EPFL - Information, Calcul et Communication - Partie théorique

Anciens examens

1 Algorithmique

Exercice 1 (automne 2020-2021).

a) Ecrire un algorithme **test**(L, n), *de complexité temporelle* $\Theta(n)$, qui prenne en entrée une liste L de nombres entiers ainsi que sa taille n, et dont la sortie soit oui si et seulement si la liste L est ordonnée dans l'ordre croissant *ou* dans l'ordre décroissant.

Exemples: Si $L = (2, 2, 2, 3, 8)$ ou $L = (10, 5, 4, 4, 2)$ (avec ici $n = 5$), alors la sortie doit être oui. Si par contre $L = (1, 4, 3, 7)$ (avec $n = 4$), alors la sortie doit être non.

b) On considère maintenant une liste L de taille n pour laquelle l'algorithme de la partie **a)** produit le résultat **test**$(L, n) =$ oui.

Ecrire un algorithme **tri**(L, n), *de complexité temporelle au plus* $\Theta(n)$, qui prenne en entrée la liste L ainsi que sa taille n, et dont la sortie soit une version de cette liste triée dans l'ordre croissant.

c) En général, il n'existe pas d'algorithme de complexité temporelle $\Theta(n)$ qui permette de trier une liste L quelconque de taille n. Ceci est dû au fait que, contrairement à la situation particulière de la partie **b)**, les éléments d'une liste L quelconque peuvent être ordonnés d'un grand nombre de façons. Par exemple, si L est une liste composée des 3 éléments 2, 4 et 7, alors ces éléments peuvent être ordonnés de 6 façons différentes:

$$L = (2, 4, 7) \quad L = (2, 7, 4) \quad L = (4, 2, 7) \quad L = (4, 7, 2) \quad L = (7, 2, 4) \quad \text{ou} \quad L = (7, 4, 2)$$

Si maintenant L est une liste composée de n éléments différents, de combien de façons différentes peut-on les ordonner?

d) Est-ce que $\Theta(n \cdot \log_2(n))$ bits suffisent pour représenter tous ces ordres différents? Justifier.

Exercice 2 (printemps 2020-2021).

On considère l'algorithme récursif suivant:

algo_rec

entrée : *nombre entier positif* $N \geq 1$
sortie : *??*

Si $N = 1$
    sortir : 1
Si N *est pair*
    sortir : **algo_rec**$(N/2)$
Sinon
    sortir : $1 +$ **algo_rec**$((N-1)/2)$

a) Que vaut la sortie de cet algorithme lorsque $N = 10$ en entrée ?

b) Pour une valeur quelconque de N en entrée, que représente la sortie de cet algorithme ?

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

d) Proposer une version itérative de cet algorithme.

Exercice 3a (automne 2021-2022).

a) Soit L une liste de nombres entiers positifs tous différents les uns des autres et ordonnés dans l'ordre croissant; soit également n la taille de la liste L et x un nombre entier positif.

Ecrire un algorithme **algo1**(L, n, x), *de complexité temporelle* $\Theta(n)$, qui prenne en entrée la liste L, sa taille n ainsi que le nombre x, et dont la sortie soit le nombre de paires $\{i, j\} \subset \{1, \ldots, n\}$ avec $i < j$ telles que $L(i) + L(j) = x$.

Exemple: Si $L = (2, 4, 6, 8, 9, 10)$, $n = 6$ et $x = 14$, alors la sortie de **algo1**(L, n, x) doit être 2.

b) Supposons maintenant que les nombres de la liste L ne soient pas forcément tous différents les uns des autres, ni triés dans l'ordre croissant. Proposez un nouvel algorithme **algo2**(L, n, x) dont la sortie soit, dans ce cas plus général, la même que celle demandée au point a). Quelle est la complexité temporelle de votre algorithme? (utiliser la notation $\Theta(\cdot)$)

Exercice 3b (automne 2021-2022).

On considère l'algorithme suivant:

algo

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

```
Si n = 0
    Sortir : 1
s ⟵ 0
Pour k allant de 0 à n − 1
    s ⟵ s + algo(k)
Sortir : s
```

a) Pour une valeur d'entrée $n \geq 1$, quelle est la sortie de cet algorithme?

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

Exercice 4 (printemps 2021-2022).

a) Écrire un algorithme **algo1**(L, n) qui prenne en entrée une liste L de n nombres entiers relatifs (par exemple, $L = (+4, -12, +17, +3, -15, +34, -8, +16)$, pour $n = 8$) et dont la sortie soit la valeur maximale de la somme

$$\sum_{k=i}^{j} L(k)$$

parmi toutes les paires d'indices $i, j \in \{1, \ldots, n\}$ avec $i \leq j$ (dans l'exemple ci-dessus, la sortie doit donc être $+47$, valeur atteinte pour $i = 3$ et $j = 8$).

b) Quelle est la complexité temporelle de votre algorithme **algo1**(L, n) ?
(utiliser la notation $\Theta(\cdot)$)

On considère maintenant l'algorithme suivant:

algo2
entrée : *liste M de n nombres entiers relatifs* sortie : *???*
Pour *i allant de 1 à $n-1$* $L(i) \longleftarrow M(i+1) - M(i)$ **Sortir : algo1**$(L, n-1)$

c) Que représente la sortie de l'algorithme **algo2**(M, n) pour une liste M donnée ?

d) Quelle est la complexité temporelle de l'algorithme **algo2**(M, n) ?
(utiliser la notation $\Theta(\cdot)$)

Exercice 5a (automne 2022-2023).

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 $i, j \in \{1, \ldots, n\}$ 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 $i \in \{1, \ldots, n\}$ et $j \in \{1, \ldots, 2^n\}$ tels que $L_1(i) = L_2(j)$? Si oui, écrivez un tel algorithme; si non, expliquez pourquoi ce n'est pas possible.

Exercice 5b (automne 2022-2023).

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 (printemps 2022-2023).

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 $i, j \in \{1, \ldots, n\}$ 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 $S \subset \{1, \ldots, n\}$ 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 7 (automne 2023-2024).

Considérons 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: (L1(1)+L2(1))/2
m ⟵ n/2
Si L1(m) > L2(m)
    Sortir: mystère(L1(1 : m), L2(m + 1 : n), m)
Sinon
    Sortir: mystère(L1(m + 1 : n), L2(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 8 (printemps 2023-2024).

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

d) 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 $\Theta(\cdot)$)

e) 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 9a (automne 2024-2025).

On considère l'algorithme suivant:

algo

entrée : *liste L de nombres entiers positifs, de taille n*
sortie : *???*

Si $n = 1$
  Sortir: $L(1)$

$m \longleftarrow \lfloor \frac{n}{2} \rfloor$
Sortir: $\frac{m}{n} \cdot$ **algo**$(L(1:m), m) + \frac{n-m}{n} \cdot$ **algo**$(L(m+1:n), n-m)$

Note: $L(a:b) = (L(a), L(a+1), \ldots, L(b-1), L(b))$ désigne la sous-liste de L composée des éléments compris entre l'indice a et l'indice b (compris).

a) Que vaut la sortie de l'algorithme **algo**(L, n) si $L = (8, 10, 5, 9)$ et $n = 4$ en entrée?

b) Que vaut la sortie de l'algorithme **algo**(L, n) pour une liste L de taille n en général?

c) Quelle est la complexité temporelle de **algo**(L, n)? (utiliser la notation $\Theta(\cdot)$ et justifier votre réponse; attention, car la réponse à la question n'est pas immédiate!)

Note: On considère ici que l'opération $\frac{m}{n} \cdot a + \frac{n-m}{n} \cdot b$ pour deux nombres a, b arbitraires compte comme une seule opération.

Supposons maintenant qu'on modifie l'algorithme précédent comme suit:

algobis

entrée : *liste L de nombres entiers positifs, de taille n*
sortie : *???*

Si $n = 1$
  Sortir: $L(1)$

$m \longleftarrow n - 1$
Sortir: $\frac{m}{n} \cdot$ **algobis**$(L(1:m), m) + \frac{n-m}{n} \cdot$ **algobis**$(L(m+1:n), n-m)$

d) Que vaut la sortie de ce nouvel algorithme **algobis**(L, n) en général? Justifier votre réponse.

e) Quelle est la complexité temporelle de ce nouvel algorithme **algobis**(L, n)? (utiliser à nouveau la notation $\Theta(\cdot)$ et justifier votre réponse)

Exercice 9b (automne 2024-2025).

a) Ecrire un algorithme qui prenne en entrée une liste L de nombres entiers relatifs, de taille n, dont la sortie soit oui si et seulement si

$$\sum_{i=1}^{k} L(i) \geq 0, \quad \forall 1 \leq k \leq n \tag{1}$$

et dont la complexité temporelle soit un $\Theta(n)$.

b) On considère maintenant l'algorithme suivant:

algo

entrée : *liste L de nombres entiers relatifs, de taille n*
sortie : *oui/non*

```
Pour i allant de 1 à n
    Si L(i) < 0
        Sortir : non
Sortir: oui
```

Ecrire un algorithme dont les entrées et sortie soient identiques à celui de la partie a), mais qui utilise explicitement l'algorithme **algo** ci-dessus comme sous-algorithme.

Note: Bien sûr, votre algorithme fera appel à **algo** avec une liste L différente de celle pour laquelle il cherche à savoir si la relation (1) de la page précédente est vérifiée ou non.