Exemples d'exercices

Exercice 1. Soient m, n deux nombres entiers positifs, A un tableau de $m \times n$ nombres entiers tels que

$$A(i,j) \leq A(i,\ell), \quad \text{pour tout} \quad 1 \leq i \leq m, \quad 1 \leq j < \ell \leq n$$

et

$$A(i,j) \geq A(k,j), \quad \text{pour tout} \quad 1 \leq i < k \leq m, \quad 1 \leq j \leq n$$

On considère l'algorithme suivant :

```
mystère
entrée : A un tableau de m × n nombres entiers vérifiant les propriétés ci-dessus et x un nombre entier
sortie : valeur binaire (oui/non)

i ⟵ 1
j ⟵ 1
Tant que i ≤ m et j ≤ n
    Si A(i,j) = x
        Sortir: oui
    Si A(i,j) < x
        j ⟵ j + 1
    Sinon
        i ⟵ i + 1
Sortir: non
```

a) Quelle est la sortie de l'algorithme **mystère** si les données d'entrée sont les suivantes ?

$$m = 3, \quad n = 4, \quad A = \begin{pmatrix} 13 & 32 & 40 & 100 \\ 12 & 17 & 39 & 65 \\ 5 & 14 & 31 & 38 \end{pmatrix} \quad \text{et} \quad x = 31$$

b) En général, dans quel cas la sortie de l'algorithme **mystère** est-elle un oui ?

c) (BONUS) Pouvez-vous donner un argument qui prouve que votre réponse à la question b) est correcte ? (et donc qui prouve que l'algorithme **mystère** fonctionne correctement)

d) Supposons que $m = n$ en entrée. Quelle est alors la complexité temporelle de l'algorithme **mystère** en fonction de n ? (utiliser la notation $\Theta(\cdot)$)

e) Supposons maintenant que $m = 2$ soit un nombre fixé. Quelle est alors la complexité temporelle de l'algorithme **mystère** en fonction de n ? (utiliser à nouveau la notation $\Theta(\cdot)$)

f) Dans ce dernier cas ($m = 2$), existe-t-il un algorithme dont la sortie soit identique à celle de l'algorithme **mystère**, mais dont la complexité temporelle soit *significativement* moindre (c'est-à-dire avec un ordre de grandeur inférieur en n) ? Si oui, écrire un tel algorithme ; si non, expliquer pourquoi un tel algorithme n'existe pas.

Exercice 2. On considère l'algorithme suivant:

mystère

entrée : *deux listes* L_1, L_2 *de nombres entiers positifs, toutes deux de taille* $n = 2^k$ *avec* $k \geq 0$*, et toutes deux ordonnées dans l'ordre croissant*

sortie : *nombre entier positif*

Si $n = 1$

  Sortir: $\frac{L_1(1)+L_2(1)}{2}$

$m \longleftarrow \frac{n}{2}$

Si $L_1(m) > L_2(m)$

  Sortir: **mystère**$(L_1(1:m), L_2(m+1:n), m)$

Sinon

  Sortir: **mystère**$(L_1(m+1:n), L_2(1:m), m)$

a) Quelle est la sortie de l'algorithme **mystère**(L_1, L_2, n) si $L_1 = (1, 3, 6, 9)$, $L_2 = (2, 4, 7, 8)$ et $n = 4$ en entrée?

b) Est-ce que la sortie de l'algorithme est modifiée si on remplace en entrée la liste $L_1 = (1, 3, 6, 9)$ par $L_1 = (1, 3, 6, 355)$?

c) Quelle est la complexité temporelle de l'algorithme **mystère**(L_1, L_2, n)? (en notation $\Theta(\cdot)$)

d) Ecrire une version itérative (c'est-à-dire non-récursive) de l'algorithme **mystère**(L_1, L_2, n).

Exercice 3. **a)** Ecrivez un algorithme qui prenne en entrée un nombre entier relatif x ainsi qu'une liste L de n nombres entiers relatifs et dont la sortie soit oui si et seulement s'il existe $i, j \in \{1, \dots, n\}$ tels que $i < j$ et $L(i) + L(j) \geq x$. De plus, la complexité temporelle de votre algorithme doit être un $\Theta(n)$.

b) On considère maintenant le problème plus général suivant:

"Etant donné un nombre entier relatif x et une liste L de n nombres entiers relatifs, existe-t-il un sous-ensemble $S \subset \{1, \dots, n\}$ tel que $\sum_{j \in S} L(j) \geq x$?"

Ce problème fait-il partie de la classe NP ? Justifiez votre réponse.

c) Sait-on si le problème de la question b) fait également partie de la classe P ? A nouveau, justifiez votre réponse.

Exercice 4. **a)** Soient L_1, L_2 deux listes de n nombres entiers positifs, chacune ordonnée dans l'ordre croissant. Ecrire un algorithme de complexité temporelle $\Theta(n)$ ou $\Theta(n \cdot \log_2(n))$ dont la sortie soit oui si et seulement s'il existe $i, j \in \{1, \dots, n\}$ tels que $L_1(i) = L_2(j)$ [*Note:* on n'exclut pas ici le cas $i = j$].

b) Si maintenant la liste L_1 est de taille n et la liste L_2 est de taille 2^n (et chacune des deux listes est toujours ordonnée dans l'ordre croissant), existe-il un algorithme de complexité polynomiale en n dont la sortie soit oui si et seulement s'il existe $i \in \{1, \dots, n\}$ et $j \in \{1, \dots, 2^n\}$ tels que $L_1(i) = L_2(j)$? Si oui, écrivez un tel algorithme; si non, expliquez pourquoi ce n'est pas possible.

Exercice 5. On considère l'algorithme suivant:

algorithme

entrée : *nombre entier strictement positif n*
sortie : *???*

Si $n = 1$
  Sortir : 1
Si n *est pair*
  Sortir : 1+ **algorithme**$(n/2)$
Sinon
  Sortir : algorithme$(n-1)$

a) Quelle est la sortie de cet algorithme si $n = 32$ en entrée?

b) Quelle est la sortie de cet algorithme si $n = 31$ en entrée?

c) Pour une valeur quelconque de $n \geq 1$ en entrée, que représente la sortie de cet algorithme?

d) Quelle est la complexité temporelle de cet algorithme? (utiliser la notation $\Theta(\cdot)$)

Exercice 6. On considère les deux algorithmes suivants:

machin

entrée : *nombre entier positif n*
sortie : *???*

Si $n = 1$
  Sortir: 1
Sortir : $2^{(\mathbf{bidule}(n-1)+1)}$

bidule

entrée : *nombre entier positif n*
sortie : *???*

Si $n = 1$
  Sortir: 0
Sortir : $\log_2(\mathbf{machin}(n-1))$

a) Quelle est la sortie de l'algorithme **bidule** lorsque $n = 4$ en entrée?

b) Quelle est la complexité temporelle de l'algorithme **bidule**? (utiliser la notation $\Theta(\cdot)$)

c) Réécrire l'algorithme **machin** de façon récursive, mais sans faire appel à **bidule**.

d) En déduire quelle est la sortie de **machin** pour une valeur quelconque de $n \geq 1$ en entrée.