EPFL - Information, Calcul et Communication - Partie théorique

Anciens examens - Solutions

1 Algorithmique

Exercice 1 (automne 2020-2021).

a)

test

entrée : *L liste de nombres entiers, n nombre entier positif*
sortie : *valeur binaire oui/non*

```
s1 ⟵ oui
Pour i allant de 2 à n
    Si L(i) < L(i − 1)
        s1 ⟵ non
s2 ⟵ oui
Pour i allant de 2 à n
    Si L(i) > L(i − 1)
        s2 ⟵ non
Si s1 = oui ou s2 = oui
    Sortir : oui
Sinon
    Sortir : non
```

b)

tri

entrée : *L liste de nombres entiers ayant passé le test ci-dessus, n nombre entier positif*
sortie : *liste triée dans l'ordre croissant*

```
Si L(1) ≤ L(n)
    Sortir : L
Sinon
    Pour i allant de 1 à n
        M(i) ⟵ L(n + 1 − i)
    Sortir : M
```

c) Le nombre d'ordres possibles vaut $n!$

d) $n! \leq n^n$, et donc $\log_2(n!) \leq n \cdot \log_2(n)$, ce qui prouve que $n \cdot \log_2(n)$ bits suffisent pour représenter tous les ordres différents.

Exercice 2 (printemps 2020-2021).

a) La sortie vaut 2.

b) Le nombre de 1 dans la décomposition binaire du nombre N.

c) $\Theta(\log_2(N))$

d)

algo_rec

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

```
s ⟵ 1
Tant que N > 1
    Si N est impair
        s ⟵ s + 1
    N ⟵ ⌊N/2⌋
sortir : s
```

Exercice 3a (automne 2021-2022).

a) (= une possibilité parmi d'autres)

algo1

entrée : *liste L, de taille n, nombre entier positif x*
sortie : *nombre entier positif ou nul*

```
s ⟵ 0
i ⟵ 1
j ⟵ n
Tant que i < j
    Si L(i) + L(j) = x
        s ⟵ s + 1
        i ⟵ i + 1
        j ⟵ j − 1
    Sinon, si L(i) + L(j) > x
        j ⟵ j − 1
    Sinon
        i ⟵ i + 1
Sortir : s
```

b) Ici, on a deux possibilités:

1) trier la liste et imiter l'algorithme précédent en gérant les égalités, mais cette gestion des égalités peut s'avérer délicate (si par exemple la liste $L = (4, 4, 10, 10)$ et $x = 14$, alors il y a 4 paires différentes $\{i, j\}$ avec $i < j$ telles que $L(i) + L(j) = x$).

2) plus simplement, chercher toutes les paires avec deux boucles imbriquées ($\longrightarrow$ complexité $\Theta(n^2)$), ce qui non seulement permet de gérer les égalités, mais ne nécessite même pas de trier la liste au préalable:

algo2

entrée : *liste L, de taille n, nombre entier positif x*
sortie : *nombre entier positif ou nul*

```
s ⟵ 0
Pour i allant de 1 à n − 1
    Pour j allant de i + 1 à n
        Si L(i) + L(j) = x
            s ⟵ s + 1
Sortir : s
```

Exercice 3b (automne 2021-2022).

a) 2^{n-1} (notez que ceci vaut pour $n \geq 1$ uniquement, car pour $n = 0$, la sortie vaut 1)

Voici les calculs successifs menant à cette conclusion:

algo(1) = algo(0) =1

algo(2) = algo(0) + algo(1) = 2

algo(3) = algo(0) + algo(1) + algo(2) = 4

et en général:

algo(n) = algo(0) + algo(1) + $\ldots$ + algo($n-1$) = algo($n-1$) + algo($n-1$) = 2 * algo($n-1$)

donc algo(n) = 2^{n-1} [formellement, si on suppose que algo(0) = 1 et algo(k) = 2^{k-1} pour $0 < k < n$, ceci donne algo(n) = $1 + 1 + 2 + \ldots + 2^{n-2} = 2^{n-1}$, montrant le résultat par récurrence].

b) $\Theta(2^n)$

Les calculs de algo(0) et algo(1) demandent chacun 1 opération. Le calcul de algo(2) = algo(0) + algo(1) demande donc 2 opérations; celui de algo(3) = algo(0) + algo(1) + algo(2) demande 4 opérations; celui de algo(4) demande 1 + 1 + 2 + 4 = 8 opérations, et ainsi de suite, donc algo(n) demande $2^{n-1} = \Theta(2^n)$ opérations.

Exercice 4 (printemps 2021-2022).

a)

algo1
entrée : *liste L de n nombres entiers relatifs* sortie : $\max_{i,j \in \{1,\ldots,n\} : i \leq j} \sum_{k=i}^{j} L(k)$
$t \longleftarrow L(1)$ **Pour** *i allant de 1 à n* $\quad s \longleftarrow 0$ $\quad$ **Pour** *j allant de i à n* $\quad\quad s \longleftarrow s + L(j)$ $\quad\quad$ **Si** $s > t$ $\quad\quad\quad t \longleftarrow s$ **Sortir :** t

b) $\Theta(n^2)$

c) La sortie vaut (notez que l'algorithme ne donne rien d'intéressant pour $n = 1$)

$$\max_{i,j \in \{1,\ldots,n-1\} : i \leq j} M(j+1) - M(i)$$

d) $\Theta(n^2)$

Exercice 5a (automne 2022-2023).

a) Deux possibilités. La première en $\Theta(n)$:

algorithme

entrée : *Listes L_1, L_2 ordonnées dans l'ordre croissant, chacune de taille n*
sortie : *oui/non*

```
i ⟵ 1
j ⟵ 1
Tant que i ≤ n et j ≤ n
    Si L1(i) = L2(j)
        Sortir : oui
    Si L1(i) > L2(j)
        j ⟵ j + 1
    Sinon
        i ⟵ i + 1
Sortir : non
```

La seconde en $\Theta(n \log_2(n))$:

algorithme

entrée : *Listes L_1, L_2 ordonnées dans l'ordre croissant, chacune de taille n*
sortie : *oui/non*

```
Pour i allant de 1 à n
    s ⟵ recherche dichotomique(L2, n, L1(i))
    Si s = oui
        Sortir : oui
Sortir : non
```

b) Oui, c'est possible. Il faut appliquer la seconde solution ci-dessus:

algorithme

entrée : *Listes L_1, L_2 ordonnées dans l'ordre croissant, de tailles n et 2^n, respectivement*
sortie : *oui/non*

```
Pour i allant de 1 à n
    s ⟵ recherche dichotomique(L2, 2^n, L1(i))
    Si s = oui
        Sortir : oui
Sortir : non
```

La complexité temporelle de cet algorithme est en $\Theta(n^2)$.

Exercice 5b (automne 2022-2023).

a) La sortie vaut 6.

b) La sortie vaut 5.

c) Le nombre de bits nécessaires pour représenter le nombre entier positif n; de manière équivalente: $\lceil \log_2(n+1) \rceil$ ou encore $\lfloor \log_2(n) \rfloor + 1$.

d) $\Theta(\log_2(n))$ [Dans le pire des cas, n est est divisé par 2 après 2 étapes de l'algorithme.]

Exercice 6 (printemps 2022-2023).

a) Un algorithme possible est le suivant:

algo

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

```
y ⟵ max(L(1), L(2))
z ⟵ min(L(1), L(2))
Pour i allant de 3 à n
    Si L(i) ≥ y
        z ⟵ y
        y ⟵ L(i)
    Sinon, si L(i) ≥ z
        z ⟵ L(i)
Si y + z ≥ x (Note: y et z sont les deux plus grands nombres de la liste L)
    Sortir : oui
Sinon
    Sortir : non
```

b) Oui. Si on nous propose un sous-ensemble S comme solution, il suffit alors d'effectuer la somme $\sum_{j \in S} L(j)$ et de comparer celle-ci à la valeur x (ce qui se fait en temps polynomial, même en $\Theta(n)$).

c) Oui, il fait partie de la classe P:

si $x \geq 0$, il suffit de parcourir une seule fois la liste L (complexité $\Theta(n)$) et de ne retenir que les nombres positifs ou nuls de celle-ci. Si leur somme est plus grande ou égale à x, la réponse est oui; sinon, la réponse est non.

si $x < 0$, il suffit à nouveau de parcourir une seule fois la liste et d'identifier s'il existe $i \in \{1, \ldots, n\}$ tel que $L(i) \geq x$, auquel cas la réponse est oui avec $S = \{i\}$; sinon, la réponse est non.

Exercice 7 (automne 2023-2024).

a) La sortie vaut 5.

b) La sortie ne change pas.

c) La complexité temporelle est un $\Theta(\log_2(n))$.

d) Voici une possibilité:

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*

Tant que $n > 1$
  $m \longleftarrow \lfloor \frac{n}{2} \rfloor$
  Si $L_1(m) > L_2(m)$
    $L_1 \longleftarrow L_1(1 : m)$
    $L_2 \longleftarrow L_2(m + 1 : n)$
  Sinon
    $L_1 \longleftarrow L_1(m + 1 : n)$
    $L_2 \longleftarrow L_2(1 : m)$
  $n \longleftarrow m$
Sortir: $\frac{L_1(1)+L_2(1)}{2}$

Exercice 8 (printemps 2023-2024).

a) oui

b) L'algorithme **mystère** sort oui si et seulement si le nombre x se trouve dans le tableau A.

c) $\Theta(n)$

d) également $\Theta(n)$

e) Oui, c'est possible. Voici un algorithme de complexité temporelle $\Theta(\log_2(n))$ qui fonctionne, car les deux lignes du tableau $A(1)$ et $A(2)$ sont chacune triées dans l'ordre croissant:

algorithme

entrée : *tableau A de $2 \times n$ nombres entiers, nombre entier x*
sortie : *valeur binaire (oui/non)*

$s_1 \longleftarrow$ **recherche dichotomique**$(A(1), n, x)$
$s_2 \longleftarrow$ **recherche dichotomique**$(A(2), n, x)$
Si $s_1 =$ *oui ou* $s_2 =$ *oui*
  Sortir: *oui*
Sinon
  Sortir: *non*

Exercice 9a (automne 2024-2025).

a) La sortie vaut 8.

b) La sortie est égale à la moyenne des valeurs de L.

c) La complexité temporelle est un $\Theta(n)$; en effet, $n-1$ additions seront effectuées au total.

d) La sortie ne change pas: c'est toujours la moyenne des valeurs de L.

e) La complexité temporelle ne change pas non plus: c'est toujours un $\Theta(n)$, car à nouveau, $n-1$ addditions sont effectuées au total.

Exercice 9b (automne 2024-2025).

a)

algo1

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

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

b)

algo2

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

```
M ⟵ (L(1))
Pour i allant de 2 à n
    M ⟵ M + (M(i − 1) + L(i))  (= "ajouter l'élément M(i-1)+L(i) à la liste M")
Sortir: algo(M, n)
```