Université des Sciences et de la Technologie Houari Boumediène
Faculté d'Electronique et d'Informatique
Département d'Informatique
LMD Master 1 RSD 2009/2010
Module ''Algorithmique avancée et complexité''

Travaux Dirigés
Série numéro 1 :
Outils mathématiques, comparaison, ordre de complexité

Exercice 1 : Utilisation de la preuve par récurrence

1. Montrer que $\sum_{i=1}^{n} i=\frac{n(n+1)}{2}$
2. Montrer que $\sum_{i=1}^{n} i^2=\frac{n(n+1)(2n+1)}{6}$
3. Montrer par récurrence (et non en recherchant des expressions séparées pour les deux sommes) que $\sum_{i=1}^{n} i^3=(\sum_{i=1}^{n} i)^2$

Exercice 2 : *Comparaison de croissances*

1. Trouver une valeur entière pour n telle que :
   a. 1 000 000 $\log_{10}(n+1)\leq n$
   b. 1 000 000 $\log_{10}(n+1)\leq n^2$
   c. 1 000 000 $\log_{10}(n+1)\leq n^3$
   Que se passe-t-il pour les valeurs supérieures ?
2. La première colonne du tableau suivant indique le temps d'exécution en microsecondes d'un programme pour une entrée de taille n. Pour chaque colonne, donner le temps d'exécution –l'ordre de grandeur ou une valeur exacte- de ce programme pour une entrée de la taille donnée :

	5	10	20	40	100	500
$\log_2 n$						
n	5 µs	10 µs	20 µs	40 µs	100 µs	500 µs
$n\log_2 n$						
n^2						
n^3						
n^5						
2^n						
3^n						
n^n						
2^{2^n}						

Exercice 3 : Comparaison

Classer les fonctions suivantes selon la vitesse de croissance. Les regrouper par ordre de croissance :

a) n-n^3+7n^5	e) $\log_2 n+n^{\frac{1}{2}}$	i) n^3	m) $(1,23)^n+n^{1,23}$
b) n^2	f) n	j) $n\log_2 n$	n) $n^{\frac{7}{4}}$+n
c) 2^n	g) $2^{\frac{n}{2}}$	k) $n^2\log_2 n$	o) $n^3+n^2(\log_2 n)^{10}$
d) ln n	h) $n^2+\log_2 n$	l) $n(\log_2 n)^2$	p) $\log_2 n+\log_2(\log_2 n)$

Exercice 4 :
Classer dans l'ordre croissant les complexités suivantes :

$O(n^2)$, $O(3^n)$, $O(2^n)$, $O(n^2\log_2 n)$, O(1), O(nlogn), $O(n^3)$, O(n!), O(log n),O(n)

Exercice 5 :

Quelles sont les complexités de :

- $T_1(n)=3n\log n+\log n$
- $T_2(n)= 2^n + n^3 + 25$
- $T_3(n)=k+n$ où $n\leq n$

Classer les fonctions dans l'ordre croissant.

Exercice 6 :

Supposons qu'on ait écrit une procédure pour additionner m matrices carrées n*n. Si l'addition de deux matrices carrées nécessite un temps un temps d'exécution en $O(n^2)$, quelle sera la complexité de cette procédure en fonction de m et n ?

Exercice 7 :

Supposons que deux algorithmes résolvent le même problème, et que l'un s'exécute en $T_1(n)=400n$ et l'autre en $T_2(n)= n^2$. Quelle sont les complexités de ces deux algorithmes ? Pour quelles valeurs de n doit-on préférer l'algorithme de complexité plus élevée ?

Exercice 8 :

Déterminer un algorithme qui teste si un tableau de taille n est un « tableau de permutation » (i.e., tous les éléments sont distincts et compris entre 1 et n) :

1. Donner un premier algorithme naïf qui soit quadratique
2. Donner un second algorithme linéaire utilisant un tableau auxiliaire

Exercice 9 :

1. Prouver que $n^2+n=O(n^2)$
2. Donner l'ordre de grandeur des expressions suivantes :
   a) $\frac{n^2+3n+1}{n+1}$
   b) $\frac{nlog(n)+n^2+\log(n)^2}{n+1}$
3. Si f(n) est une fonction en O(n), les affirmations suivantes sont-elles vraies ?
   a) $(f(n))^2=O(n^2)$
   b) $2^{f(n)} = O(2^n)$

Exercice 10 :

Soient f(n) et g(n) deux fonctions positives asymptotiques. En s'aidant de la définition de base de la notation , prouver que max(f(n),g(n))=θ(f(n)+g(n))

Corrigé

Exercice 1 :

1. Montrer que $\sum_{i=1}^{n} i=\frac{n(n+1)}{2}$

Preuve par récurrence :

- Vérification de l'égalité pour les premières valeurs de n
- Hypothèse de récurrence (HR) : supposer que l'égalité reste vérifiée jusqu'à une certaine valeur k (pour tout $n\leq k$, l'égalité est vérifiée)
- Montrer que l'égalité est alors vérifiée pour n=k+1

Vérification de l'égalité pour n=1 :

$\sum_{i=1}^{n} i$=1 et $\frac{n(n+1)}{2}$=$\frac{1(1+1)}{2}$=1

Vérification de l'égalité pour n=2 :

$\sum_{i=1}^{n} i$=1+2=3 et $\frac{n(n+1)}{2}$=$\frac{2(2+1)}{2}$=3

Hypothèse de récurrence :

L'égalité reste vérifiée jusqu'à un certain k≥2 (pour tout n≤k, $\sum_{i=1}^{n} i$=$\frac{n(n+1)}{2}$)

On montre que l'égalité est alors vérifiée pour n=k+1 :

$\sum_{i=1}^{k+1} i$=($\sum_{i=1}^{k} i$)+(k+1)

D'après l'hypothèse de récurrence, on a : $\sum_{i=1}^{k} i$=$\frac{k(k+1)}{2}$

Donc $\sum_{i=1}^{k+1} i = \frac{k(k+1)}{2}$+(k+1)= $\frac{(k+1)[(k+1)+1]}{2}$

Conclusion : Pour tout n, $\sum_{i=1}^{n} i$=$\frac{n(n+1)}{2}$

2. Montrer que $\sum_{i=1}^{n} i^2$=$\frac{n(n+1)(2n+1)}{6}$

   Preuve par récurrence :

   n=1 : $\sum_{i=1}^{1} i^2$=1 et $\frac{1(1+1)(2*1+1)}{6}$=1

   n=2 : $\sum_{i=1}^{2} i^2$=5 et $\frac{2(2+1)(2*2+1)}{6}$=5

   Hypothèse de récurrence : Egalité vraie jusqu'à n=k≥2 (pour tout n≤k, $\sum_{i=1}^{n} i^2$=$\frac{n(n+1)(2n+1)}{6}$)

   On montre que l'égalité est alors vérifiée pour n=k+1 :

   $\sum_{i=1}^{k+1} i^2$=($\sum_{i=1}^{k} i^2$)+$(k+1)^2$

   D'après l'hypothèse de récurrence, on a : $\sum_{i=1}^{k} i^2$=$\frac{k(k+1)(2k+1)}{6}$

   Donc $\sum_{i=1}^{k+1} i^2$=$\frac{k(k+1)(2k+1)}{6}$+$(k+1)^2$=$\frac{(k+1)[k(2k+1)+6(k+1)]}{6}$

   Il faut montrer que $k(2k+1)+6(k+1)$=((k+1)+1)(2(k+1)+1)

   1er membre : 2k^2+K+6k+6=2k^2+7k+6

   2ème membre : (k+2)(2K+3)= 2k^2+7k+6

   Conclusion : Pour tout n, $\sum_{i=1}^{n} i^2$=$\frac{n(n+1)(2n+1)}{6}$

3. Montrer par récurrence (et non en recherchant des expressions séparées pour les deux sommes) que $\sum_{i=1}^{n} i^3$=$(\sum_{i=1}^{n} i)^2$

   Pour n=1 : $\sum_{i=1}^{1} i^3 = 1^3$+1 et $(\sum_{i=1}^{1} i)^2$=$(1)^2$=1

   Pour n=2 : $\sum_{i=1}^{2} i^3$=1^3+2^3=9 et $(\sum_{i=1}^{2} i)^2$=$(1+2)^2$=9

   Hypothèse de récurrence : Egalité vraie jusqu'à n=k≥2 (pour tout n≤k, $\sum_{i=1}^{n} i^3$=$(\sum_{i=1}^{n} i)^2$)

   On montre que l'égalité est alors vérifiée pour n=k+1 :

   $\sum_{i=1}^{k+1} i^3$=$\sum_{i=1}^{k} i^3$+$(k+1)^3$

   D'après l'hypothèse de récurrence, on a : $\sum_{i=1}^{k} i^3$=$(\sum_{i=1}^{k} i)^2$

   De plus, d'après la réponse à la question 1. on a : $\sum_{i=1}^{k} i$=$\frac{k(k+1)}{2}$

   Donc $\sum_{i=1}^{k+1} i^3$=$(\frac{k(k+1)}{2})^2$+$(k+1)^3$

   Il faut montrer que $(\frac{k(k+1)}{2})^2$+$(k+1)^3 = (\sum_{i=1}^{k+1} i)^2$

   D'après la réponse à la question 1. on a : $\sum_{i=1}^{k+1} i$=$\frac{(k+1)(k+2)}{2}$

   Donc il faut montrer que $(\frac{k(k+1)}{2})^2$+$(k+1)^3 = (\frac{(k+1)(k+2)}{2})^2$

1^er^ membre : $\frac{(k+1)^2(k^2+4k+4)}{4}=\frac{(k+1)^2(k+2)^2}{4}=\left(\frac{(k+1)(k+2)}{2}\right)^2$

Donc il y a bien égalité entre les 1er et 2ème membres

Conclusion : Pour tout n, $\sum_{i=1}^{n} i^3=(\sum_{i=1}^{n} i)^2$

Exercice 2 : *Comparaison de croissances*

1. Trouver une valeur entière pour n telle que :
   a. 1 000 000 $\log_{10}(n+1)$≤n
      $\log_{10}(n+1)^{1000000}$≤n
      La fonction exponentielle de base 10 étant strictement croissante, on a : $10^{\log_{10}(n+1)^{1000000}}\leq 10^n$
      La fonction exponentielle de base 10 et la fonction logarithme de base 10 étant réciproques l'une de l'autre, on a : $(n+1)^{1000000}\leq 10^n$
      Réponse à la question : n_0=9
   b. 1 000 000 $\log_{10}(n+1)\leq n^2$
      De la même façon qu'à la question précédente : $(n+1)^{1000000}\leq 10^{n^2}$
      Réponse : n_0=9
   c. 1 000 000 $\log_{10}(n+1)\leq n^3$
      De la même façon qu'aux questions précédentes :
      $(n+1)^{1000000}\leq 10^{n^3}$
      Réponse : n_0=9

   Que se passe-t-il pour les valeurs supérieures ?
2. La première colonne du tableau suivant indique le temps d'exécution en microsecondes d'un programme pour une entrée de taille n. Pour chaque colonne, donner le temps d'exécution –l'ordre de grandeur ou une valeur exacte- de ce programme pour une entrée de la taille donnée :

	5	10	20	40	100	500
$\log_2 n$	2,32 µs	3,32 µs	4,32 µs	5,32 µs	6,64 µs	8,96 µs
n	5 µs	10 µs	20 µs	40 µs	100 µs	500 µs
$n\log_2 n$	11,6 µs	33,2 µs	86,4 µs	212,8 µs	664 µs	4480 µs
n^2	25 µs	100 µs	400 µs	1600 µs	10 000 µs	250 000 µs
n^3	125 µs	1000 µs	8000 µs	64000 µs	1 s	125 s
n^5	3125 µs	100000 µs	3,2 s	1,7 m	2,77 h	361,69 j
2^n	32 µs	1024 µs	1,05 s	12,72 j	40,2 millions de milliards d'années	-
3^n	243 µs	59049 µs	58,1 m	385517 années	-	-
n^n	3125	2,77	3325 milliards d'années	-	-	-
2^{2^n}	1,19 h	-	-	-	-	-

Exercice 4 :

Classer dans l'ordre croissant les complexités suivantes :

O(n^2), O(3^n), O(2^n), O($n^2\log n$), **O(1),** O(nlog n), **O(n^3),** O(n!), O($\log n$),**O(n)**

$\lim_{n\to\infty}\frac{n^k}{n^{k+l}}$=0, pour tout l≥1. Donc O(1)≤O(n)≤O(n^2) ≤O(n^3)

$\lim_{n\to\infty}\frac{k^n}{l^n}$=0, pour tous k et l tels que k<l. Donc O(2^n) ≤O(3^n)

Rappel sur les fonctions exponentielles et les fonctions logarithmes

On appelle fonction exponentielle, l'unique fonction *f* dérivable sur IR, telle que *f'=f* et *f(0)=1*. On note *f(x)=exp(x)*.

La fonction exponentielle est une bijection de IR sur]0,+∞[. L'image de 1 est unique et est notée *e* : *exp(1)=e*. Le nombre e est irrationnel et proche de 2,718. On peut donc adopter la notation *exp(x)=*e^x

$\lim_{x\to+\infty} e^x=+\infty$ $\lim_{x\to-\infty} e^x=0$ $\lim_{x\to 0}\frac{e^x-1}{x}=1$

$\lim_{x\to+\infty} \frac{e^x}{x}=+\infty$ $\lim_{x\to-\infty} x\,e^x=0$

Soit a un nombre réel strictement positif. Le logarithme népérien de a, noté $\ln(a)$ ou $\ln a$, est l'unique réel solution de l'équation exp(x)=a

La fonction logarithme népérien est la fonction notée ln, associant à tout réel dans $]0, +\infty[$ le nombre $\ln(x)$

$\ln(1)=0$ $\ln e=1$

Pour tout réel x>0, pour tout réel x : y=$\ln x \Leftrightarrow x=e^y$, exp($\ln x$)=x, $\ln e^y$=y

Pour tous réels $a>0$ et $b>0$: $\ln(ab)=\ln(a)+\ln(b)$

$\lim_{x\to+\infty} \ln x=+\infty$ $\lim_{x\to 0^+} \ln x=-\infty$ $\lim_{x\to+\infty} \frac{\ln x}{x}=0$ $\lim_{x\to 0^+} x\ln x=0$

On appelle fonction exponentielle de base a, a réel strictement positif, la fonction f définie, pour tout réel x, par f(x)=$a^x=e^{x\ln a}$

Pour tout réel a strictement positif, on appelle logarithme de base a la fonction notée $\log_a$, définie, pour tout réel x strictement positif, par : $\log_a(x)=\frac{\ln(x)}{\ln(a)}$

Le logarithme de base 10 (ou logarithme décimal) est aussi noté log : pour tout réel x>0, $\log x=\log_{10} x$