info719 : Rappels d'algorithmique et programmation C
TD 2 : complexité

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22, poste : 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/
wiki : http://www.lama.univ-savoie.fr/wiki

Exercice 1 : classes de complexité

Question 1. À l'aide d'une calculatrice ou d'un ordinateur, remplissez le tableau suivant. La première colonne indique la complexité en microseconde d'un algorithme pour une entrée de taille n. Pour chaque colonne, estimez le temps d'exécution de cet algorithme pour une entrée de la taille donnée.

	5	10	20	40	100	500
n	5×10^{-6} s	10^{-5} s	2×10^{-5} s	4×10^{-5} s	10^{-4} s	5×10^{-4} s
$n\log(n)$						
n^2						
n^3						
n^5						
2^n						
3^n						
n^n						
2^{2^n}						

Question 2. Quelles sont les classes de complexité "utilisables" ?

Si on estime que la loi de Moore est valide (la puissance de calcul double tous les deux ans), quelles sont les classes de complexité qui peuvent basculer du "infaisable" dans le "raisonnable" ?

Exercice 2 : Un exemple fondamental

Question 1. En cryptographie, on manipule régulièrement des nombres entiers de quelques centaines de chiffres. Il faut donc utiliser une structure de données différente du type `int` de votre langage favori. (Un `int` n'est stocké que sur un ou deux octets...)

Donnez un type de données que vous pourrez utiliser pour ces "grands entiers".

Question 2. Donnez les algorithmes de multiplication et addition de grands entiers. Quelles sont leurs complexités ?

Réfléchissez à la division et au modulo...

Question 3. Une autre opération fondamentale en cryptographie est la puissance. Étant donné un grand entier x et un autre grand entier n, on veut calculer le résultat de

$$x^n \pmod m$$

où m est un grand entier.

Essayez de programmer cette opération, et estimez sa complexité *en fonction du nombre d'opérations arithmétiques sur les grands entiers.*

Qu'en pensez-vous ? Cette opération est-elle utilisable sur des grands entiers ? Pouvez-vous l'améliorer ?

Exercice 3 : polynômes

Question 1. combien de multiplications sont nécessaires pour évaluer un polynôme de degré n sur un entier ? (On suppose que tous les coefficients sont non-nul.)

Question 2. quel est le nombre (exact) de multiplications et additions utilisées si on calcule les puissances "normalement", mais en réutilisant les calculs. (On suppose à nouveau que tous les coefficients sont non-nuls.)

Question 3. même question si on utilise la règle de Horner :

$$c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n \quad = \quad \Big(\ldots \Big(\big((c_n x + c_{n-1})x + c_{n-2} \big) x + c_{n-3} \Big) \ldots \Big) x + c_0$$

Exercice 4 : analyse du tri fusion

Le tri fusion est un algoritme de tri particulier qui a complexité théorique optimale. (On ne le démontrera probablement pas.) En pratique, d'autres tris sont généralement préferrables (tri par tas ou tri rapide).

Le problème est le suivant : on veut trier un tableau contenant les n premiers nombres entiers $(0, \ldots, n-1)$ et on ne peux que faire des comparaisons entre les différents éléments du tableau.

Question 1. un tri naif consiste à récupérer l'élément minimal du tableau et le mettre en première position. Ensuite, on s'intéresse à la suite du tableau et on recommence...

Écrivez l'agorithme correspondant (tri par selection) et donnez une analyse asymptotique du nombre de comparaisons utilisées en fonction de la taille du tableau. (Dans le pire cas, le meilleur cas et la moyenne...)

Question 2. le tri fusion est basé sur le fait que fusionner deux tableaux triés en un seul tableau trié est relativement aisé. Pour trier un tableau, on commence donc par le diviser en deux parties égales, on trie chaque morceau et on fusionne les résultats.

Écrivez la fonction `fusionne(t1,t2)` correspondante et analysez son comportement (nombre de comparaisons utilisées) en fonction de la taille des tableaux. (Meilleur cas, pire cas et moyenne.)

Écrivez l'algorithme `tri-fusion(t)` correspondant. Donnez une relation de récurrence pour le nombre de comparaison effectuées dans le pire cas et trouvez une approximation asymptotique de ce nombre. Qu'en est-il pour le meilleur cas ? Et en moyenne ?

Question 3. discutez des détails pouvant affecter l'efficacité de ces deux algorithmes.