TD Algorithmique n°5

PTSI B Lycée Eiffel

27 novembre 2012

Pour ce cinquième TD de l'année, nous allons revenir à des choses plus élémentaires que la dernière fois, en écrivant quelques programmes manipulant les nombres entiers. Comme d'habitude, Maple sait faire énormément de choses tout seul, mais on va tout reprogrammer à la main. Les seules commandes spécifiques que nous utiliserons dans nos programmes sont les commandes **irem(n,p)** et **iquo(n,p)**, qui donnent respectivement le reste et le quotient de la division euclidienne de l'entier n par l'entier p. Commençons avec quelques exercices classiques sur des histoires de diviseurs :

- Écrire une procédure Maple qui prend en argument un nombre entier n et dresse la liste de ses diviseurs (on se contentera de tester la division par tous les entiers plus petits que $\sqrt{n}$, 1 et n n'étant pas considérés comme diviseurs de n). Améliorer le programme pour qu'il donne également le nombre de diviseurs.
- Écrire une procédure Maple qui teste si un nombre entier n est premier (divisible uniquement par 1 et par lui-même) en testant sa divisibilité par tous les entiers compris entre 2 et $\sqrt{n}$.
- Écrire une procédure affichant la liste de tous les couples de nombres premiers jumeaux (dont la différence vaut 2) compris entre 1 et 100 000.
- Écrire une procédure Maple qui donne la décomposition en facteurs premiers d'un nombre entier (sous la forme d'une liste de diviseurs répétés autant de fois qu'ils apparaissent dans la décomposition).
- Un nombre entier n est parfait si la somme de ses diviseurs (autres que 1 et lui-même) est égale à $n-1$. Écrire une procédure déterminant si un nombre entier donné est parfait. Écrire une procédure donnant la liste des nombres parfaits inférieurs à 100 000.
- Écrire une procédure Maple qui calcule le pgcd de deux nombres entiers donnés à l'aide de l'algorithme d'Euclide (qui consiste à utiliser le fait que le pgcd de n et p est égal au pgcd de p et de r, où r est le reste de la division de n par p, et à recommencer jusqu'à obtenir un reste nul).
- Lé théorème de Bezout affirme que pout tout couple d'entiers n et p il existe deux entiers (qui ne sont pas uniques) pour lesquels $un + vp = pgcd(n,p)$. Pour les déterminer, on peut compléter l'algorithme d'Euclide de la façon suivante : on pose $u_0 = 1$, $u_1 = 0$; et $v_0 = 0$, $v_1 = 1$, et à chaque étape on calcule $u_{i+1} = u_{i-1} - qu_i$ et $v_{i+1} = v_{i-1} - qv_i$, où q est le quotient de la division euclidienne effectuée à cette étape de l'algorithme. Compléter la procédure de la question précédente pour calculer les deux coefficients de Bezout en plus du pgcd.

Enchaînons avec quelques autres exercices plus anecdotiques mais qui vous feront travailler un peu votre programmation :

- Écrire une procédure Maple qui prend en argument un entier n et renvoie le nombre obtenu en renversant son ordre d'écriture (par exemple, 14536 devient 63541).
- En déduire une procédure qui détermine si un nombre est un palindrome ou non.
- On considère l'algorithme suivant : on part d'un entier multiple de 3, et à chaque étape, on calcule la somme des cubes des chiffres du nombre précédent. Écrire une procédure Maple affichant tous les nombres obtenus, jusqu'à ce que la suite devienne stationnaire.
- Écrire une procédure Maple calculant $\sum_{k=1}^{k=n} \frac{1}{k}$, pour la plus petite valeur de n pour laquelle cette somme devient supérieure à nombre réel A donné en argument.

- On souhaite jouer avec Maple au jeu suivant : l'ordinateur choisit au hasard un nombre entre 0 et 100 (on pourra utiliser la commande rand(n) qui tire un nombre au hasard compris entre 0 et $n-1$), et le joueur propose des nombres jusqu'à tomber sur le bon. À chaque essai, l'ordinateur nous indique si le nombre proposé est trop petit ou trop grand (sinon on n'est pas près de gagner). On utilisera à cet effet la commande **readstat** qui permet de lire une valeur saisie par l'utilisateur. Ainsi, la commande a :=readstat("Quelle heure est-il ?") affichera la phrase « Quelle heure est-il ? » (on a le droit de mettre quelque chose de plus intelligent) et attendra qu'on tape un nombre pour l'affecter à la variable a. On écrira une procédure Maple qui indique le nombre d'essais effectués avant de trouver le bon nombre. Question subsidiaire : quelle est la meilleure tactique à ce jeu ? Combien d'essais environ devrait-on avoir pour trouver un nombre compris entre 0 et 100 ? Entre 0 et 1 000 000 ?