Correction du TD1

Exercice 1. Montrons que $\Theta(g(n)) = \Omega(g(n)) \cap O(g(n))$.
Une technique classique pour montrer une égalité d'ensembles est de montrer la double inclusion. Par exemple, ici il s'agit de montrer que :

$$\Theta(g(n)) \subseteq \Omega(g(n)) \cap O(g(n)) \quad (1)$$
$$\Omega(g(n)) \cap O(g(n)) \subseteq \Theta(g(n)) \quad (2)$$

1. Soit $f(n) \in \Theta(g(n))$.
   Montrons $f(n) \in \Omega(g(n))$ et $f(n) \in O(g(n))$:

   Puisque $f(n) \in \Theta(g(n))$, par definition, il existe une constante $c = c_1 > 0$ et un rang n_0 à partir duquel $\forall n \geq n_0, 0 \leq c_1 g(n) \leq f(n)$ donc $f(n)$ satisfait la definition de $\Omega(n)$.

   Puisque $f(n) \in \Theta(g(n))$, par definition, il existe une constante $c = c_2 > 0$ et un rang n_0 à partir duquel $\forall n \geq n_0, 0 \leq f(n) \leq c_2 g(n)$ donc $f(n)$ satisfait la definition de $O(n)$.

2. Soit $f(n)$ tel que $f(n) \in \Omega(g(n))$ et $f(n) \in O(g(n))$.
   Montrons que $f(n) \in \Theta(g(n))$:

   La definition de Ω nous donne une constante $c^{(\Omega)}$ et un rang $n_0^{(\Omega)}$. La definition de O nous donne une constante $c^{(O)}$ et un rang $n_0^{(O)}$. En prenant $c_1 = c^{(\Omega)}$, $c_2 = c^{(O)}$, et $n_0 = \max\{n_0^{(\Omega)}, n_0^{(O)}\}$, la fonction $f(n)$ satisfait la definition de $\Theta(g(n))$, car a fortiori, pour tout n plus grand que n_0, on a n plus grand que $n_0^{(\Omega)}$ et $n_0^{(O)}$, donc:

$$c^{(\Omega)} g(n) = c_1 g(n) \leq f(n) \leq c_2 g(n) = c^{(O)} g(n)$$

Donnons des exemples de fonctions f et g telles que $f(n) = O(g(n))$: $10n^2 = O(n^2), n^3 + n^2 + n + 1 = O(n^3), n^4 - 10n^3 = O(n^4)$.

La fonction $f(n) = n^4$ est bornée inferieurement par $g(n) = n^3$, donc $f(n) = \Omega(n^3)$, mais pas superieurement, donc $f(n) \notin O(n^3)$ et donc $f(n) \notin \Theta(n^3)$.

Exercice 2. Les notations asymptotiques s'étendent aux fonctions de plusieurs variables: $\Theta(g(n,m)) = \{f(n,m) : \exists c_1, c_2 > 0, \exists N_0, \forall n, m \geq N_0, c_1 g(n,m) \leq f(n,m) \leq c_2 g(n,m)\}$

A $\Theta(\min\{n,m\})$ car l'algorithme s'arrete dès que une des deux conditions n'est plus satisfaite.

B $\Theta(\max\{n,m\})$ car l'algorithme s'arrete lorsque les deux conditions ne sont plus satisfaites.

C $\Theta(n+m)$ car l'algorithme procède en deux phases: une première où il incrémente `i` jusqu'à `m`, et une deuxième phase où il incrémente `j` jusqu'à `n`.

D $\Theta(nm)$, car les indices `i` et `j` parcourent l'équivalent d'une matrice `n x m`.

Exercice 3. Concernant la premiere fonction, chaque iteration i effectue de l'ordre de m operation pour le bloc d'instruction `for j in range(0,m)`. Il y a n iterations sur la variable i. Donc la fonction effectue de l'odre de $\Theta(nm)$ operations.

Si $n \leq m$, la seconde fonction, apres 2 affectations, effectue $f(n,m) = \sum_{i=0}^{n-1}(m-i)$ additions et affectations. Notons que:

$$\begin{aligned} f(n,m) &= nm - \sum_{i=1}^{n-1} i \\ &= nm - n(n-1)/2 \\ &= n(m-(n-1)/2) \end{aligned}$$

Comme $n \leq m$ implique $m/2 \leq (m-(n-1)/2) \leq m$, il s'en suit que $f(n,m) \in \Theta(nm)$.

Si $n > m$, le cout $f_i(n,m)$ de chaque iteration `for i` differe suivant que $i < m$ ou $i \geq m$. Si $i < m$ alors $f_i(n,m) = m - i$ additions et affectations. Si $i \geq m$ alors une operation en temps constant permet de determiner que `range(`i,m`)` est vide. Par consequent, nous avons donc:

$$\begin{aligned} f(n,m) &= \sum_{i=0}^{m-1} f_i(n,m) + \sum_{i=m}^{n-1} f_i(n,m) \\ &= \sum_{i=0}^{m-1}(m-i) + \sum_{i=m}^{n-1} 1 \\ &= \sum_{i=1}^{m} i + (n-m-1) \\ &= m(m+1)/2 + (n-m-1) \\ &= m(m-1)/2 + (n-1) \\ &\in \Theta(m^2+n) \end{aligned}$$

Exercice 4.

1. $n^2 = O(10^{-5}n^3)$. Prenons $c = 10^5$ et $n_0 = 1$. Alors pour tout $n \geq n_0$ nous avons $n^2 \leq 10^5 10^{-5} n^3$.
   $25n^4 - 19n^3 + 13n^2 = O(n^4)$. Prenons $c = 25$ et $n_0 = 1$. Comme $-19n^3 + 13n^2 = n^2(13 - 19n) \leq 0$ pour $n \geq 1$, on a bien $25n^4 - 19n^3 + 13n^2 \leq 25n^4$.
   $2^{n+100} = O(2^n)$. Prenons $c = 2^{100}$. Alors $2^{n+100} = 2^{100}2^n$. Discuter de la pertinence de cette assertion.
2. $O(1) \subseteq O(\log_2(n)) \subseteq O(n) \subseteq O(n\log_2(n)) \subseteq O(n^2) \subseteq O(n^3) \subseteq O(2^n)$
3. Si $f(n) = O(g(n))$ alors il existe une constante c telle que a partir d'un certain rang, on a: $f(n) \leq cg(n)$. Et donc on a $(1/c)f(n) \leq g(n)$ a partir du meme rang. Donc $g(n) = \Omega(f(n))$ avec $c' = 1/c$.
4. Si $f(n) = O(g(n))$, alors il existe une constante $c > 0$ telle que $f(n) \leq cg(n)$ a partir d'un certain rang. Donc $f(n) + g(n) \leq (c+1)g(n)$ et donc $f(n) + g(n) \in O(g(n))$.
5. $f(n) + g(n) = \Theta(\max\{f(n), g(n)\})$ decoule du fait que:
   $$\max\{f(n), g(n)\} \leq f(n) + g(n) \leq 2\max\{f(n), g(n)\}$$
6. $O(f(n) + g(n)) = O(\max\{f(n), g(n)\})$. D'une part,
   $$f(n) + g(n) \leq 2\max\{f(n), g(n)\}$$
   donc $O(f(n) + g(n)) \subseteq O(\max\{f(n), g(n)\})$. D'une autre part
   $$\max\{f(n), g(n)\} \leq f(n) + g(n)$$
   donc $O(\max\{f(n), g(n)\}) \subseteq O(f(n) + g(n))$.
7. Si $S(n) = O(f(n))$ et $T(n) = O(g(n))$ et $f(n) = O(g(n))$ alors:
   il existe des constantes $c_1, c_2, c_3 > 0$ telles que a partir de rangs $n_1, n_2, n_3 \in \mathbb{N}$, on ait pour tout $n \geq \max\{n_1, n_2, n_3\}$:
   $$S(n) \leq c_1 f(n), \quad T(n) \leq c_2 g(n), \quad f(n) \leq c_3 g(n)$$
   Ainsi, a partir du meme rang, on a:
   $$S(n) + T(n) \leq c_1 f(n) + c_2 g(n) \leq (c_3 + c_2) g(n)$$
   Et donc $S(n) + T(n) \in O(g(n))$.

8. En outre:
   $$S(n)T(n) \leq c_1 f(n) T(n) \leq c_1 c_2 f(n) g(n)$$
   Et donc $S(n)T(n) \in O(f(n)g(n))$.

Exercice 5. Soit n la longueur du tableau A. La complexite dans le pire des cas corresponds à la situation où le tableau donné en entrée est trié dans l'ordre inverse. Dans quel cas, à l'iteration `j`, l'insertion de `A[j]` coute de l'ordre de $O(j)$ operations. Au cours de l'algorithme, `j` croit de 2 à n, donc il y a de l'ordre de $O(n^2)$ operations.

1. Enoncé: $P(j)$: À la fin de l'iteration `j`, le tableau `A[1,..,j]` est trié.
2. Initialisation: À la fin de l'iteration `j=2`, le tableau `A[1,2]` est trié.
3. Récurrence: On suppose $P(j-1)$ pour $j > 2$ et on montre $P(j)$.
   `A[1,..,j-1]` est trié. La boucle interne insère l'élément `A[j]` au bon endroit. Elle trouve le plus grand indice `i` tel que pour tout `i' > i` on ait `A[i'] > A[i]` décale les éléments concernés, et insère `A[j]` en place. Ainsi, le nouveau tableau des j premiers elements `A[1,..,i],A[j],A[i+1,..,j-1]` est trié.
4. Conclusion: A la fin de l'algorithme, le tableau est `A[1,..,n]` est trié.

Exercice 6. Soit $p, q \geq 2$ deux entiers. Soit $f(n) \in O(\log_p(n))$, donc à partir d'un certain rang:

$$f(n) \leq C\log_p(n) = C\frac{\ln(n)}{\ln(p)} = C\frac{\ln(q)}{\ln(p)}\frac{\ln(n)}{\ln(q)} = C'\log_q(n)$$

où $C' = C\frac{\ln(q)}{\ln(p)}$. Donc $f(n) \in O(\log_q(n))$. Ainsi $O(\log_p(n)) = O(\log_q(n))$.

Exercice 7. Traditionnellement, un algorithme est dit "efficace" lorsqu'il prend un temps au pire polynomial en la longueur de l'entrée, et est dit "insurmontable" lorsqu'il prend un temps exponentiel en la longueur de l'entrée. Les algorithmes de l'exercice 2 sont-ils efficaces ou insurmontables ? Au premier abord, des temps en $\Theta(\min\{n, m\})$, $\Theta(\max\{n, m\})$, $\Theta(n+m)$ et $\Theta(nm)$ sont au pire polynomiaux. Toutefois la longueur des entrées est logarithmique en n et m. Donc en fait les temps sont respectivement:

$$\Theta(\min\{2^a, 2^b\}), \quad \Theta(\max\{2^a, 2^b\}), \quad \Theta(2^a + 2^b), \quad et \quad \Theta(2^{a+b})$$