

# Solutions : Chapitre 5

## Question # 1

Soit  $A[0 \dots n - 1]$ , un tableau de  $n$  éléments comparables (Pour simplifier, on suppose ici que les éléments sont tous distincts). Dans le tri par insertion (voir Algorithme 1), une paire d'indices  $(i, j)$  est une inversion si  $i < j$  et  $A[i] > A[j]$ .

---

### Algorithme 1 : TriParInsertion( $A[0..n - 1]$ )

---

```

1 pour  $i = 1..n - 1$  faire
2    $v \leftarrow A[i]$  // Élément à insérer
3    $j \leftarrow i - 1$  // Positions possibles d'insertion
4   tant que  $j \geq 0 \wedge A[j] > v$  faire
5      $A[j + 1] \leftarrow A[j]$ 
6      $j \leftarrow j - 1$ 
7    $A[j + 1] \leftarrow v$ 
8 retourner  $A$ 
```

---

- A) Décrivez le tableau ayant le plus grand nombre d'inversions pour une taille  $n$  donnée. Quel est ce nombre d'inversions ?

**Solution :**

Le plus grand nombre d'inversions que peut avoir  $A[i]$  ( $0 \leq i \leq n - 1$ ) avec les éléments à la droite de  $A[i]$  est  $n - 1 - i$ . Ceci arrive lorsque  $A[i]$  est plus grand que tous les éléments à sa droite. Donc, le plus grand nombre d'inversions pour un tableau complet arrive lorsqu'on a un tableau trié en ordre décroissant. Le nombre total d'inversions est donc :

$$\sum_{i=0}^{n-1} (n - 1 - i) = \frac{n(n - 1)}{2}$$

- B) Décrivez le tableau ayant le plus petit nombre d'inversions pour une taille  $n$  donnée. Quel est ce nombre d'inversions ?

**Solution :**

Le plus petit nombre d'inversion pour  $A[i]$  ( $0 \leq i \leq n - 1$ ) est clairement 0. Ceci arrive lorsque  $A[i]$  est plus petit ou égal que tous les éléments à sa droite. Le plus petit nombre d'inversion pour un tableau complet est donc de 0 pour les tableau en ordre croissant.

- C) Analysez l'efficacité de l'algorithme en cas moyen en prenant la comparaison « $A[j] > v$ » comme opération de base. Supposez que tous les éléments de  $A$  sont distincts.

**Solution :**

Nous avons que  $C_{avg}(n)$  est le nombre moyen de comparaisons. Pour un tableau  $A$  quelconque, l'insertion de l'élément  $A[i]$  à la position  $j \in \{0, \dots, i\}$  est équiprobable pour tout  $j$ . Soit  $C_{avg}^{insertion}(i)$  le nombre moyen de comparaisons nécessaires pour insérer  $i$  à la bonne position. Alors, nous avons

$$\begin{aligned}
 C_{avg}^{insertion}(i) &= \sum_{j=0}^i \frac{1}{i+1} (i - j + 1) = \frac{1}{i+1} \sum_{j=0}^i (i + 1) - \frac{1}{i+1} \sum_{j=0}^i j \\
 &= \sum_{j=0}^i 1 - \frac{1}{(i+1)} \frac{i(i+1)}{2} = i - \frac{i}{2} + 1 = \frac{i}{2} + 1
 \end{aligned}$$

Par conséquent, nous avons que le nombre moyen de comparaisons est

$$\begin{aligned}
 C_{avg}(n) &= \sum_{i=1}^{n-1} C_{avg}^{insertion}(i) = \sum_{i=1}^{n-1} \left( \frac{i}{2} + 1 \right) \\
 &= \frac{1}{2} \sum_{i=1}^{n-1} i + \sum_{i=1}^{n-1} 1 = \frac{n(n-1)}{4} + (n-1-1+1) \\
 &= \frac{1}{4}n^2 + \frac{3}{4}n - 1 \in \Theta(n^2)
 \end{aligned}$$

### Question # 2

Écrivez un algorithme non-récursif qui résout le problème de sélection en se basant sur les partitions. Le problème de sélection est celui de trouver le  $k$ ième plus petit élément dans un tableau de  $n$  nombres.

**Solution :**

---

**Algorithme 2 :** Selection( $A[1 \dots n]$ ,  $k$ )

---

```

1  $l \leftarrow 1$ 
2  $r \leftarrow n$ 
3 tant que  $l < r$  faire
4    $s \leftarrow Partition(A[l \dots r])$ 
5   si  $s = k$  alors
6     retourner  $A[s]$ 
7   sinon si  $s > k$  alors
8      $r \leftarrow s - 1$ 
9   sinon
10     $l \leftarrow s + 1$ 
11 retourner  $A[l]$ 

```

---

### Question # 3

Vous avez un ensemble de pièces d'or  $P[1, \dots, n]$ . Toutes ces pièces sont identiques à l'exception d'une unique fausse pièce, qui est supposée plus légère que les autres. Vous avez à votre disposition une balance pouvant uniquement comparer deux poids l'un contre l'autre. Vous pouvez mettre plusieurs pièces sur la balance. Par conséquent, l'algorithme Balance( $A, B$ ) retourne 0 si les poids sont égaux et  $-1$  si  $A$  est plus léger que  $B$  et 1 si  $A$  est plus lourd que  $B$ . Écrivez un algorithme qui trouve la fausse pièce avec le minimum d'appel à l'algorithme Balance. Votre algorithme doit être en  $\Theta(\log(n))$ . Faites l'analyse de l'efficacité de votre algorithme.

**Solution :**

---

**Algorithme 3 :** TrouverFaussePiece( $P[1..n]$ )

---

```
1 //  $P$  est un ensemble non vide de  $n$  pièces d'or, dont une seule est fausse.  
2 si  $n = 1$  alors  
3   retourner  $P[1]$   
4 si  $n = 2$  alors  
5   si  $\text{Balance}(P[1..1], P[2..2]) = -1$  alors  
6     retourner  $P[1]$   
7   sinon  
8     retourner  $P[2]$   
9 suivant  $\text{Balance}(P[1..\lfloor n/3 \rfloor], P[(\lfloor n/3 \rfloor + 1)..2\lfloor n/3 \rfloor])$  faire  
10  cas où  $-1$  faire  
11    retourner TrouverFaussePiece( $P[1..\lfloor n/3 \rfloor]$ )  
12  cas où  $1$  faire  
13    retourner TrouverFaussePiece( $P[(\lfloor n/3 \rfloor + 1)..2\lfloor n/3 \rfloor]$ )  
14  cas où  $0$  faire  
15    retourner TrouverFaussePiece( $P[(2\lfloor n/3 \rfloor) + 1..n]$ )
```

---

Premièrement, il faut remarquer que peu importe les pièces dans  $P[1..n]$ , on réduit la taille de l'instance de  $2/3$  à chaque appel récurrent. Par conséquent, l'efficacité de l'algorithme dépend uniquement de la taille de l'instance  $n$ .

L'opération de base choisie est l'appel à l'algorithme  $\text{Balance}(A,B)$ , puisque c'est ce que l'on nous demande de minimiser dans l'énoncé. La relation de récurrence associée est la suivante.

$$C(n) = \begin{cases} 0 & \text{si } n = 1 \\ 1 & \text{si } n = 2 \\ C(\lfloor n/3 \rfloor) + 1 & \text{si } n > 2 \end{cases}$$

En supposant que  $n = 3^k$  et en utilisant le théorème général, puisque  $r = 1$ ,  $b = 3$  et  $d = 0$ , nous obtenons que  $C(n) \in \Theta(n^d \log(n)) = \Theta(\log(n))$ ,  $\forall n \in \mathbb{N}$ .

**Question # 4**

Un groupe de personnes se réunissent à l'occasion d'une oeuvre de charité. Vous êtes en charge de remettre un colis à une personne selon les indications suivantes :

1. elle ne connaît personne parmi les invités ;
2. elle est connue de tous les invités.

Votre mission est de trouver le destinataire du colis en posant le moins de questions possibles aux invités, histoire de ne pas trop vous faire remarquer. Écrivez un algorithme trouvant la personne cherchée. Vous pouvez faire appel à l'algorithme  $\text{Connait}(A,B)$  qui retourne vrai si  $A$  connaît  $B$  et faux sinon.

**Solution :**

Supposons que l'algorithme  $\text{Pile}(A)$  transforme un tableau  $A$  en pile. Ceci peut être fait en temps linéaire par rapport à la taille de  $A$ . Nous utilisons également les méthodes  $\text{pile.pop}$  pour retirer l'élément sur le dessus de la pile ainsi que  $\text{pile.push}(a)$  pour ajouter l'élément  $a$  sur le dessus de la pile. Ces deux méthodes s'effectuent en temps constant.

---

**Algorithme 4 :** TrouverDestinataire( $P[0..n - 1]$ )

---

```
1 candidat ← 0;
2 // On trouve un candidat;
3 pour i = 1..n - 1 faire
4   si ¬Connait(P[i], P[candidat]) alors
5     candidat ← i;
6 // Tout le monde connaît le candidat;
7 pour i = 0..candidat - 1 faire
8   si ¬Connait(P[i], P[candidat]) alors
9     retourner-1;
10 // Le candidat ne connaît personne;
11 pour i = 0..n - 1 faire
12   si i ≠ candidat ∧ Connait(P[candidat], P[i]) alors
13     retourner-1;
14 retourner candidat;
```

---

L'opération de base choisie est le nombre de «questions», c'est-à-dire le nombre d'appels à l'algorithme Connait( $A, B$ ).

La première boucle fait  $n - 1$  appels. La deuxième boucle en fait au plus  $n - 1$ . La troisième boucle fait  $n - 1$  appels. En pire cas, il y a  $3n - 3 \in \Theta(n)$  appels. En meilleur cas, la deuxième boucle s'arrête à la première itération. Il y a donc  $n - 1 + 1 = n \in \Theta(n)$  appels.

**Question # 5**

Vous devez analyser l'algorithme Mystère1. Si la complexité de l'algorithme varie parmi les instances de même taille, identifiez le pire et le meilleur cas. Procédez ensuite à une analyse en pire cas, en meilleur cas et en cas moyen. Si la complexité de l'algorithme est la même pour toutes les instances d'une même taille, alors donnez la complexité de cet algorithme en fonction de  $n$ . Toutes vos réponses doivent utiliser la notation  $\Theta$ .

---

**Algorithme 5 :** Mystère1( $A[0..n - 1]$ )

---

```
1 ρ ← Uniforme(1, n)
2 c ← 0
3 pour i from 1 to nρ faire
4   c ← c + A[i mod n]
5 retourner c
```

---

**Solution :**

La complexité de l'algorithme Mystère1 dépend de  $\rho$  généré de façon pseudo-aléatoire et de la taille de l'instance  $n$ . Cependant, pour une taille d'instance  $n$ , il est impossible de modifier l'instance de façon à accélérer ou ralentir l'exécution de l'algorithme. Il n'y a donc ni pire ni meilleur cas. L'opération de base est l'addition. La complexité de l'algorithme dans tous les cas est donc la suivante.

$$T(n) = E_\rho[n\rho] = \sum_{\rho=1}^n \frac{1}{n} n\rho = \sum_{\rho=1}^n \rho = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n \in \Theta(n^2) \quad \text{Règle du maximum}$$

**Question # 6**

Vous devez analyser l'algorithme Mystère2. Si la complexité de l'algorithme varie parmi les instances de

même taille, identifiez le pire et le meilleur cas. Procédez ensuite à une analyse en pire cas, en meilleur cas et en cas moyen. Si la complexité de l'algorithme est la même pour toutes les instances d'une même taille, alors donnez la complexité de cet algorithme en fonction de  $n$ . Toutes vos réponses doivent utiliser la notation  $\Theta$ . Vous pouvez supposer que la probabilité d'avoir  $i$  bits d'allumés (pour  $0 \leq i \leq n$ ) dans la séquence  $B$  est de  $\frac{1}{n+1}$ .

---

#### **Algorithme 6 :** Mystère2( $B[0..n - 1]$ )

---

```

1 //  $B$  est un tableau de bits :  $B[i] \in \{0, 1\}, \forall i \in \{0, \dots, n - 1\}$ 
2  $s \leftarrow 0$ 
3 pour  $i$  from 0 to  $n - 1$  faire
4    $s \leftarrow s + B[i]$ 
5  $\rho \leftarrow \text{Uniforme}(1, n)$ 
6  $c \leftarrow 0$ 
7 pour  $i$  from 1 to  $s\rho$  faire
8    $c \leftarrow c + B[i \bmod n]$ 
9 retourner  $c$ 
```

---

#### **Solution :**

En meilleur cas, nous avons en entrée le vecteur  $B = [0, \dots, 0]$ . La première boucle s'exécute en temps  $\Theta(n)$  alors que la deuxième boucle prend un temps  $\Theta(1)$ . Nous avons donc  $T_{BEST}(n) \in \Theta(n)$ .

En pire cas, nous avons en entrée le vecteur  $B = [1, \dots, 1]$ . L'opération de base est  $c + B[i \bmod n]$ . Notez que puisque  $s = n$ , cette opération est exécutée aussi souvent que toutes les autres instructions. Nous avons donc la complexité suivante.

$$T_{WORST}(n) = E_\rho[n\rho] = \sum_{p=1}^n \frac{1}{n} n\rho = \sum_{p=1}^n \rho = \frac{n(n+1)}{2} \in \Theta(n^2) \quad \text{Règle du maximum}$$

En cas moyen, il faut calculer la moyenne sur toutes les instances de taille  $n$ . La première boucle de l'algorithme prend un temps  $n$  à s'exécuter alors que la deuxième boucle prend un temps  $s\rho$ . On dit dans l'énoncé que la probabilité d'avoir un nombre  $i$  de bits allumés est de  $\frac{1}{n+1}$ . La solution est donc la suivante.

$$\begin{aligned} T_{avg}(n) &= n + E_s E_\rho(s\rho) \\ &= n + E_s \left( \sum_{\rho=1}^n \frac{1}{n} s\rho \right) \\ &= n + \sum_{s=0}^n \frac{1}{n+1} \left( \sum_{\rho=1}^n \frac{s\rho}{n} \right) \end{aligned}$$

On factorise  $\frac{s}{n}$  de la deuxième sommation

$$= n + \sum_{s=0}^n \frac{s}{n(n+1)} \sum_{\rho=1}^n \rho$$

On factorise  $\frac{1}{n(n+1)}$  de la première sommation

$$\begin{aligned} &= n + \frac{1}{n(n+1)} \sum_{s=0}^n s \sum_{\rho=1}^n \rho \\ &= n + \frac{1}{n(n+1)} \sum_{s=0}^n s \frac{n(n+1)}{2} \\ &= n + \frac{1}{n(n+1)} \frac{n(n+1)}{2} \sum_{s=0}^n s \\ &= n + \frac{1}{2} \sum_{s=0}^n s \\ &= n + \frac{1}{2} \frac{n(n+1)}{2} \\ &= n + \frac{n(n+1)}{4} \\ &= n + \frac{n^2}{4} + \frac{n}{4} \in \Theta(n^2) \quad (\text{Par la règle du maximum.}) \end{aligned}$$

## Question # 7

Vous devez analyser l'algorithme Mystère3. Si la complexité de l'algorithme varie parmi les instances de même taille, identifiez le pire et le meilleur cas. Procédez ensuite à une analyse en pire cas, en meilleur cas et en cas moyen. Si la complexité de l'algorithme est la même pour toutes les instances d'une même taille, alors donnez la complexité de cet algorithme en fonction de  $n$ . Toutes vos réponses doivent utiliser la notation  $\Theta$ . Vous pouvez supposer que la probabilité d'avoir  $i$  bits d'allumés (pour  $0 \leq i \leq n$ ) dans la séquence  $B$  est de  $\frac{1}{n+1}$ .

### Algorithme 7 : Mystère3( $B[0..n - 1]$ )

```

1 //  $B$  est un tableau de bits :  $B[i] \in \{0, 1\}, \forall i \in \{0, \dots, n - 1\}$  ;
2  $s \leftarrow 0$ 
3 pour  $i$  from 0 to  $n - 1$  faire
4    $s \leftarrow s + B[i]$ 
5  $a \leftarrow \text{Uniforme}(3, n)$ 
6  $c \leftarrow 0$ 
7 si  $s = 0$  alors
8   pour  $i$  from 0 to  $n - 1$  faire
9      $c \leftarrow c + B[i \bmod n]$ 
10 sinon si  $0 < s \leq \lfloor n/2 \rfloor$  alors
11   pour  $i$  from 1 to  $s$  faire
12      $c \leftarrow c + B[i \bmod n]$ 
13 sinon
14    $c \leftarrow 4$ 
15 retourner  $c$ 
```

### Solution :

La formule générale pour le temps d'exécution d'un algorithme randomisé pour une instance  $x$  est :

$$\mathcal{C}(x) = E_a[C(x, a)] = \sum_a p(a)C(x, a).$$

Les valeurs possibles pour le paramètre aléatoire  $a$  sont les entiers  $\{3, \dots, n\}$  et toutes ces valeurs sont équiprobables. Par conséquent, nous avons  $p(a) = \frac{1}{n-2}$  pour toute valeur de  $a$ . Donc,

$$\mathcal{C}(x) = \sum_{a=3}^n \frac{1}{n-2} C(x, a) = \frac{1}{n-2} \sum_{a=3}^n C(x, a).$$

Nous devons maintenant identifier  $C(x, a)$ . Or,  $C(x, a)$  ne dépend pas uniquement de la taille de  $B$  et de la variable aléatoire  $a$ .  $C(x, a)$  dépend également des éléments du tableau  $B$ , plus particulièrement de leur somme  $s$ . Pour cette raison, nous le noterons  $C(n, s, a)$ .

Le Tableau 1 illustre les détails de complexité pour chacune des valeurs de  $a$  et pour chaque type d'instance pour les lignes 5 à 15. Les valeurs de  $a$  sont représentées dans les colonnes et les lignes représentent les différents types d'instance. Il y a trois types d'instances, un pour chaque condition si dans l'algorithme. Notez qu'il faut additionner  $n$  au temps d'exécution total de l'algorithme afin de prendre en compte les lignes 2 à 4 qui s'exécutent à chaque appel et cela, peu importe l'instance.

Faisons maintenant l'analyse complète de la complexité en prenant l'addition comme opération de base. En meilleur cas, nous avons en entrée un tableau  $B$  tel que  $s > \lfloor n/2 \rfloor$ , c'est-à-dire qu'au moins

$s$	$C(n, s, a)$					$\mathcal{C}(n, s)$
$s = 0$	$n$	$\dots$	$n$	$\dots$	$n$	$\frac{1}{n-2} \sum_{a=3}^n C(n, 0, a) = n$
$0 < s \leq \lfloor n/2 \rfloor$	$s \times 3$	$\dots$	$s \times a$	$\dots$	$s \times n$	$\frac{1}{n-2} \sum_{a=3}^n C(n, s, a) = \frac{s(n+3)}{2}$
$s > \lfloor n/2 \rfloor$	0	$\dots$	0	$\dots$	0	$\frac{1}{n-2} \sum_{a=3}^n C(n, s, a) = 0$

TABLEAU 1 – Tableau illustrant les détails de l’analyse de complexité des lignes 5 à 15 de l’algorithme *Mystère3*.

$\lfloor n/2 \rfloor + 1$  éléments de  $B$  sont égaux à 1. Prenons par exemple  $s = n$ . Alors, nous avons

$$\begin{aligned}\mathcal{C}_{BEST}(n) &= n + \frac{1}{n-2} \sum_{a=3}^n C(n, n, a) \\ &= n + \frac{1}{n-2} \sum_{a=3}^n 0 \\ &= n \in \Theta(n).\end{aligned}$$

En pire cas, nous avons en entrée un tableau  $B$  dont la somme est  $s = \lfloor n/2 \rfloor$ . Alors, nous avons donc

$$\begin{aligned}\mathcal{C}_{WORST}(n) &= n + \frac{1}{n-2} \sum_{a=3}^n C(n, \lfloor n/2 \rfloor, a) \\ &= n + \frac{1}{n-2} \sum_{a=3}^{\lfloor n/2 \rfloor} \lfloor n/2 \rfloor a = n + \frac{\lfloor n/2 \rfloor}{n-2} \sum_{a=3}^{\lfloor n/2 \rfloor} a \\ &= n + \frac{\lfloor n/2 \rfloor}{n-2} \left[ \sum_{a=1}^{\lfloor n/2 \rfloor} a - 1 - 2 \right] = n + \frac{\lfloor n/2 \rfloor}{n-2} \left[ \frac{n(n+1)}{2} - 3 \right] \\ &= n + \frac{\lfloor n/2 \rfloor}{n-2} \left[ \frac{n(n+1)-6}{2} \right] = n + \frac{\lfloor n/2 \rfloor}{n-2} \times \frac{n^2+n-6}{2} \\ &= n + \frac{\lfloor n/2 \rfloor}{n-2} \times \frac{(n-2)(n+3)}{2} \\ &= n + \frac{\lfloor n/2 \rfloor(n+3)}{2} \in \Theta(n^2) \quad (\text{Par la règle du maximum.})\end{aligned}$$

En cas moyen, il faut calculer la moyenne sur toutes les instances de taille  $n$ . Nous pouvons nous aider de la Table 1. Nous avons trois types d’instances : celles où nous avons  $s = 0$ , celles où  $0 < s \leq \lfloor n/2 \rfloor$  et finalement celles où  $s > \lfloor n/2 \rfloor$ . Par hypothèse, nous avons que chaque valeur de  $s$  est équiprobable.

Nous avons donc

$$\begin{aligned}
\mathcal{C}_{AVG}(n) &= n + \sum_{s=0}^n \frac{1}{n+1} \mathcal{C}(n, s) \\
&= n + \frac{1}{n+1} \mathcal{C}(n, 0) + \sum_{s=1}^{\lfloor n/2 \rfloor} \frac{1}{n+1} \mathcal{C}(n, s) + \sum_{s=\lfloor n/2 \rfloor + 1}^n \frac{1}{n+1} \mathcal{C}(n, s) \\
&= n + \frac{1}{n+1} n + \sum_{s=1}^{\lfloor n/2 \rfloor} \frac{1}{n+1} \frac{s(n+3)}{2} + \sum_{s=\lfloor n/2 \rfloor + 1}^n \frac{1}{n+1} 0 \\
&= n + \frac{n}{n+1} + \frac{n+3}{2(n+1)} \frac{\lfloor n/2 \rfloor (\lfloor n/2 \rfloor + 1)}{2} \\
&\geq n + \frac{n}{n+1} + \frac{n+3}{2(n+1)} \frac{(n/2 - 1)(n/2)}{2}
\end{aligned}$$

Puisque  $n \geq 0$  et  $\frac{n}{n+1} \geq 0$

$$\begin{aligned}
&\geq \frac{n+3}{2(n+1)} \frac{(n/2 - 1)(n/2)}{2} \\
&\geq \frac{n}{2(n+n)} \frac{(n/2 - 1)(n/2)}{2} \quad \text{Pour } n \geq 1 \\
&\geq \frac{1}{8} (n/2 - 1)(n/2) \\
&\geq \frac{1}{8} \left( \frac{n^2}{4} - \frac{n}{2} \right) \\
&\geq \frac{1}{8} \left( \frac{n^2}{4} - \frac{n}{2} \frac{n}{4} \right) \quad \text{Pour } n \geq 4 \\
&\geq \frac{1}{64} n^2 \quad \text{Pour } n \geq 4 \\
&\in \Omega(n^2)
\end{aligned}$$

Puisque  $\mathcal{C}_{AVG}(n) \leq \mathcal{C}_{WORST}(n) \in O(n^2)$

$$\begin{aligned}
\mathcal{C}_{AVG}(n) &\in O(n^2) \\
\mathcal{C}_{AVG}(n) &\in \Theta(n^2)
\end{aligned}$$