TD Algorithmique n°3

PTSI B Lycée Eiffel

16 octobre 2012

Listes et tableaux

Il existe plusieurs façons de définir des structures en Maple permettant d'accueillir plusieurs objets, et non un seul comme peut le faire une variable de type numérique ou même de type fonction. Nous nous contenterons pour cette fois-ci de définir ces différentes structures et nous verrons la prochaine fois comment les manipuler, notamment en quoi elles permettent de représenter facilement des polynomes.

Séquences

Une séquence Maple est juste un objet constitué de plusieurs objets (le nom du type Maple est exprseq), simplement séparés par une virgule. Par exemple, la commande

[> $s := a, 1.5, 2 * p - q, sqrt(12)$;

définit une séquence constituée de quatre éléments de types différents. Pour accéder à un élément de la séquence, on utilise les crochets. Ainsi, [> $s[4]$;, après la définition précédente, va renvoyer $\sqrt{12}$. Attention, on ne peut pas affecter individuellement les éléments d'une séquence, c'est-à-dire que [> $s[4] := 0$; renverra une erreur. On peut par contre utiliser lors de l'affectation d'une séquence l'objet NULL ([> $s := NULL$ crée une séquence vide), l'opérateur \$ pour répéter plusieurs fois un élément [> $s := a\$5$ crée la séquence a, a, a, a, a), et surtout la commande *seq*, faite pour créer des listes, dont la syntaxe est la même que celles des sommes par exemple : [> $s := seq(i^2, i = 1..4)$ crée la séquence $1, 4, 9, 16$.

Listes

Une liste est simplement une séquence placée entre crochets (le type Maple s'appelle list). Par exemple [> $l := [1, 2, 5]$ crée une liste à trois éléments. À la différence des séquences on peut affecter individuellement les éléments d'une liste. Pour le reste, ça marche pareil.

Ensembles

Un ensemble (de type set) est une séquence placée entre accolades. Un ensemble ne peut pas contenir deux fois le même élément (si c'est le cas, il n'apparaitra qu'une fois). On dispose de toutes les opérations mathématiques sur les ensembles : union, intersection (intersect), complémentaire (minus) notamment. Maple dispose par ailleurs d'un certain nombre d'outils combinatoires mais ce n'est pas vraiment notre sujet aujourd'hui.

Tableaux

Les tableaux, type array en Maple, sont (de loin) les structures les plus utilisées dans la liste que je viens de vous donner. Les tableaux ressemblent beaucoup aux listes, à la différence près qu'ils peuvent être multi-dimensionnels. Quand on crée un tableau, on a le choix entre préciser ses dimensions ou donner les éléments le constituant :

$[> t := array(1..4)$ crée un tableau à 4 éléments, pour l'instant non initialisé. $[> t := array(1..4, 2, 3)$ crée un tableau dont les deux premiers éléments valent 2 et 3, et les deux derniers restent non initialisés. Enfin, $[> t := array([[2, 3][1, 4]])$ crée un tableau à deux lignes et deux colonnes.

Premiers exercices

1. Écrire une instruction créant un tableau à 10 lignes et 10 colonnes, contenant (dans l'ordre de lecture usuel) les entiers de 1 à 100 (avec une boucle for, bien sûr, et pas à la main).
2. Écrire une procédure $diagonale(T, n)$, qui prend en argument un entier n et un tableau à n lignes et n colonnes, et calcule la somme des termes de sa diagonale.
3. Écrire une procédure vérifiant si un tableau est un carré magique ou non (somme des lignes, colonnes et deux grandes diagonales constante).

Un peu de polynômes

On peut utiliser des tableaux (ou plutôt des listes) pour représenter des polynômes : il suffit d'y stocker les $n + 1$ coefficients du polynôme (n étant le degré du polynôme). On peut alors effectuer divers types de calculs classiques sur les polynômes en écrivant des petits programmes faisant intervenir des tableaux.

1. Écrire un algorithme permettant de calculer le polynome dérivé d'un polynôme P (c'est-à-dire de créer un tableau stockant les coefficients de P').
2. Écrire un algorithme permettant de calculer le produit de deux polynômes données.
3. Écrire un premier algorithme permettant d'évaluer le polynome en un réel x donné, c'est-à-dire de calculer $P(x)$ (n faisant comme vous feriez à la main).
4. Le procédé de Hörner permet d'évaluer les polynomes de façon légèrement plus efficace que la procédure « normale » en faisant l'opération suivante : $p_0 + x(p_1 + x(p_2 + x(\ldots(p_n + x))))$. Vérifier que la valeur calculée est bien égale à $P(x)$. Écrire un nouvel algorithme utilisant le procédé de Hörner pour calculer $P(x)$. Comparer le nombre d'opérations nécessaire en utilisant chacune des deux méthodes.

Coefficients binomiaux

Vous avez déjà du croiser dans votre scolarité antérieure les coefficients binomiaux $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, nous en reparlerons en cours de maths un de ces jours. Pour l'instant, notre objectif est simplement de les calculer de façon efficace.

1. Écrire une première procédure calculant $\binom{n}{k}$ directement à l'aide de la définition (en écrivant explicitement le calcul des factorielles, sans utiliser la commande préexistante en Maple).
2. Écrire un deuxième programme utilisant la relation de Pascal : $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$. On utilisera une seule liste dont on modifiera au fur et à mesure les valeurs pour obtenir après i parcours de la boucle la liste de tous les coefficients binomiaux $\binom{i}{k}$. Tentez de comparer l'efficacité relative des deux programmes.