et ce sous-tableau est trié.

c. Pourquoi cet algorithme doit-il s'exécuter uniquement pour les $n-1$ premiers éléments, plutôt que pour l'ensemble des n éléments ?

Après les $n-1$ itération, le sous-tableau $A[1 \,..\, n-1]$ contient les $n-1$ plus petit éléments triés entre eux, donc l'élément $A[n]$ est forcément le plus grand.

d. Donner les ordres de croissance du tri de sélection dans le meilleur et le pire des cas.

Les temps d'exécution est toujours de l'ordre de $\Theta(n^2)$

Exercice 4 – Ordre de croissance

Soit deux constantes a et b telles que $a \in \mathbb{R}$, $b \in \mathbb{R}$, et $b > 0$. Montrer que :

$$(n + a)^b \in \Theta(n^b)$$

Exercice 4 – Ordre de croissance

Soit deux constantes a et b telles que $a \in \mathbb{R}$, $b \in \mathbb{R}$, et $b > 0$. Montrer que :

$$(n+a)^b \in \Theta(n^b)$$

1. Pour montrer que $(n+a)^b \in \Theta(n^b)$, il nous faut des constantes $c_1, c_2, n_0 > 0$ telles que $0 \leq c_1 n^b \leq (n+a)^b \leq c_2 n^b$ pour tout $n \geq n_0$.

2. Notons que

$$\begin{aligned} n + a &\leq n + |a| \\ &\leq 2n \qquad \text{quand } |a| \leq n \ , \end{aligned}$$

et que

$$\begin{aligned} n + a &\geq n - |a| \\ &\geq \frac{1}{2}n \qquad \text{quand } |a| \leq \tfrac{1}{2}n \ . \end{aligned}$$

Donc quand $n \geq 2\,|a|$,

$$0 \leq \frac{1}{2}n \leq n + a \leq 2n \ .$$

3. Comme $b > 0$, l'inégalité est encore valable lorsqu'on élève le tout à puissance b:

$$0 \leq \left(\frac{1}{2}n\right)^b \leq (n+a)^b \leq (2n)^b \ ,$$

$$0 \leq \left(\frac{1}{2}\right)^b n^b \leq (n+a)^b \leq 2^b n^b \ .$$

Donc, $c_1 = (1/2)^b, c_2 = 2^b,$ et $n_0 = 2\,|a|$ satisfont la définition 1