CORRIGÉ DU TD2 , NOTIONS DE COMPLEXITÉ

Lien web :
`http://www-sop.inria.fr/members/Stephane.Perennes/TD2/corrige.pdf`

1 Notations asymptotiques

Les réponses à cet exercice seront données dans le fichier `exercice_1.txt`. Pour les symboles, il suffit d'écrire `Omega(n)` pour $\Omega(n)$, `Theta(n)` pour $\Theta(n)$, ...

1. Application des définitions du cours *Notions de complexité* (diapositives 8 à 13)

   (a) En utilisant la définition de O montrer que $3n^2 + 4n + 6 = O(n^2)$.

   Réponse :
   On peut écrire que si $n \geq 4$ alors $4n \leq n^2, 6 \leq n^2$ et donc $3n^2 + 4n + 6 \leq 5n^2$. Donc on peut utiliser la définition, pour $n \geq n_0 = 4$ et $C = 5$ car alors $\forall n \geq n_0, 3n^2 + 4n + 6 \leq Cn^2$.

   La constante C ici n est pas bonne, en effet $\frac{3n^2+4n+6}{n^2} \leq 3 + \frac{4n+6}{n^2}$. La fonction $\frac{4n+6}{n^2}$ tend vers zero quand $n \to +\infty$. Donc pour toute constante $\varepsilon_0 > 0$ (pex 10^{-10}) il existe un nombre $n_0(\varepsilon)$ tel que pour $n \geq n_0(\varepsilon)$ on a $\frac{4n+6}{n^2} \leq \varepsilon$. Ceci signifie que pour $n \geq n(\varepsilon)$ on a $3n^2 + 4n + 6 \leq (3 + \varepsilon)n^2$. On peut donc rendre C aussi proche de 3 que voulut.

   On peut aussi tout simplement prendre $n_0 = 1$ et dire que pour $n \geq n_0$ on à $4n \leq 4n^2$ et $6 \leq 6n^2$ et donc $3n^2 + 4n + 6 \leq 13n^2$. La constante C est ici 13. Pour $n_0 = \frac{1}{100}$ on aurait une constante C_0 mais encore plus mauvaise.

   Nous avons en fait $3n^2 + 4n + 6 \sim 3n^2$, la notation $\sim$ signifie que le rapport des deux fonctions tends vers 1, c est une notion bien plus précise que Θ, O, ω mais elle les implique toute les 3.

   (b) En utilisant la définition de Ω montrer que $3n^2 + 4n + 6 = \Omega(n^2)$

   Réponse possible
   On peut simplement dire que $3n^2 + 4n + 6 \geq 3n^2$ si $n \geq 0$, ceci est la définition de $3n^2 + 4n = 6 = \Omega(n^2)$ avec $C = 3, n_0 = 0$.

   (c) En utilisant la définition de Θ montrer que $3n^2 + 4n + 6 = \Theta(n^2)$

   Réponse
   On a montré en (1) que $3n^2 + 4n + 6 = O(n^2)$ ($3n^2 + 4n + 6$ est au plus de l'ordre de n^2) et en (2) que $3n^2 + 4n + 6 = \Omega(n^2)$ ($3n^2 + 4n + 6$ est au moins de l'ordre de n^2) c est la définition de $3n^2 + 4n + 6 = \Theta(n^2)$ (être du même ordre).

2. Application des propriétés du cours *Notions de complexité* (diapositive 14)
   Caractériser les fonctions suivantes en termes de notation Θ :

(a) $2^{1024} + 5n^8 + 2087n^7$
Réponse : n^8 car la constante 2^{1024} est un $\Theta(1)$ et $2087n^7 = \Theta(n^7)$. Quand les expressions sont des polynomes c est celui de plus grand degré qui compte pour la classe Θ. .

(b) $1024n + 2048\sqrt{5n}$
Réponse $\Theta(n)$, même raisons $\sqrt{5n} = \sqrt{5}n^{1/2}, 2048 = 2048n^0$. C'est automatique. N'oubliez pas que les constantes multiplicatives *ne comptent pas*.

(c) $2n^8 + 6789^8 + \frac{3}{2}n^9$ **Réponse $\Theta(n^9)$ à nouveau tout les termes sont des polynomes, les degrés respectifs étant $8, 0, 9$**

(d) $\frac{n^2}{3} + 10^{100}n$
Réponse $\Theta(n^2)$ à nouveau tout les termes sont des polynômes, les degrés respectifs étant $2, 1$. l'énorme constante 10^{100} ne doit pas vous leurrer on regarde le comportement pour n très grand. Notez la différence entre théorie et pratique, pour des n raisonnables l expression vaut en gros $10^{100}n$. Mais quand $n >> 10^{100}$ c'est le terme $n^2/3$ qui l'emporte.

(e) $4\,log(5n) + \sqrt{3n}$ **Réponse : $\Theta(\sqrt{n})$. Le logarithme croit très lentement et est dominé par toute fonction de la forme $n^a, a > 0$, donc ici le terme en $\sqrt{n} = n^{0.5}$ domine et détermine l'ordre de la fonction.**

(f) $2n\,logn + 10\sqrt{n}$
Réponse : $\Theta(n \log n)$, n domine $\sqrt{n}$ (1 >0.5) et $\log n$ tend vers l'infini. Donc c'est bel est bien le terme en $n \log n$ qui l'emporte.

(g) $2n \sum_{i=1}^{2n}(3i + 1) + \sum_{p=1}^{n} 2^p$.
Réponse : On sait que : $\sum_{j=0}^{j=n} j = \frac{n(n+1)}{2}$ et donc $\sum_{j=0}^{j=n} j = \Theta(n^2)$. De même $\sum_{j=0}^{j=2n} j = \Theta((2n)^2) = \Theta(4n^2) = \Theta(n^2)$ (par définition quand on considère l'ordre on peut supprimer les constantes). On peut alors conclure que que $\sum_{i=0}^{i=2n} 3i + 1 = \Theta(\sum_{i=0}^{i=2n} i = \Theta(n^2)$. On peut aussi develloper la somme $\sum i = 0^{i=2n} 3i+1 = 3 \sum i = 0^{i=2n} i + \sum i = 0^{i=2n} 1 = 3\Theta((2n)^2) + (2n + 1)$, ce qui donne à nouveau $\Theta(n^2)$.
Le second terme vaut $2^{n+1} - 1$. Ce dernier domine donc très largment le premier qui est polynomial en $\Theta(n^2)$.

Remarque 1 *Notons que pour un réel $\alpha > 0$, on a bel et bien $\sum_{i=0}^{i=n} i^\alpha = \Theta(i^{\alpha+1}$, ceci est très lié au fait que la* primitive [1]*. de i^α soit $\frac{i^{\alpha+1}}{\alpha+1}$.*
En effet la somme discrète $sum_{i=0}^{i=n} i^\alpha$ est voisine de la la somme continue $I = int_{i=0}^{i=n} i^\alpha$ or l'intégrale I vaut $F(n) - F(0)$ ou F est une primitive i^α. Or ici nous connaissons une telle primitive, en l'occurence $\frac{i^{\alpha+1}}{\alpha+1}$. Ce qui donne le résultat.
La technique est assez génerale et sous certaines conditions on a :

$$\sum_{i=0}^{i=n} f(i) = F(n)$$

Où F est une primitve de f.
On appelle cette méthod formule d'Euler-Mac-laurin*, voir* `https://en.wikipedia.org/wiki/Euler%E2%80%93Maclaurin_formula`.

1. Autrement dit la dérivée de $i^{\alpha+1}$ est $(\alpha + 1)i^\alpha$

3. Comparer les fonctions suivantes (qui est en $O()$ de qui ?) :
   (a) 2^n
   (b) n^n
   (c) $n!$

   Réponse : (a)= $2 \times 2...2$ **(**n **fois),** $(b) = n \times n \dots n$ **(**n **fois) et** $c = n \times n - 1 \dots 1$**. Donc** $b \geq c \geq a$**. On a pour tout** $n \geq n_0$ **:**$b \geq n_0 c$ **(effet pur** b **le dernier terme est** n **tandis qu'il est** 1 **pour** a**) et donc** $c \leq \frac{1}{n_0} b$ **donc** $c = O(b)$**.**
   De même pour $n \geq n_0$ **nous avons** $a \leq c/n_0$ **et donc** $a = O(c)$**. Et comme** O **est une relation d'ordre** $a = O(O(b)) = O(b)$**.**
   Notons que que nous avons le résultat plus fort qui est $c = o(b)$ **le** o **signifie négligeable, autrement dit le rapport** c/b **doit tendre vers zéro. Comme** $c \leq \frac{1}{n_0} b$ **(pour** $n \geq n_0$**) c est bien le cas, plus** n **est grand plus** c **est petit devant** b**.**

4. En utilisant la définition de Θ montrer que $f(n) + g(n) = \Theta(max(f(n), g(n)))$.
   Réponse : Ici implicitement les fonctions sont supposées positives. Notons que alors :

$$max(f(n), g(n)) \leq f(n) + g(n) \leq 2max(f(n) + g(n)$$

   Autrement dit à un facteur 2 **près le max et la somme sont égaux, et donc si on charche juste des ordre de grandeur on peut remplacer l'un par l'autre. Remplacer les max par des sommes est souvent utile car cela simplifie les choses puisque la somme est associative et commutative. Et on ne perd qu'un facteur** 2**.**
   La propriété est fausse si on ne suppose pas les fonctions positives, par exemple si on choisit $g(n) = -f(n)$ $f(n) + g(n) = 0$**. Donc pour** $f(n) = n$ **ou toute fonction** f **tendant vers l'infini on a un contre exemple.**

2 Complexité des algorithmes du TD1

1. Exercice 1 du TD1. Exprimer en notations Ω, Θ et O le nombre de fois où s'affiche « Bonjour 1 » et « Bonjour 2 ». **Réponse : nous avons déjà donné des estimation plus précises à savoir** n^2 **et** $n^2/2$**. Donc on trouve** $O(n^2), \Omega(n^2)$ **et** $\Theta(n^2)$ **pour bonjour 1 et bonjour 2. Notez qu'ecrire** $O(n^{17})$ **n est pas faux vu que** n^2 **est dominé par** n^{17}**. De même on pourrait écrire** $\Omega(n)$ **car** n^2 **domine** n**, Mais pour** Θ **nous n avons pas ce choix, l'ordre est** n^2**.**
2. Exercice 2 du TD1. Exprimer en notations Ω, Θ et O la complexité du tri par sélection.
   Réponse :
   La selection dans une liste de taille i prends $\Theta(i)$, On le fait pour des listes de taille $1, 2, \dots n$ ce qui donne $\Theta(\sum_{i \in [0,n]} i) = \Theta(\frac{n^2}{2})$. Donc la réponse est $\Theta(n^2)$ et donc par exemple $O(n^2), \Omega(n^2)$.
3. Exercice 3 du TD1.
   Réponse

   Le temps de la recherche obéit à l'équation $T(n) = T(n/2) + \Theta(1), T(1) = \Theta(1)$ ce qui en devellopant donne $T(n) = T(\frac{n}{2^k}) + k\Theta(1), n \leq 2^k$. On trouve donc $T(n) = \Theta(\log n)$ et donc aussi $\Omega(\log n)$, $O(\log n)$.

3 Exercice 4 du TD1

Terminer l'exercice 4 du TD1 et écrire la complexité de chacune des trois méthodes dans les commentaires, juste à la suite des Pre et Post conditions.

Pour le tri par selection on écrira juste :

— avant la boucle interne ligne 9 : (prec-ondition) que le tableau est trié pour indices de l'intervale $[0.i]$

— à la find la boucle interne ligne 17 ou 18 : (post-condition) que le tableau est trié pour indices de l'intervale $[0.i+1]$.

— Notez bien que la post condition est en fait la pre-condition pour l'itération suivante)i.e. pour $i+1$). C est donc le couple pre/post-condition qui valide le schéma algoritmique.

— Entre les lignes 8-9 on ajoutera que la boucle selectionne le minimium du troncon final du tableau, et que donc elle est de complexité $\Theta(n-i)$ ou n est la taille du tableau.

— au debut de la fonction on ajouter que la complexité est $somme_{j=0,n}j = \Theta(n^2)$. Si l'on souhaite être plus rigoureux on peut détailler cela en $\sum_{i=0,n-1} B(i)$ où $B(i)$ le temps la ième boule. Ce temps est dominé par le temps de la boucle interne. Donc on a $B(i) = \Theta(n-i)$ de part le commentaire sur la boucle interne. Au final on trouve $\sum_{j=n,n-1,...1} \Theta(j) = \Theta(n^2)$.

— Notons aussi que sachant que nous effectuons des estimations la différence entre $n, n-1$ ou $1, 0$ est sans impact on peut donc être un peu approximatif. Cependant dans certain cas (rares) on a pas $\Theta(f(n)) = \Theta(f(n-1))$ par exemple pour la fonction factorielle. Pour le polynomes on a toujours $\Theta(P(an+b)) = \Theta(P(n))$, je vous incite d'ailleurs à le prouver.

4 Au voleur !

Un voleur cherche à dévaliser un magasin. Comme il a mal au dos, il ne peut pas porter plus qu'un poids maximum `poidsMax` mais il cherche cependant à emporter le butin de plus grande valeur possible. Le butin est choisi parmi les `Articles` du `Magasin`, pris en 0 ou 1 exemplaire.

On considère la classe `Magasin` donnée sur Moodle : `m415-skel-td2.zip`

— Pour l'instance de problème du magasin bio (constructeur par défaut de la classe `Magasin`), quelle est la valeur optimale du butin qu'il emportera pour des valeurs de poids maximum de 30, 311 et 1000 ?

Réponse :

— Je trouve 52 pour le volume de 30, le volume effectif est de 11, saffran et purée.

— Je trouve 5065 pour le volume de 311, le volume effectif est de 301 avec : collier, amandes, saffran, faux filet.

— Je trouve 5077 pour le volume de 1000, le volume effectif est de 611 avec : collier, amandes, saffran, faux filet, purée, merguez.

4.1 Questions de Code

— Voici un code python possible ou tous les sacs sont générés en binaire. J'ai séparé la recherche du meilleur sac afin que ce soit plus lisible, bien entendu on peut cherche les max à la volée.

— on peut garder la liste L des meilleurs sac en cas d'égalité, si le sac courant s est strictement meilleur la liste devient $[s]$ si il est égal L devient $L + [s]$ et sinon L est inchangée.

— Au lieu de coder les sacs possibles en binaire on peut aussi avoir une liste de triplets $(liste\,objets, volume, valeur)$. on génère alors les nouveaux sac en ajutant l'objet courant à la liste (si le volume ne depasse pas). Ca va deux fois plus vite.

— J'ai mis la version binaire car elle est générique mais probablement deux fois plus lente.

```
1
2  class Article:
3      def __init__(self, p, val, name):
4
5          self.valeur=val
6          self.volume= p
7          self.nom = name
8
9      def __str__(self):
10         return "{}:Volume {}, Valeur {}".format(self.nom, self.volume, self.
               valeur)
11
12 def Opt_exhaustif(Mag, Vmax):
13     # mag is a list of items
14     print( "Algo exhaustif, pour le magasin")
15     for a in Mag:
16         print(a)
17     k=0
18     cases= [([],0,0)]
19     for article in Mag:
20         # On parcoure les articles
21         # cases est la liste des sacs faisables avec les articles deja vus
22         # un element de cases  est sac
23         # un Sac est un  triplet (code binaire, volume , valeur)
24         # le code binaire [0,0,1,0,1] repr snte le cas avec les objets 2 et
               5
25
26         next_cases=[]
27         #Variable ou la liste des sac suivante est stock e
28         print("Generation de toutes les solutions valables avec les {}
               premiers objets".format(k+1))
29
30         for x in cases:
31             # on parcoure la liste courante des sacs
32             #Chaque sac peut donner deux nouveaux sacs
33             bincode=x[0]
34             volume=x[1]
35             valeur=x[2]
36             # on ajoute le meme sac, avec + 0 au code binaire car l article
                   n est pas choisi
37             next_cases+=[ (list(bincode) +[0], volume, valeur )]
38
39             if (volume+article.volume < Vmax):
40                 # si on peut ajouter l article    l'ancien sac on cree le
                       nouveau sac
41                 # on ajoute 1 au code binaire (objet choisi)
42                 # le volume et la valeur du nouveau sac son ceux de l ancien
                       + ceux de l objet
```

```
43                  next_cases+=[ (list(bincode) +[1], volume+article.volume ,
                        valeur+article.valeur,) ]
44
45          cases=list(next_cases)
46          #print (next_cases)
47          k+=1
48
49      # Recherche d'un meilleur sac
50      best= cases[0]
51      for somebag in cases:
52          if somebag[2] > best[2]:
53              best=somebag
54      print ("Opt is {}  de volume {} , One best is". format(best[2],best[1]))
55
56      for Art_index in range( len (best[0])):
57          if best[0][Art_index]==1:
58              print (Magasin[Art_index])
59
60
61  Magasin=[None]*8
62  Magasin[0] =  Article(100, 5000, "collier en or");
63  Magasin[1] =  Article(1000, 1, "spaghetti");
64  Magasin[2] =  Article(100, 5, "amandes d cortiqu es biologiques");
65  Magasin[3] =  Article(1, 50, "saffran");
66  Magasin[4] =  Article(1000, 1, "flageolets en conserve");
67  Magasin[5] =  Article(100, 10, "faux filet");
68  Magasin[6] =  Article(10, 2, "pur e biologique");
69  Magasin[7] =  Article(300, 10, "merguez");
```