TP3 MPSI : Tableaux

fabien.viger@ens.fr
2 février 2005

But du TP

Dans ce TP, on va apprendre à manipuler les *tableaux* en Caml. Les questions **bonus** sont facultatives, ne les abordez que si vous pensez avoir le temps de finir le TP. Et bien sûr, **à n'importe quel moment, toute question est la bienvenue.**

Question 1 : Création

Laquelle de ces syntaxes marche bien pour créer le tableau $[1, 2, 3, 4]$? Attention, l'une d'elles *semble* marcher mais ne fait pas la chose voulue.

```
[| 1 2 3 4 |];;
[| 1 , 2 , 3 , 4 |];;
[| 1; 2; 3; 4 |];;
```

- En utilisant `make_vect`, créer un tableau de 1000 nombres flottants.
- À l'aide de `random__float` (tester la fonction avant de l'utiliser!), écrire une fonction `random_vect` qui crée un tableau de taille arbitraire et le remplit de nombres aléatoires entre 0 et 1.

Question 2 : Modification

`Caml` ne considère pas les tableaux et les autres variables de la même manière : une fois créés, ils peuvent être *modifiés* [1]. Cela rend parfois les choses moins claires au début. Pour se familiariser, essayer :

```
let x = make_vect 10 "a";;
let y = x;;
x.(2) <- "yo";;
x;;
y;;
```

Se rappeler à l'avenir que copier des vecteurs, ce n'est pas si simple. Le cas échéant, il faut utiliser `copy_vect`, qui crée une *nouvelle* copie.

Question 3 : Opérations de base

Écrire la fonction `sub`, de type `'a vect -> int -> int -> 'a vect`, qui extrait le sous-vecteur des éléments compris entre deux indices (inclus).

1. Cela est vrai aussi pour les chaînes de caractères

Écrire la fonction `cat`, de type `'a vect -> 'a vect -> 'a vect`, qui concatène deux vecteurs. Vérifier qu'on obtient le même résultat qu'avec les fonction intégrées de Caml `sub_vect` et `concat_vect`.

Essayer la fonction `map_vect` :

```
let v = make_vect 5 3;;
map_vect (fun x->2*x) v;;
```

Se servir de `map_vect` pour coder une fonction `rand_vect` "toute petite" telle que `rand_vect n x` renvoie un vecteur de n entiers aléatoires entre 0 et $x-1$.
Pour les plus curieux, aller voir la petite liste des fonctions intégrées de Caml, pour les vecteurs, dans l'Aide en ligne, "the core library", "vect".

Question 4 : Opérations

Écrire les fonctions `vect_min` et `vect_max` qui trouvent le minimum et le maximum d'un vecteur. Essayer de les coder pour que *tous les types* de vecteurs soient admis.

Question 5 : Polynômes

On peut identifier les polynômes aux vecteurs : par exemple, $x^4 + 2x^2 - \frac{1}{4}x + 3$ sera `[|3.0; -0.25; 2.0; 0.0; 1.0|]`, par ordre de degrés croissants : ainsi, le k-ème élément correspond au terme de degré k.
Écrire la fonction d'addition de deux polynomes (attention, les degrés peuvent être différents).
Écrire la fonction de multiplication de deux polynomes.
Écrire une fonction `deriv` qui calcule la dérivée d'un polynôme.

Bonus Écrire une fonction `string_of_poly` de type `float vect -> string` convertissant un polynôme (sous forme de vecteur) en jolie chaîne de caractères, du style `"4.0*x^3 - 3.5*x^2 + 1.2*x + 7.8"`.

Question 6 : Polynômes 2

Écrire la fonction `poly_val` d'évaluation d'un polynome en un réel x.
S'en servir pour coder une fonction `racine` qui, étant donnés deux réels a et b tels que $P(a)P(b) < 0$, calcule une racine de P par dichotomie.

Bonus Se servir de `poly_val` et `deriv` pour coder la méthode de la tangente de Newton. Comparer la rapidité des deux méthodes (Newton et dichotoomie) sur un gros polynôme, du style $x^{10001} + 10^{100}$

Question facultative (*): PGCD

Écrire une fonction `poly_div` qui fait la division euclidienne de deux polynômes, i.e. qui renvoie le couple (quotient, reste).
Écrire une fonction `poly_pgcd` qui calcule le pgcd de deux polynômes.
Les tester sur `[|1.0 0.0 0.0 0.0 1.0|]` et `[|1.0 0.0 1.0|]`

Question 7 : Tri par selection

Implémenter le tri par séléction pour un vecteur de *type quelquonque*. On pourra commencer par des fonctions auxiliaires, comme `min_sub_vect`, telle que `min_sub_vect v i j` renvoie la *position* du plus petit élément du vecteur $[v.(i), v.(i+1), \cdots, v.(j)]$; ou encore la fonction `swap_vect`, telle que `swap_vect v i j` échange les i-ème et j-ème éléments de v.

Tester la fonction sur un vecteur aléatoire (cf Question 1), puis sur
`[|"caml";"c";"'";"est";"super"; "cool"|]`

Bonus Implémenter le tri à bulle (bubble-sort)

Question 8 : Matrices

On peut faire des matrices en Caml. Par exemple, pour une matrice $a \times b$:

```
let m a b = make_vect a (make_vect b 0.0);;
```

Essayer cette fonction :

```
let test = m 3 5;;
test.(1).(3) <- 1.0;;
test;;
```

Essayer de comprendre le résultat des opérations ci-dessus. En fait, pour bien créer une matrice, on devrait plutôt faire :

```
let m a b =
 let v = make_vect 3 [| |] in
  for i=0 to (a-1) do v.(i) <- make_vect b 0.0; done;
  v;;

let test = m 3 5;;
test.(1).(3) <- 1.0;;
test;;
```

Heureusement, Caml nous donne une solution encore plus simple :

```
let test = make_matrix 3 5 0.0;;
```

Écrire la fonction de multiplication de deux matrices : si $M_{i,j}$ est l'élément sur le i-ème rang et la j-ème colonne de M, alors le produit matriciel des

matrices A $(p \times q)$ et B $(q \times r)$ est une matrice C $(p \times r)$ définie par :

$$\forall i \in \{0, \cdots, p-1\}, \quad \forall j \in \{0, \cdots, r-1\}, \quad C_{i,j} = \sum_{k=0}^{q-1} A_{i,k} B_{k,j}$$

La tester sur `[| [| 0.0; 1.0 |]; [| -1.0; 0.0 |] |]`.

Bonus Écrire une fonction `copy_matrix`. Attention! vérifier que ca marche bien `:-)`

Question 9 : Permutations

Écrire une fonction qui permute aléatoirement les éléments d'un tableau. Il faut donc qu'après l'opération, chaque élément ait une chance égale de se retrouver à n'importe quelle place. **Attention**, ce n'est pas si facile!

Question 10 : Sous-tableau de somme maximale

Étant donné un tableau d'entiers (positifs et négatifs) v, on veut trouver le *sous-tableau de somme maximale* : il s'agit donc de trouver le couple d'indice (i, j) maximisant $\sum_{k=i}^{j} v.(k)$.

- Écrire une fonction qui trouve ce sous-tableau de somme maximale, et qui renvoit juste cette somme.
- Si ce n'est pas déjà le cas, l'optimiser pour qu'elle tourne en temps *quadratique*
- (*) Écrire une fonction qui tourne en temps *linéaire*.

On testera les fonctions codées sur des tableaux d'entiers aléatoires (Question 3), de petite taille (pour vérifier "à la main"), et de grande taille (pour voir la vitesse des algorithmes).