Séquence 9 - TD2 - Nouveaux outils théoriques pour les preuves de complexité - Corrigé

Exercice 1 (Quelle stratégie DPR ?).
Pour résoudre un problème donné, vous avez le choix entre trois algorithmes : sur un entrée de taille n :

- `algo1` divise le problème en 5 sous-problèmes de taille $\lceil \frac{n}{2} \rceil$ et combine les solutions en temps $\Theta(n)$
- `algo2` divise le problème en 2 sous-problèmes de taille $n-1$ et combine les solutions en temps $\Theta(1)$
- `algo3` divise le problème en 9 sous-problèmes de taille $\lceil \frac{n}{3} \rceil$ et combine les solutions en temps $\Theta(n^2)$ Quel algorithme choisissez-vous ? Justifier...

Corrigé de l'exercice 1. *[Retour à l'énoncé]*

	modèle	a	b	β	cas	Ordre de grandeur
`algo1`	$T_1(n) = aT(\lceil \frac{n}{b} \rceil) + n^\beta$	$a=5$	$b=2$	$\beta=1$	$a > b^\beta$	$T_1(n) \in \Theta\left(n^{\log_2(5)}\right) \approx \Theta\left(n^{2.32}\right)$
`algo2`	$T_2(n) = aT(n-1) + O(1)$	$a=2$	$-$	$-$	f polyn.	$T_2(n) \in \Theta\left(2^n\right)$
`algo3`	$T_3(n) = aT(\lceil \frac{n}{b} \rceil) + n^\beta$	$a=9$	$b=3$	$\beta=2$	$a = b^\beta$	$T_3(n) \in \Theta\left(n^2 \log_3(n)\right)$

L'algorithme 2 est disqualifié : il est exponentiel et bien moins performant que les deux autres. Il nous reste à comparer l'évolution asymptotique de la complexité pour les algorithmes 1 et 3 :

$$\frac{T_3(n)}{T_1(n)} \in \Theta\left(\frac{n^2 \log_3(n)}{n^{\log_2(5)}}\right) \in \Theta\left(\frac{\log_3(n)}{n^{\log_2(5)-2}}\right)$$

et comme $\log_2(5) - 2 \approx 0.32 > 0$, on a $\lim_{n\to\infty} \frac{T_3(n)}{T_1(n)} = 0$ (le logarithme perd face à n'importe quel n^α, $\alpha > 0$).
On en conclut donc que l'algorithme 3 est le plus performant asymptotiquement : c'est celui que je choisis.

Exercice 2 (Résolution de récurrence).
Résoudre la relation de récurrence suivante pour $n = 2^p$:

$$u(n) = 4\, u(\lfloor \frac{n}{2} \rfloor) + n^2$$

Corrigé de l'exercice 2. *[Retour à l'énoncé]*

Comme n est une puissance de 2, $\frac{n}{2^k}$ sera toujours un entier et on peut donc se débarrasser des parties entières.

On déroule la récurrence p fois :

$$\begin{aligned}
u(n) &= 4 \times \left(4 \times u\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)^2\right) + n^2 \\
&= 4 \times \left(4 \times \left(4 \times u\left(\frac{n}{8}\right) + \left(\frac{n}{4}\right)^2\right) + \left(\frac{n}{2}\right)^2\right) + n^2 \\
&= \dots \\
&= 4^p u(1) + \sum_{k=0}^{p-1} \left(4^k \frac{n}{2^k}\right)^2 \\
&= (2^p)^2 \, u(1) + n^2 \sum_{k=0}^{p-1} 4^k \left(\frac{1}{4}\right)^k \\
&= n^2 u(1) + n^2 \sum_{k=0}^{p-1} 1 \\
&= n^2 u(1) + n^2 \times p \\
&= n^2 \left(u(1) + \log_2(n)\right) \\
&= n^2 \left(1 + \log_2(n)\right) \text{ car } u(1) = 1 \text{ d'après la formule de l'énoncé}
\end{aligned}$$

On s'assure que cette formule et celle donnée par l'énoncé sont cohérente pour les valeurs de n qui sont des puissances de 2 :

- pour $n = 0$, les deux formules donnent bien 0
- pour $n = 1$, la formule de l'énoncé donne $4 \times u(0) + 1^2 = 1$, la notre donne $1^2(u(1) + \log_2(1)) = 1$, ce qui est cohérent.
- pour $n = 2$, la formule de l'énoncé donne $4 \times u(1) + 2^2 = 4 + 4 = 8$ et notre formule donne $2^2(1 + \log_2(2)) = 4 \times (1 + 1) = 8$
- pour $n = 4$, la formule de l'énoncé donne $4 \times u(2) + 4^2 = 4 \times 8 + 16 = 48$ et notre formule donne $4^2(1 + \log_2(4)) = 16 \times (1 + 2) = 48$

Cela semble cohérent.

Exercice 3 (Complexité moyenne du tri rapide : nombre d'entrées possible infini). Nous allons effectuer une analyse de complexité moyenne du tri rapide. On ne comptera que le nombre de comparaisons pour évaluer la complexité temporelle de l'algorithme.

1. Combien y a-t-il d'entrées différentes possibles pour cet algorithme ?
2. Quelle est la situation la plus défavorable pour l'algorithme de tri rapide ? Quelle est la complexité temporelle dans ce cas ?
3. Quelle semble être la situation la plus favorable ? Quelle est la complexité temporelle dans ce cas ?
4. En dehors de ce cas exceptionnel, quelle est la situation la plus favorable ? Donner un ordre de grandeur asymptotique de la complexité dans ce cas.
5. Nous allons maintenant étudier la complexité moyenne du tri rapide.
   a. En fonction de quelle quantité s'exprime la complexité temporelle de l'algorithme de tripartition ? Combien de comparaisons sont effectuées par cet algorithme ?
   b. Pour un tableau de taille n sans doublon, combien y a-t-il de tripartitions possibles ?
   c. En vous aidant de la question précédente, exprimez la complexité **moyenne** $T_{\text{moy}}(n)$ de l'algorithme de tri rapide par une formule de récurrence. On fera une moyenne des complexités sur les différentes configurations dénombrées à la question précédente.
   d. Résoudre cette récurrence pour obtenir une expression explicite de $T_{\text{moy}}(n)$ en fonction de n.
   e. *Indications :*
      — *On commencera par multiplier la relation obtenue pour se débarrasser des dénominateurs*
      — *On procédera ensuite par une technique de combinaison linéaire pour se débarrasser des complexité moyennes $T_{moy}(k)$ intermédiaires.*
      — *On utilisera ensuite une technique de télescopage*
      — *On conclura en se rappelant un équivalent d'une somme « classique »*
   f. Donner un équivalent de $T_{\text{moy}}(n)$ lorsque $n \to +\infty$.
   g. Conclure, en lien avec la question 2.

Corrigé de l'exercice 3. *[Retour à l'énoncé]*

1. Il y a une infinité d'entrées possibles
2. La situation la plus défavorable correspond au cas d'un tableau déjà trié (par ordre croissant ou décroissant). Si on choisit toujours le premier élément du tableau comme pivot, à chaque appel récursif, l'appel récursif sur le segment gauche se fait avec un tableau vide mais l'autre appel récursif sur le tableau droit se fait sur un tableau dont la taille a seulement diminué de 1 élément. On a donc un algorithme quadratique en la taille du tableau.
3. Le cas le plus favorable est celui d'un tableau ne contenant qu'une valeur, la même dans toutes les cases. Le segment central prend toute la place et, dès le premier appel

récursif, on tombe sur le cas de base car les appels récursifs se font sur des segments gauche et droit qui sont vides. La complexité globale de l'algorithme de tri rapide est alors linéaire en la taille du tableau.

4. En dehors de ce cas exceptionnel, la situation la plus favorable se produit lorsque les deux appels récursifs se font sur des segments de même taille. Si on note $T(n)$ la complexité temporelle de l'algorithme de tri rapide pour un tableau de taille n, on a alors, dans ce cas équilibré, la relation de récurrence suivante :

$$T(n) = \underbrace{n}_{\text{coup de la tripartition sur le tableau entier de taille } n} + \underbrace{2 \times T(\lfloor \frac{n}{2} \rfloor)}_{\text{2 appels récursifs équilibrés}}$$

Ainsi :

$$\begin{cases} T(0) & = 0 \\ T(1) & = 0 \\ T(n) & = n + 2 \times T(\lfloor \frac{n}{2} \rfloor) \end{cases}$$

On a affaire à un modèle DPR, avec $a = 2$, $b = 2$ et $\beta = 1$. On est dans le cas $b^\beta = a$ et on s'attend à un ordre de grandeur asymptotique en $\Theta(n\log_2(n))$.

Comme nous n'avons pas le droit d'invoquer le théorème maître, on doit quand même faire un calcul pour justifier :

Limitons nous au cas où n est une puissance de 2 : $n = 2^p \Leftrightarrow p = \log_2(n)$.

$$\begin{aligned} T(2^p) &= 2^p + 2 \times T(2^{p-1}) = 2^p + 2 \times (2^{p-1} + 2 \times T(2^{p-2})) \\ &= 2^p + 2^p + 4 \times (2^{p-2} + 2 \times T(2^{p-3})) \\ &= 3 \times 2^p + 2^3 \times T(2^{p-3}) = \dots \\ &= p \times 2^p + 2^p T(1) = \Theta(p2^p) = \Theta(\log_2(n)n) \end{aligned}$$

On peut ensuite généraliser ce résultat pour n'importe quelle valeur de $n \in \mathbb{N}$ et montrer que, dans le cas favorable équilibré, la complexité temporelle est toujours en $\Theta(n\log_2(n))$, quelque soit la taille n du tableau initial.

5. a. La complexité de l'algorithme de tripartition s'exprime en fonction de la taille $n = r - \ell$ de la fenêtre de travail. Cet algorithme effectue $n - 1$ comparaisons, il est linéaire.

b. Pour un tableau de taille n sans doublon, le segment central est de taille 1 (il ne peut pas y avoir plusieurs valeurs égales au pivot). Il y a n tripartitions possibles selon la position de ce singleton central.

c. Soit $n \in \mathbb{N}$. La probabilité d'obtenir un doublon dans un tableau de taille n est négligeable si on a le choix entre une infinité de valeurs pour chaque case.

On ne considère donc que des tableaux sans doublon : le segment central est donc forcément de taille 1 (une seule valeur égale au pivot dans le tableau)

On note T_n l'ensemble des tableaux de taille n sans valeurs doublons.

Pour $k \in [\![0; n-1]\!]$, on note $T_{n,k}$ la classe d'équivalence contenant tous les tableaux de taille n sur lesquels l'algorithme de tripartition décrit ci-dessus aboutit à un segment gauche de taille k et un segment droit de taille $n - 1 - k$. Ces classes d'équivalences forment une partition de T_n :

$$\bigcup_{k=0}^{n-1} T_{n,k} = T_n$$

On fait l'**hypothèse de modélisation probabiliste** suivante : un tableau de taille k sans doublon, formé de valeurs indépendantes piochées au hasard a autant de chance d'appartenir à chacune des n classes d'équivalences $(T_{n,k})_{k\in[\![0;n-1]\!]}$. Autrement toutes les classes d'équivalences sont équiprobables :

$$\mathbb{P}(T_{n,k}) = \frac{1}{n}$$

On considère maintenant la variable aléatoire X_n qui associe, à chaque tableau t de taille n, la complexité temporelle évaluée en nombre de comparaisons résultant de l'algorithme de tri rapide.
Soit $t = [t_0; t_1; \ldots; t_{n-1}] \in T_{n,k}$. On note $X_{n,k}$ la restriction de X_n à la classe d'équivalence $T_{n,k}$. On a alors, d'après l'algorithme :

$$X_{n,k}(t) = \underbrace{n-1}_{\text{nombre de tests dans la tripartition}} + \underbrace{X_k([t_0; t_1; \ldots; t_{k-1}])}_{\text{coût appel récursif segment gauche}} + \underbrace{X_{n-1-k}([t_{k+1}; t_{k+2}; \ldots; t_{n-1}])}_{\text{coût appel récursif segment droit}}$$

La complexité moyenne correspond à l'espérance de la variable aléatoire X_n. Comme le nombre de tableaux de taille n que l'on peut donner en entrée de l'algorithme est infini, on calcule cette espérance en réalisant une somme sur les classes d'équivalences, qui sont, elles, en nombre fini (il y en a n) :

$$T_{\text{moy}}(n) = E[X_n] = \sum_{k=0}^{n-1} \mathbb{P}(T_{n,k}) E[X_{n,k}]$$

Or, par linéarité de l'espérance :

$$E[X_{n,k}] = \underbrace{E[n-1]}_{=n-1 \text{ car la moyenne d'une constante est... cette constante}} + E[X_k] + E[X_{n-1-k}]$$

$$= n - 1 + T_{\text{moy}}(k) + T_{\text{moy}}(n-1-k)$$

On a donc :

$$T_{\text{moy}}(n) = E[X_n] = n - 1 + \sum_{k=0}^{n-1} \frac{1}{n} \left(T_{\text{moy}}(k) + T_{\text{moy}}(n-1-k)\right)$$

$$T_{\text{moy}}(n) = n - 1 + \frac{1}{n} \sum_{k=0}^{n-1} \left(T_{\text{moy}}(k) + T_{\text{moy}}(n-1-k)\right)$$

d. Il s'agit maintenant d'utiliser des techniques sur la manipulation de sommes pour aboutir à une expression explicite de $T_{\text{moy}}(n)$.
Un manipulation très simple d'indice sur le deuxième terme de la somme à droite permet d'écrire (symétrie) :

$$T_{\text{moy}}(n) = n - 1 + \frac{2}{n} \sum_{k=0}^{n-1} T_{\text{moy}}(k)$$

On utilise ensuite une technique d'élimination par combinaisons linéaires pour essayer de faire disparaître tous le termes $T_{\text{moy}}(k)$ intermédiaires.

On multiplie par n l'égalité ci-dessus :

$$n \times T_{\text{moy}}(n) = n(n-1) + 2\sum_{k=0}^{n-1} T_{\text{moy}}(k)$$

On la réécrit au rang $n-1$ (en substituant $n \leftarrow n-1$) :

$$(n-1) \times T_{\text{moy}}(n-1) = (n-1)(n-2) + 2\sum_{k=0}^{n-2} T_{\text{moy}}(k)$$

Puis on soustrait les deux expressions obtenues, ce qui permet de se débarrasser de tous les termes intermédiaires $T_{\text{moy}}(k)$ pour $0 \leq k \leq n-2$:

$$nT_{\text{moy}}(n) - (n-1)T_{\text{moy}}(n-1) = n(n-1) - (n-1)(n-2) + 2T(n-1)$$

$$\Leftrightarrow nT_{\text{moy}}(n) - (n+1)T_{\text{moy}}(n-1) = 2(n-1)$$

Divisons cette égalité par $n(n+1)$:

$$\Leftrightarrow \frac{T_{\text{moy}}(n)}{n+1} - \frac{T_{\text{moy}}(n-1)}{n} = \frac{2(n-1)}{n(n+1)}$$

On commence à apercevoir une possibilité de somme télescopique, écrivant la relation pour $n \to k$:

$$\frac{T_{\text{moy}}(k)}{k+1} - \frac{T_{\text{moy}}(k-1)}{k} = \frac{2(k-1)}{k(k+1)}$$

$$\sum_{k=3}^{n} \left(\frac{T_{\text{moy}}(k)}{k+1} - \frac{T_{\text{moy}}(k-1)}{k} \right) = \sum_{k=3}^{n} \frac{2(k-1)}{k(k+1)}$$

$$\Leftrightarrow \frac{T_{\text{moy}}(n)}{n+1} - \frac{T_{\text{moy}}(2)}{3} = \sum_{k=3}^{n} \frac{2(k-1)}{k(k+1)}$$

Nous allons maintenant décomposer la fraction rationnelle $F(X) = \dfrac{2(X-1)}{X(X+1)}$ en éléments simples :

$$\frac{2(X-1)}{X(X+1)} = \frac{4}{X+1} - \frac{2}{X}$$

et on a évalué XF en 0 pour trouver le coefficient associé à la valeur interdite 0 et $(X+1)F$ en (-1) pour trouver le coefficient associé à la valeur interdite -1.
Nous avons donc montré que :

$$\frac{T_{\text{moy}}(n)}{n+1} - \frac{T_{\text{moy}}(2)}{3} = \sum_{k=3}^{n} \left(\frac{4}{k+1} - \frac{2}{k} \right)$$

et on a $T_{\text{moy}}(2) = 1$ car un tableau à deux éléments nécessite une seule comparaison pour effectuer la tripartition.

$$\frac{T_{\text{moy}}(n)}{n+1} = \frac{1}{3} + 4\sum_{k=3}^{n} \frac{1}{k+1} - 2\sum_{k=3}^{n} \frac{1}{k} \underset{n\to+\infty}{\sim} 2\ln(n)$$

et on a reconnu deux sommes harmoniques à droite.
Ainsi, on a :

$$T_{\text{moy}}(n) \underset{n\to+\infty}{\sim} 2n\ln(n) \approx 1.39\, n \log_2(n)$$

e. On observe que la complexité moyenne se comporte asymptotiquement de la même manière que le cas équilibré favorable, en $\Theta(n\log_2(n))$. C'est une très bonne nouvelle !

Exercice 4 (Études asymptotiques).
Donner un ordre de grandeur asymptotique en Θ si pour, ou en O sinon, pour les complexités suivantes :

1. $T(n) = 5\,T(n-1) + 3n + 2$ et $T(0) = 7$

2. $T(n) = 7\,T(\lfloor\frac{n}{2}\rfloor) + O(n^2)$. (le $\mathrm{O}(\cdot)$ est supposé être une fonction croissante).

3. $T(n) = 2\,T(\lceil\frac{n}{2}\rceil) + n\log_2(n)$

Corrigé de l'exercice 4. *[Retour à l'énoncé]*

1. On est sur un modèle $T(n) = aT(n-1) + f(n)$ avec $a = 5$, où f est linéaire (polynomiale) donc $T(n) \in \Theta(5^n)$.

2. On est sur un modèle diviser pour régner (DPR) $T(n) = aT(\lfloor\frac{n}{b}\rfloor) + f(n)$ avec $a = 7$, $b = 2$. On ne pourra donner au mieux qu'un grand O car on ne possède qu'une information en O sur f. Le grand O est quadratique donc $\beta = 2$. $b^\beta = 2^2 = 4 < 7 = a$ donc on a une complexité asymptotique en $O(n^{\log_b(a)})$ soit $O(n^{\log_2(7)}) \approx O(n^{2.8})$

3. On est sur un modèle diviser pour régner (DPR) $T(n) = aT(\lfloor\frac{n}{b}\rfloor) + f(n)$, avec $a = b = 2$. En utilisant la domination grossière $n\log_2(n) \in O(n^2)$ on peut déjà donner un O pour la complexité asymptotique : on a alors $\beta = 2$ et $b^\beta = 2^2 = 4 > 2$. On conclut donc $T(n) \in O(n^\beta)$ soit $T(n) \in O(n^2)$: l'algorithme est au pire quadratique.
On peut cependant affiner cette étude en reprenant la preuve du théorème maître (ce que l'on vous demandera au concours de toute façon !). On se limite au cas où n est

une puissance de 2, $n = 2^p$:

$$
\begin{aligned}
T(n) &= 2T(\left\lceil \frac{n}{2} \right\rceil) + n\log_2(n) \\
&= 2^p T(\left\lceil \frac{n}{2^p} \right\rceil) + \sum_{k=0}^{p-1} 2^k \left\lceil \frac{n}{2^k} \right\rceil \log_2(\left\lceil \frac{n}{2^k} \right\rceil) \\
&= 2^p T(1) + \sum_{k=0}^{p-1} 2^k 2^{p-k} \log_2(\lceil 2^{p-k} \rceil) \\
&= 2^p T(1) + 2^p \sum_{k=0}^{p-1} (p-k) \\
&= 2^p T(1) + 2^p \sum_{k=1}^{p} k && \text{(ré-indiçage } k \leftarrow p-k\text{)} \\
&= 2^p T(1) + 2^p \frac{p(p+1)}{2} && \text{(somme classique)} \\
&= nT(1) + n\frac{\log_2(n)(\log_2(n)+1)}{2} && \text{on ré-exprime en fonction de } n \\
&\in \Theta\left(n\left(\log_2(n)\right)^2\right) && \text{car le second terme est dominant quand } n \to +\infty
\end{aligned}
$$

On a donc $T(n) \in \Theta\left(n\left(\log_2(n)\right)^2\right)$. On remarque que $\Theta\left(n\left(\log_2(n)\right)^2\right) \in O(n^2)$, ce qui est cohérent avec notre première étude plus grossière.